This is a repository copy of *Urban energy simulation based on 3d city models: a service-oriented approach*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/156725/

Version: Published Version

URBAN ENERGY SIMULATION BASED ON 3D CITY MODELS : A SERVICE-ORIENTED APPROACH

P. Wate, P. Rodrigues,* E. Duminil , V. Coors

University of Applied Sciences Stuttgart
Schellingstraße 24,
70174 Stuttgart, Germany -
(parag.wate, preston.rodrigues, eric.duminil, volker.coors)@hft-stuttgart.de

KEY WORDS: Energy Simulation, Solar Potential Analysis, Web Service, SOAP, 3D City Models, CityGML

ABSTRACT:

Recent advancements in technology has led to the development of sophisticated software tools revitalizing growth in different domains. Taking advantage of this trend, urban energy domain have developed several compute intensive physical and data driven models. These models are used in various distinct simulation softwares to simulate the whole life-cycle of energy flow in cities from supply, distribution, conversion, storage and consumption. Since some simulation software target a specific energy system, it is necessary to integrate them to predict present and future urban energy needs. However, a key drawback is that, these tools are not compatible with each other as they use custom or propriety formats. Furthermore, they are designed as desktop applications and cannot be easily integrated with third-party tools (open source or commercial). Thereby, missing out on potential model functionalities which are required for sustainable urban energy management. In this paper, we propose a solution based on Service Oriented Architecture (SOA). Our approach relies on open interfaces to offer flexible integration of modelling and computational functionality as loosely coupled distributed services.

1. INTRODUCTION

Information & Communication Technology (ICT) have progressed to a point, that it is now possible to store, process and exchange large amount of data seamlessly. Furthermore, with decreasing hardware and software prices it is relatively easy to purchase storage and time-consuming computational processes (AWS, 2016) (GCP, 2016). This has provided the necessary impetus for the urban energy domain to look for solutions beyond conventional simulations. Towards this objective, one solution currently being looked at is proposing simulation strategies based on information from different sources. These information sources can be building registers, census, cadastre records and geospatial datasets. CityGML has shown a lot of promise in this regard. Furthermore, geospatial data of most cities are now readily available as open data in CityGML compliant virtual 3D city models (Berlin 3D - Download Portal, 2015) (3D City Model of New York City, 2015). This has empowered energy experts with an unique opportunity to use their knowledge to target specific energy field and model its behaviour (eg., variable voltage demand loads for appliances or energy conversion systems in distribution networks). This evolution is envisioned towards a broader goal to achieve sustainable urban energy management.

Urban energy management is an integrated approach. Its key objective is to provide a balanced sustainable energy solution to predict current and future urban energy needs. This approach has paved the way to many physical and numerical models (static and dynamic) with a focus on demand response, energy storage, district heating and cooling networks and supply strategies (Boyer et al., 1996) (Clarke, 2001). Energy domain experts have transformed their knowledge about physical models into efficient energy simulation systems. The developed energy simulation systems are computational software programs, tools and platforms providing different use-case scenarios under certain boundary conditions. Thus we have efficient simulation tools and softwares

operating independently to provide desired results. Nonetheless, in order to have accurate and realistic results it is highly advantageous to integrate them as a distributed simulation environment. However, these tools in general are standalone software programs supporting heterogeneous data formats making them difficult to interoperate.

Recently, Service Oriented Architecture (SOA) have been gaining a lot of momentum. SOA is an enabler that exposes software functionalities as *Web Services*. In order to realize this, SOA uses three key standard technologies (i.e. SOAP, WSDL, UDDI). Simple Object Access Protocol (SOAP) (Gudgin et al., 2016) defines a messaging standard based on XML and, with the help of HTTP, provides a communication protocol for accessing web services. Web Service Description Language (WSDL) (Christensen et al., 2016) is used to describe the web service access interface. Universal Discovery, Description and Integration (UDDI) (Clement et al., 2016) is a registry that allows advertisement and discovery of web services thereby providing the opportunity to dynamically bind a web service at runtime. Furthermore, to address the need of the geo-spatial community, the Open Geospatial Consortium (OGC) have proposed a standardized interface specification known as Web Processing Service (WPS) (Matthias Müller and Cauchy, 2016). WPS describe geo-computation processes along with their associated inputs and outputs requirements. Additional, it also defines the execution mechanism. A basic WPS implementation is published as WSDL description and made available as web services, where the WPS implementation acts as middleware.

