BERGISCHE UNIVERSITÄT WUPPERTAL

Control logic developed in building performance simulations for the operation of HVAC systems

Options, applications, and opportunities
in a digital twin framework

Dissertation
zur Erlangung eines Doktorgrades
(Dr.-Ing.)

in der
Fakultät für Architektur und Bauingenieurwesen
der
Bergischen Universität Wuppertal

vorgelegt von
Karl Walther
aus Herdecke

Wuppertal 2023

Vorsitz der Prüfungskomission:	Prof. Prof. Dr. rer. nat. Lukas Arnold
1. Gutachter:	Prof. Dr.-Ing. Karsten Voss
2. Gutachter:	Prof. Dr.-Ing. Jens Pfafferott
Mitglied der Prüfungskomission:	Prof. Dr. Jochen Müller

Tag der Einreichung:	09.11.2023
Tag der Verteidigung:	09.04.2024

Abstract

At around 40 %, the building sector makes a significant contribution to final energy consumption and CO_2 emissions in the EU. At the same time, user comfort and well-being play a major role in building operation. Building automation systems (BAS) with intelligent controls enable the efficient operation of heating, ventilation, and air conditioning (HVAC) systems to ensure the desired comfort level. However, the increasing complexity of HVAC systems in buildings is a major challenge for the development of intelligent controls.

In building practice, a number of deficits can be observed at the interface between HVAC systems and BAS: Firstly, the level of detail of controls often remains low in the design phase and programming only takes place shortly before commissioning. Secondly, textual and graphical formats are used to communicate controls between planning offices and contractors, which are often ambiguous. Finally, the control actually implemented during operation is often unclear. As a result, this leads to discrepancies between expectations and measured performance and increased energy consumption overall. These deficits also make the efficient use of digital twins more difficult. Such digital twins are a promising approach that links models of buildings with data from live operation. This promotes a variety of model-based use cases in operation such as performance gap analyses, fault detection, or "what-if" analyses.

Against this background, this work is dedicated to the question of how to ensure that HVAC systems are operated exactly according to specifications from the design phase. The tools and methods required for this are described in a three-step approach in planning, commissioning and operation. The starting point is the detailed development and definition of controls for HVAC systems at code level as early as the design phase. These controls are tested on building and HVAC models in Building Performance Simulation (BPS) environments. This creates a detailed Digital Twin of buildings, HVAC systems and controls as early as the planning phase. The direct implementation of the previously developed controls in the commissioning phase eliminates the performance gap caused by textual and graphical communication formats. Various approaches are described and compared for this transfer. Such an approach opens up a number of opportunities, above all the development of efficient controls that are tailored to the building characteristics. At the same time, it ensures that the Digital Twin and the real building have identical controls, which enables the efficient implementation of model-based Digital Twin applications in building operation.

This approach is validated in three large-scale tests on air handling units (AHUs) in an industrial production building under realistic conditions. The experiments differ in how the control logic developed in BPS environments is implemented in building operation. The use of graphical schemas as a normative standard initially illustrates the basic suitability of the controls developed in BPS tools. For the first time, a prototype tool chain that allows the simulation of control code according to IEC 61131 in the BPS environment IDA ICE was then successfully implemented in a large-scale building application. The PLCopen XML format then enables the digital transfer from the planning phase to building controllers. In addition, the direct execution in the loop of controllers in IDA ICE for the operation of AHUs was also successfully implemented.

In all three implementations, considerable savings were achieved compared to the original operation. The results and experiences thus underline the potential of model-based control development. Performance analyses of simulation and measurement at control, system and room level illustrate deviations that can be attributed to model simplifications. The analyses also include organizational aspects at the interface between planning and execution.

The described and validated methods contribute to increasing the quality of the planning phase with regard to controls for HVAC systems. The combination of model-based design and model-based applications in operation provides important impetus for the establishment of Digital Twins in the building sector.

Zusammenfassung

Der Gebäudesektor trägt mit rund 40 % einen erheblichen Beitrag zum Endenergieverbrauch und CO_2-Emissionen in der EU bei. Gleichzeitig haben Nutzerkomfort und Wohlbefinden einen großen Stellenwert im Gebäudebetrieb. Gebäudeautomationssysteme mit intelligenten Regelungen ermöglichen den effizienten Betrieb der Technischen Gebäudeausrüstung (TGA) zur Gewährleistung des angestrebten Komfortlevels. Die zunehmende Komplexität von TGA-Systemen, insbesondere von HLK-Systemen (Heizung, Lüftung, Klima), ist allerdings eine große Herausforderung für die Entwicklung von intelligenten Regelungen.

In der Gebäudepraxis sind an der Schnittstelle von TGA-Systemen und Gebäudeautomation eine Reihe von Defiziten zu beobachten: Zum einen bleibt der Detailgrad von Regelungen in der Entwurfsphase oft gering und eine Programmierung findet erst kurz vor der Inbetriebnahme statt. Zum anderen werden für die Kommunikation von Regelungen zwischen Planungsbüros und ausführenden Firmen textliche und grafische Formate verwendet, die häufig nicht eindeutig sind. Schließlich ist die tatsächlich implementierte Regelung im Betrieb oft unklar. Im Ergebnis führt dies zu Abweichungen zwischen Erwartungen und gemessener Performance und insgesamt erhöhten Energieverbräuchen. Zudem erschweren diese Defizite den effizienten Einsatz von Digitalen Zwillingen. Solche Digitalen Zwillinge sind ein vielversprechender Ansatz, der Modelle von Gebäuden mit Daten aus dem Live-Betrieb verknüpft. Dies fördert eine Vielzahl modellbasierten Anwendungsfälle im Betrieb wie Performance Gap Analysen, Fehlererkennung oder "what-if"-Analysen.

Vor diesem Hintergrund widmet sich diese Arbeit der Frage, wie erreicht werden kann, dass TGA-Systeme exakt nach Festlegungen aus der Entwurfsphase betrieben werden. Die dazu erforderlichen Werkzeuge und Methoden sind in einen dreistufigen Ansatz in Planung, Inbetriebnahme und Betrieb beschrieben. Ausgangspunkt ist die detaillierte Entwicklung und Definition von Regelungen für TGA-Systeme auf Code-Ebene bereits in der Entwurfsphase. Die Prüfung dieser Regelungen erfolgt an Gebäude- und Anlagenmodellen in Building Performance Simulationen (BPS). So entsteht bereits in der Planungsphase ein detaillierter Digital Twin von Gebäuden, TGA und Regelungen. Die direkte Implementierung der zuvor entwickelten Regelungen in der Inbetriebnahmephase beseitigt den durch textliche und grafische Kommunikationsformate verursachten Performance

Gap. Für diesen Transfer werden verschiedene Ansätze beschrieben und miteinander verglichen. Ein solches Vorgehen eröffnet eine Vielzahl von Chancen, allen voran die Entwicklung von effizienten Regelungen, die auf die Gebäudecharakteristik abgestimmt sind. Gleichzeitig wird so sichergestellt, dass der Digital Twin und das reale Gebäude über identische Regelungen verfügen, was die effiziente Durchführung modellbasierter Digital Twin-Anwendungen im Gebäudebetrieb ermöglicht.

Dieser Ansatz wird in drei großmaßstäblichen Versuchen an Lüftungsanlagen in einem industriellen Produktionsgebäude unter praxisnahen Bedingungen validiert. Die Versuche unterscheiden sich darin, wie die in BPS-Umgebungen entwickelte Regelungslogik im Gebäudebetrieb implementiert wird. Die Verwendung grafischer Schemata als normativem Standard verdeutlicht zunächst die grundsätzliche Eignung der in BPS entwickelten Regelungen. Zum ersten Mal wurde dann eine prototypische Toolkette, die die Simulation von Regelungscode nach IEC 61131 in der BPS-Software IDA ICE erlaubt, erfolgreich in einer großmaßstäblichen Gebäudeanwendung umgesetzt. Das PLCopen XML Format ermöglicht dann den digitalen Transfer aus der Planungsphase auf Gebäudecontroller. Darüber hinaus wurde auch die direkte Ausführung "in the loop" von Reglern in IDA ICE für den Betrieb der Lüftungsanlagen erfolgreich umgesetzt.

