SoftwareX 9 (2019) 271–281

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

SimApi, a smartgrid co-simulation software platform for benchmarking building control algorithms

Fabiano Pallonetto [a,*], Eleni Mangina [b], Federico Milano [c], Donal P. Finn [d]

[a] Energy Institute, University College Dublin, Ireland
[b] School of Computer Science, University College Dublin, Ireland
[c] School of Electrical & Electronic Engineering, University College Dublin, Ireland
[d] School of Mechanical and Materials Engineering, University College Dublin, Ireland

ARTICLE INFO

Article history:
Received 3 April 2018
Received in revised form 18 February 2019
Accepted 6 March 2019

Keywords:
Smart grid
Control algorithm
Building simulation software

ABSTRACT

This paper describes an open source smart grid software infrastructure for co-simulation between cloud-based energy management systems and a building energy model. The core component of the infrastructure is an API, which provides a protocol abstraction as a decoupling mechanism between the control algorithms and the building, thereby facilitating the development and testing of intelligent controllers. The open-source infrastructure can be utilised for the development and benchmarking of smart grid demand response algorithms aimed at reducing the energy consumption and the carbon footprint of buildings, while facilitating the integration of renewable energies into the power system.

Code metadata

Current code version	Mediator version 0.4, Platform version 0.8
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2018_29
Legal Code License	MIT License
Code versioning system used	Git
Software code languages, tools, and services used	Laravel 4.2, PHP, MySQL 5.4, Java
Compilation requirements, operating environments	BCVTB (https://simulationresearch.lbl.gov/bcvtb)
If available Link to developer documentation/manual	https://github.com/UCD-ElectricityResearchCenter/SimApi/tree/master/docs
Support email for questions	fabiano.pallonetto@ucdconnect.ie

1. Introduction

Building energy consumption depends on a significant number of factors, such as building envelope insulation, air infiltration, internal and solar gains, the efficiency of the heating system and occupant behaviour. Estimating the effect of each element requires an in-depth knowledge of the interactions between these factors and the related physics.

In recent years, harnessing the exponential growth of hardware computational capabilities, different approaches for estimating building energy consumption have been developed [1], such as Building Energy Simulation (BES) (white box), hybrid approaches (grey box) and statistical-based tools (black box) [2]. At the same time, the penetration of Renewable Energy Systems (RES) at the building level and the electrification of thermal loads have shifted the attention of system operators from generation to load controls via Demand Response (DR) programs. DR has been defined as "changes in electricity use by demand-side resources from their normal consumption patterns in response to changes in the price of electricity or to incentive payments designed to induce lower electricity use at times of high wholesale market prices or when system reliability is jeopardised" [3]. The objective of DR is to intentionally reshape the electricity demand in response to a signal sent by an aggregator or Transmission System Operators (TSO) providing energy flexibility to the power system. The changes can be quantified using different metrics and are directly correlated to building thermal characteristics [4],

* Corresponding author.
E-mail address: fabiano.pallonetto@ucdconnect.ie (F. Pallonetto).

https://doi.org/10.1016/j.softx.2019.03.003

occupant behaviour and the presence of storage or RES at a building level [5]. DR assets can alter the electricity demand curve dynamically, providing peak shaving, load shifting and load forcing measures [6]. Therefore, electricity loads are becoming a controllable resource for ensuring the resilience of the power system [7].

Using buildings as a controllable load implies the adoption of smart devices and EMS that can respond to a communication signal from system operators. However, the assessment of the benefits of the large-scale adoption of controllable load and building DR measures requires long testing cycles and considerable investment [8].

Furthermore, innovative combinations of technologies such as heat pumps, phase change materials and thermal or electric storage combined with renewable energy systems, can provide demand response capability to the building stock which become an energy flexible load. The power flexibility of these complex system, coupled with the thermal mass of the building can be analysed using advanced building energy modelling [9]. These software are effective to analyse the DR potential because they model the energy usage of building assets under different operational and environmental scenarios. Building modelling software can be also used to assess baseline and DR scenarios using large-scale simulation [10].

Although the use of BES software can facilitate algorithm testing, the development of control algorithms using BES software is constrained by the capabilities of the integrated development environments embedded in BES software architecture that revealed various limitations such as the lack of advanced instructions, data structures and debugging features. Moreover, the developed DR algorithms are building model agnostic without a complete redesign of the overall software infrastructure. Therefore, an open source smart grid co-simulation infrastructure to benchmark and decouple DR control algorithms for building simulation software can represent a solution in producing detailed and realistic studies of DR measures, in order to assess the benefits and to reduce the duration of smart-grid enabled appliances testing cycles [11].

Furthermore, in a smart grid, the implementation of a bidirectional communication channel from end users to generators still represents an open challenge due to the lack of standards and the high capital costs involved [12]. Moreover, the integration of dynamic demand-side assets and high penetration of RES generators add variability to supply/demand balance of the power grid [13].

Open architecture and co-simulation systems can boost the adoption of the technology and reduce the development and implementation life cycle of smart grid architectures [14]. In this context, buildings controlled by EMS can become a valuable resource to enable DR measures [11]. Hence, a co-simulation infrastructure for assessing control strategy performance on buildings in a smart grid scenario represents a fundamental research vehicle to accelerate the advancement of the smart grid concept and intelligent DR controllers [15].

2. Software objectives and requirements

This section describes the state of the art for smart grid co-simulation frameworks. The selected frameworks presented have been analysed on the basis of the requirements for smart grid simulation tools reported in [8] and detailed in Table 1.

The requirements are divided in three main groups: general, syntactic and control requirements. The general requirements concern the basic functionality and the extendibility of a co-simulation software such as the data logging capabilities [GR2] or the flexibility to different single data types and compositions of data. The syntactic requirements describe the communication protocol and the availability of the resources. The last group is the control requirements. This category illustrates the synchronisation and control capabilities of the system

In the last ten years, several frameworks for smart grid simulation have been developed, and associated research reports have been published. In Table 2, six packages have been examined based on the criteria selected. As noted in Table 1, none of the co-simulation software evaluated meets all the criteria. However, the MosaikApi [16] is the most flexible and scalable middleware. Such an infrastructure can be used for simulation of a smart grid scenario where building models are simple and do not require complex computational capabilities or interfaces.

Nevertheless, the integration with white box building energy simulation packages, such as TRNSYS,[1] EnergyPlus[2] or ESP-R,[3] often requires a considerable effort in software integration [17]. The presence of modelling features, the software usability and the computational capabilities are among the characteristics whose define different BES packages [18]. For instance, ESP-R is primarily used for design decision support and not for testing of control systems [19]. Similarly, TRNSYS is a software package that implements a component-based simulation [20]. Each component could be a fundamental element of the system, such as a valve or a pipe or it could be a sophisticated building model with a complex control algorithm [21] .