This paper makes the following contributions:

- Identify the need for integrated urban simulation and the advantages of distributed simulation environment.

- Define an energy simulation use-case targeting urban scale.

- Proposes a Service Oriented Architecture based simulation enviroments.

---

*Corresponding author

• Show the applicability of our solution with a proof-of-concept.

The rest of this paper is organized as follows. Section 2. discusses the related works. Section 3. highlights the need for an integrated urban energy simulation framework and its advantages. Section 4. describe our distributed simulation concept with the help of an use case. Section 5. describes the implementation of our approach with a proof-of-concept. Finally, Section 6. concludes and presents future work

2. RELATED WORKS

2.1 Component-Based software Development (CBD)

Component-Based software Development (CBD) relies on reusability of well documented existing software components. The documentation addresses coding formats, standards, functionality and specifies well-defined interfaces (provided and required) with fault handling capability. In component based software development process, a system is divided into well-defined loosely coupled sub-systems or distributed components which are integrated at runtime without modifying them. The integration methods and guidelines are provided in Component Based Software Engineering (CBSE). The authors (Crnkovic and Larsson, 2001) proposed a component-based structure having three layers; namely: presentation, business and data. The business layer constitutes the software component which implements an underlying business logic as a useful reusable entity of new developing system. A component defines multiple interfaces to interact with user or other components as binary interfaces in the form of Interface Definition Language (IDL). IDL is parsed by intended component or user to generate stubs or proxy ports at their end in order to access the functionality or business logic implemented by serving component. A component model specifies the standard implementation of component and standard interfaces between components (Councill and Heineman, 2001) . There are three basic component models mentioned in CBSE literature: Component Object Model (COM) (Box, 1998), Enterprise JavaBeans (EJB) and Common Object Request Broker Architecture (CORBA) with their interface standards: Distributed Component Object Model (DCOM), Remote Method Invocation (RMI) and Internet Inter Object Request Broker Protocol (IIOP) respectively.

The well-known concept of accessing the native method implementation from Plain Old Java Object (POJO) class by loading pre-compiled object files as shared or Dynamic Linked Libraries (DLL) possess certain potential drawbacks due to lack of documentation availability, broken DLLs, their update frequency and availability. In a view of service-oriented architecture, this approach of accessing the native computation is not practical because updating broken DLL will require updating DLLs at all POJO classes location who are accessing the native computation. However, on other hand in service-oriented architecture, the native component (POJO class or Java Bean) offers a service through standardised interface and publishes it in central registry so that it can be fetched and consumed by the service client. Since, the provided interface is a standardised interface, it is independent of service implementation, which means even if native implementation is changed or updated, the service client need not to be updated because interface remains the same. In our case of web service implementation, the idea is to invoke the executable of simulation software from service implementation class and publish this class & its interface as web service. Whenever, this service is requested by the service client, the server will only need to have a simulation software installed and running in order to respond the client request.

2.2 Software Component and Web Services based approaches in Urban Energy Simulation

A software component based approach was initially developed for optimization of engineering systems using Component Architecture for Design of Engineering Systems (CADES) framework (Delinchant et al. 2007) (Delinchant et al., 2013). The physical models representing optimization system were converted into CADES component using component generator tool and were offered as an optimization service. (Gaaloul et al., 2011) have introduced the software component standard for building performance simulation using *black box* interoperability approach. In *black box* approach, the model features are accessed through *plug-in* and *plug-out* mechanisms. A *plug-out* allows models to be exported as software components and *plug-in* is other way round. These two mechanisms are interfaces to the models encapsulating their features and implemented functionality. The authors have proposed the ICAR norm for demonstration of building modelling and simulation. A ICAR norm was also used in service-oriented mechanisms for sharing of services between system components to provide service-oriented smart home architecture (Wu et al., 2007). In another context of business service approach, the TRNSYS based models concerning single zone building simulations were implemented in generic software framework and were made accessible through simple web application interface (De Coninck et al., 2011). The interface component was developed to define simulation information related to input and output variables, and simulation model in Domain Specific Language (DSL). The framework was presented as simulation tool for building product manufactures to demonstrate the impact of use of their products in energy saving and comfort improvement. In MEU (Management of Energy systems in Urban environment) project, a web service oriented architecture (WSOA) was implemented to develop web based decision support tool for urban energy planning and management (Rager et al., 2013). A WSOA offered three services: One intermediate service and two computational services. An intermediate service was used by client interface for orchestration of other two computational services. The computational services, an EnerGis (Girardin et al., 2010) and CitySim solvers were accessed as library for WSOA with centralized database to perform energetic calculations. (Delinchant et al., 2013) have created software component for heat conduction model and coupled it with an optimizer as web service. The library resource of building simulation models were encapsulated into a component offering web service and were consumed through plugins into analysis tools in a cloud computing environment.