In allen drei Implementierungen konnten erhebliche Einsparungen gegenüber dem ursprünglichen Betrieb erreicht werden. Die Ergebnisse und Erfahrungen unterstreichen damit das Potential von modellbasierter Regelungsentwicklung. Performance-Analysen von Simulation und Messung auf Regelungs-, Anlagen- und Raumebene verdeutlichen Abweichungen, die auf Modellvereinfachungen zurückzuführen sind. Die Analysen umfassen auch organisatorische Aspekte an der Schnittstelle von Planung und Ausführung.

Die beschriebenen und validierten Methoden tragen dazu bei, die Qualität in der Planungsphase in Bezug auf Regelungen für TGA-Systeme zu erhöhen. Die Verbindung modellbasierter Planung und modellbasierten Anwendungen im Betrieb leistet wichtige Impulse für die Etablierung von Digital Twins im Gebäudesektor.

Acknowledgements

This thesis was written during my work as a research associate at the Chair of Building Physics and Technical Building Services at the University of Wuppertal, Germany. First and foremost, I would like to thank my supervisor Prof. Karsten Voss for his continuous availability for technical discussions, his critical guidance and feedback, and his support and encouragement in the implementation of my ideas. I am very grateful to Prof. Jens Pfafferott for taking on the second supervisor's report.

I would like to thank my colleagues at the b+tga team, especially Işıl Kalpkırmaz-Rızaoğlu, Tuğçin Kırant-Mitic, and Tjado Voß, for the friendly and humorous working atmosphere. The work is thematically linked to the research project VEProB, funded by the Bundesministerium für Wirtschaft und Klimaschutz (BMWK). Many thanks to Matthias Schmude, Cebastien Foumouo-Tsakou, Jens Oppermann, and Stephan Schmied from the Wilo company for the goal-oriented and trusting cooperation in the VEProB project which supported the transfer of scientific approaches into practice. I would also like to thank Michael Krüttgen from the TH Köln University of Applied Sciences for the professional exchange in the field of building automation.

This work has been significantly enriched by a research stay in Stockholm. I would like to thank Marco Molinari for hosting me at the Royal Institute of Technology KTH and for many stimulating discussions. I am also thankful to Sven Moosberger and Per Sahlin for the opportunity for in-depth collaboration with the software developer Equa. Special thanks to Patrik Skogqvist and Markus Högberg for their support in applying new features in IDA ICE.

Many thanks to Daniel, Julia, Tobias and Christian for their feedback on the text.

My family and friends accompanied me during the work on my dissertation and were and are a great joy to me, for which I am very grateful.

Contents

List of Figures

List of Tables

Acronyms

AHU air handling unit iv, xiv, 1, 13, 17–19, 23, 39, 40, 43, 44, 46–54, 58, 61, 62, 70, 77, 78, 80–83, 91–93, 95, 97, 108, 112, 114, 116, 117, 119, 120, 122–125

ASHRAE American Society of Heating, Refrigerating and Air-Conditioning Engineers 19, 62, 65, 68, 101, 116, 122

BA building automation 54, 67, 69, 114

BACnet Building Automation and Control Networks 17, 40, 44, 70

BAS building automation system iii, 2, 3, 5, 6, 8–11, 14, 16–18, 20, 24, 26, 46, 53, 55, 63, 68, 70, 78, 80, 109, 116, 117, 119–121, 123

BCVTB Building Controls Virtual Test Bed 23

BIM Building Information Modeling 10, 28, 34, 62, 117, 124

BMS building management system 24, 35, 43, 58, 71, 92

BMWK Federal Ministry for Economic Affairs and Climate Action vii

BPS Building Performance Simulation iii, iv, 3, 7–11, 13, 14, 17, 20–22, 25, 26, 31, 35, 36, 46, 48, 49, 53–58, 60, 62, 63, 65, 68–72, 74, 75, 77–80, 86, 95, 106, 108–110, 112, 114–125

CDL Control Description Language 8, 19, 26, 36, 64, 65, 67, 71, 72, 74, 80, 109, 116, 120, 121, 124

CHP combined heat and power 4, 17, 19, 59

CXF Control Exchange Format 8, 74

DAE Differential Algebraic Equation 22, 24, 68, 71

DIN Deutsches Institut für Normung xiii, 5, 14, 15, 78, 85, 104

DIN V Deutsches Institut für Normung Vornorm 13

DLL Dynamic-link library 65, 116

DSM demand side management 60

DTE Digital Twin Environment xiii, 34, 35, 92, 116

DTI Digital Twin Instance xiv, xvii, 27–33, 55, 56, 71–74, 85, 86, 90, 92, 116, 123

DTP Digital Twin Prototype xiv, 27, 30, 33, 55, 56, 72–74, 85, 86, 90, 115, 116, 123

EBC Energy in Buildings and Communities 60

eFMI Functional Mock-up Interface for embedded systems 63, 124
EN Europäische Norm xiii, 5, 14–16, 78, 85, 104
EPBD Energy Performance of Buildings Directive 13, 60, 117
EU European Union 1, 13, 60, 64, 67, 68

FBD Function Block Diagram 17, 65, 85, 91
FDD Fault Detection and Diagnosis 2, 9, 31, 33–35, 71, 74, 115
FLC Fuzzy Logic Control 16
FMI Functional Mock-up Interface 23

GEG Gebäudeenergiegesetz 13
GUI graphical user interface 35

HIL Hardware-In-the-Loop 20, 24, 55, 63, 69
HOAI Fee Structure for Architects and Engineers 6, 54
HVAC heating, ventilation, and air conditioning iii, iv, 2–6, 8–11, 13, 14, 16, 18–22, 24, 35, 36, 39, 46, 48, 53–63, 68–71, 80, 81, 107, 108, 113–115, 118–121, 123–125

ICT information and communications technology 43, 82
IDE Integrated Development Environment 18, 20, 21, 43, 44, 49, 63, 65–69, 71, 77, 78, 85, 86, 90, 109, 111, 112, 116, 121, 124
IEA International Energy Agency 60
IEC International Electrotechnical Commission iv, vi, xiv, 8, 17, 18, 20, 26, 40, 43, 44, 49, 56, 64, 65, 67, 68, 77, 80, 82, 85–87, 93, 109, 120, 121, 124
IL Instruction List 17
IoT Internet of Things 17
ISO International Organization for Standardization xiii, 14–16, 78, 85, 104
IT Information technology 69, 80, 90, 116

KTH KTH Royal Institute of Technology 43

LBNL Lawrence Berkeley National Laboratory 64
LD Ladder Diagram 17

MPC Model Predictive Control 3, 9, 16, 28, 29, 31, 33–35, 74, 85, 115, 124

NMF Neutral Model Format 23
NNC Neuronal Network Control 16

OBC OpenBuildingControl 8, 26, 64, 116

OPC UA Open Platform Communications Unified Architecture 17, 29, 69–71, 77, 79, 80, 90, 91, 109, 112, 116, 121, 125
OSCAT Open Source Community for Automation Technology 18, 25, 82, 86, 90, 108

PLC programmable logic controller 8, 17, 18, 20, 36, 43, 44, 64–69, 77–80, 85, 86, 90, 91, 93, 109, 110, 112, 114, 121, 124
PLM Product Lifecycle Management 27
PM Predictive Maintenance 31, 34, 115
PV photovoltaic 1

RL Reinforced Learning 16
RMSE Root Mean Square Error 102
RQ research question 9
RTE Runtime Environment 18

SFC Sequential Flow Chart 17
SIL Software-In-the-Loop 20, 24, 55, 63
SQL Structured Query Language 43
SRI Smart Readiness Indicator 60
ST Structured Text 17, 49, 82, 85, 86, 91

TABS thermally activated building system 33, 81, 86, 90
TMon Technical Monitoring 2

UI user interface 71
US United States 2, 19, 64, 67, 68

VAV variable air volume 19, 43, 51, 54, 82, 83, 91, 97, 99, 101, 108
VDI Verein Deutscher Ingenieure xiii, 14–16, 58, 85
VEProB Connected Energy Flows of Production and Office Buildings vii, 39, 49
VOB Construction Contract Procedures 6
VOC Volatile Organic Compounds 83

XML Extensible Markup Language iv, vi, 8, 20, 21, 36, 43, 64–69, 71, 72, 74, 77–80, 90, 95, 99, 109, 111, 112, 120, 121, 124

1 Introduction

1.1 Background and motivation

Buildings play a key role in our society for the responsible use of natural resources and for climate protection, as well as a place to live and work. Two figures illustrate this: Today, the share of buildings in final energy consumption and CO_2 emissions in the European Union (EU) is about 40 % (European Commission 2020). Moreover, people spend 90 % of their time indoors (Klepeis et al. 2001). In the context of climate change, the problem of summer overheating and the increasing need for cooling is also coming to the fore in regions with generally temperate climates.

