Coupling Building Simulation with Virtual Building Management System for Advanced Test and Validation

Patrick Beguery\(^1\), Oskar Nilsson\(^2\), Thomas Murienne\(^1\), Van Lap Ngo\(^3\)
\(^1\)Schneider Electric, Grenoble, France
\(^2\)Schneider Electric, Lund, Sweden
\(^3\)RH Solutions, Grenoble, France

Abstract
Building Energy Simulation can be used for control design, but it is very seldom re-used in control development, commissioning and performance follow-up in operation. This paper describes the investigation of the coupling between a software instance of a real Building Management System with two simulations tools. Comparison between the two solutions is provided and the main challenges are analysed. Full demonstration of the co-simulation capacity is achieved, opening the door for further works on pre-commissioning, testing third parties advanced control solutions and advanced fault detection and diagnostic in operation.

Key Innovations
- Coupling an instance of a real Building Management System with a Building Energy Simulation software.

Practical Implications
This approach open opportunities for pre-commissioning of the building control as well as more robust testing of advanced energy management third party solutions.

Introduction
A Building Management System (BMS) is a hardware and software solution for control and monitoring of building systems. In this paper, we introduce the concept of Virtual BMS, an instance of the software part of the BMS that will be couple to simulation instead of real building. BMS control is traditionally commissioned during the first year after its implementation. During this phase of error correction and tuning, the building performance is typically much lower than the expected one. One reason for this is that the BMS was not configured properly, with errors and control parameters not set at their optimal value. To tune and test the BMS a building with working electricity, network, air duct pressure, hot water among other things is needed and that happens just before the handover. Since there is little time to commission the BMS in the building construction phase an interesting approach is to use co-simulation for virtual commissioning before the building is available.

In this paper, we explore the capacity to replace the real building by a simulation. The Virtual BMS is connected to a building model to produce an environment that allows control to be validated and tuned, and other functionalities such as alarms and graphics to be created and tested.

Another application of this environment is the validation of advanced third-party control solutions that interact with the BMS. In that case, the testbench consisting of co-simulating the building physics with the Virtual BMS will provide a testing environment with both the power of detailed simulation and the real complexity of a full BMS system and its integration. Further, no real building needs to be found, the weather can be chosen, and the testing will have no tenant impact. Finally, a high-fidelity digital twin of both the physics and controls also offers interesting opportunities for the operations phase of the building, for instance evaluating the impact of a control parameter change.

Several building simulation environments have co-simulation capacity, some of them using the popular FMI standard. In this paper, IES Virtual Environment and EQUA IDA-ICE software co-simulation features are tested on a simple case consisting of part of a real building. A virtual instance of the real Building’s BMS software is connected to BES through a Python based orchestrator. The two solutions are compared, and the challenges of co-simulation analysed in details.

Building Simulation and Control
Building Energy Simulation (BES) has been used for building design optimization since decades, and there is a trend to extend its usage to the commissioning and operation phases. However, adoption of BES software by the building control community has been slow, for different reasons. First, the development of a building model at design stage, and even more its calibration at operation phase, are complex tasks that many control development teams cannot afford. And although BIM methods are making progress, they are far from providing homogeneous and robust approaches, especially for building systems and controls.

Second, most of the BES Software are modelling control in a simplified way. A good example can be given with heat/cool emitters which are typically modelled with a setpoint that is assumed to be achieved continuously if the emitter capacity is not exceeded. Indeed, except for IDA-ICE or research environment like Modelica, most BES use fixed step time that are not sufficient to allow the modelling of the real closed loop control. Consequently,
for a given simplified control model, several detailed control solutions can be considered. This not only creates a potential gap between simulation and real world, but also prevents the BMS development team to use the simulation platforms to test their detailed control and compare possible variants.

Finally, in most building simulation tools, the control is distributed at the level of each actuator. While being understandable at design stage, this approach is not aligned with the layered layout of BMS control, thus making the use of building simulation for control validation even more difficult. A more control-oriented building simulation architecture has been proposed in (Beguery, 2013), with zone level and building level advance controller macro in which centralized control can be defined to affect multiple actuators. Although this environment theoretically allows Model-In-the-Loop validation of full control, there is no direct support to link this BES architecture to the effective real control implementation in a BMS.

As the full BMS control algorithm cannot be easily integrated in BES, an obvious solution should be to co-simulate the building physics and building control. This approach is at the heart of the product model based design methods and start to be considered in the commissioning of industrial control solution (see for example https://virtualcommissioning.com/).