Compared to the previous software, the use of EnergyPlus in a co-simulation environment is facilitated by the presence of an Functional MockUp Unit (FMU) component [22]. The component allows the lazy coupling with a third party software, requiring only the interface description. The TRNSYS implementation of the same interface is more complex as noted in [23]. Therefore, TRNSYS was not considered as the first choice for the development of the co-simulation API.

The BES integration requirement and the need for data logging capabilities with a web based user interface in order to benchmark control algorithms is identified as a research gap within the literature [24]. Logging capabilities and BES integration feature are an essential requirements to assess the benefit of a smart grid ready building to the overall power infrastructure. Consequently, such technical requirements and the lack of protocol abstraction found in the co-simulation frameworks examined, represent a research gap to address in the current paper. The contribution of the paper to address such a gap is the development of a novel open source software infrastructure for building control and communication which will be presented in the next section.

3. Software description

The SimApi platform is designed to interface a building simulation system to a control algorithm in a programming language agnostic method exploiting a cloud interface based on the API. The objective is to abstract the operation of the building actuators from the controller, thereby decoupling the control algorithm from the building model. A simulation can be triggered via the API or the associated web dashboard to assess the baseline scenario.

3.1. Software architecture

The objective of the developed framework is to provide a co-simulation interface between an EnergyPlus calibrated building model and a generic cloud-based control algorithm. The software implements a restful API [30] to allow for remote control of a building simulation. The API was developed using a Model View Control (MVC) design pattern. The software stores all the requests

[1] TRNSYS homepage: http://www.trnsys.com/.

[2] EnergyPlus homepage: https://energyplus.net.

[3] ESP-R homepage: https://github.com/ESP-rCommunity.

Table 1
Smart grid co-simulation framework requirements.

Category	Requirement	Description
General Requirements	[GR1] Variable Entity I/O	The system is flexible on the number of I/O required depending on the parameter configuration of the model.
General Requirements	[GR2] Data logging	The tool must log all data that is provided by the simulators for later evaluation.
General Requirements	[GR3] Execution on multiple servers	The tool must be able to start and execute simulators across multiple computation nodes and servers.
General Requirements	[GR4] Integration with BES	The tool should be able to integrate with BES or other simulation environments.
Syntactic Requirements	[SR1] Standardised simulator Application Program Interface (API)	A defined minimalistic API is defined with basic CRUD functionalities.
Syntactic Requirements	[SR2] Data available via API	The API exposes all available model data, including time-invariant data, such as configurations and settings.
Syntactic Requirements	[SR3] Transmission of complex data types	The API can submit (and receive) complex data types from the entities of the participating simulators.
Syntactic Requirements	[SR4] Time-stepped simulations	The tool should support time-step simulations.
Control Requirements	[CR1] Control strategy API	The defined API allows control strategies to interact with the simulated physical part of the smart grid. The objective is to decouple physical and ICT objects and allows to develop control strategies independent of specific simulation models.
Control Requirements	[CR2] Agent synchronisation	Two ways control strategies synchronisation of the simulations.

Table 2
State of the art of smart-grid co-simulation frameworks.

Requirement description	Variable I/O	Data Logging	Execution on multiple server	Integration with BES	Standardised APi	Data Access API	Transmission Complex Data	Time stepped simulation	Control Strategy API	Agent synchronisation
Requirement	[GR1]	[GR2]	[GR3]	[GR4]	[SR1]	[SR2]	[SR3]	[SR4]	[CR1]	[CR2]
Venus-C [25]	×	✓	×	×	×	×	✓	✓	×	×
Jade [26]	✓	×	×	×	✓	✓	✓	✓	×	×
Smart Cities [27]	✓	×	×	×	×	×	×	×	×	×
ILIas [28]	×	×	×	×	×	×	×	×	✓	✓
EPOCHS [29]	✓	×	×	×	×	×	×	×	✓	✓
Mosaik API [16]	✓	✓	✓	×	✓	✓	✓	✓	✓	✓

from the API in a relational database [31], while it retrieves the sensor data from the building simulation. The communication is ensured by a BCVTB background process. Both the BCVTB and the SimApi are connected to the database via Java Database Connectivity (JDBC) technology which defines the access protocol to the database.

The implemented interface reads and writes data to the database for each time step simulated. The database is connected to the calibrated building model [32] and tested using EnergyPlus [33]. The layout of the system is illustrated in Fig. 1 and the main components are:

1. EnergyPlus BES building model. This is a software model of a physical building. The building simulation takes into account the occupancy profile, the weather data and the renewable energy system installed at building level.
2. BCVTB framework. This provides a connection between the simulation model and the database. The role of the actors is to store data points from the sensors and send the actions to the models.
3. API interface. This provides the conceptual protocol abstraction of the implemented API, for both CRUD operations and control flow management.
4. Web server. This exposes the API endpoints defining the data exchange protocol with the building simulation components; it also ensures data validation and connection to the database and implements the transport layer to and from the components.
5. MySql Database. This provides a persistent layer to store the data points for each time step and synchronises the simulations with the control. The persistent layer is also used for analytics.
6. Mediator agent. This is a background process that retrieves a control token over an instance and triggers the BCTVB framework and the building model simulation.
7. EMS. This embeds the control algorithm for the building model. It can be language agnostic, and consumes the API endpoints while receiving the synchronisation signal from the system.

The web server (4) accepts HTTP requests from one or multiple EMS (7) connected to the cloud. The requests are stored in a relational database (5) and are retrieved by the simulation agent (2), called actor. The actor sends the runtime requests to the EnergyPlus simulation (1). EnergyPlus uses the data as control input for each time step. The software platform is divided in three main components:

1. Web interface. This component allows the interaction with the building simulation model, the user management and the data exploration of the stored simulation instances.
2. Initialisation and halt. These two endpoints manage the interaction with the mediator component. They trigger the execution of the building model simulation and the end of the simulation, resetting the environment variables.
3. Flow control. The flow control endpoints interact with the building simulation environment via the database tables and are responsible of receiving, storing and sending data to the simulation.

The next section describes in more detail each of the components from implementation and software functionality perspective.