Energy consumption and emissions in the building sector are divided into the construction and operation phases. In addition, the consumption of resources for the production of building materials must also be taken into account. This work focuses on the energy consumption and emissions that occur during the operation of buildings. Although the goal in the EU is to decarbonize the energy supply by using renewable energy sources, high expenditures are required to install wind power, photovoltaic (PV) and biomass plants and to build the necessary transmission infrastructure. Therefore, reducing consumption, first by eliminating unnecessary consumption and then by using efficient components, remains an essential goal for achieving sustainable buildings.

Over the past few decades, fundamental results have been achieved in research and development that enable energy-efficient operation as well as an increase in indoor comfort. These include high-performance materials for insulation, glazing and shading to improve the quality of the building envelope. In addition, energy efficient components such as heat recovery in air handling units (AHUs), heat pumps, variable speed pumps and fans, and LED lighting are now available on the market and commonly used in buildings.

Parallel to the development of components, advances in sensor technology, automation systems and data analytics, as well as more precise modeling and dynamic simulation methods, have brought the expected and measured performance, especially in terms of energy and comfort, to the foreground (Voss et al. 2016; Wilde 2018). Although buildings are increasingly equipped with the above mentioned innovative and high performing components, the measured performance of buildings when operating often falls below expectations. This so-called “performance gap” (Figure 1.1) is related to several aspects

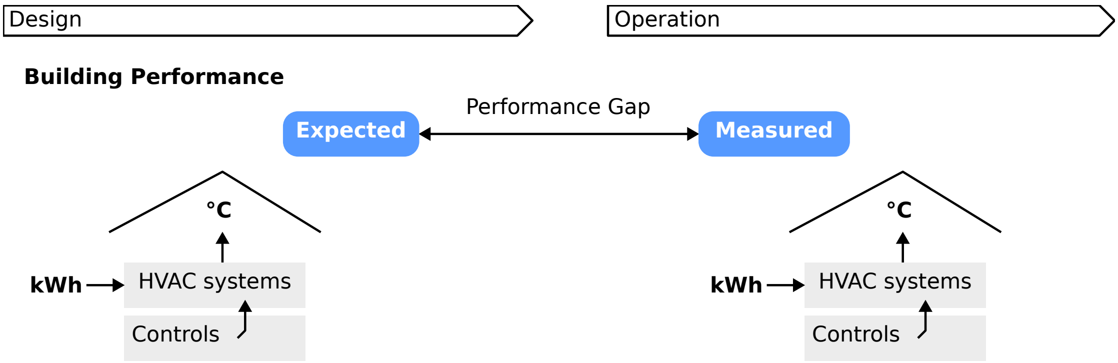


Figure 1.1: Performance gap of HVAC systems between design and operation

including energy efficiency, thermal, visual and acoustic comfort and indoor air quality (Wilde 2014). Causes for the performance gap, such as incorrect assumptions, changes during implementation, user behavior, or measurement uncertainties, can be attributed to the design, construction, or operation phases (Wilde 2014).

A key factor influencing building performance is the control of heating, ventilation, and air conditioning (HVAC), lighting and shading systems (Figure 1.1). Even "low-tech" buildings such as the famous "2226" building, which does not use active heating and cooling systems, require sophisticated control strategies for natural ventilation and temperature management (Walther et al. 2021). The importance of controls is expected to increase in the coming years as renewable energy sources require flexible buildings in interaction with heat and electricity grids (Schluck, Kräuchi, and Sulzer 2015; Sommer et al. 2020). In addition, with the greater proliferation of heat pumps, generators will be installed that are more demanding in terms of operational management.

However, poor control performance due to programming and implementation errors has been reported for more than two decades (Barwig et al. 2002; Waide et al. 2014). According to a survey of practitioners by Fütterer, Schild, and Müller (2017), 40 % of respondents cited errors in the design and implementation process of controls as the main reason for problems related to building automation systems (BAS). Fernandez et al. (2017) estimate possible energy savings of 29 % for the US commercial building sector if optimized control sequences are implemented.

In order to address the poor performance of control systems during operation, services for Technical Monitoring (TMon) and Fault Detection and Diagnosis (FDD) have been developed and established on the market in the last few years (Plesser et al. 2010; Granderson et al. 2018). However, the assignment and correction of faults remains complex because the underlying causes can be located in different technical systems, such as generator components, hydraulic distribution network, pumps or valves, as well as in the programming and configuration of controls. In order to efficiently address the

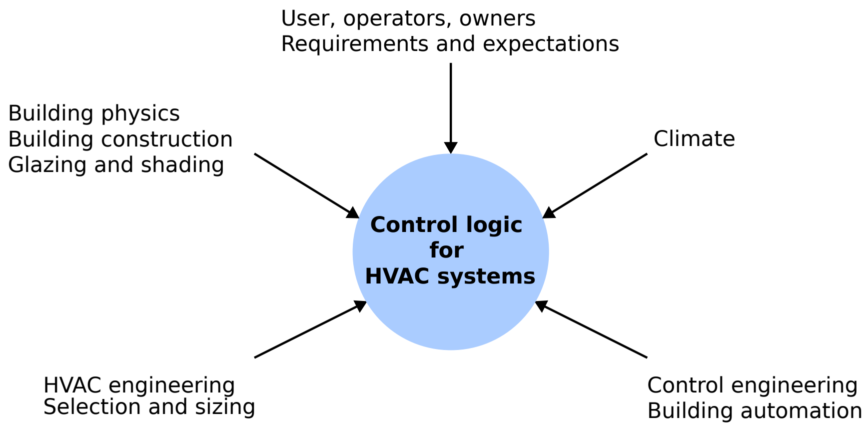


Figure 1.2: Complexity of the control design task

root causes of problems during operation, the quality and level of detail in the design phase must be increased (Egger et al. 2013). With respect to the operation of HVAC systems, fundamental innovations are needed in the way controls are developed, tested and deployed in buildings (Sahlin, Bring, and Eriksson 2009). Ideally, controls and HVAC systems are developed and tested in Building Performance Simulation (BPS) environments during the planning phase and implemented automatically in BAS. Such an approach would also promote the use of Digital Twins in the building sector. Digital Twins are a promising approach from Industry 4.0 that links models with measured data during operation. This enables a variety of model-based use cases in operation such as fault detection, "what-if" analyses or Model Predictive Control (MPC).

1.2 Challenges and problems

This work is generally concerned with all building energy systems that can be controlled automatically. This typically includes HVAC systems, movable shading and lighting (Harish and Kumar 2016). This thesis focuses specifically on the operation and control of HVAC systems. Compared to lighting and shading systems, they are much more heterogeneous and generally have a greater impact on overall building performance. Two main challenges have been identified with respect to the development of HVAC controls, which are outlined in the following sections.