Several urban energy simulation studies based on 3D city models have demonstrated the methodology for derivation of required spatio-semantic data by evaluating the geometry of 3D city models (Carrión et al., 2010) (Nouvel et al., 2014) (Strzalka et al., 2010) have stated two models to estimate heating energy demand at city scale and building level: Degree day model considers only transmission losses through outer building envelope while Energy balance model considers overall energy balance with transmission and ventilation losses along with solar and internal heat gains. The studies have also been carried out to utilize the spatio-semantic properties of city models encoded in CityGML standard for heating demand (Krüger and Kolbe, 2012) and total energy demand estimations (Kaden and Kolbe, 2013) at city scale. The process involves the extraction of geometric parameters such as boundary area, envelope volume, boundary surface tilt and orientation from the respective semantic components (*WallSurface*, *RoofSurface* and *GroundSurface*) represented in CityGML. Then, the implemented building energy simulation programs are used to compute heating energy demand using those parameters.

3. DISTRIBUTED URBAN SIMULATION APPROACH

Currently, urban simulation tools focus on modelling and simulating specific entity of urban domain (eg., Buildings). However, apart from buildings other urban entities (eg., Bridges, Vegetation, Roads etc.) should also be considered to achieve sustainable urban energy management. The heterogeneity within the urban energy domain makes it quite difficult to simulate all entities of an urban environment with a single simulation tool. This necessitates the need for an integrated framework. A framework designed from scratch could be a way forward. However, it is a time and resource consuming activity. Furthermore, the entire process may end up with redundant and erroneous implementations of similar functionalities. Additionally, as modelling experts (using efficient tools) are located at different places, it would be very helpful if they can use their preferred tools. A feasible alternative could be to extend the useful (valid, tested and well documented) model implementations by encapsulating them into software component through open interfaces.
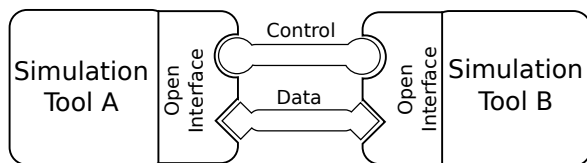

Figure 1: Open Interface

As shown in Figure 1 an open interface (control & data) acts as a communication point between simulation tools. The data interfaces enables exchange of information and functionalities during simulation run-time (i.e. common data models, processing time requirements). While on the other hand, the control interface manages the runtime simulation operations (eg., start, pause, resume and stop) among connected simulation tools. Open interfaces can potentially be used to connect different simulation tools to addresses complex urban simulation scenarios. As a consequence, the chain of connected tools results in a distributed urban simulation infrastructure. A distributed urban simulation infrastructure is not only the need of the hour, but is also highly advantageous. Such a simulation infrastructure can be beneficial at three different levels, namely; *computational, decision and business*

Computational Level: The sharing of data resources and functionality in a distributed computing environment could exploit the high performance computing capabilities of processors and hardware to predict future energy scenarios at high granular scale.

Decision Level: An integrated system consisting of individual expert's knowledge from their respective domain could facilitate urban energy planners and managers to assess the overall and specific primary energy requirements at varying spatial scales.

Business Level: An integrated system could bring new business opportunities for building equipment manufactures and energy supply companies by bringing simulation results as services depicting energy saving potential and comfort living to their customers.