In the building domain, there have been several attempts to use co-simulation for the development and validation of real control software. One of the most interesting was the Building Control Virtual Test Bed (BCVTB) developed by LBNL (Wetter, 2011). This advanced, multiple platforms co-simulation environment is targeting the Hardware-In-the-Loop validation of real controller.

The solution presented in this paper is a much simpler Software-In-the-Loop approach (e.g. we do not use a real BMS hardware but directly run a software instance of the BMS and the BES on a single computer). We made a first successful co-simulation experimentation based on Modelica. While promising, this environment is still dedicated to research projects, with little usage by building control design teams. So, it was decided to investigate co-simulation capacity of commercial BES.

### BES and Virtual BMS co-simulation

#### Principle of co-simulation and Virtual BMS instance

The concept of co-simulation concerns the integration and synchronisation of different simulation software, thereby taking advantage of each platform’s expertise. For instance, building energy & physics models can be modelled and simulated using model-based tools (Modelica, Energy+, IDAICE, IES, …) while the control part is handled by real control software or research one based on Matlab or Python.

In our case, the building controls will be handled by the Virtual BMS and the building simulation will be performed by the two BES considered, namely Virtual Environment (VE) and IDA-ICE.

Figure 1 describes the global software architecture. On the left, the Virtual BMS, which is a virtual copy of the Schneider Electric Ecosstruxure Building Operation system implemented on the real building. On the right, the building’s Digital Twin model developed in BES software and exported for co-simulation purpose.

The two sides are connected via an orchestrator whose tasks are to: (1) ensure correct data format, (2) administer the communication process and (3) synchronise the simulated time steps.

On the Virtual BMS side, communication can only be done using a dedicated Python package. On BES side, Python is also a natural choice, both for interaction with IES Python environment and availability of PyFMI package for integration of IDA-ICE FMU model.

![Figure 1: Co-Simulation loop process](https://doi.org/10.26868/25222708.2021.30712)
In this project, all the software are running on a single computer. In future development, it might be useful to consider having one or both of the simulation environment running in the cloud or even considering direct coupling with the BMS hardware.

The data flow in Figure 1 corresponds to the simple use case that will be described in the result part, namely a typical Air Handling Unit control.

The following two sections will describe in greater detail the co-simulation structure with IDA-ICE and IES.

**Co-simulation framework with IDA-ICE**

IDA Indoor Climate and Energy (IDA-ICE) is a commercial simulation tool for building energy consumption, indoor air quality and thermal comfort, which was developed by the Swedish company EQUA Simulation AB (first released in 1998). The mathematical models of IDA-ICE are written in the program-independent language for dynamical systems - Neutral Model Format (NFM) (Sahlin and Sowell, 1989).

A new feature under development will allow to create from IDA-ICE a package compliant with the widely adopted Functional Mockup Interface (FMI) standard. FMI (Modelica Association, 2020) is a standard interface for exchanging simulation models and basically standardises the data format and formulation of simulation models so that simulation platforms that adopt it can communicate with each other’s. The main object of the FMI is the Functional Mockup Unit (FMU), which contains all information and files required to run a simulation of the tool from which the FMU is created.

There are two types of FMU: Model Exchange and co-simulation. Model Exchange FMUs only include the mathematical models that will be simulated directly in the master platform. Co-simulation FMUs include all the files required to run the model with its own solver. This later approach is the one implemented in IDA-ICE.

Figure 2 illustrates the general co-simulation structure and data flow between Virtual BMS and IDA-ICE FMU. Notice that there are two packages being used in the Python Orchestrator: (1) a proprietary Python package developed to communicate with Virtual BMS and (2) a modified version of PyFMI (an open source Python package for interacting with FMUs, part of Jmodelica.org project).

PyFMI offers capabilities to perform single open-looped FMU simulations with predefined input as well as co-simulation between different FMUs (loaded using FMUModelCS class). However, these capabilities would not allow a co-simulation between an FMU and a non-FMU third party platform (in our case between IDA-ICE FMU and the Virtual BMS). Hence, an extended version of the FMUModelCS class has been developed to allow IDAICE FMU to be loaded and communicate with the Virtual BMS at every time step of the co-simulation.