1.2.1 Complexity of buildings and systems

The first and most common problem is complexity, which occurs on multiple levels:

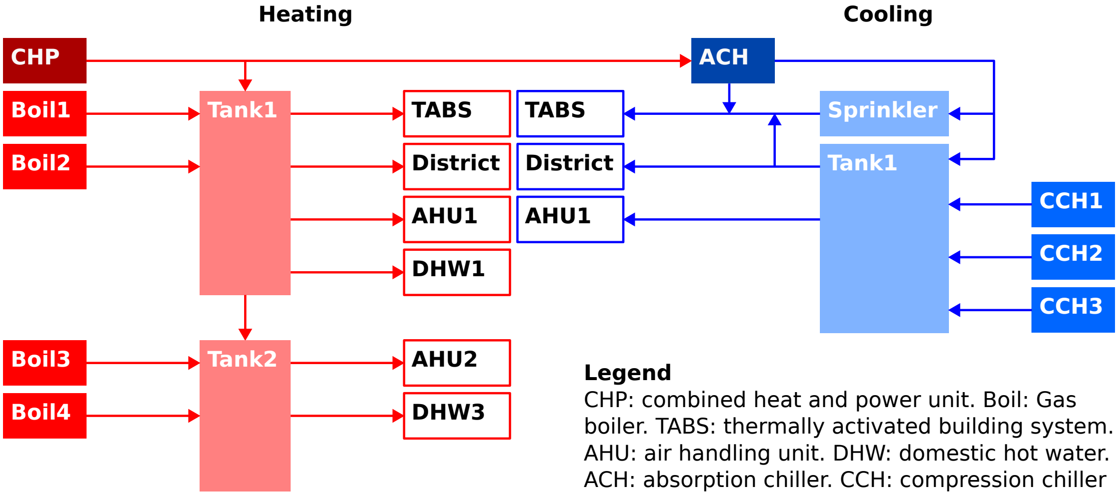


Figure 1.3: Simplified energy flow scheme for a complex HVAC system with trigeneration and multiple heat and cold generators and consumers

1. *Complexity of buildings*

   Understanding the dynamic interaction between building structure, climate, occupants, HVAC systems, shading, lighting, and controls is a highly complex task. Specific knowledge from these very different domains is needed to develop integrated and optimized solutions (Figure 1.2). For example, cooling systems can be omitted if overheating is limited by intelligent operation of movable shading systems or natural ventilation. In today's practice, however, components and systems in buildings are typically designed and sized separately, without consideration of their interaction. As a result, systems may be oversized or even unnecessary in actual operation. In addition, when several different systems are combined, the overall complexity increases. This in turn makes control design very challenging and often leads to problems during commissioning and operation.

2. *Complexity of HVAC systems*

   Building level HVAC systems typically consist of supply, distribution and consumer systems. While a system in a residential building typically has only one type of supply and consumer system, plants in larger commercial building may have several different types with different characteristics and requirements. As an example, Figure 1.3 depicts a highly simplified energy flow scheme for the heating and cooling of an industrial building (see Section 3.1.1). The plant has systems operating at different temperature levels (for example gas boilers at 70 °C and a combined heat and power (CHP) unit at 98°C on the supply side) and a particularly challenging trigeneration with a CHP unit and an absorption chiller. Developing a working and efficient control strategy for such plants requires detailed design considering multiple operating modes under dynamic conditions. In addition, HVAC systems in larger buildings include several hundred or thousands of individual elements, such as duct sections, valves, pumps, dampers, merging and dividing components, inlets and outlets. All of these elements affect the system characteristics in terms of inertia, response times, and delays that affect the behavior of the system within a control loop.

3. *Interaction of HVAC design with control design*

   The design of controls and the selection and sizing of HVAC systems are closely related and require integrated engineering: For example, a heat pump can be selected smaller in size or with lower temperature levels if intelligent peak shaving or load shifting is applied along with thermal storage. In today's practice, HVAC engineering and control engineering are often separated, which hinders optimized solutions. The selection and sizing of HVAC systems is done by my mechanical engineers, using static calculations. The size of heat generators, for example, is determined by a quasi-static heat load calculation according to EN 12831 (DIN 2020). For the development of operation and control strategies, both HVAC and automation engineers can be involved. For example, according to the national standard DIN 18386:2019-09 (DIN 2019a), function lists, plant schematics, functional diagrams, and functional descriptions are a required design output for automation engineering. For the HVAC design, DIN 18380:2019-09 (DIN 2019b) and DIN 18379:2019-09 (DIN 2019c) lists, among others, function diagrams, functional descriptions, and automation schemes as required planning output. The fact that development and documentation of controls are described in both HVAC and automation engineering shows that the required tasks and responsibilities are not clearly assigned.

4. *Heterogeneity of building automation systems*

   Controls for HVAC systems are implemented on BAS. BAS themselves are highly heterogeneous with respect to hardware, communication interfaces, or programming languages (Domingues et al. 2016; Mishra and Wen 2018). Controls can be implemented on different devices and layers depending on the BAS architecture (Béguery, Kissavos, and Sahlin 2013). For example, a common separation is made between supervisory control and local loop control (Wang and Ma 2008). At the building level, supervisory control is used for the coordinated operation of HVAC systems, lighting and shading, while local loop control is applied to subsystems and individual actuators. The dependencies and interactions of controls at different levels must be considered and understood when developing integrated, project-specific solutions.

An important organizational aspect in the context of complexity that cannot be neglected are false incentives. In Germany, for example, according to the Fee Structure for Architects and Engineers (HOAI) (Bundesregierung der Bundesrepublik Deutschland 2020) the fees for engineering services depend on the construction sum. Thus, a higher construction sum leads to a higher fee for the design service provider. This supports the design of extensive HVAC systems without the need to prove the overall functionality in operation.

1.2.2 Design and implementation of controls

The second fundamental problem is a gap between the development of controls in the design phase and their implementation on BAS in the commissioning phase. In Germany, this gap is closely related to a separation of responsibilities for design tasks before and after the contract is awarded: according to the phase model of the HOAI and Construction Contract Procedures (VOB), design tasks are performed both by a design office before the contract is awarded and by the contractor performing the work (Figure 1.4).

Initially, design offices for the HVAC and control engineering are commissioned to provide conceptual and basic engineering services. The output is a so-called “execution planning”, which is used as a basis for tender documents. Although these phases are formally still product neutral, HVAC manufacturers are often involved to support the design offices and to select suitable products. These components often promise particularly energy-efficient operation. As a result, many technical details about components are already known (see bottom in Figure 1.4). In contrast, the level of detail for operational management and control strategies at the building and system level is usually still relatively low. Once the contract is awarded, the risk is transferred to the contractors who install the HVAC and automation systems and implement the controls (see top in Figure 1.4). According to the VOB, these executing contractors are also responsible for further detailed design and the preparation of a product-specific so-called “assembly planning” (Werk- und Montageplanung).

This construct is problematic for two reasons: firstly, design offices may be tempted not to develop and describe controls in full detail, leaving this task to the executing contractors. Secondly, the detailed and explicit definition and programming of controls takes place after the contract is awarded, when the HVAC systems are finally defined (bottom in Figure 1.4). In contrast, in the process industry, where HVAC systems are also used, execution and assembly planning are integrated into the “detail engineering” to avoid these problems (NAMUR 2020).

