LINKING INTERACTIVE MODELICA SIMULATIONS TO HTML5 USING THE FUNCTIONAL MOCKUP INTERFACE FOR THE LEARNHPB PLATFORM

Xiufeng Pang[1], Raj Dye[2], Thierry S. Nouidui[1], Michael Wetter[1] and Joe J. Deringer[3]
[1]Lawrence Berkeley National Laboratory, Berkeley, USA
[2]SRI International, USA
[3]Institute for Sustainable Performance of Buildings, Berkeley, USA

ABSTRACT

Learn High Performance Buildings (LearnHPB) is an eLearning platform that is currently being developed to enable users to learn about high-performance, energy efficient building design, construction, and operations. In the development of this eLearning platform, the Functional Mockup Interface (FMI) was used to address a need for a two-way data communication between the building simulation and the Graphic User Interface (GUI) and animations in the web browser. WebGL (Khronos 2011) was employed to implement the interactive 3D graphics, which are rendered in the web browser directly, without the use of any plug-ins. The 3D interactive environments provide the users with cut-away views of the building system as well as building performance and energy utilization data. The software architecture and the preliminary results are presented in this paper.

INTRODUCTION

Saving energy in buildings involves more than developing new technologies. There is a critical need to train people in the design, construction, operation, and maintenance of increasingly complex buildings and systems.

Learn High Performance Buildings (LearnHPB) is an eLearning platform that is currently being developed for this purpose. LearnHPB uses Problem-Based Case Study (PBCS) scenarios supported by 3D visualizations and game-based approaches to address real life situations found in key phases of an office building's life cycle. It integrates four building systems - envelope, HVAC (heating, ventilation, and air-conditioning), lighting, and daylighting (Deringer et al., 2012). This paper focuses on the HVAC system.

Interactive animations representing the HVAC system operation need to receive simulation results from a running simulation. This can be achieved in various ways. In the predecessor of LearnHPB, the LearnHVAC software, a BSD socket connection was employed. The major challenge when using the socket was the data mapping from the simulation results to the user interface. The interface developer needed to know the exact order of the data string that comes from the simulation engine. It was thus easy to make mistakes during this process especially when dealing with large amount of data to be exchanged. In the current approach, we use the Functional Mock-up Interface (FMI) standard (MODELISAR-Consortium, 2008-2012a). FMI is an open, tool independent standard for model exchange and co-simulation of dynamic models using a combination of xml-files, complied code and optional C-code. There are currently more than 38 modeling and simulation environments which support or plan to support FMI, which is an indicator of the impact of the technology in research and industry.

In the development of LearnHVAC, the 2D animation engine was developed in Adobe Flash, and the tool had to be downloaded and installed onto a user's computer, including the program's simulation engines which created issues related to system incompatibility. In the development of LearnHPB, WebGL (Khronos 2011) was employed to render 3D assets natively in a web browser without the need of plug-ins. The simulation engines were migrated from the user's local computer onto a remote server. Two-way communication between the server-based simulation engines and the user's local web browser was achieved using WebSockets (IETF 2011).

This paper presents the software architecture and the preliminary resting results.

SOFTWARE ARCHITECTURE

The software is comprised of four key elements as shown in Figure 1.

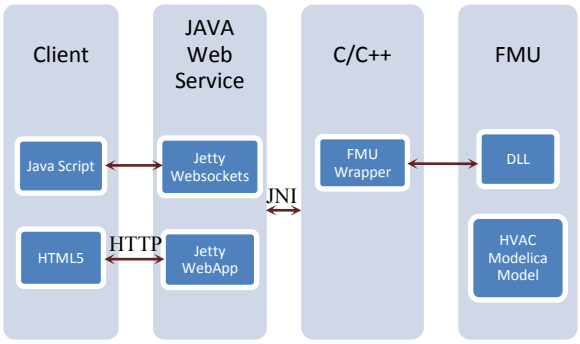

Figure 1 software architecture

HVAC Modelica Model

One of the key features of LearnHPB is to address the HVAC system operation, control and troubleshooting. We implemented a Variable Air Volume (VAV) system with a central chiller and a boiler plant. The building characteristics are obtained

from the new construction medium office building as described in the set of U.S. Department Of Energy (DOE) Commercial Building Benchmarks (Deru et al., 2009). The number of stories is selectable from 3, 5 or 7. The building geometry and the floor plan are shown in Figure 2. There are eight perimeter zones and one core zone in each floor.