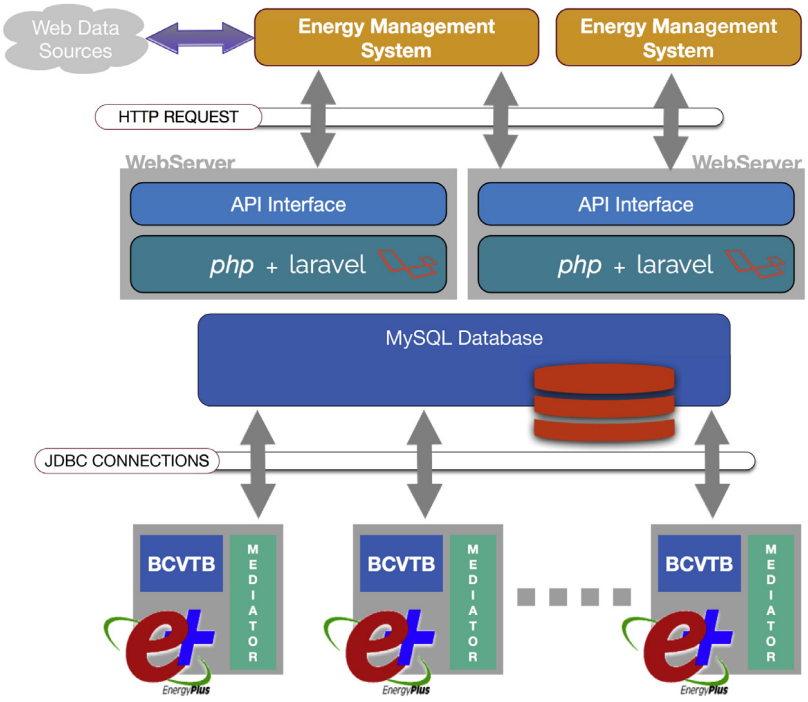

Fig. 1. Architecture of the cloud base system interface.

3.2. Software functionalities

The API can be consumed by the control algorithm on a Hypertext Transfer Protocol (HTTP) transport layer over the cloud an it represents the abstraction of the protocol communication between the agents involved in the co-simulation infrastructure, the building model and the EMS. The API resources are described in the API Blueprint, using a standard description format [34].

A web user interface has been developed for the visualisation of simulation results and for running the building simulation. Fig. 2 shows the web interface captured after a simulation. On the left hand side, the user can evaluate the inside temperature of a building zone and the associated energy consumption is given on the right hand side.

The web user interface can also manage the user type and settings. The system is designed to give access to two user categories; the *admin* users and the *general* user. In addition to all the capabilities of the general user, the *admin* user has access to the endpoints for user creation, change of details and deletion of generic users. The general user has permission to:

1. Authentication and password management;
2. Create a new simulation instance;
3. Enlist all the instances created;
4. Delete an instance and all the data points;
5. Extract the data points of each simulation instance;
6. Trigger initialisation and flow control endpoints.

The *general* user authentication and privileges have been used to test different scenarios and algorithms with the one residential building simulation model. However, it should be noted that the framework is designed to handle various and heterogeneous building models.

The initialisation and halt endpoints have been developed to interact with the mediator component and set the initial configuration data. After the initialisation of a new instance, the API responds with an identification number (`instance-id`). In a multitasking environment, different instances can begin in parallel and exchange data at each time step following the execution of the building simulation model associated. The authenticated user agent can start a building simulation at any point using the `instance-id` reference. The objective of the initialisation endpoints is also to set or retrieve initial parameters such as the location of the building simulation model or the BCVTB executable. Three available endpoints show the configuration data and can be used to override the default settings. Two main HTTP methods can be invoked on the initialisation endpoints: `GET` and `PUT`. The `GET` method reads the parameters while the `PUT` overwrites the parameters default. The initialisation endpoints are:

1. `sensor`: this enlists the sensors and actuators available for the selected building model.
2. `appliance`: this shows a list of controllable appliances and the control interface for each appliance.
3. `inside-temperature`: this endpoint enlists the temperature setpoint for each temperature sensor.

As illustrated in Fig. 3, after the initialisation is complete, the `begin` command triggers the simulation and sends the value to the initialisation value to the building model. Additionally, the `halt` command is a feature to force a simulation to stop at any point after the warm up of the model. When the initialisation is complete, then the main client–server control flow controller interacts with the algorithms.

After the `begin` command is invoked, the building model simulation starts and the control flow component drives the simulation exchanging data with the EMS. During the execution, the user agent can check the simulation status using the command `getBegin`. It is also possible to retrieve the time step progress using the `time` command. The `getsensor` endpoint retrieves the sensor readings for the actual simulation time step. The `next` endpoint pushes a control action to the building model. Typical actions are represented by binary operations or threshold value rules. At the building level, the actuator can control a circulation pump, the heating system or a particular appliance. The instruction is sent through a JavaScript Object Notation (JSON) format in the body of the call. The command also allows the simulation

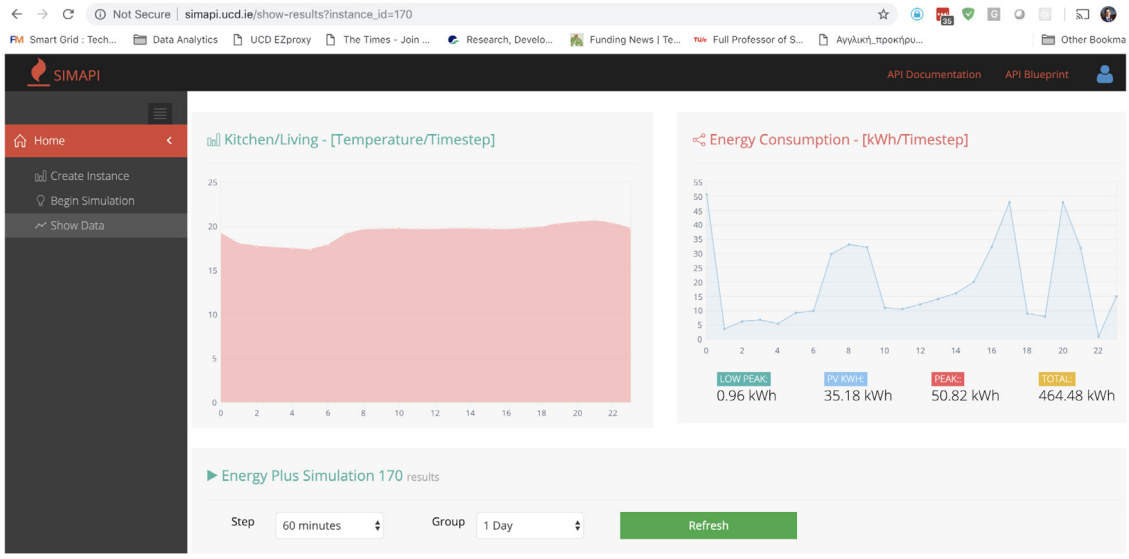

Fig. 2. Platform web interface to analyse building simulation data.

