The Digital Twin as a Base for the Design of Building Control Strategies

Christoph Nytsch-Geusen¹, Werner Kaul¹, Jörg Rädder¹, Lucas Westermann¹,
Vishvesh Shenoy², Pruthviraj Balekai²

¹Berlin University of the Arts, Institute of Architecture and Urban Development, Berlin, Germany
²Technische Universität Berlin, Zentralinstitut El Gouna, Berlin, Germany

Abstract

This contribution describes a simulation-based method for the design and evaluation of building control strategies for new planned buildings. The approach consists of a combination of a digital twin of the real building and its HVAC system, modelled as a physical Modelica model and the openHAB software package, for the implementation of correspondent building control strategies. For this purpose, openHAB and the Modelica model is bi-directional coupled in real-time. In this way, new control strategies can be tested and improved under consideration of the dynamic system behavior in the building design phase. If the system behavior of the building simulation model is close enough to the real system, the digital twin can be substituted afterwards by the built building and the control strategy will behave during the operation of the real system almost unchanged.

Introduction

Simulation models of real buildings are often too much simplified in comparison to the reality, which leads to significant differences in the system behavior between the “simulation world” and the “real world”. A digital twin, which reflects the technical structure, the functionalities and the physical behavior of the real system in a high precision (Nytsch-Geusen et al., 2018), can work as a base for the development of control strategies already in the “simulation world”. It is much easier and cheaper to test different variants of control strategies in advance in the early planning phase of buildings.

Figure 1 illustrates this approach. During the design phase of the building energy system a detailed simulation model of the building construction and the HVAC system – the digital twin - serves as the controlled system. The simulation model is delivering the necessary systems states as input signals for the rule-based control strategy. Thereafter, the calculated control signals are sent back to the simulation model. In our approach this bi-directional data exchange takes place between a home automation software, the open Home Automation Bus (openHAB) software platform (openHAB, 2019) and the Modelica simulation tool Dymola in real-time within the local network. For this purpose, the network User Datagram Protocol (UDP, https://tools.ietf.org/html/rfc768) is used. Later, if the genuine building energy system runs under real operation, the installed sensors, actors and bus systems in the building and of its HVAC system are connected with the same openHAB software platform. Under the assumption that the detailed Modelica system model has reflected precise enough the dynamic and logical system behavior within the previous design phase, the rules of the openHAB system can be reused for the real system without any changes.

There are some related industrial solutions and research approaches, which touch similar issues, most of them in the HIL area. For instance, the company Typhon HIL offers an integrated software and hardware platform that includes a so called “Virtual HIL Testbed” which is able to substitute real HIL testbeds, e.g. during the development of controller strategies for micro grids (https://typhoon-hil.com).

In Schneider et al. (2015) a HIL testbed is described, in which parts of the building energy system (e.g. the hydraulic part of the HVAC system) consists of real hardware and the rest of the problem (the building, the heating system, the control strategy) is reflected in a Modelica model. Hereby, the hardware and the virtual part are coupled in real-time over a TCP/IP data interface.

In De La Cruz et al., 2017, an implementation of a HIL/real-time simulation test bench for an air to water heat pump is presented. In this approach, a real heat pump and real weather conditions were combined with a virtual Modelica model of the building and occupant’s response, for optimizing parameters of the control strategy.

In opposite to the two mentioned Modelica approaches, in which only parts of the building energy system are
represented by a virtual Modelica model, the method in this paper works as described above with a pure virtual model for the controlled system. The first development steps in this direction were made within two master theses (Shenoy, 2018 and Balakei, 2018). In both theses, prototypically real-time couplings between openHAB and Modelica were investigated in context of the Rooftop building (see more details about the Rooftop building in the case study paragraph). Outgoing from these first experiences, the research approach, described in the next paragraph, was developed by the authors.