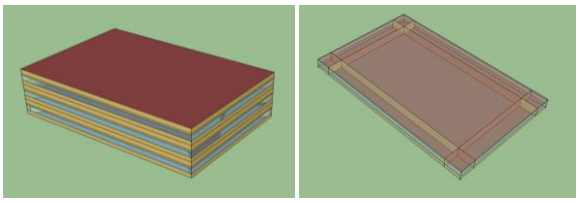

Figure 2 Building geometry and floor plan

The building envelope heat transfer, multi-zone air exchange and the HVAC system model was implemented using the Modelica Buildings Library (Wetter, 2009). A model of nine interconnected rooms is used to model the heat transfer through the building envelope as shown in Figure 3. The thermal room model computes transient heat conduction through walls, floors and ceilings and long-wave radiative heat exchange between surfaces. There is also a layer-by-layer short-wave radiation, long-wave radiation, convection and conduction heat transfer model for the windows. The model is similar to the Window 5 model and described in TARCOG 2006.

Each thermal zone can have airflow from the HVAC system, through leakages of the building envelope (except for the core zone) and through bi-directional air exchange through open doors that connect adjacent zones. The bi-directional air exchange is modeled based on the differences in static pressure between adjacent rooms at a reference height plus the difference in static pressure across the door height as a function of the difference in air density. There is also wind pressure acting on each facade. The wind pressure is a function of the wind speed and wind direction. Therefore, infiltration is a function of the flow imbalance of the HVAC system and of the wind conditions.

The Air Handling Unit (AHU) and the terminal boxes are shown in Figure 4. The Sequences of Operation for Common HVAC Systems (ASHRAE, 2006) are adopted as the default control sequence. In this control sequence, the supply fan speed is regulated based on the duct static pressure. The return fan controller tracks the supply fan airflow rate reduced by a fixed offset. The duct static pressure is adjusted so that at least one VAV damper is 90% open. The economizer dampers are modulated to track the setpoint for the mixed air dry bulb temperature. Priority is given to maintain a minimum outside air volume flow rate. In each zone, the VAV damper is adjusted to meet the room temperature setpoint for cooling, or fully opened during heating. The room temperature setpoint for heating is tracked by varying the water flow rate through the reheat coil. In the VAV model, all airflows are computed