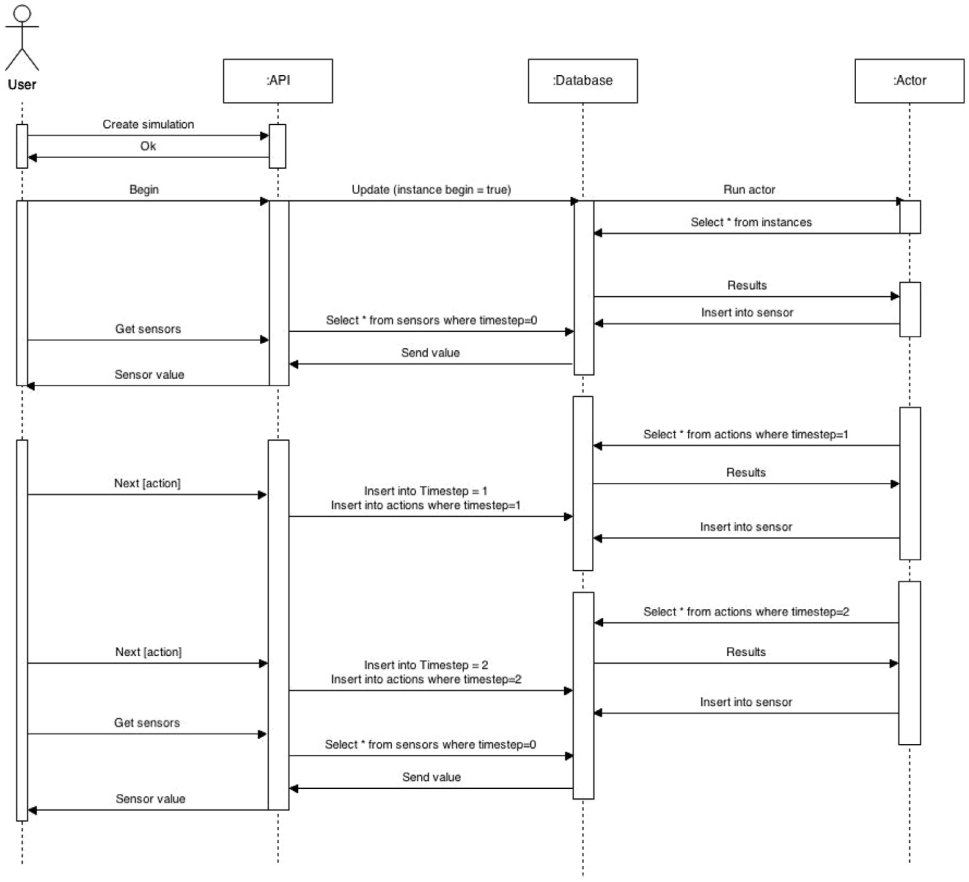

Fig. 3. UML, data exchange between an EMS and the BCVTB actor connected to the building simulation via the database.

director component to continue the simulation to the next time step. The JSON data structure attached to the command has the objective of describing the list of systems controlled and, for each system, the value set for each controlled parameter.

The simulation ends releasing the control to the initialisation and halt component that triggers the mediator component to clear the environment variables and reset the status of the instance to complete. As illustrated in Table 3, the developed infrastructure satisfies all the requirements identified from the literature and described in Table 1. Although the development has focused on the control strategies (requirements [CR1] and [CR2]), the system inherited the functionalities and capabilities of the foundation layers such as BCVTB, Relational DataBase Model (RDBMS) such as the transmission of complex data [SR3] and the execution on multiple server [GR3] ensured by the database capabilities.

Table 3
Analysis of the developed co-simulation framework.

Requirement description	Variable I/O	Data Logging	Execution on multiple server	Integration with BES	Standardised APi	Data Access API	Transmission Complex Data	Time stepped simulation	Control Strategy API	Agent synchronisation
Requirement	[GR1]	[GR2]	[GR3]	[GR4]	[SR1]	[SR2]	[SR3]	[SR4]	[CR1]	[CR2]
SimApi	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Consequently, requirement [SR2] has been check marked because of the standard JDBC capabilities embedded in the RDBMS that exposed a standard data access API. The same assumption is valid for [SR4] that is delegated to the combination of BCVTB and EnergyPlus.

4. Implementation example of a rule-base algorithm

A simple rule-based algorithm was developed to test the functionalities of the software platform on a calibrated residential building model developed using EnergyPlus. The building has an electric heating system connected to a Thermal Energy Storage (TES) and PhotoVoltaics (PV) system of 6 kWp. The selected building was fully instrumented and the EnergyPlus model has been calibrated with metered data. Fig. 4 shows the hourly based cumulative simulated electricity consumption for 2014 versus the measured data. Additionally, on the right vertical axis, the average Mean Bias Error (MBE) index for the calibration is reported on hourly basis.

The control system implemented has the objective of reducing the energy expenditure using a time of use electricity price while maintaining the thermal comfort in the building. The time of use electricity price is divided in three tariffs depending on time of consumption, 0.30 (€ · kWh) during the peak (17:00–19:00), 0.1 (€ · kWh) during night tariff (00:00–06:00) and 0.2 (€ · kWh) during off-peak (06:00–17:00 and 19:00–00:00). The algorithm simply charge the TES two hours before peak tariff (15:00–17:00) and exploit the thermal energy stored during the peak time. The algorithm control flow is illustrated in Fig. 5 and it was implemented in Java.

The rule-based algorithm creates and uses a single instance of a HTTP client class during its execution for all communication to the API. In the code repository, a utility class that outputs all the initial settings and the static variables such as temperature, server name and others was developed.

The program has two conditions based on the sensor reading at each timestep. The first condition checks whether the data point for the timestep is produced or not. The timestep counter is incremented only if a new data point is generated. Whereas the second condition checks if the control needs to disable the heating system.

The second condition is verified only between 09:00 and 15:00, which represents the period when the heating system is disabled if there is no electricity generation from the PV. Between 15:00 and 17:00, the controller enables the heating system, charging the thermal storage. The control instructions and sensor readings are stored in the SimApi database during the whole simulation.

The rule based algorithm shows an expected behaviour, shifting the electricity consumption to off-peak hours. The baseline system is controlled only by the thermostatic set point, so it exhibits a greater overall electricity cost during the day-time hours. Additionally, as illustrated in Fig. 6, the baseline operation of the heating system during price peaks (1700 to 1800 hrs) results in high expenditure periods due to the structure of the price tariff, which penalises peak consumption. These peaks are not present for the rule based controller demonstrating the effectiveness of the simulation in reducing the peak expenditure. A more extensive example of the use of the cloud platform for the assessment of DR algorithms in residential buildings can be found in [35].