Data Exchange within the FMI standard (with the current version 2.0) is limited in term of data type. For example, it is not possible to send a vector of temperature setpoint for zones using a single FMU port: instead, it will be needed to create as many ports as variable, which will quickly become complex for a full Virtual BMS.
To create a coupling interface, it is necessary to create a specific macro in IDA-ICE. The FMU block will be manually connected to all the sensors/actuators used by the Virtual BMS. Once all data exchange links have been properly setup, the model would be ready to be exported to an FMU that can be called by any platform that support FMI without the presence of IDA-ICE environment. An example of such an implementation is given in next part (see Figure 8). In this simple example the FMU macro is located directly in AHU model, but in a larger example, it will be possible to take full advantage of the control-oriented software architecture to distribute control in the whole BES model from this Building Controller interface (see Figure 3).

The manual creation of the links between the FMU block and the BES actuators/sensors is of course something that will need to be improved for a commercial application, with a GUI to facilitate the definition of inputs/outputs and an automatic creation of the FMU macro.

**Co-simulation framework with IES**

Virtual Environment (VE) is another BES that accurately models the building, its systems, and building performance analysis. It is developed by the Scottish company IES, funded in 1994.

Co-simulation capability within IES VE is still at an early stage of development, and not available in the commercial package. The research approach does not rely on FMI standard and it is necessary to open IES VE during the co-simulation. IES VE has its own Python Environment which is primarily designed for scripting with simple reports and visualisation functionalities. However, it does have an object class called Scripted Profile that allows co-simulation to be conducted. Note that only variables belonging to Profile class (mostly internal gains schedule or system setpoints) are available for external control through the Scripted Profile solution.

Scripted Profiles are generic Python programs that would be executed iteratively for every step by IES VE’s internal Python environment. To use these Scripted Profiles for co-simulation, it is necessary to develop another Python class (called Profile Manager) to handle the data communication tasks which are described below.

Figure 4 shows the communication principle between the Virtual BMS and VE which is handled directly by the Profile Manager through one single socket (using the native Python module socket that provides access to the BSD socket interface). Once it has received / accumulated all the data (commands and sensor readings), the Profile Manager redistributes the controls to the corresponding scripted profiles (managed by Scripted Profile type and zone id) and collect back sensor data to be returned to the Virtual BMS.

This architecture, along with a graphical user interface that allows users to quickly assign scripted profiles within VE control, makes it rather easy to implement the coupling, even for large models. However, preliminary investigations regarding co-simulation with Virtual BMS has shown several limits:

- **Limited Controllable Variables:** only profiles can be replaced by external control with scripted profiles.
- **Socket-based data exchange with VE Python environment offers flexibility but is quite complicated to develop due to limited display and debugging capabilities.**

**Comparing the two solutions**

First, it must be mentioned an important limitation of the Virtual BMS in its current state. It can only be run in real-time, which will limit the efficiency of testing.

On the BES side, it is worth explaining in more details several issues. Firstly, both IDA-ICE and VE require a start-up phase before the real (co)simulation begins. IDA-ICE performs this start-up phase after the FMU instantiation, which means that the FMU establishes the communication between Virtual BMS and IDA-ICE first and then the start-up phase is run. This is a real problem considering the current real-time constraint on the Virtual BMS. VE, on the other hand, offers more flexibility over the co-simulation progress. It is possible to use a variable that indicates that the simulation is in the start-up phase and start the effective co-simulation after this period is finished. However, this approach raises the question of what predefined control value will be used during the start-up.

Secondly, managing time-related parameters in order to ensure a proper time synchronization in the co-simulation is a delicate task. First, we have to establish a connection sampling time since there are differences in the time step taken by these platforms: the Virtual BMS operates in real time, IDA-ICE simulates using variable step time and VE runs with fixed step time with a minimum value of 60s. For VE, the synchronisation time step was set to 60s, which might be exceedingly long for some closed loop control implemented in the Virtual BMS. For IDA-ICE,
the lower limit on synchronisation step time will be fixed by the real-time constraint. It must be noted that IDA-ICE uses its own internal synchronisation between its variable step-time simulation and the fixed step-time data exchange. This mechanism sometime creates unexpected behaviour or even numerical errors.

Apart for the co-simulation issues, one important aspect is the capacity to effectively create the coupling for the input/output (I/O) of the Virtual BMS with the BES data. In VE case, the variables that can be externally controlled are limited. Furthermore, the simple modelling of system does not allow to create links at actuator I/O level. Although its standard models are of the same level, IDA-ICE offer the capacity to create more realistic models with the advanced system editor, thus enabling us to recreate in the BES any I/O of the Virtual BMS.