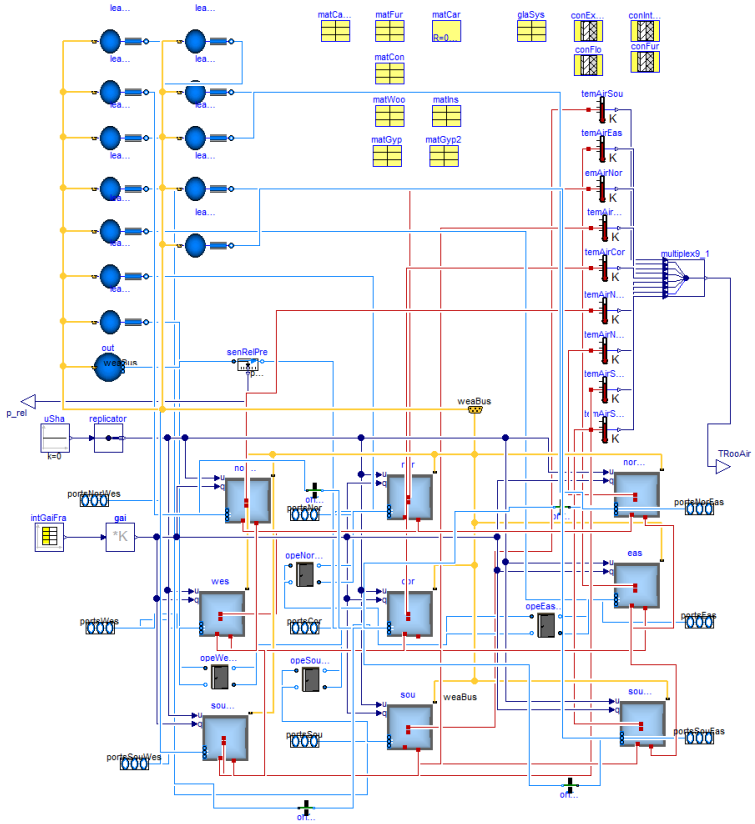

Figure 3 Building envelope model

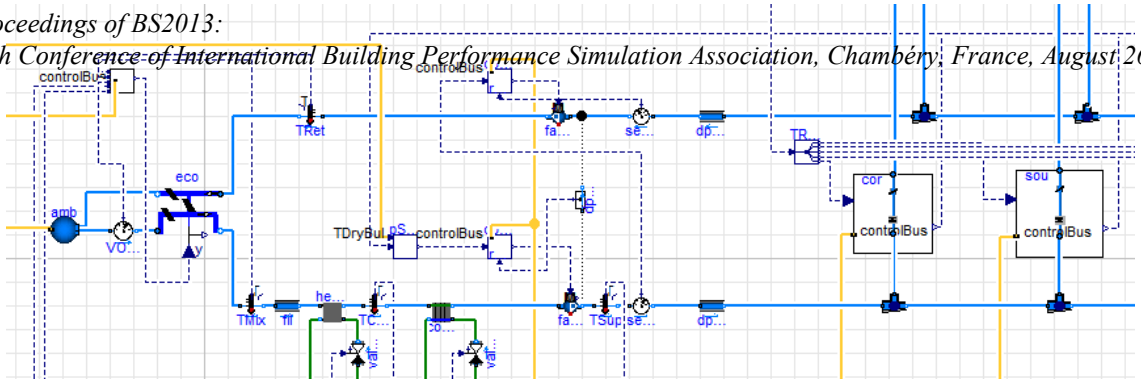

Figure 4 AHU and terminal units model (only two terminal units are shown)

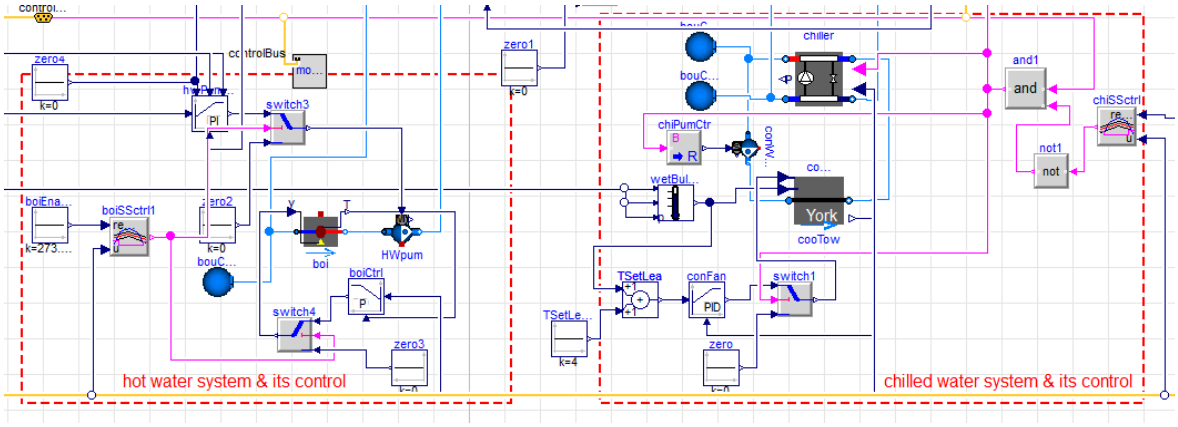

Figure 5 Chiller and boiler models

based on the duct static pressure distribution and the performance curves of the fans. Local loop control is implemented using proportional and proportional-integral controllers, while the supervisory control is implemented using a finite state machine.

The chiller, boiler and their associated hydronic system models are shown in Figure 5. Both the chiller model and the boiler model are steady-state models using performance curves. The default control sequences are obtained from the continuous commissioning guidebook (Liu et al., 2001). The chilled water pump and hot water pump are modulated to maintain the cooling coil valve and heating coil valve at 90% open. The condenser water temperature set point is reset based on outdoor wet-bulb temperature. The chilled water and hot water supply temperature set points are taken from the user's input through the web interface.

Functional Mockup Unit

The FMI is a tool independent and non-proprietary standard for model exchange and co-simulation of dynamic models. Version 1.0 of the standard, which is used in this framework, consists of three different parts: a) FMI for Model Exchange (MODELISAR-Consortium, 2008-2012c), b) FMI for Co-Simulation (MODELISAR-Consortium, 2008-2012b), and c) FMI for Product Lifecycle Management (PLM) (MODELISAR-Consortium, 2008-2012d). In our framework, we use the FMI for Co-Simulation Application Programming Interface (API) which requires a simulation model to be exported with its solver. Co-simulation refers in this context to a technique that allows individual subsystem models, possibly described by differential algebraic or discrete equations, to be simulated by different simulation programs running simultaneously and exchanging data during run-time.