5. Experimental results and discussion

The developed infrastructure is a cloud-based platform for co-simulation of EMS and BES system. The approach used to develop the infrastructure required the interaction of different components such as the API, web server, BCVTB and EnergyPlus. Each component was tested in isolation during the development phase. However, assembling the components in a single cloud-based system can lead to deadlocks and inefficiency caused by the interaction. Therefore, a cloud-based testing was necessary to evaluate the overall system performance based on standard criteria.

The objective of the test was to ensure the reliability of the software, identifying bottlenecks and synchronisation errors which, if not monitored, can output altered information. As best practise, a standard performance environment was used for testing the stability and the scalability of the cloud-based software [36].

The test was performed using a dataset of API calls extracted from a simulation execution of the rule-based algorithm described in the previous section embracing a one week period, from 1st August to 7th August 2017. The evaluation of the algorithm shows the average computational time for each timestep is approximately 5.4 s. Such a time interval gives a sufficient time gap for the EMS to update the settings and to compute a new action. In the latter case, the sensor readings are retrieved and a new action is added to the database. The overall average simulation time for a week is 1.1 h. The API calls were invoked by a client using Jmeter package [37] to ensure the parallelisation of the test. The results were stored in a test database for the analysis. The following section describes the physical infrastructure of the test environment.

5.0.1. Physical infrastructure

The Performance environment is composed of two servers; an App Server and a Database Server. All co-simulation components are deployed upon the App Server and the Database Server hosts a single MySQL database instance. The Performance environment DB Server was configured to manage all the data requirements involved for the simulation. The Performance environment is detailed in Table 4. The servers used were standard test instances of the Amazon AWS Cloud Platform [38]. The data retrieved from the Performance environment for the analysis of the results were:

- Database Server: temporal (1 min) snapshots of Database (DB) utilisation (percentage)
- Application Server: temporal (1 min) snapshots of CPU utilisation (percentage)

In the following sections, the results from the stability analysis and the scalability analysis are outlined.

Fig. 4. Cumulative annual electricity consumption on an hourly basis: simulated, metered and MBE (2014).

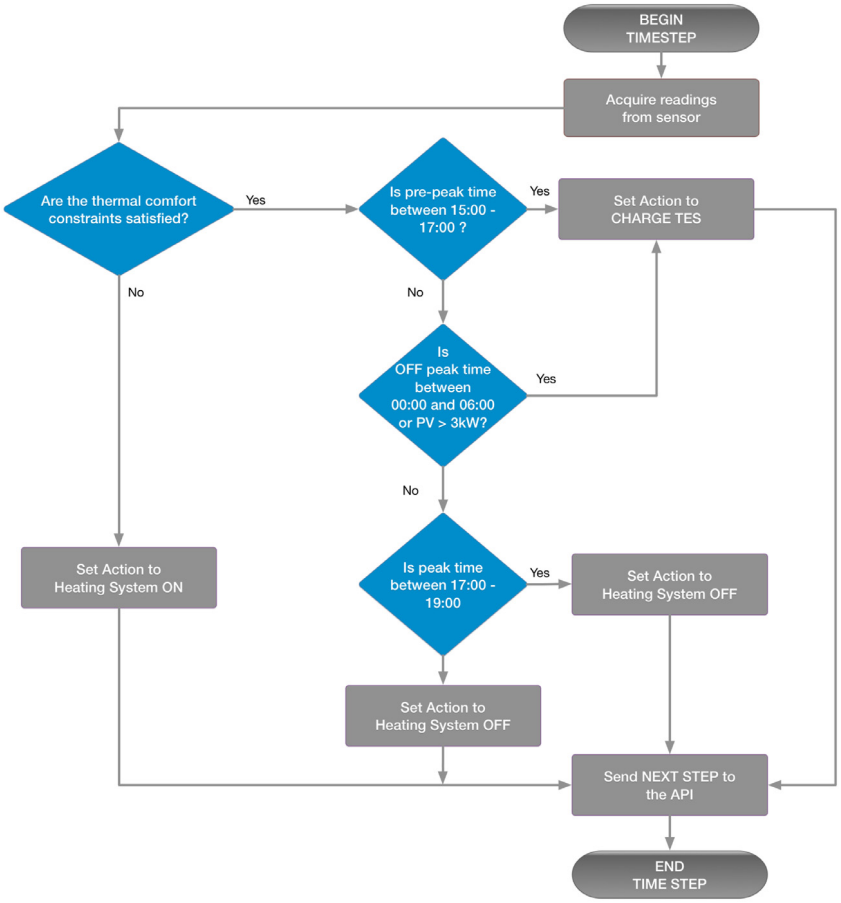

Fig. 5. Rule-based control flow algorithm.

5.1. Stability analysis

A test harness of 58 simulations was used to evaluate the stability of the API response and the framework. Each simulation had 5762 API endpoint calls. The total number of test calls were 334,196 calls. The test has two objectives:

- Ensure that 100% of the API calls were stored in the database (DB);
- Evaluate the presence of bottlenecks or server unresponsiveness in the execution via the analysis of the Round Trip Time (RTT) with an established threshold of 1000 ms.

The output of the test was a binary value passed/failed.

5.1.1. Test results

The number of API calls sent to the web server was compared with the number of rows in the *Timestep* in the database. If the number of rows and API calls matched consequently the test was deemed passed.

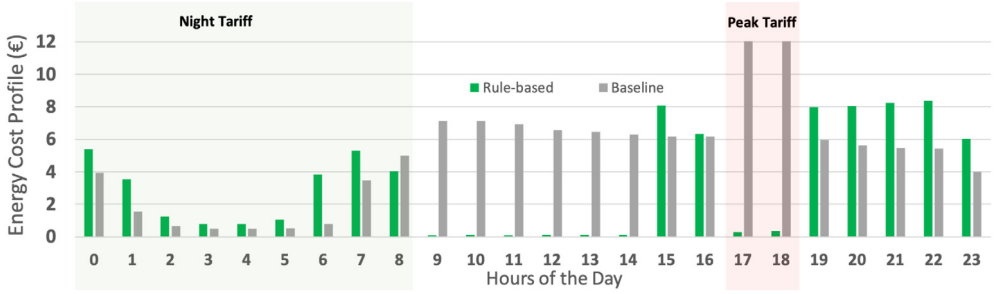

Fig. 6. Rule-based versus baseline, electricity expenditure over a week using time of use tariff.

Table 4
Performance environment, servers configuration.