4. USE CASE & CONCEPT

4.1 Use-case

As illustrated in Figure 2, to estimate solar heat gains for a building, shortwaves irradiation simulation algorithm is used with building geometry as input. However, at urban scale, the task becomes highly challenging. In order to estimate solar heat gains at urban scale we need to address three key complexities; ❶ *data processing and exchange*, ❷ *modeling* and ❸ *computational*. *Data processing* at an urban scale requires automated extraction and processing of building geometry parameters. Furthermore, the extracted data needs to be modelled into a compatible format required by different simulation programs. This is necessary as it will ensure that the generated data model can be easily fed to any simulation algorithm. *Modeling* complexity involves practical considerations of adjacency and proximity between buildings to account for detailed modeling of shadowing effects, while *computational* complexity relates to graphics hardware available for processing of large urban scenes. In order to realize our use
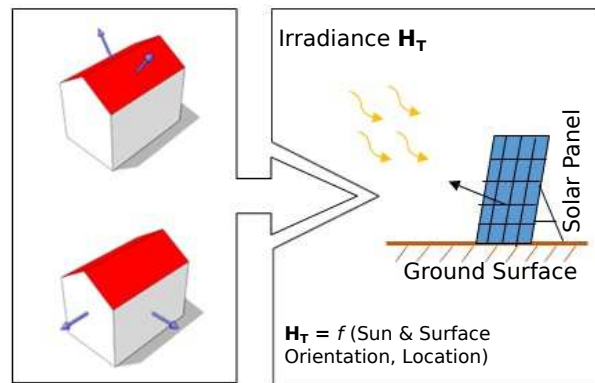

Figure 2: Coupling between processing and modeling

case we will use two simulation tools namely; a simulation platform SIMSTADT and INSEL. Using web services, we will show how to couple these two tools to perform processing and modeling functionalities on-the-fly and run a simulation targeting urban scale. The following subsections provides a brief overview about the simulation tools and our concept.

4.2 Simulation tools

SIMSTADT
Is an urban energy simulation platform jointly developed by the department of Energy and Geo-informatics lab of the University of Applied Sciences (HFT) Stuttgart (Nouvel et al., 2014). Energy simulations in SIMSTADT is accomplished using *workflows*. A *workflow* is made-up of many *workflow steps*. SIMSTADT however, defines four default *workflow steps* namely; `input`, `preprocessing`, `simulation` and `reporting`. Each *workflow step* is responsible to trigger the next *workflow step* in the chain.
INSEL
INSEL (INSEL, 2016) is a commercial simulation software developed by zafh.net research center. It provides a graphical programming language for the simulation of renewable energy systems. Furthermore, it uses a template based system to execute simulations. Using INSELS programming language, users can implement customs simulation templates on-the-fly and execute them. Key application fields of INSEL include but not limited to solar photovoltaics and solar thermal.

4.3 Concept

SIMSTADT is an urban energy simulation platform. It aims to empower urban planners and city managers alike to define low-carbon energy strategies with a variety of multi-scale energy analyses. Based on open 3D city models and with its modular workflow-driven design make SIMSTADT highly extensible simulation plat-

form. Figure 3 depicts the overall workflow-driven concept utilized by SIMSTADT. Functionality of each *workflow step* is explained below:
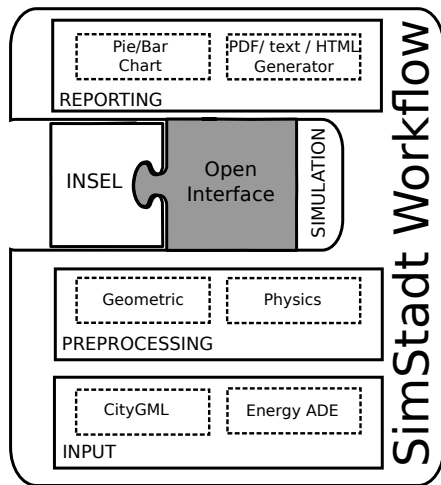

Figure 3: SimStadt

Input step

The `input` *workflow step* reads a CityGML model and extracts building information from *CityObject*. This information is then mapped into a internal data structure. The internal data structure assigns unique IDs to each wall and maintains the accuracy of buildings' geometric information with the help of vertices, edges and polygons. This step is essential to remove redundant information and provide spatio-semantic coherent data structure for energy simulation. Furthermore, the coherency and quality assurance of geometry data aids in reducing the uncertainty of simulation results. On successful completion of the `input` *workflow step*, `preprocessing` *workflow step* is triggered.