Research approach
The research approach is based on following components: the openHAB software platform, a Modelica library for energy building simulation and a real-time coupling between openHAB and the Modelica tool, which performs the simulation experiment with the digital twin.

openHAB
The open Home Automation Bus (openHAB) is a Java based software framework, which integrates components for building automation from different vendors independent from a specific communication protocol on one software platform (openHAB, 2019). Because it works as a server solution on a Java virtual machine, openHAB runs on different operating systems (Linux, Windows, MacOS) and hardware (personal computers, single-board computers). It is related with the Eclipse SmartHome project (Eclipse SmartHome, 2019), which delivers the technical core of openHAB. openHAB itself is focused on so-called bindings to more than 200 different technologies (physical hardware, protocols, software technologies, web services etc.). Well-known examples are the integration of technical devices over the Modbus protocol, the integration of commercial software solutions for smart home applications such as Apple HomeKit, Nest or Philips Hue and the integration of weather data services, e.g. the Yahoo weather service.

Figure 2 illustrates the software architecture of openHAB, which includes as the base the runtime for running the Java server and a lot of related system components and add-ons for the configuration of a specific problem.

Figure 2: openHAB software architecture (Source: https://de.wikipedia.org/wiki/OpenHAB)

System variables in openHAB are defined as Items whose states can be updated during the runtime by events. An Item represents the current states of continuous variables (e.g. temperatures or flow rates) and of discrete variables (e.g. the number of operation modes of a technical device). The different Items are logically interrelated by Rules that define the control strategy of an openHAB problem. On the top, the Sitemaps configure the graphical appearance in the user interfaces in a web browser or in an openHAB app (see the example in Figure 11).

In our context, the openHAB REST service with its Representational State Transfer (REST) API is a very important component, because it supports the read and write access to the state of every Item during runtime.

The communication within the openHAB system is taking place on the openHAB Event Bus in two directions (compare with Figure 3).

Figure 3: openHAB Event Bus
(Source: https://de.wikipedia.org/wiki/OpenHAB)
1. Control signals (top down): if a Rule changes the state of an Item, this event is registered by the event bus and a command is send over the specific binding to the respective device (e.g. the rule changes the state for a bulb from OFF to ON and set its brightness to 50 percent).

2. Measured signals (bottom up): a sensor registers a change of the value of the measured signal (e.g. the measured air temperature or the relative moisture of a space). This change is sent over the sensor binding to the openHAB Event Bus, which triggers an update of the state of the respective Item.

Modelica libraries for the digital twin
The digital twin – the modelled building energy system – is configured as a system simulation model by a combined use of following three Modelica libraries:
1. the BuildingSystems library
   https://github.com/UdK-VPT/BuildingSystems
   (Nytsch-Geusen et al., 2016) for the modelling of the building construction and the HVAC system,
2. the Modelica_DeviceDrivers library
   https://github.com/modelica/Modelica_DeviceDrivers
   (Thiele et al., 2017) for the bi-directional data exchange between the simulated digital twin and the parallel running openHAB system and
3. the Modelica standard library
   (https://github.com/modelica/ModelicaStandardLibrary) with a lot of additional needed standard components like signal sources, math components etc.

A fourth library, the Modelica IBPSA library (successor of the Annex 60 library, see Wetter et al., 2015) is integrated within the BuildingSystems library.

Following the idea that the digital twin can be exchanged against the real existent building energy system while the control strategy and the related home automation software shall remain the same, a real-time data exchange between openHAB and Modelica/Dymola is required.

**Real-time coupling between openHAB and Modelica**

The bi-directional real-time data exchange is realized with two Python interfaces (Modelica2openHAB and openHAB2Modelica), whereby the REST API of openHAB and the UDP sockets to the Modelica simulation tool are used as the technical base (see Figure 4). The implementation of the Python interfaces were done by using the library python-openHAB (python-openhab, 2019), which supports the access to the REST API. For the data transfer from the Python to the Modelica real-time simulator (Dymola) a Python binding to two UDP sockets for sending and receiving data based on the C libraries of the Modelica_DeviceDrivers library (Thiele et al., 2017) was programmed.