	Application Server	Database Server
Hardware	Intel Xeon E5-2670 v2 (Ivy Bridge) Processors	Intel Xeon E5-2676 v3 (Haswell) processors
Processor (Base Freq.)	2.6 GHz	2.5 GHz
Processor (Turbo Freq.)	3.3 GHz	3.3 GHz
Memory	16 GB	4 GB
Hard disk	30 GB SSD	300 GB SSD
Cores	8	10
Threads	16	20
Thermal design power	115 W	115 W

The second objective was to analyse the upper limit of the RTT of a single API endpoint call. The server average response time was calculated to be 310 ms. The maximum and minimum response times were 450 ms and 259 ms, respectively. The standard deviation was 40.9 ms. The network latency from the test harness location situated in the UK was on average equivalent to 25% of the response time. The maximum, minimum and average RTT were 600 ms, 330 ms and 387 ms, respectively. Although the maximum RTT was almost equivalent to the double of the average, the established threshold was never met and so the test was considered passed.

Moreover, after the system evaluation, a time out waiting feature was added to the framework control flow to avoid server hang events. If the remote controller does not send an instruction within a threshold time equal to the maximum response time plus a standard deviation, the simulation will continue using the current settings. Other metrics to assess the number of cycles in the code base and the scalability of the data access were analysed using an open source package for API test assessment [37]. According to [39], scalability is the ability to continue to function with acceptable performance when the workload has been significantly increased. The authors characterised an interval of demand in which the system would perform acceptably. Their analysis revealed that if the testing interval were sufficiently large (i.e., it covered a significantly wide range of workloads), the system would be scalable.

Scalability failures occur when some resource is overloaded or exhausted and adding capacity to the resource does not result in a commensurate ability to handle significant additional demand. In particular, by determining likely times that performance problems may occur when workloads are significantly increased, it is possible to ensure the consistency and the integrity of the simulation output.

Consequently, the objective of the analysis was to evaluate the scalability performance of the co-simulation framework application for simulation data integrity reasons. The analysis was performed using data from a live environment (the Performance environment) with 210 instances running in parallel. The infrastructure delivering the service includes the following components:

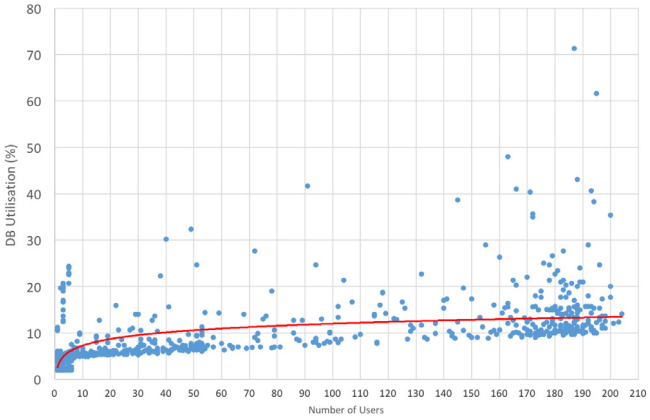

Fig. 7. Database (DB) usage over number of users for date interval.

- Framework server-side API;
- Database.

The test is considered passed, if the result shows a CPU and DB utilisation less or equal to 100%. In case of utilisation percentage close to the limit, the code base could be affected by redundant operations or inefficient loops. The test is performed, as previously mentioned, to ensure the consistency of the output under the use of the Object Relational Model (ORM) open source package. The assessment was performed in four main phases:

1. Identification and configuration of resources;
2. Analysis of the resources utilisation over a time period;
3. Curve fitting of the resource utilisation and statistical analysis;
4. Scalability evaluation.

The passing threshold is an average CPU utilisation and DB utilisation of less than 90%.

Table 5
Performance Environment, servers configuration.

	Max	Min	Average	Standard Deviation
Application Server % CPU Usage	50.16	0.46	10.75	3.24
Database Server % CPU Usage	71.33	1.97	6.71	2.94

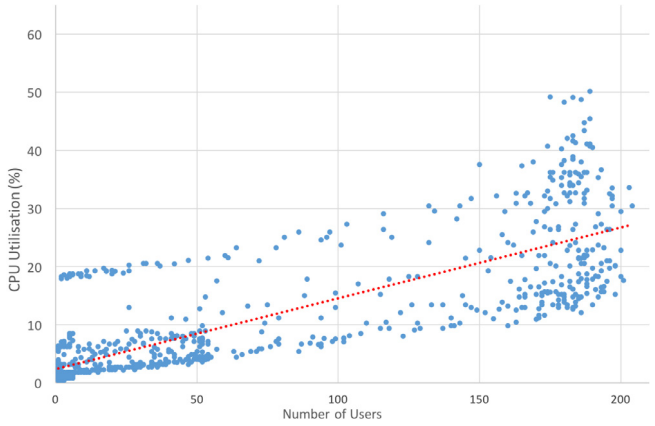

Fig. 8. CPU usage over number of users for date interval.

5.1.2. Summary analysis

During the test period, the Performance Environment Server supported a minimum of 10 and maximum of 210 concurrent connected users of which in excess of 98% were connected via the co-simulation framework. The data resolution is one minute and each data point represents the maximum resource utilisation during the time step.

As illustrated in Table 5, the average App Server CPU Usage was 10.75% with a standard deviation of 3.24. Throughout the test period of 1 week, the App Server CPU Usage peak of 50.16% was recorded; evidence that the computation capabilities of the App Server are deemed more than sufficient to meet the demand. Similarly the DB Server reached a utilisation peak of 71.33% with an average of 6.71% and standard deviation of 2.94. It is evident that the computational capabilities of the DB Server are more than sufficient to support additional users, consequently the test was deemed passed.

A deeper analysis was necessary to assess the causes of the CPU and DB metric spikes. However occasional, these spikes align often with the scheduling of cron jobs which are regular database maintenance routines.

Fig. 7 illustrates that a logarithmic relation exists between the number of concurrent users and DB Utilisation. It is noted that the spikes above 50% occurred when the Performance Environment reached the threshold of 10% from the maximum number of concurrent users and often coincided with Extract, Transform, Load (ETL) processes. Fig. 8 shows a linear relationship can be established between the CPU usage of the App Server and the number of users. Compared to the DB usage there is a higher variance when the number of users is above 160. However, the CPU Usage is greater than 45% only on five occurrences. For CPU usage, the impact of the cronjobs is also relevant, serialisation of the four top cronjobs could reduce the peak size and increase the overall performance.

The overall co-simulation system, utilising the Performance Environment, can support more than two hundreds instances in parallel, therefore all the tests passed without any exception. The stability and the scalability performance of the infrastructure indicates a robust code base that can be used without data loss or computational bottlenecks for the EMS assessment.