Preprocessing step

The `preprocessing` *workflow step* has a flexible design. This flexibility allows user defined modules to be integrated as plugins. SIMSTADT by default provides two plugins, namely; *geometric* and *physics*. The *geometry plugin* computes geometric parameters such as surface area, orientations, gross volume, building heights and window-to-wall ratios (Alam et al., 2014). While, building physics related parameters like construction type are derived from the *physics plugin*. Depending on a particular simulation requirement either or both plugins can be enabled.

Simulation step

On successful completion of `preprocessing` *workflow step*, `simulation` *workflow step* is triggered. Unlike `preprocessing`, which can integrate user defined plugins, `simulation` *workflow step* can integrate existing simulation tools. To showcase the flexibility of SIMSTADT and its advantage as a distributed energy simulation platform, we have coupled INSEL into SIMSTADTS' `simulation` *workflow step*.

Reporting step

The `reporting` *workflow step* is used to show the results of simulation as visual data. The current version of SIMSTADT supports 2D charts and 3D landscapes. Inaddition, results can also be exported as PDF, HTML, XML or as a plain text file. However, a planned future release will add support for 3D charts.

The following section explain in details the implementation of our proposed solution with the help of the use case defined in Section 4.

5. IMPLEMENTATION

5.1 System configuration

As described in section 4.3, the SIMSTADT platform can be customised to address different simulation requirements. To realize our use-case, we customized the `preprocessing` *workflow step* and enabled only the *geometric plugin*. The output of the `preprocessing` *workflow step* is fed as an input to INSEL via open interface through `simulation` *workflow step*. On successful simulation, the result is configured to be shown as a time series plot on a monthly scale. It is worthmentioning that the `input` *workflow step* in our configuration is using LOD2 CityGML model. To evaluate solar potential of surfaces we need to calculate their irradiance values. This is highly advantageous for space heating to assess contribution of energy source to predict monthly energy demand. Irradiance of surfaces is calculated with the help of a radiation processor. Using INSEL we have created a radiation processor template, it calculates monthly average daily irradiance (W/m^2) incident for a given orientation (surface azimuth and tilt) Optionally, to address weather conditions in a particular urban quarter we also provide as input the geo-coordinates of the building. Figure 4 shows a snapshot of INSELS monthly irradiation template. As shown, the template provides unique place holders for all the required inputs.

Figure 4: INSEL Template

5.2 Proof-of-concept prototype

This subsection explains the development of our prototype implementation. Figure 5 illustrates the flow of data at simulation runtime. As shown, the `Input` *workflow step* loads LOD2 data from the CityGML file. The extracted LOD2 data is passed as an input to the `Preprocessing` *workflow step* where the *geometric plugin* calculates the geometric parameters. The output of `preprocessing` *workflow step* is passed as an input to the `Simulation` *workflow step*. As INSEL is a remote web service, the required input parameters are sent to the remote server to generate an INSEL template and subsequently execute it.

The INSEL service is implemented as a POJO and deployed on Tomcat8 web server using Apache Axis2
(Apache Software Foundation, 2016). Apache Axis2 is a web services framework with support for hot deployment of services. Furthermore, it also supports automatic generation a WSDL. This WSDL is used to build clients in order to access remote services. Additionally, it can easily be extended to support new WS-*specifications. Moreover, it can be configured to define how clients can access the services with the help of *services.xml* file. It has support for two messaging paradigm, namely; `In-Only` and `In-Out`. Our implementation supports the `In-Out` messaging paradigm. Hence, for every request received a response (success or failure) will be returned. However, in order to send service
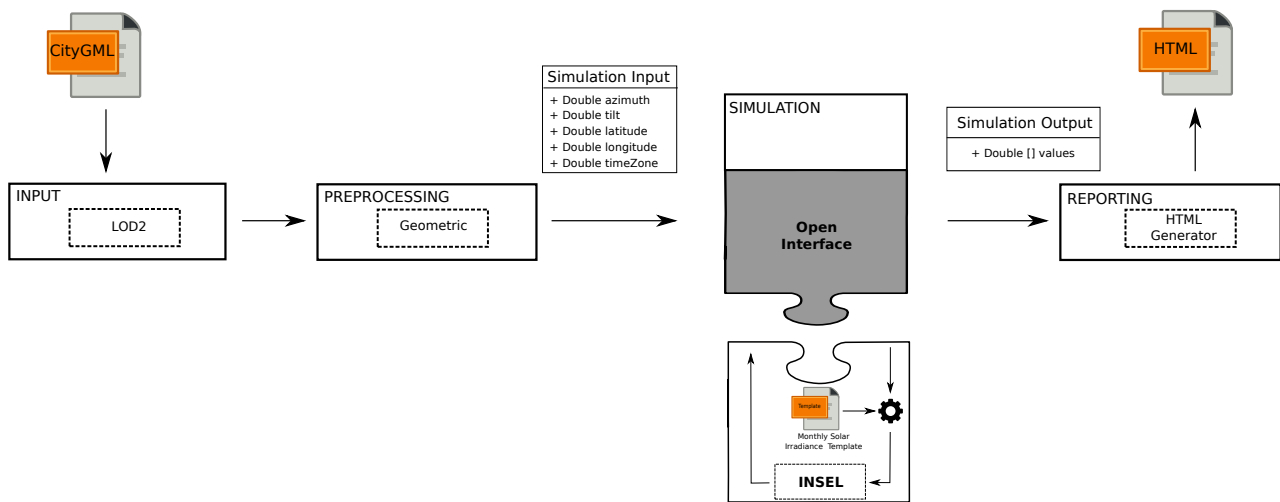
Figure 5: Runtime Data Flow