![Figure 4: Building automation test bed based on openHAB and the digital twin (Modelica)](image)

openHAB and the real building system

Figure 6 shows a typical configuration of an openHAB building automation system for a real building.

In this case, the data transfer between the different devices is routed over the Local Area Network (LAN) by using a standard internet router. Three openHAB bindings are used: the Modbus binding to connect about 100 signals of a heat pump, the Netatmo binding to connect the signals...
of a Netatmo weather station and the serial bus binding which is used for the data exchange with several Arduino Mega micro controllers. The Arduino’s are responsible for gaining measurement values from different types of sensors (solar radiation, ultrasonic distance, temperature, moisture, flow rates etc.) and for setting control values of actors (e.g. to control the servo-motors of a façade shading system).

Case study

Rooftop building

The described approach was evaluated using the example of a solar plus energy rooftop building unit for the building stock (Figure 7), which was developed and built within the international competition Solar Decathlon Europe (SDE) 2014 (Figure 8 and Team Rooftop, 2014).

In 2017 and 2018, the Rooftop building was reconstructed on the university campus Berlin-Charlottenburg, where the both universities UdK Berlin and TU Berlin use it for research and education. The large room on the right side serves as a seminar room and the smaller room on the left side as a lab.

Figure 7: Rooftop building as a solar plus energy living unit for the building stock (Team Rooftop, 2014)

Figure 8: Rooftop building on the SDE 2014 competition site in Versailles, France (Team Rooftop, 2014)

The building envelope is realized as a wooden lightweight construction with a high-energy efficiency (U-values of the opaque building elements less than 0.14 W/m²K). The seven sliding windows on both façade sides (Figure 9) are triple glazed and have a U-value of 1.1 W/m²K.

A reversible air to water heat pump (type Viessmann LWZ 304 SOL) with a capacity of 4.3 kW heating power (A2/W35) and 3.6 kW cooling power (A35/W7) heats the building over a ventilation and floor heating system in winter and enables also an active cooling during hot summer periods by the use of a cooling ceiling system. The heat pump and the PV system work in combination with a 200-liter buffer tank and an electric battery with 4.2 kWh capacity for an optimized energy management. A small solar thermal system of 3.2 m² roof integrated vacuum tube collector generates the hot water for the sanitary room and the kitchen. It also supports the heating function of the heat pump during sunny days in the heating period.

The building automation system of the Rooftop building is based on an openHAB server. Its runs on a Raspberry Pi using openHabian, a preconfigured Linux version of the openHAB software platform. The before mentioned building technology and further distributed sensors in the building are integrated and controllable by the openHAB system on a higher building operation level.

During the reconstruction process, many additional sensors within the building construction, the inner air space and the building technology were added, for performing an energy monitoring of the overall Rooftop building and for generating reference values for a comparison with its digital twin.

The digital twin

The entire system, including the building construction and the building technology was modelled as a detailed digital twin in Modelica based on the BuildingSystems library. This system simulation model of the Rooftop building with all its sub system modules – the building model, the PV system model and the HVAC model is explicitly described in Nytsch-Geusen et al., 2018.
Control scenario

The control scenario specifies an evaluation of an openHAB rule, which triggers a reaction of the HVAC system dependent on the user behavior. The evaluation takes place during a cold winter day with an outdoor air temperature about -0.7 °C and an indoor air temperature about 21.8 °C. Following rule was investigated: if one or more persons enter the heated building with a “normal behavior” – that means he or she opens and closes one of the seven sliding windows within 10 seconds – then this behavior shall not have an impact on the control strategy of the HVAC system. If one or more of the sliding windows are staying open longer than 10 seconds with a common opening width more than 30 cm, then the ventilation system and the circulation pump of the floor heating system is switched off for avoiding unnecessary energy losses. If the sliding windows are closed again, the ventilation system and circulation pump are reactivated.