5.2. Discussion

Evaluation of the effectiveness of control algorithms in real buildings often require considerable periods of analysis before a consistent validation can be arrived at, and test conditions are not easy to reproduce because of the unpredictable nature of human behaviour or weather conditions. When the evaluation also involves critical infrastructure, such as a power grid or advanced heating and cooling equipment, trial-and-error-approaches can compromise the integrity of the test bed and the relating systems. Therefore, the ability to test control algorithms in a co-simulation environment on BES software can reduce the test cycle time, reduce the hardware infrastructure and also allows for the replication of the experimental conditions [15].

The developed framework also acts as a test benchmark and a logging tool for the evaluation of thermostatic settings, controllers, new retrofit measures, installation of RES and new electricity tariffs which can affect the energy consumption and profile. Furthermore, it can become an assessment tool for the calibration of BES model using sensor data. The implemented analytic dashboard can provide information on electricity patterns and total consumption with the objective of identifying inefficiency in the operation of the heating system. Moreover, the logging capabilities of the interface can facilitate the assessment of connected devices that may affect the thermal comfort or the energy expenditure. The software also acts as a data repository for the simulations performed and the available interface allows the custom creation of energy expenditure graphs based on the data stored.

The infrastructure decouples the building from the controller reducing the overhead of direct in situ control. The API is described as an abstraction paradigm, representing a scalable solution to connect a Home Area Network (HAN) to the smart grid infrastructure. Moreover, the framework enables the bi-directional communication between a centralised or distributed controller and a residential building, exposing the control building interface as an external DR asset to utilise through the API abstraction. The framework also enables the convergence of end user and utility objective which is increasing the efficiency of the power system while reducing the overall operational costs.

At the environmental level, the co-simulation may provide indirect relevant benefits. The software architecture abstraction can represent the communication infrastructure between the smart grid operators and single residential units facilitating DR events for the integration of RES and therefore reducing the overall carbon footprint of the building.

6. Conclusions

A smart grid co-simulation software which focused on the BES integration and established on API abstraction paradigm, has been developed and released as an open source resource repository. The co-simulation architecture facilitates the implementation and testing of control algorithms in the building sector that can be updated through the server and centrally combined. The use of the co-simulation framework developed can reduce the testing cycle of smart grid components such as smart thermostats or advanced heating systems enabling Demand Side Management (DSM) measures such as DR.

The co-simulation framework will be of interest to several stakeholders in the energy arena. The primary purpose of the system is to become a test benchmark and a logging tool for the evaluation of thermostatic settings, controllers, new retrofit measures, installation of RES and new electricity tariffs which can affect the energy consumption and profile. The analytic dashboard can provide information on electricity patterns and total

consumption with the objective of identifying inefficiency in the operation of the heating system. Moreover, the logging capabilities of the interface can facilitate the assessment of devices that may affect the thermal comfort or the energy expenditure.

Therefore, the target users of the system are mainly energy policymakers, utilities and research institutions. From the utility perspective, the co-simulation framework offers a modular software architecture for the exchange and distribution of data in a connected smart grid system. The framework can regularly push data to the utility which then uses the exposed API endpoints to signal DR events. Energy and technology policymakers can use the tool to test and evaluate the scalability of new technologies and their environmental, energy and sustainability impact. It can also be utilised by utilities to gather aggregated information on flexibility, facilitating demand response for geographic regions.

The co-simulation framework has broader applications from an integration perspective. The infrastructure decouples the building from the controller reducing the overhead of direct in situ control which requires equipment capital costs, regular sensor maintenance and accurate experiment planning. Further development of the API will aim to adapt the code base thereby facilitating a more scalable solution for connection between a HAN and a smart grid infrastructure. Additionally, the provision of an interface for the assessment and optimisation of building design features would enhance the capabilities of the API. Therefore, such as technological innovation through the integration of information technology, communications, and circuit infrastructure can lead to a higher penetration of RES, increase the asset efficiency and reduce the overall carbon emissions of the power system.

Acknowledgements

This work was conducted in the Electricity Research Centre, University College Dublin, Ireland, which is supported by the Commission for Energy Regulation, Ireland, Bord Gis Energy, Ireland, Bord na Mna Energy, Ireland, Cylon Controls, Ireland, EirGrid, Ireland, Electric Ireland, Energia, Ireland, EPRI, Ireland, ESB International, Ireland, ESB Networks, Ireland, Gaelectric, Ireland, Intel, Ireland, SSE Renewables, Ireland, and UTRC, Ireland. This publication has emanated from research conducted with the financial support of PRLTI, Ireland [R12681]. The authors would like to thank the building owner for his essential support.

Conflict of interest

The authors declare that there is no conflict of interest.