Under the FMI standard, models or simulators are packaged as a Functional Mockup Unit (FMU). An FMU is a zip file with the extension .fmu. This zip -file contains a model description file in xml format, the needed run-time libraries used in the model, and/or binaries for one or several target machines, as well as resources files needed to run or describe the model or simulator.

The model description file is an XML-file which contains a list and definition of all variables of the FMU that are exposed to the environment in which the FMU shall be used, as well as other model information.

The runtime libraries contain the set of C-functions needed to interface with the model. These functions are used in a co-simulation scenario to set the inputs of the FMU, get its outputs and advance time.

In our application, before creating an FMU, we need to define the input and outputs of the model, which will be exposed for data -exchange. This can be done in the Modelica model with *RealInput*, *BooleanInput*, *RealOutput*, and *BooleanOutput* connectors. These connectors must be linked to input and output variables that will be interfaced and displayed through the web interface. We distinguish in the framework between three data types: parameters, inputs and outputs. Parameters are sent to the model before the simulation starts, and are constant during the simulation, e.g. the window to wall ratio, the building location, the construction material properties. Inputs are time-variant and sent to the

model during simulation run time, e.g. the control set points. Outputs are the simulated results computed by the model. They are sent to the 3D graphics to animate the HVAC system operation.

To export the FMU, we used Dymola 2013 which supports FMU for Co-Simulation export.

Since Dymola 2013 does not support for external resources like data files and tables etc. in the FMU, the weather file must be supplied with the FMU and put in the working directory.

FMU Wrapper

An FMU must be unzipped before it can be simulated. In our case, the archive is unzipped at authoring time and stored on the server as a folder that contains several files.

The binary file is a dynamically linked library which contains compiled native code that must be loaded into memory and executed using C function calls. These functions cannot be called directly by the Java servelet and therefore must be wrapped to allow access from the Java Virtual Machine, hence the name 'FMU Wrapper.'

The FMU SDK provided by Qtronic (Qtronic 2010) served as a starting point for developing the wrapper. The SDK code was combined with custom written C++ code to assist with session management. The Model View Controller (Trygve 1979) and Finite State Machine (Wright 2005) design patterns were used when developing this part of the FMU wrapper. The top layer of the FMU Wrapper is a set of C function which conform to the Java Native Interface (Oracle 2012) standards and can be called from Java using Java Native Access (JNA) (Wall 2012).

Java Web Service

A multi-threaded web service is implemented using the Jetty HTTP Java Servlet as its core. The web service uses JNA function calls to access the FMU. The server implements a WebSocket server which conforms to RFC 6455 (IETF 2011).

When a new request is received, a new session is started and the server initializes the simulation through the FMU Wrapper API. Results are encapsulated in Object Oriented JavaScript Object Notation (JSON) messages which conform to RFC 4627 (IETF 2006). These JSON messages are then pushed to the client.

Client

The e-learning client runs entirely in the student's web browser. It is built using HTML5 (W3C 2012) and JavaScript. The client uses standard HTTP requests to open the initial page and embedded scripting code. A WebSocket client implemented in JavaScript then sends JSON requests to the Java Web Service described above. The client also renders 3D content in the web browser using the THREE.JS JavaScript Library which conforms to the WebGL standard (Khronos 2012).

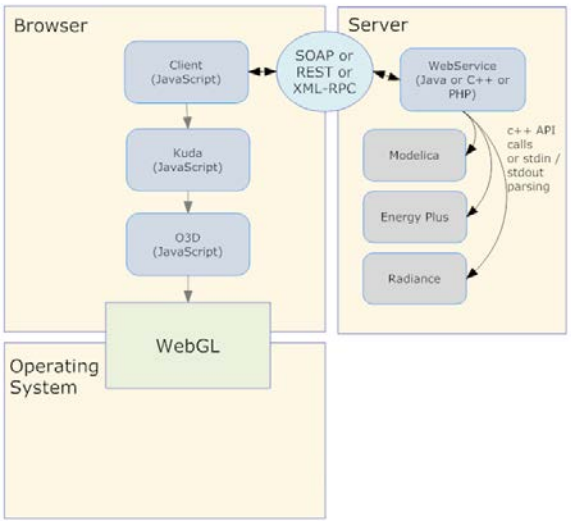

Figure 6 Web service architecture

PRELIMINARY TESTING

Two tests have been conducted to verify that 1) the FMU wrapper can run the FMU, and 2) the simulation results from the FMU agree with the simulation results generated when simulating the model directly in Dymola using the Radau solver, which is a fifth order differential algebraic equation solver.