The user behavior is tracked by ultrasonic sensors, which are individually installed for each sliding window (Figure 10). These sensors measure the window opening width between 0 cm (fully closed) and 160 cm (fully opened).

![Figure 10: The ultrasonic sensor, which measures the opening width of the sliding windows](image)

The wished change of the HVAC system behavior is realized with the switch of the heat pump operation mode from “Automatic mode” to “Manual mode”:

- “Automatic mode”: Ventilation rate 140.0 m³/hour; flow rate circulation pump 6 liter/minute
- “Manual mode”: Ventilation rate 0.0 m³/hour; flow rate circulation pump 0.0 liter/minute

Because the ventilation rate and the flow rate for each operation mode are predefined values of the heat pump, the rule in openHAB has to trigger only the change of the operation mode:

```
rule "Initialization of parameters"
when
System started
then
  operMode.sendCommand(11) // automatic mode
  Thread::sleep(1000)
end
```

```
rule "Set operating mode from automatic mode to manual mode and back"
when
Member of Doors changed
then
  logInfo("RooftopSystem.rules", "start")
  var dst = Dst1.state as DecimalType +
             Dst2.state as DecimalType +
             Dst3.state as DecimalType +
             Dst4.state as DecimalType +
             Dst5.state as DecimalType +
             Dst6.state as DecimalType +
             Dst7.state as DecimalType;
  var Number minSum = 30
  var Number operMode2nd = 14 // manual mode
  var Number ruleContr = 20
  if (dst >= minSum)
    Thread::sleep(10000)
    if (dst >= minSum)
      if (operMode.state != operMode2nd)
        postUpdate(operMode_Proxi, operMode.state);
        sendCommand(operMode, operMode2nd);
    } }
  if (dst < minSum)
    Thread::sleep(10000)
    if (dst < minSum)
      if (operMode.state == operMode2nd &&
           operMode_Proxi.state != ruleContr)
        sendCommand(operMode, operMode_Proxi.state as DecimalType);
        postUpdate(operMode_Proxi, ruleContr);
  } }
end
```

![Figure 11: openHAB GUI of the control scenario](image)
Simulation based evaluation (digital twin)

In the first step, the openHAB rule is evaluated with the simulated system states of the digital twin, modelled in Modelica (compare with Figure 12). For this purpose, 12 values are sent over the Modelica2openHAB interface:

- ambience and the room air temperature,
- opening widths of sliding windows 1 to 7,
- mass flow rate of the circulation pump and ventilation supply flow rate and extract flow rate.

The digital twin receives only one value, the operation mode of the heat pump over the openHAB2Modelica interface.

Figure 13 illustrates the structure of the HVAC system, which includes also the heat pump model. This model obtains its operation mode as in input signal by openHAB.

If it is working in the automatic mode then the two local controllers are regulating the valve positions of the hydraulic distributor for the four loops of the floor heating system related to the set point temperature of the respective room. When the heat pump switches to the manual mode then the distributor valves are totally closed and the flow rates in the floor heating loops are reduced to zero.

Figure 14 shows on a more detailed level the inner structure of the heat pump model. The two integrated fans are responsible for providing the building with fresh air and transporting the used air into the environment. They are thermally connected via a heat recovery system for minimizing the ventilation energy losses. In the automatic mode, the fans are working with the predefined air flow rates (in the scenario both with 140.0 m$^3$/hour). If the manual mode is active, both fans are switched off and the airflow rates are set to zero.
In the simulation experiment, the outdoor air temperature is set to a constant value of -0.7 °C and the direct and diffuse solar radiation to 0 W/m², because the outside air temperature during the real experiment was approximately constant and the experiment took place after sunset.