References

[1] Bianco V, De Rosa M, Scarpa F, Tagliafico LA. Analysis of energy demand in residential buildings for different climates by means of dynamic simulation. Intl J Ambient Energy 2016;37(2):108–20.
[2] Coakley D, Raftery P, Keane M. A review of methods to match building energy simulation models to measured data. Renew Sustain Energy Rev 2014;37:123–41. http://dx.doi.org/10.1016/j.rser.2014.05.007, http://www.sciencedirect.com/science/article/pii/S1364032114003232.
[3] Gils HC. Assessment of the theoretical demand response potential in Europe. Energy 2014;67:1–18.
[4] De Rosa M, Bianco V, Scarpa F, Tagliafico LA. Impact of wall discretization on the modeling of heating/cooling energy consumption of residential buildings. Energy Efficiency 2016;9(1):95–108. http://dx.doi.org/10.1007/s12053-015-9351-5, https://doi.org/10.1007/s12053-015-9351-5.
[5] Chen Y, Xu P, Gu J, Schmidt F, Li W. Measures to improve energy demand flexibility in buildings for demand response (DR): A review. Energy Build 2018;177:125–39. http://dx.doi.org/10.1016/j.enbuild.2018.08.003, http://www.sciencedirect.com/science/article/pii/S0378778818310387.
[6] Nolan S, O'Malley M. Challenges and barriers to demand response deployment and evaluation. Appl Energy 2015;152:1–10. http://dx.doi.org/10.1016/j.apenergy.2015.04.083.
[7] Palensky P, Dietrich D. Demand side management: Demand response, intelligent energy systems, and smart loads. IEEE Trans Ind Inform 2011;7(3):381–8.
[8] Rohjans S, Lehnhoff S, Schutte S, Andren F, Strasser T. Requirements for Smart Grid simulation tools. In 2014 IEEE 23rd international symposium on industrial electronics; 2014. p. 1730–6. http://dx.doi.org/10.1109/ISIE.2014.6864876.
[9] Lizana J, Friedrich D, Renaldi R, Chacartegui R. Energy flexible building through smart demand-side management and latent heat storage. Appl Energy 2018;230:471–85. http://dx.doi.org/10.1016/j.apenergy.2018.08.065, http://www.sciencedirect.com/science/article/pii/S0306261918312170.
[10] Curtis M, Torriti J, Smith ST. A comparative analysis of building energy estimation methods in the context of demand response. Energy Build 2018;174:13–25. http://dx.doi.org/10.1016/j.enbuild.2018.06.004, http://www.sciencedirect.com/science/article/pii/S0378778817336393.
[11] Mets K, Ojea JA, Develder C. Combining power and communication network simulation for cost-effective smart grid analysis. IEEE Commun Surv Tutor 2014;16(3):1771–96. http://dx.doi.org/10.1109/SURV.2014.021414.00116.
[12] Farhangi H. The path of the smart grid. IEEE Power Energy Mag 2010;8(1):18–28. http://dx.doi.org/10.1109/MPE.2009.934876.
[13] IEA. Technology roadmap smart grids. Tech. rep., International Energy Agency; 2011, https://www.iea.org/publications/freepublications/publication/technology-roadmap-smart-grids.html.
[14] Podmore R, Robinson MR. The role of simulators for smart grid development. IEEE Trans Smart Grid 2010;1(2):205–12.
[15] Schloegl F, Rohjans S, Lehnhoff S, Velasquez J, Steinbrink C, Palensky P. Towards a classification scheme for co-simulation approaches in energy systems. In 2015 international symposium on smart electric distribution systems and technologies; 2015. p. 516–21. http://dx.doi.org/10.1109/SEDST.2015.7315262.
[16] Schatte S, Scherfke S, Traschel M. Mosaik: A framework for modular simulation of active components in Smart Grids. In 2011 IEEE first international workshop on smart grid modeling and simulation; 2011. p. 55–60. http://dx.doi.org/10.1109/SGMS.2011.6089027.
[17] Jones A, Finn D. Co-simulation of a HVAC system-integrated phase change material thermal storage unit. J Build Perform Simul 2017;10(3):313–25.
[18] Crawley DB, Hand JW, l Kummert M, Griffith BT. Contrasting the capabilities of building energy performance simulation programs. Build Environ 2008;43(4):661–73. http://dx.doi.org/10.1016/j.buildenv.2006.10.027, http://www.sciencedirect.com/science/article/pii/S0360132306003234, Part Special: Building Performance Simulation.
[19] Clarke J, Cockroft J, Conner S, Hand J, Kelly N, Moore R, OBrien T, Strachan P. Simulation-assisted control in building energy management systems. Energy Build 2002;34(9):933–40.
[20] EMD University of Wisconsin. TRNSYS. Thermal energy system specialists. University of Wisconsin, Engineering Mechanical Department; 2013, http://sel.me.wisc.edu/trnsys/.
[21] Ruiz-Calvo F, De Rosa M, Monzó P, Montagud C, Corberán JM. Coupling short-term (B2G model) and long-term (g-function) models for ground source heat exchanger simulation in TRNSYS. Application in a real installation. Appl Therm Eng 2016;102:720–32.
[22] Ferroukhi M-Y, Belarbi R, Limam K, Bosschaerts W. Experimental validation of a HAM-BES co-simulation approach. Energy Procedia 2017;139:517–23. http://dx.doi.org/10.1016/j.egypro.2017.11.247, http://www.sciencedirect.com/science/article/pii/S1876610217356588, Materials & Energy I (2015).
[23] Widl E, Müller W, Basciotti D, Henein S, Hauer S, Eder K. Simulation of multi-domain energy systems based on the functional mock-up interface specification. In: Smart electric distribution systems and technologies (EDST), 2015 international symposium on. IEEE; 2015, p. 510–5.
[24] Schütte S. Simulation model composition for the large-scale analysis of smart grid control mechanisms (PhD diss.), BIS der Universit?t Oldenburg.; 2013.
[25] Naboni E, Zhang Y, Maccarini A, Hirsch E, Lezzi D. Extending the use of parametric simulation in practice through a cloud based online service. In Proceedings of 1st IBPSA-Italy conference: BSA2013–building simulation applications conference, vol. 30; 2013. p. 105–12.
[26] JADE. Java agent Development framework. 2015, http://jade.tilab.com/.
[27] Knudsen H, Nielsen JN. Introduction to the modeling of wind turbines. Wind Power Power Syst 2005;525–85.
[28] Chinnow J, Tonn J, Bsufka K, Konnerth T, Albayrak S. A tool set for the evaluation of security and reliability in smart grids. In: International workshop on smart grid security. Springer; 2012, p. 45–57.
[29] Hopkinson K, Wang X, Giovanini R, Thorp J, Birman K, Coury D. EPOCHS: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components. IEEE Trans Power Syst 2006;21(2):548–58.
[30] Jorgensen A, Whittaker JA. An api testing method. In Proceedings of the international conference on software testing analysis & review; 2000.

[31] Nijssen GM, Halpin TA. Conceptual schema and relational database design: a fact oriented approach. Prentice-Hall, Inc.; 1989.
[32] Pallonetto F, Oxizidis S, Finn D. Exploring the demand response potential of a smart-grid ready house using building simulation software. In IBPSA building simulation conference; 2013.
[33] Ellis PG, Torcellini PA, Crawley DB. Simulation of energy management systems in EnergyPlus. National Renewable Energy Laboratory; 2008.
[34] Pallonetto F, Mangina E, Finn D, Wang F, Wang A. A restful API to control a energy plus smart grid-ready residential building: demo abstract. In: Proceedings of the 1st ACM conference on embedded systems for energy-efficient buildings. New York, NY, USA: ACM; 2014, p. 180–1. http://dx.doi.org/10.1145/2674061.2675023.
[35] Pallonetto F, Milano F, Finn D. Demand response algorithms for smart-grid ready residential buildings using machine learning models. Appl Energy 2019. http://dx.doi.org/10.1016/j.apenergy.2019.02.020.
[36] Gao J, Bai X, Tsai W-T. Cloud testing-issues, challenges, needs and practice. Software Eng Intl J 2011;1(1):9–23.
[37] Nevedrov D. Using JMeter to performance test web services. Published on dev2dev (http://dev2dev.bea.com/); 2006. p. 1–11.
[38] Bermudez I, Traverso S, Mellia M, Munafo M. Exploring the cloud from passive measurements: The Amazon AWS case. In: INFOCOM, 2013 proceedings IEEE. IEEE; 2013, p. 230–4.
[39] Weyuker EJ, Avritzer A. A metric to predict software scalability. In: Software metrics, 2002. Proceedings. Eighth IEEE symposium on. IEEE; 2002, p. 152–8.