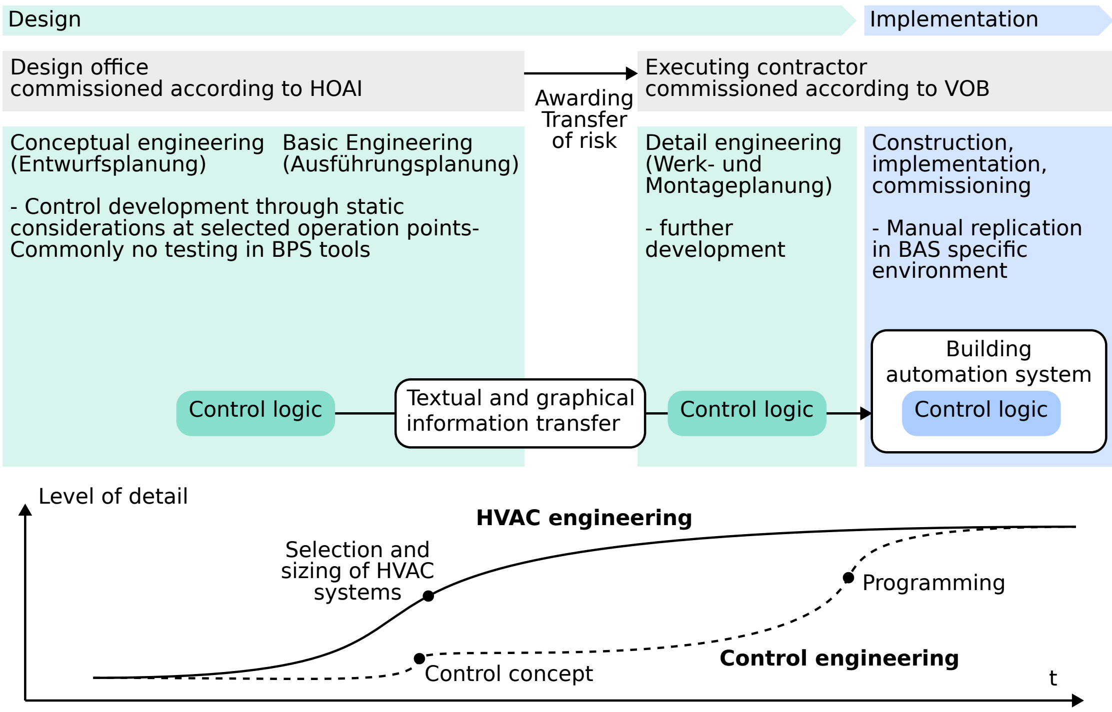


Figure 1.4: Development of control logic in planning and commissioning today

The following aspects are particularly problematic from a methodological and technical point of view (Figure 1.4):

1. *Development of controls*

   In today's practice, controls for building energy systems are drafted in the design phase on the basis of static considerations in single operating modes (left in Figure 1.4). This approach has remained unchanged for decades. With very few exceptions, controls are not developed and tested under dynamic conditions. However, BPS tools would allow the integrated simulation of building, systems and controls, and the advantages are documented for example by Kramer, van Schijndel, and Schellen (2017).

2. *Communication of controls*

   The communication of controls from a design office to an executing company is a critical step that determines whether the controls are implemented as intended (center in Figure 1.4). Today, graphical automation schematics and textual descriptions are commonly used for this communication. These formats are known to be error-prone, incomplete, and ambiguous, leaving room for interpretation in implementation (Fisch et al. 2017).

3. *Implementation of controls*

   To implement the controller in BAS, the information from the graphical and textual documentation is translated into program code (right in Figure 1.4). This is a manual process and is typically done in controller-specific programming environments.

In recent years, increased research and development efforts have been made to develop digital formats for the transmission of controls from BPS environments in order to automate their implementation in BAS. A major contribution has been made by the OpenBuildingControl (OBC) project. It defines a Control Description Language (CDL) and a Control Exchange Format (CXF) as a digital specification of control logic (Wetter et al. 2022). Sahlin, Skogqvist, and Högberg (2018) present a method to simulate control programs according to the industry standard IEC 61131-3 in the BPS tool IDA ICE. File exchange using the standardized PLCopen XML then allows the digital transfer of controls from IDA ICE to programmable logic controllers (PLCs).

1.3 Research gap

Based on the descriptions in Section 1.2, the main problems in the development, communication, and implementation of control logic for HVAC systems are:

1. The development of controls for HVAC systems considering building physics, dynamic loads, user interaction, and climate in the design phase is not done in an integrated manner.
2. The level of detail of the control development in the design phase (before the contract is awarded) is usually relatively low. In contrast, the level of knowledge about HVAC components is often higher.
3. Textual and graphical formats are used to communicate control strategies from a design office to an executing company.
4. The actual programming is only carried out after the contract has been awarded.

The overarching goal should be that HVAC systems are operated according to controls that have been developed, tested and optimized on building and system models during the design phase. Digital exchange formats should be used in order to address the performance gap associated with an unclear, ambiguous, analog description of control logic. Such formats are becoming increasingly important in the building sector. However, toolchains are still under development and need to be compared and, above all, validated under real conditions.

The increased use of BPS for control development in the design phase is promising, but a higher modeling effort is expected. The continued use of these models in operational applications such as FDD or MPC should be pursued in order to increase their value over the life cycle. The use of models both for the development and optimization of a physical object and for operational applications is a key aspect of the Digital Twin approach (Grieves and Vickers 2017). Digital Twins are a key technology in Industry 4.0, but consistent applications in the building sector are still rare. This is especially true for HVAC systems and their controls.

1.4 Objectives and research questions

To address the challenges described in Section 1.2 and the research gaps described in Section 1.3, the following objectives and research questions (RQ) are defined:

Objective 1 *Description, analysis and practical implementation of options to deploy controls developed in BPS environments for the operation of HVAC systems.* This research objective addresses the link between BPS environments and BAS as a key technical aspect. This involves identifying different options, describing them in detail, and validating them in the context of a practical demonstration. The overall goal is to operate HVAC systems according to exactly the same control logic that has been previously developed and tested in BPS environments. The control development must be performance-based along testable metrics. This requires that the simulation models correctly reflect the behavior of the real building, including the HVAC systems. In terms of implementation, the main goal is an accurate, efficient, and automated process. As a result, the engineering effort in the implementation phase would be drastically reduced compared to today's practice. This leads to the following research questions:

- **RQ 1.1** What options exist to implement control logic developed in BPS environments in buildings for the operation of HVAC systems and how do they differ methodologically?
- **RQ 1.2** Which discrepancies between simulated and measured performance on building, system and control level can be observed when control logic developed in BPS environments is used for the operation of HVAC systems against the background of model simplifications?
- **RQ 1.3** How does the development of controls in BPS tools during a design phase change currently established processes and the effort at the interface of design and execution services?

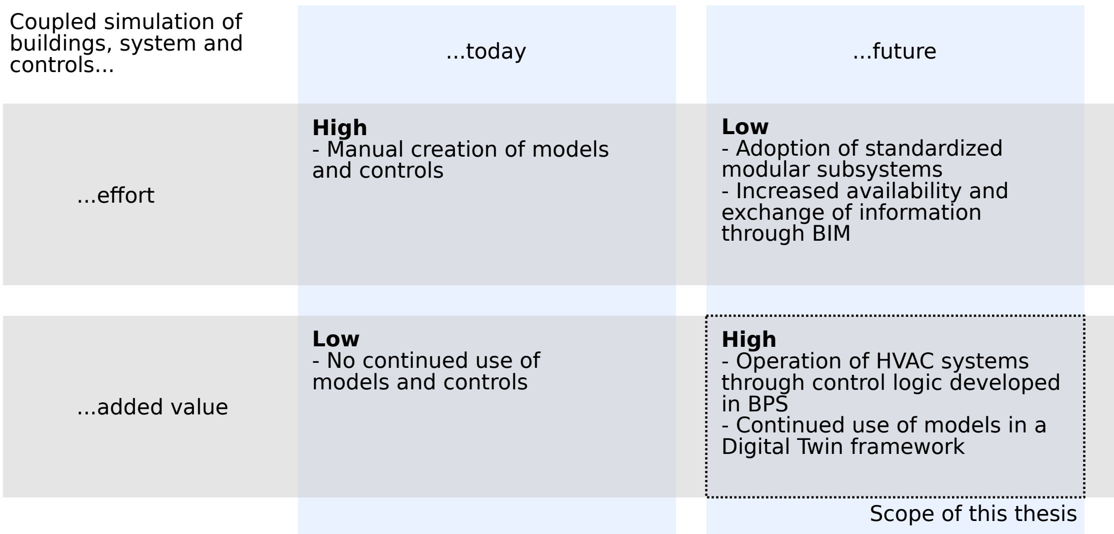


Figure 1.5: Effort and added value for building energy modeling including controls in BPS today and in the future

Objective 2 *Embedding of the workflow aimed for in Objective 1 into a Digital Twin framework.* Objective 1 aims to exchange control logic between BPS tools and BAS. Ideally, the control logic applied to simulation models will be identical to the control logic implemented in BAS. This bridges the gap between models and building operations, which directly promotes the use of Digital Twins in operations. This leads to the following research questions:

RQ 2.1 What is the role of controls in Digital Twins in relation to HVAC systems and buildings?