requests, we need to develop a client. Apache Axis2 provides efficient tools to generate native java code from a service WSDL. The generated clients can now send service requests as SOAP messages. For each incoming request, INSEL service has to accomplish two tasks. ❶ it uses a pre-defined template and fills the place holders on-the-fly to generate an *.insel* file. ❷ it invokes INSEL engine to executes the generated *.insel* file to run the simulation. The result of the simulation is passed to the `Reporting` *workflow step*. Since we have configured the `Reporting` *workflow step* to display the result as a web page, the *HTML generated* transforms the results and present them as a html page. Figure 6 shows the output as a time series plot of the amount of irradiance incident on a test building in Ludwigsburg, Stuttgart.
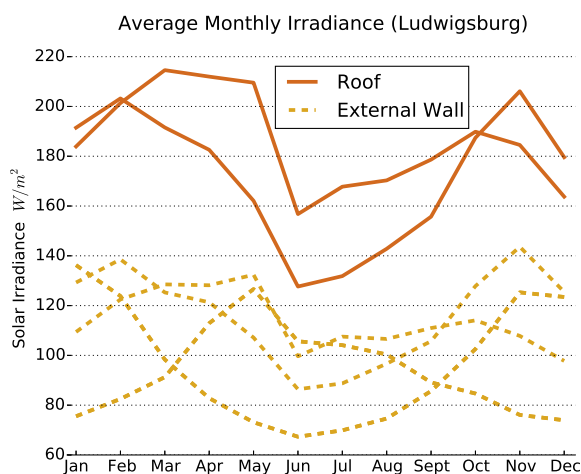

Figure 6: Reporting Output

6. CONCLUSION

In this paper, we introduced a service-oriented approach for urban energy simulation. Our approach uses the concept of open interfaces by connecting two different simulation tools, namely; SIMSTADT and INSEL. As proof-of-concept a prototype was developed to illustrate solar irradiation calculation as web service. The prototype demonstrated the feasibility of our approach by invoking the INSEL engine with an INSEL template generated from user inputs. The flexibility of generating dynamic INSEL template is highly suitable to carry out solar potential analysis in urban environment. Since a given urban façade possess arbitrary tilt and azimuth, which can be directly fed to the web service in order to have a quick primary assessment of solar potential of urban surfaces. While our solution provides a way forward, it only support CityGML LOD2 models. In furture we plan to support LOD3 and LOD4 CityGML models. Furthermore, we plan to extendend our approach to address two more aspects: namely; Automation and Orchestration.

1. Automation

   - Retrieve CityGML model from a database (3DcityDB) rather then individual file.
   - Automatically extract parameters from building boundary surfaces and provide solar radiation values per boundary surface.
   - Invoke multiple INSEL engines one per building boundary surface and combine the result as a single output.

2. Orchestration

   - Extend the SOA concept to other *workflow steps* namely; `Input`, `Preprocessing` and `Reporting`.
   - Propose a web-based user friendly tool to design custom user defined *workflows*.