![Figure 15: Simulated based evaluation of the control scenario in Modelica](image)

Figure 15 shows a section of the real-time experiment between 400 and 2,400 seconds. At time 600 seconds, the sliding window 7 is opened to an opening width of 155 cm for 20 minutes. This triggers the rule in openHAB to change the operation mode of the heat pump from automatic (11) to manual (14) that means the flow rates for the ventilation system and for the hydraulic system of the floor-heating system are set to zero. At time 1,800 seconds, the sliding window is closed again, so the operation mode switches back to automatic and the ventilation system and the floor heating system is switch on again. Because of this, the room air temperature is now increasing to its set value of 21.8 °C.

**Evaluation with the real building**

In the second step, the same openHAB rule is applied to the real energy system of the Rooftop building. The experiment with the real building energy system was performed with similar boundary conditions as the virtual experiment with its digital twin (outside air temperature, set air temperature, opening time and opening width of the sliding window 7). Figure 16 demonstrates that the behavior of the control strategy remains the same (switching the operating mode from automatic to manual after the sliding window was opened and back to automatic when it was closed later).

![Figure 16: Evaluation of the control scenario with the real building](image)

**Discussion**

The comparison of the experiments for the digital twin and for the real building (both evaluated with the same openHAB rule) indicates a similar system behavior. The natural ventilation effect through the almost complete opened sliding window over 20 minutes drastically reduces the room air temperature. In both cases, the heat pump operation mode is changing corresponding to the same pattern. Therefore, it can be stated that the approach works in practice as theoretically formulated before. However, the speed of lowering the room air temperature still shows clear differences: in the simulation model, it drops significantly faster than measured and remains on a higher temperature level as in the experiment (about 8 °C in the simulation experiment and 4 °C measured). These differences could be caused by following reasons:

- The simulation model calculates only one thermal node for the enclosed air in the considered room. This calculated mean value can differ from the local situation, where the temperature sensor in the practical experiment was placed (in a height of 70 cm on the table in the middle of the room). Caused by the natural ventilation process, the warm air leaves the room through the upper half of the full opened sliding window and the cold air from the environment enters the room in the lower half of the opening. This leads to a strong thermal stratification within the room during the ventilation process, which can not be precisely reflected by an ideal-mixed air model and only one measurement point in the real experiment.

- All numerical nodes in the model of the building construction are initialized with 20 °C. Perhaps the built-up time of the simulation experiment is still too short for developing the typical temperature gradient.
from warm (room) to cold (ambience) in the layered models of the opaque building constructions. In this case, more thermal energy would be transferred from the “warmer” building construction to the air, when the floor heating system is switched off in the manual mode of the heat pump and finally this would result in a higher indoor air temperature level as in the reality.

Conclusion and outlook
The described approach that a digital twin can serve as the base for the evaluation of building control strategies was successfully demonstrated in a first case study of the Rooftop building. The control algorithm formulated as an openHAB rule and the controlled system show in both worlds - the virtual world of the simulation model and the real physical world – a similar behavior.

There are still significant differences in the respective indoor air temperatures. This could be improved by the use of a spatial resolved Modelica thermal room model (Mucha et al., 2015) and also physical measurements of the air temperature in different heights of the room in the Rooftop building.

Future research will focus on the evaluation of control strategies for other control topics of the Rooftop building. In this context, the complexity of the formulated openHAB rules will be increased. For example, control strategies for the optimal positions of the 14 adaptable façade elements with the integrated photovoltaic elements should be evaluated: which algorithm can maximize the generated electricity by an one-axis tracking on the one hand and can also regulate the passive solar gains through the transparent façade of the building (e.g. optimized sun protection in summer and maximum use of solar energy in winter) on the other hand?

Acknowledgments
The research described in this paper was conducted within the research project “WaveSave: Planung und Steuerung von dezentralen Energiesystemen in Gebäuden zur nachhaltigen, ressourcenschonenden und wirtschaftlichen Strom- und Wärmeerzeugung” and funded by the Federal Ministry for Economic Affairs and Energy in Germany (reference: 03ET1312B).

References


python-obenhab (2019). Python library for accessing the openHAB REST API. GitHub repository https://github.com/sim0hn/python-obenhab