The HVAC system model used in our tests contains 515 differentiated variables and 19,527 equations, and the maximum size after manipulation of the nonlinear systems by Dymola is 20. The PC that was used to run the tests has a Windows 7 64-bit Operation System, with Intel Core i7 CPU 870@2.93GHz and 8GB memory.

One summer day of the Typical Meteorological Year 3 (TMY3) weather data on July 1st for Chicago, IL was used for the tests. The FMU wrapper successfully initialized and started the simulation, and the simulation completed in 30 seconds using the SUNDIALS CVODE solver, which is used by Dymola for generating co-simulation FMUs.

The original HVAC system model was simulated in Dymola using the Radau solver for the same time period and completed in 79 seconds.

A few simulation results, i.e. the cooling coil valve position, the south zone air temperature, and the chiller electric power, were selected to compare the simulation results that were generated by the FMU wrapper and by Dymola. These comparisons are presented in Figure 7-9. As can be seen from the figures, the results match closely. Slight differences are seen at the start of the simulation.

CONCLUSION

We implemented a software architecture for a simulation-driven, 3D web animation interface for e-learning of high performance buildings. FMI was used for the data communication between the

simulation and the web server. A representative HVAC model as well as the envelope model were created using the Modelica Buildings Libray. The completed model is a fairly complex model with about 20,000 equations and 500 state variables. The FMU wrapper successfully initialized and started the FMU, and the simulation completed in 30 seconds, which is less than half of the time Dymola used. The simulation results matched closely with that generated natively in Dymola.

REFERENCES

Deringer, J.J., Nahman, J.E., Heming, K., Wetter, M., Pang, X. and Konstantoglou, M. 2012. LearnHPB and eLAD - Two Related Online eLearning Platforms for High Performance Buildings. 2012 ACEEE Summer Study on Energy Efficiency in Buildings, Aug 12-17, 2012, Pacific Grove, CA USA.

Deru, M. et al. 2009. DOE commercial building research benchmarks for commercial buildings. Technical report, U.S. DOE, Washington, DC, 2009.

Wetter, M. 2009. Modelica-based Modelling and Simulation to Support Research and Development in Building Energy and Control Systems. Journal of Building Performance Simulation, v2(2): 143-161.

TARCOG. 2006. Mathematical models for calculation of thermal performance of glazing systems with or without shading devices, technical report, Carli Inc., Oct, 2006

ASHRAE. 2006. Sequence of operation for common HVAC systems. ASHRAE, Atlanta, GA 2006.

Liu, M., Claridge, D. and Turner, W. 2001. Continuous Commissioning Guidebook for Federal Energy Managers. Federal Energy Management Program, U.S. Department of Energy.

IETF (2011) - Internet Engineering Task Force Request for Comments: 6455

http://tools.ietf.org/html/rfc6455

IETF (2006) - Internet Engineering Task Force Request for Comments: 4627

http://tools.ietf.org/html/rfc4627

Khronos (2011) – Khronos Group WebGL Specification

https://www.khronos.org/registry/webgl/specs/1.0/

Qtronic (2010) – QTronic GmbH - FMU SDK 1.0.2

http://www.qtronic.de/en/fmusdk.html

Trygve (1979) - The Model-View-Controller (MVC)

Its Past and Present - Trygve Reenskaug, University of Oslo.

http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf

Wright (2005) - Finite State Machines - David R. Wright - N. Carolina State Univ

http://www4.ncsu.edu/~drwrigh3/docs/courses/csc216/fsm-notes.pdf

Oracle (2012) - Java Native Interface Specification – Oracle, Inc.

http://docs.oracle.com/javase/7/docs/technotes/guides/jni/index.html

Wall (2012) - JNA API Documentation - Timothy Wall

http://twall.github.com/jna/3.5.1/javadoc/

Modelisar-Consortium. 2008-2012a. *Functional Mock-up Interface* [Online]. Available: https://fmi-standard.org/ [Accessed January 14 2013].