REFERENCES

3D City Model of New York City, 2015. https://www.gis.bgu.tum.de/projekte/new-york-city-3d/.

Alam, N., Wagner, D., Wewetzer, M., von Falkenhausen, J., Coors, V. and Pries, M., 2014. Towards automatic validation and healing of citygml models for geometric and semantic consistency. In: *Innovations in 3D Geo-Information Sciences*, Springer, pp. 77–91.

Apache Software Foundation, 2016. Apache Axis2. http://axis.apache.org/axis2/java/core/index.html.

AWS, 2016. Amazon Web Services. https://aws.amazon.com/.

Berlin 3D - Download Portal, 2015. http://www.businesslocationcenter.de/en/downloadportal.

Box, D., 1998. *Essential Com*. Addison-Wesley Professional.

Boyer, H., Chabriat, J.-P., Grondin-Perez, B., Tourrand, C. and Brau, J., 1996. Thermal building simulation and computer generation of nodal models. *Building and environment* 31(3), pp. 207–214.

Carrión, D., Lorenz, A. and Kolbe, T. H., 2010. Estimation of the energetic rehabilitation state of buildings for the city of berlin using a 3d city model represented in citygml. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 38, pp. 31–35.

Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S., 2016. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

Clarke, J. A., 2001. *Energy simulation in building design*. Routledge.

Clement, L., Hately, A., von Riegen, C. and Rogers, T., 2016. Universal Description Discovery & Integration (UDDI) version 3.0.2. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm.

Councill, B. and Heineman, G. T., 2001. Definition of a software component and its elements. *Component-based software engineering: putting the pieces together* pp. 5–19.

Crnkovic, I. and Larsson, M., 2001. Component-based software engineering-new paradigm of software development. *Invited talk and report, MIPRO* pp. 523–524.

De Coninck, R., Devriendt, D., Thiesse, S. and Huberlant, B., 2011. Online software platform for dedicated product related simulations. *Proceedings of IBPSA*.

Delinchant, B., Gibello, P.-Y., Verdière, F. and Wurtz, F., 2013. Cloud computing services for the design and optimal management of buildings. In: *13th Conference of IBPSA 2013*.

Gaaloul, S., Delinchant, B., Wurtz, F., Verdière, F. et al., 2011. Software components for dynamic building simulation. In: *Building simulation conference, Sidney, Australie*.

GCP, 2016. Google Cloud Platform. https://cloud.google.com/.

Girardin, L., Marechal, F., Dubuis, M., Calame-Darbellay, N. and Favrat, D., 2010. Energis: A geographical information based system for the evaluation of integrated energy conversion systems in urban areas. *Energy* 35(2), pp. 830–840.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Karmarkar, A. and Lafon, Y., 2016. Simple Object Access Protocol (SOAP) version 1.2 part 0: Primer (second edition). http://www.w3.org/tr/2007/rec-soap12-part0-20070427/.

INSEL, 2016. http://www.insel.eu/index.php?id=301&L=1.

Kaden, R. and Kolbe, T., 2013. City-wide total energy demand estimation of buildings using semantic 3d city models and statistical data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 2, pp. W1.

Krüger, A. and Kolbe, T., 2012. Building analysis for urban energy planning using key indicators on virtual 3d city modelsthe energy atlas of berlin. In: *Proceedings of the ISPRS Congress*.

Matthias Müller, B. P. and Cauchy, A., 2016. Web Processsing Service (WPS) version 2.0. http://www.opengeospatial.org/standards/wps.

Nouvel, R., Zirak, M., Dastageeri, H., Coors, V. and Eicker, U., 2014. Urban energy analysis based on 3d city model for national scale applications. In: *Presented at the IBPSA Germany Conference*, Vol. 8.

Rager, J., Rebeix, D., Cherix, G., Maréchal, F. and Capezzali, M., 2013. Meu: An urban energy management tool for communities and multi-energy utilities. *Proceedings to CISBAT* pp. 4–6.

Strzalka, A., Eicker, U., Coors, V. and Schumacher, J., 2010. Modeling energy demand for heating at city scale. In: *SimBuild–Fourth National Conference. New York: IBPSA-USA*, pp. 358–364.

Wu, C.-L., Liao, C.-F. and Fu, L.-C., 2007. Service-oriented smart-home architecture based on osgi and mobile-agent technology. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 37(2), pp. 193–205.