RQ 2.2 How do the options described in RQ 1.1 support the application of Digital Twins?

1.5 Contribution and outline

The work contributes to the further development of methods that enable energy-efficient operation of HVAC systems. In general, the focus is on increasing quality in the planning phase. The increase in quality is achieved through a higher level of detail in the development and definition of controls for HVAC systems in BPS environments. Today, the simulation of HVAC systems and their controls is associated with high modeling effort (top left in Figure 1.5). On the other hand, the added value of such a simulation in practice is low because of the lack of interfaces for operational use cases (bottom left in Figure 1.5). It is expected that the wider use of the Building Information Modeling

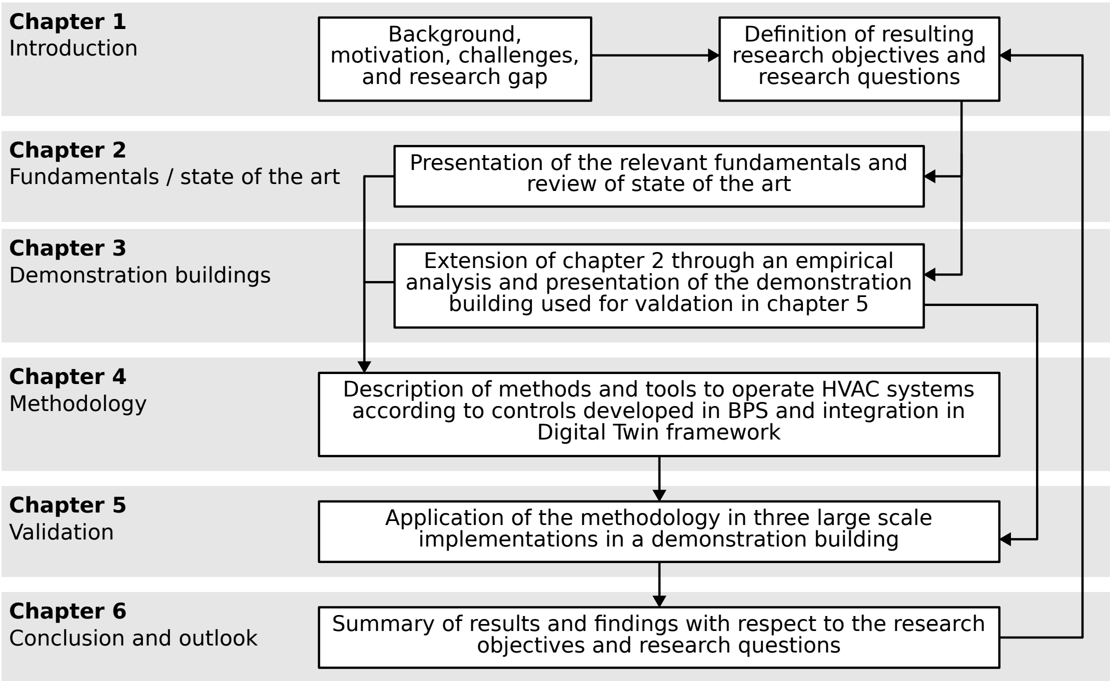


Figure 1.6: Overview over the structure of this thesis

(BIM) method will provide more information and reduce the modeling effort (top right in Figure 1.5). In addition, standardization of HVAC subsystems and controls may further simplify modeling. While the reduction of modeling effort is not the focus of this thesis, the goal is to identify and demonstrate ways to increase the added value of BPS (bottom right in Figure 1.5). This will be achieved through the use of tested control logic in BAS (Objective 1) and the continued use of building and system models for model-based use cases in operations (Objective 2).

This thesis is organized as follows (see Figure 1.6):

In chapter 2, an overview of the fundamentals and the state of the art is given. This includes performance requirements at the building level, building automation systems, controls in the context of BPS environments, and Digital Twins. These aspects are enriched in chapter 3 by an analysis of HVAC system controls in demonstration buildings.

A three-step methodology is proposed in chapter 4 to address the identified problems and research gaps. First, the scope and requirements for control development and building energy modeling in BPS environments are defined. Then, different options for linking the BPS tool and BAS are described, analyzed, and compared. Finally, these options are discussed in the context of Digital Twins.

As part of the work for this thesis, three large-scale implementations of the proposed methodology have been carried out under real conditions in an industrial building. The implementation and the results are presented and discussed in chapter 5. The results of this thesis are summarized in chapter 6 in relation to the objectives and research questions.

2 Fundamentals and state of the art

2.1 Performance requirements on building level

In the EU, the Energy Performance of Buildings Directive (EPBD) (European Parliament and Council of the European Union 2018) sets the legal framework for building performance requirements. The following comparison of the national implementations in Germany and Sweden provides a brief overview of the relationship between building performance requirements and HVAC system controls.

In Germany, the building energy law GEG (Gebäudeenergiegesetz) (Deutscher Bundestag 2020) requires to verify a lower primary energy demand of the planned building compared to the primary energy demand of a reference building with the same geometry and standardized technical equipment in the design phase (Horward and Rosenberger 2020) (left column in Table 2.1). The energy demand for the planned and the reference building is calculated according to DIN V 18599 (DIN 2018c) based on a monthly energy balance with defined boundary conditions, e.g. for climate and use profiles. HVAC systems are reflected in a standardized way, mostly by simplified energy balances. Controls for AHUs for example are considered in DIN V 18599-7:2018 by performance factors in tables (DIN 2018a). Intelligent operational management strategies or project-specific controls cannot be taken into account in this way. A performance check that the target demand values match the measured consumption is not required. Dynamic simulations (Section 2.3) are commonly used in Germany for project-specific consulting, but are not eligible to be used to verify normative requirements.

Table 2.1: Comparison of building performance verification in Germany and Sweden

	Germany	**Sweden**
Verification approach	$Q_{p,project} < Q_{p,reference\ building}$	$Q_{p,project} < Q_{p,threshold}$
Tool	Monthly energy balance	Dynamic simulation with time steps < 1 hour
Modeling of controls	Standardized performance factors in tables	Custom controls in BPS possible

In contrast, according to the Swedish regulation, the primary energy demand must remain below fixed thresholds (Hjorth et al. 2021) (right column in Table 2.1). The verification can be done either by calculations or by measurements in the finished building. If calculations are used, dynamic simulations with time steps shorter than one hour are required for non-residential buildings (Swedish National Board of Housing, Building and Planning Boverket 2018). In such simulations, operational management and control of HVAC systems can be modeled on a project-specific basis. Intelligent controls can then help ensure that the required primary energy limit is not exceeded. This illustrates that a verification architecture with fixed performance thresholds, and the requirement to use dynamic simulations, implicitly encourages the design of project-specific controls in BPS.

2.2 Control logic and building automation systems

This section provides an overview of the control of HVAC systems with a focus on the methods used in control design and implementation in BAS.