IES VE co-simulation capacity might be enough for high level control loop (typically temperature setpoint management proposed by some third-party advanced control solutions) but for full Virtual BMS integration the only possible turnaround will be to develop additional data conversion code in the intermediate Python orchestrator. This will allow to take the control signal provided by the Virtual BMS and translate them into data that can be delivered to IES VE through the scripted profile variables. However, this approach introduces an additional simulation environment that will make the global solution even more complex.

Table 1 summarizes some results obtained from our experiments regarding co-simulation capabilities of both platforms. The use of a standardized interface together with the more accurate modelling of the control loop give a significant advantage for IDA-ICE with respect to co-simulation to support BMS development.

## Experimentation

### Case Study

A 3-storey office building located in Grenoble, France, was used as a case study. The as-built model of the building (see Figure 5) was developed with both IES VE and IDA-ICE.

![Figure 5: Building Model, Digital Twin with IDA ICE](image)

In this first experiment, the coupling between Virtual BMS and BES was limited to a single Air Handling Unit. Figure 6 shows the schematic of the real AHU system in the BMS graphical interface. To be more specific, the Virtual BMS sends control to the AHU simulated in BES based on the returned temperature and pressure obtained from simulation model. Table 2 describes the variables being communicated between the two platforms.

<table>
<thead>
<tr>
<th>Virtual BMS to BES</th>
<th>BES to Virtual BMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recovery Wheel Speed (%)</td>
<td>Return Air Pressure (Pa)</td>
</tr>
<tr>
<td>Hot Valve Position (%)</td>
<td>Supply Air Pressure (Pa)</td>
</tr>
<tr>
<td>Cold Valve Position (%)</td>
<td>Return Air Temp (°C)</td>
</tr>
<tr>
<td>Return Fan Speed (%)</td>
<td>Supply Air Temp (°C)</td>
</tr>
<tr>
<td>Supply Fan Speed (%)</td>
<td>Discharge Air Temp (°C)</td>
</tr>
<tr>
<td></td>
<td>Outdoor Temp (°C)</td>
</tr>
</tbody>
</table>

**Table 2: Data exchange between BES and Virtual BMS.**

<table>
<thead>
<tr>
<th>Method</th>
<th>IDAICE</th>
<th>IESVE</th>
</tr>
</thead>
</table>
| Controllable Variable                       | Any IDA-ICE model input     | Profiles only and high-level controls (flow rate, temperature, heat transfer…)
| Adequacy of BES model with control I/O      | High fidelity can be obtained with advanced level modeling | Limited to setpoint I/O (no actuator model) |
| Start-up Period                             | Start-up is included in the co-simulation time horizon (Virtual BMS needs to exchange data during this period) | Can be run apart using VE start-up flag (this raise question about Virtual BMS control setting during start-up) |
| Data exchange                               | Standardized FMI data exchange (with limitation, like no arrays or string) | Any type of data but requires programming of the intermediate layer to connect Scripted Profiles to socket |
| Simulation Time step                        | Variable, but the FMU internal synchronization with the fixed step time data exchange may create unexpected behavior | Only 6 options (1, 5, 10, 30 and 60 minutes) |
| Debugging of the communication              | No need for debugging as IDA-ICE use the standardized FMI communication | Debugging inside VE Apache's Python environment VE |

**Table 1 - Software co-simulation capacity comparison.**
Figure 6: AHU system in the BMS interface visualisation

Figure 7: AHU system in IDA ICE

Figure 8: AHU control box
Although the co-simulation was successfully run with both BES environment, comparing their capacity highlight the advantage of IDA-ICE, and the experimentation was continued with this single software.

Figure 7 shows the AHU system designed in IDA ICE. The AHU is composed of heating and cooling coils, supply and return fan, and a heat recovery rotary wheel. A pressure and temperature sensors collect data to send them to the Virtual BMS. The pre-conditioned air is then sent to the rooms. Room air temperature control is managed by additional zone units.

Return air is extracted from the rooms and go through a pressure and temperature sensors. It shares its calories with the supply air thanks to the heat exchanger and then the air is discharge to the exterior.

As we are considering a single AHU, a control macro specifically designed to establish communication between the FMU block and the was created in this system. Its manually designed schematic is given in Figure 8. In term of co-simulation step-time, different values were tested before selecting 9s, which offer a good compromise between real time constraint and accuracy of the closed loop.

Results

The results are promising even if the process to build advanced level systems and to link every variable is complex and time consuming.

Figure 9 shows an example of a 4 hours co-simulation process in a cold period (outdoor temperature < 3°C). In this period, the wheel exchanger is always at 100% capacity to maximize the heat recovery from the return air. The heating valve position is handled by the Virtual BMS algorithms to control the supplied air temperature.