Modelisar-Consortium. 2008-2012b. *Functional Mock-up Interface for Co-Simulation* [Online]. Available: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_CoSimulation_v1.0.pdf [Accessed January 14 2013].

Modelisar-Consortium. 2008-2012c. *Functional Mock-up Interface for Model-Exchange* [Online]. Available: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_ModelExchange_v1.0.pdf [Accessed January 14 2013].

Modelisar-Consortium. 2008-2012d. *Functional Mock-up Interface for Product Lifecycle Management* [Online]. Available: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_PLM_v1.0.pdf [Accessed January 14 2013].

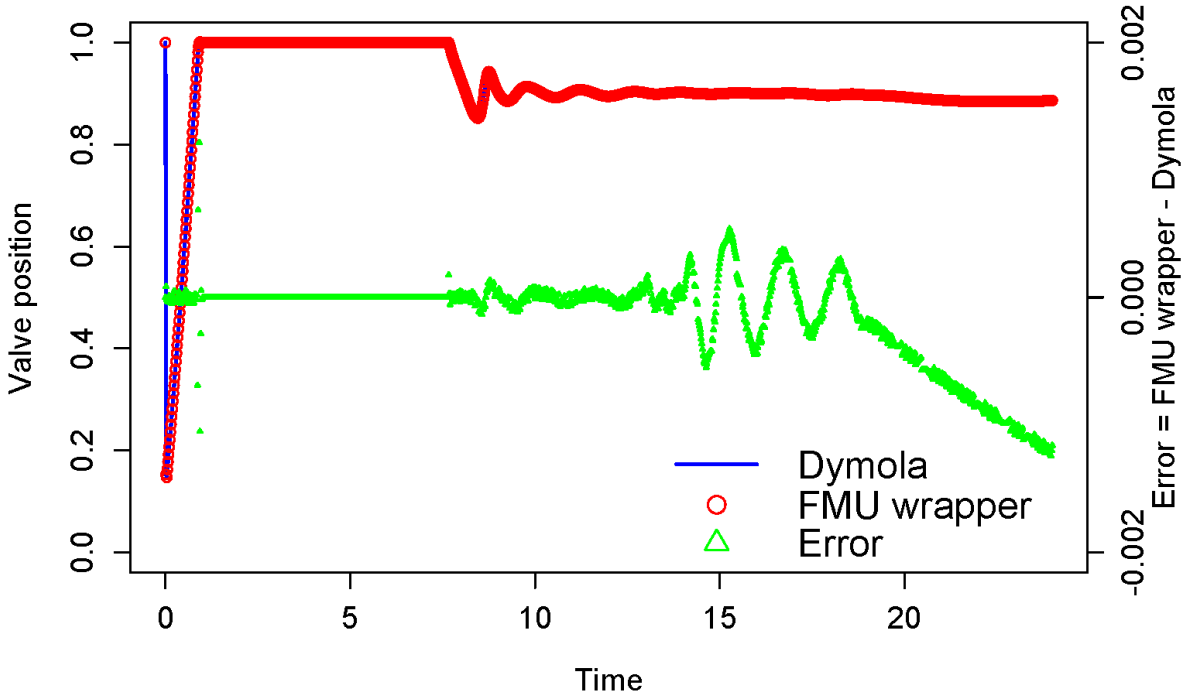

Figure 7 Cooling coil valve position comparison

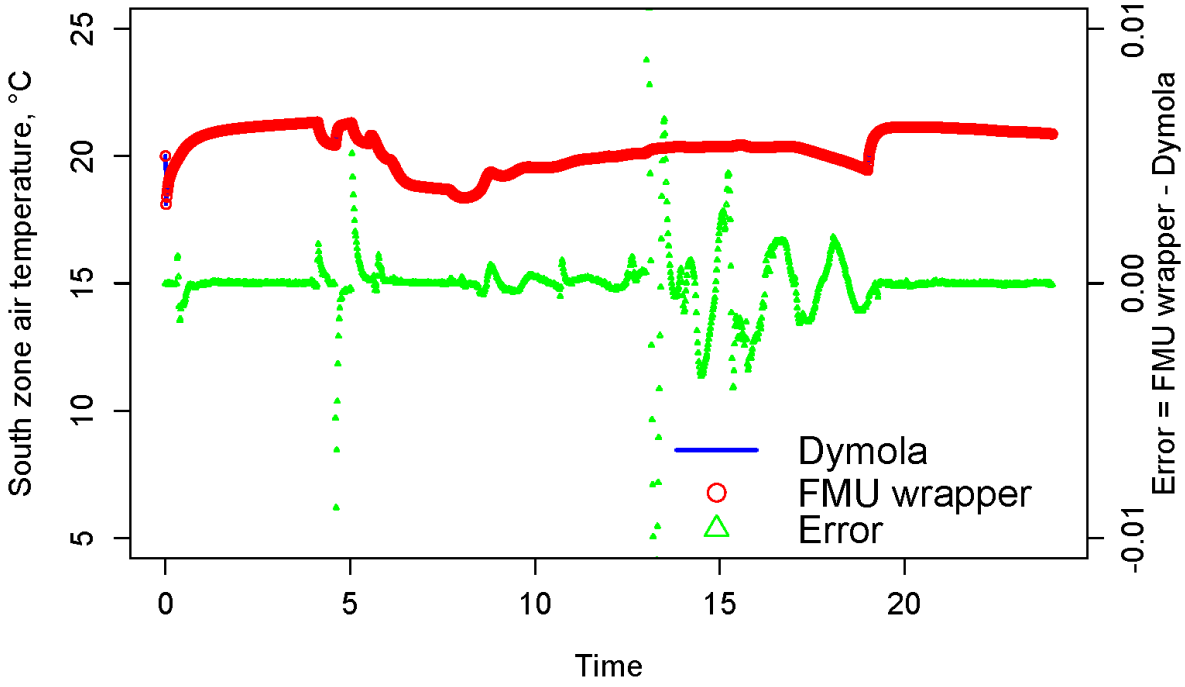

Figure 8 Room air temperature comparison

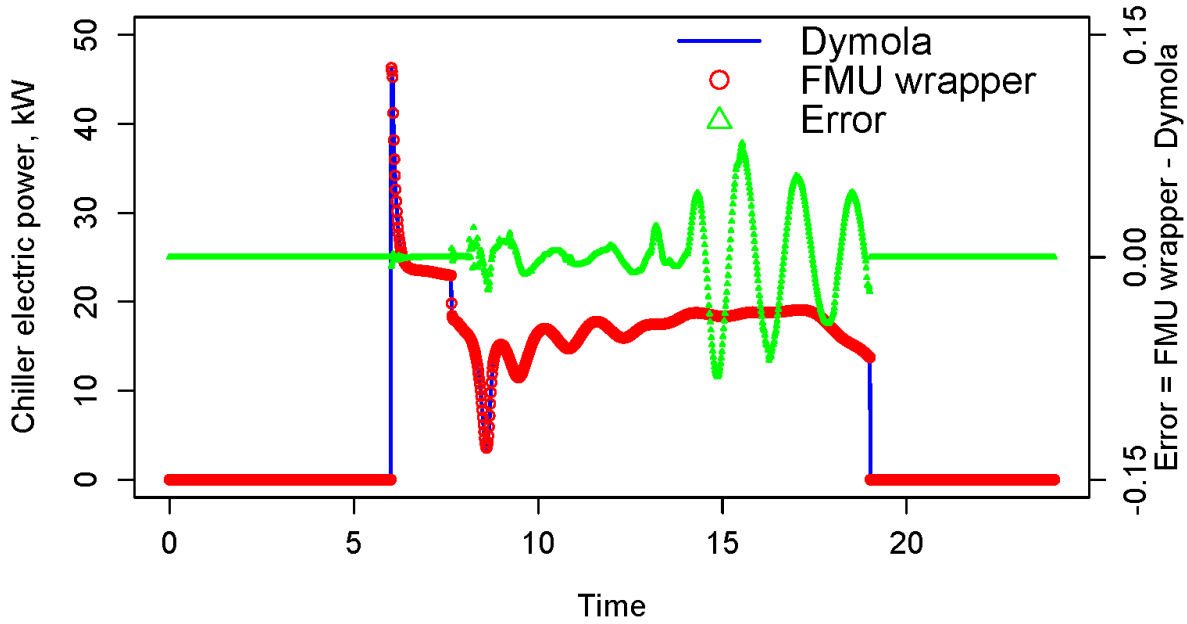

Figure 9 Chiller electric power comparison