2.2.1 Development of controls in the design phase and design documentation

The initial control design up to the awarding is carried out by planning offices (see Section 1.2.2). The tasks to be performed are defined in several standards and guidelines, such as DIN EN ISO 16484-1:2011 (DIN 2011) and VDI 3814-2.2:2019 (VDI 2019c). The following documents are used to define and describe controls:

- *Automation schemes*

  The format of automation schemes, function lists and functional diagrams is defined in DIN EN ISO 16484-3:2005 (DIN 2005) and VDI 3814-4.3:2022 (VDI 2022). According to these standards, automation schemes should contain a scheme with the physical system and sensors (center in Figure 2.1), a functional diagram (bottom in Figure 2.1), and characteristic curves (top in Figure 2.1). The main information about the planned control logic is contained in the functional diagram and the characteristic curves. Function blocks, their relationships, and their connection to inputs and outputs are shown in the functional diagrams. The behavior of the function blocks is described in the characteristic curves.

- *Function lists*

  In the function list, control functions are assigned to data points. VDI 3814-4.3:2022 separates input/output functions, application functions, and control/display functions. These functions are further specified in VDI 3814-3.1:2013 (VDI 2019a).

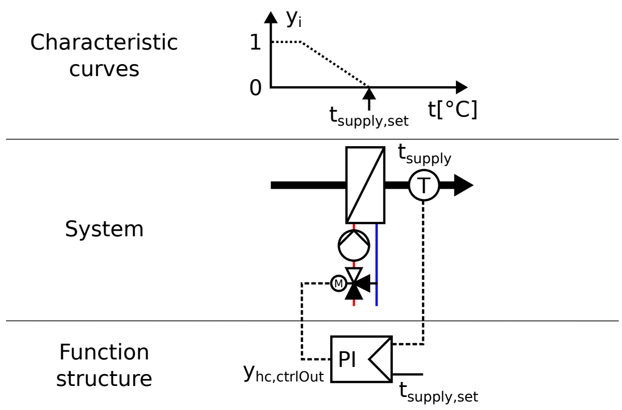


Figure 2.1: Exemplary automation schemes for the control of a heating coil. Automation scheme according to DIN EN ISO 16484-3:2005 / VDI 3814-4.3:2022.

- *Functional descriptions*

  The functional description is a textual document that should contain general descriptions of the system and the target values. VDI 3814-6:2003 (VDI 2008)[1] stated as early as 2003 that "the textual description frequently used up to now quickly reaches its limits, even for simple tasks". However, the textual functional description is still common practice for describing controls, and sometimes the only document available (see chapter 3).

- *Graphical representations*

  While automation schemes and functional diagrams are an abstract representation of controls, it is generally useful to have a more general, high-level graphical representation, especially for communicating with owners and operators. For this purpose, VDI 3814-6:2003 defines a state chart as a graphical representation of control tasks. The disadvantages of this type of state chart are the mandatory manual creation and the potentially high complexity as it does not offer features such as sub-states or parallel processes. There are few examples for applications in research projects (Lechner et al. 2018) and only some building owners like the German Federal Armed Forces (Bundeswehr 2019) require these state charts as design output.

In theory, controls should be comprehensively and unambiguously described by these documents. In practice, however, the quality of the design output is often poor, as reported by Fisch et al. (2017).

1. withdrawn

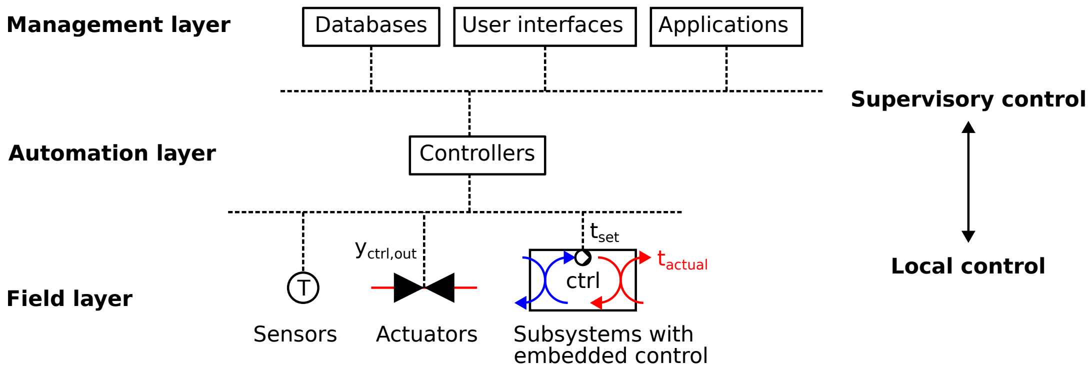


Figure 2.2: Exemplary location of control inputs and outputs in a hierarchical automation architecture.

From a methodological point of view, the cited standards for control development in the design phase define in great detail the required tasks and the format of the output. However, they do not contain or describe methods how to test the performance of controllers in interaction with HVAC systems.

Regarding the control functions, classical rule-based controls, mostly based on PID controllers, dominate in today's building practice (Royapoor, Antony, and Roskilly 2018). Advanced controls such as MPC, Fuzzy Logic Control (FLC), Neuronal Network Control (NNC) or Reinforced Learning (RL) are intensively studied in a scientific context, but are rarely used in commercial construction practice (Schild et al. 2019).

2.2.2 Building automation systems

According to VDI 3814-1:2019 (VDI 2019b), BAS are defined as "all products and services for the goal-oriented operation of building services" (VDI 2019b). This definition emphasizes that BAS combines a physical dimension (products) and a non-physical dimension (services). BAS are typically structured and described in different layers: According to EN ISO 16484-2:2004 (DIN 2004), classical hierarchical BAS consist of a field layer, an automation layer, and a management layer (Figure 2.2). In VDI 3814-1:2019 on the other hand, BAS are structured in a spatial dimension in portfolio, property, building, area, room, and segment (VDI 2019b).

Regarding the control functions implemented on BAS, a distinction can be made between supervisory control and local loop control (Salsbury 2005; Wang and Ma 2008; Roth et al. 2022). Supervisory control coordinates different HVAC systems at the building level. It processes inputs from low level measurements, user inputs, and high level signals from subordinate grids and sends signals to subsystems. On the other hand, local controls, also referred to as local loop controls or subsystem controls, determine the behavior of individual actuators, such as valves, or subsystems, such as AHUs. While supervisory control logic is mainly implemented on servers and controllers at the management and automation layer, subsystem control is implemented on controllers at the automation layer or embedded in subsystems at the field layer (Domingues et al. 2016). In the field of complex heat and cold generators, such as compression chillers, absorption chillers, heat pumps, or CHP, such embedded controls are already standard. It is not always possible to make a clear distinction between supervisory and local controls, or to assign them to specific layers or devices.

Controllers at the automation level are microcomputers designed for use in an industrial or building environment. These controllers are often proprietary systems that use vendor-specific programming languages and environments. IEC 61131 provides standards for interfaces and programming of so-called PLCs. The basic and common principle is the execution of control programs in discrete time with cycles in the range of seconds and milliseconds (see Section 2.3.2 for a comparison with computational methods in BPS).

Communication standards, such as Modbus or BACnet, enable communication between these layers and different devices (Domingues et al. 2016). As a key technology for Industry 4.0 applications, the OPC UA standard has also become increasingly popular in building automation. OPC UA enables communication from the lowest field layer up to cloud-based supervisory services (OPC Foundation 2020; Drgoňa et al. 2020). While OPC UA is already widely used for communication between management and automation layers, fieldbus protocols such as Modbus or BACnet still dominate on the field layer (Veichtlbauer, Ortmayer, and Heistracher 2017).

In recent years, technologies have emerged that soften the hierarchical structure of BAS. For example, IoT (Internet of Things) devices with embedded controls communicate directly with higher-level, often cloud-based applications (Stluka, Mařík, and Endel 2014; Brümmendorf, Ziegeldorf, and Fütterer 2019; Storek et al. 2019). Despite the emergence of these new technologies, classical hierarchical BAS are still the standard in practice (see chapter 3).