1. Co-simulation is starting: a few minutes after the wheel exchanger is rotating at 100% of his speed capacity, the supply temperature increases higher than the heating temperature setpoint. In consequences, the Virtual BMS PID reacts from this information and decrease the heating coil operation.
2. Supply temperature decreases thanks to the heating coil valve lowering his operation.
3. As the supply temperature get lower than the temperature setpoint, the Virtual BMS command the valve to open again.
4. Supply temperature is stabilized after 1 hour.

We demonstrated a working co-simulation describing with high fidelity how the Virtual BMS algorithm will achieve control of the supplied temperature through closed loop control. With such a testing environment, we can imagine a BMS developer being able to tune its PID controller or analyse what will be the behaviour in case conditions are modified (for example if the rotating wheel is switch off).

Perspectives

There is still a wide range of additional developments required to make the use of co-simulation an acceptable solution for control development teams. On the BES side, easy creation of the input/output co-simulation interface will be key, as well as specific features like hot start, debug option and data storage. On the Virtual BMS side, the current most painful limitation is the real-time constraint that prevent us to run accelerated testing. It might also be useful to consider an FMI-ready BMS that might remove the need for development of a specific intermediate orchestrator.

An interesting application of this co-simulation approach will be the evaluation of start-up companies offering supervisory control on top of traditional BMS solutions, using e.g. AI. Typical example of such application is the management of temperature setpoint in order to leverage the building flexibility, for example to answer Demand Response event. This advanced controls are typically provided by third parties that need to interact with the BMS to apply their strategy. The main benefits of this co-simulation would be to enable faster evaluation, and the possibility to accurately benchmark different solution with actual control. Direct usage of BES to test the performance of test such solutions is possible in early phase, but a commercial offer is typically intended to be real-time connected to BMS, thus leveraging the interest of the Virtual BMS and BES co-simulation approach.
Another possible usage of the co-simulation will be advanced fault detection and diagnostic. AFDD has received a lot of interest since several decades, with both data driven and simulation-based approaches (see Woohyun (2018) for an extensive review). Among the various challenges faced by AFDD are the difficulties to analyze the different root causes of a given symptom. Most approaches solve this by looking only at fault in part of the system, making the assumption that the other parts are behaving correctly. In (Gao, 2020), a more global solution is proposed, based on Bayesian Network integrating data driven and simulation-based models. This works well for normal operation, with a given BMS control. But what will happen if there is a significant update of the BMS? The standard approach will be to translate the BMS new logic into a BES for simulation-based model, and to wait a few weeks/months to accumulate new training data for data-driven models.

In this context, the solution proposed in this paper will make a lot of sense. After having been used to allow a smoother commissioning, the co-simulation environment can be re-used in operating phase, acting as a detailed digital twin of the building control. When a BMS update happens, the co-simulation environment can be used in two ways. First, if the update only includes small changes, not supposed to change the behavior, it will be possible to test the two instances of the Virtual BMS (before and after changes) on the same BES model to check that there is effectively no change in the building performance. On the other hand, when a major update is made to building control, then the Virtual BMS and BES co-simulation can be used to provide the new normal mode data for the AFDD method.

**Conclusion**

This study allows us to successfully demonstrate the co-simulation capacity of both VE and IDA-ICE detailed building models with an instance of a real BMS Software. IES provide an easy way to define the data coupling, but is more limited in term of possible input/output than IDA-ICE. Moreover, the use of the FMI standard facilitates the integration in the co-simulation framework.

Although there remains large number of challenges to address to make such an approach easily usable in commercial project, such a co-simulation environment opens broad opportunities for control design, validation, optimization, and performance follow-up. Having the possibility to test the BMS solution before the implementation in the building would enable proper verification tests and optimization to assure that the design intent is well executed.

Apart from the co-simulation related challenges, one key issue will be to identify the right level of BES model details and calibration. For simple detection of human errors, like a bad link or a not appropriate change in the BMS setting, a simple digital twin model will be enough. On the other hand, detailed control design and optimization will require a fully detailed and calibrated BES model.

**Acknowledgment**

The authors would like to thank EQUA and IES teams for sharing beta version of their on-development co-simulation solutions, and for the support they provide in this work.

**References**


Gao, T. (2020), Integrated building fault detection and diagnosis using data modeling and Bayesian networks, PhD Report, Université de Lille, France.