2.2.3 Implementation of controls on building automation systems

After the contract has been awarded, executing companies must adapt the planning documents received (see Section 2.2.1) and implement the control logic on BAS (see Section 1.2.2). The programming of PLCs is standardized in IEC 61131-3 (DIN 2014). The standard defines the three graphical languages Ladder Diagram, Function Block Diagram, and Sequential Flow Chart and the two textual languages Structured Text and Instruction List. In practice, a mixture of these languages is common.

The translation of the graphical and textual representations is done by a manual reproduction in the BAS specific software environment. Three steps are required for implementation and execution on a controller: first, an editor to replicate the communicated control logic in control code. Second, a compiler to translate this code into machine-readable code. Third, a Runtime Environment (RTE) for the execution on the controller. These software components are often combined in a so-called IDE (Integrated Development Environment), which can be vendor-specific or a cross-platform solution such as Codesys (CODESYS; Hanssen 2015). An attempt to provide cross-platform open source solutions is the PLCopen project. For replication and implementation, executing companies have several options (Hydeman, Taylor, and Eubanks 2015):

1. *Project-specific individual programming*

   In theory, a control program, including control functions such as PID or hysteresis control, can be written entirely from scratch using basic mathematical and logical expressions in the BAS-specific programming language. However, recurring control functions are typically provided in libraries to facilitate programming.

   Basic algebraic, logical, data type conversion, and control functions such as a simple PID controller are defined in IEC 61131-3. However, since this library is very limited, controller manufacturers provide extended function libraries along with their IDEs. The proprietary nature of these libraries and thereby the inaccessibility of the underlying controller code is a common drawback. An attempt to provide vendor independent open source libraries for PLCs are the OSCAT "Basic" and "Building" libraries (Mühlbauer 2015a, 2015b).

   Function block libraries allow an individual, project-specific control implementation according to the design documentation (Section 2.2.1). However, the engineering effort is high, and testing and debugging are required during commissioning to ensure a properly functioning control program.

2. *Configurable control macros for common HVAC systems*

   Configurable macros can be used instead of project-specific custom programming to simplify implementation, increase productivity, and reduce errors. Such macros are usually provided by automation manufacturers in libraries as part of their IDEs. The manufacturer Wago, for instance, offers 9 macros for AHUs, 3 macros for heating circuits, 4 macros for domestic hot water, 1 macro for district heating, and 2 macros for boilers (WAGO 2019). A drawback of these vendor-specific macros is that the underlying code is typically proprietary and not visible. Engineers and operators often have to rely on textual and graphical descriptions in manuals to understand the underlying control logic.

An approach to standardizing control logic for common HVAC systems has been taken in ASHRAE Guideline 36-2021 (Hydeman, Taylor, and Eubanks 2015; ASHRAE 2021). ASHRAE Guideline 36-2021 provides control sequences for AHUs and supply systems commonly used in the US. The estimated energy savings of ASHRAE Guideline 36-2021 control sequences compared to current building practice for an exemplary commercial building are around 31 % (Zhang et al. 2022). ASHRAE Guideline 36-2021 control sequences can be implemented in simulation environments as described by Wetter, Grahovac, and Hu (2018) for variable air volume (VAV) systems using the CDL in Modelica. This enables testing of such standardized control sequences coupled with building models in the design phase.

In general, the successful application of control macros depends on the individuality of the respective HVAC system. If an HVAC engineer has selected a system that matches the configuration for which the control macro was designed and that has the appropriate inputs and outputs, such control macros can greatly simplify and streamline the implementation process. Unfortunately, false incentives encourage HVAC engineers to design project-specific, complex HVAC systems (see note for the regulation in Germany in Section 1.2.1) that are often incompatible with these macros. If standards such as ASHRAE Guideline 36-2021 were generally accepted throughout the building industry, which is not the case in Germany at present, HVAC system manufacturers might be forced to offer products whose configuration matches the control macros.

3. *Proprietary controls embedded in HVAC subsystems*

   Proprietary controls developed by HVAC system manufacturers can be used to completely avoid individual, project-specific programming. These controls have been developed and tested along with the product design, often using domain-specific simulations (Wetter 2009). They typically receive an activation signal and setpoints from a supervisory controller. The embedded controller then regulates the internal components to achieve the setpoints. Proprietary controls are typically not accessible to operators or automation engineers because they are part of the manufacturer's intellectual property. While for certain subsystems, such as heat pumps, CHP units, or chillers, proprietary controls are already standard, project-specific custom programming and embedded proprietary controls are common for AHUs. It should be noted that proprietary controls are typically limited to subsystems. Due to the heterogeneity of installed HVAC systems in commercial buildings they are typically not available at the building level.

The commissioning phase is a critical step to ensure that the control program works as intended and to eliminate software failures and errors before the real building is operated (Visier 2004). A common approach is to connect inputs and outputs of BAS to building and system models to emulate the behavior of the physical counterparts (Mansson and McIntyre Don 1997; Clarke et al. 2002). Software-In-the-Loop (SIL) methods allow testing of the control program while Hardware-In-the-Loop (HIL) methods also include testing of the controller hardware. SIL or HIL implementations can differ significantly in the level of detail of the building model, also called the “emulator”. For example, Togashi and Miyata (2019) use a set of equations to mimic the behavior of the building. In contrast, Sahlin, Skogqvist, and Högberg (2018) present a toolchain for using validated building models in the BPS tool IDA ICE (Section 2.3). Although SIL and HIL testing is generally a useful and powerful method, it is not widely used in commissioning practice. This is mainly due to the high effort required to create building energy models in a phase with typically tight time schedules and limited financial resources. Building and system models from the design phase are often missing due to the low penetration of BPS in the design phase, or cannot be used for HIL and SIL due to the lack of standardized interfaces.

2.2.4 Interoperability

Interoperability is an important aspect of enabling the exchange of control logic between different environments and applications[2]. Standardized interchange formats and interfaces are required to enable interoperability.

An XML format has been developed by the PLCopen consortium and standardized in IEC 61131-10 to exchange control logic between PLCs following the IEC 61131-3 standard. The PLCopen XML contains all information about used function blocks and their relations, user defined functions, variables, and parameters. The structure of the XML file is described in detail by Da Silva (2018), Schaper (2011), and Marcos et al. (2009). A general requirement for a successful application is that the respective IDEs support the import and export of the PLCopen XML (Simros, Theurich, and Martin Wollschlaeger 2012).

Despite the existence of such digital exchange formats, graphical and textual formats, as described in Section 2.2.1, are still the standard in building practice for exchanging control logic between planning offices and executing companies. This is due to controls

2. As described by Drath et al. (2023), two dimensions of interoperability can be distinguished: First, design and development interoperability refers to the exchange of information, in this case control sequences, during the design phase and from the design phase to the implementation phase. Second, operational interoperability refers to the ability of components to interact and communicate with each other. Within this section, the first dimension of interoperability is in the focus.

not yet being specified down to code level by planning offices (see Section 1.2.2). During the implementation of controls (Section 2.2.3), there are practically no use cases, because automation engineers can handle all tasks within the controller-specific IDE.

Semantic modeling has gained importance in recent years to enable knowledge exchange between different applications in engineering and operations (Schneider 2019; Ihlenburg et al. 2020; Ihlenburg, Benndorf, and Réhault 2022; Roth et al. 2022). These approaches are often linked to exchange formats such as the PLCopen XML. By adding semantic information about e.g. inputs and outputs, they support the exchange between different applications.

2.3 Simulation of buildings, HVAC systems and controls

Mathematical models allow to simulate the behavior of buildings under dynamic influences (Beausoleil-Morrison 2021). The use and applications of such BPS to support design, commissioning, and operation with respect to thermal comfort, energy performance, or daylighting are extensively documented (Hensen and Lamberts 2019). A variety of software tools are available today.

In practice, building simulations are widely used to support design decisions, for example, by comparing performance indicators for different insulation or glazing options. Due to the high effort and limited resources in design, simulations of HVAC systems including their controls (Trčka and Hensen 2010) are less frequently used in practice. However, especially the integrated simulation including HVAC systems and controls has great potential to support the model-based development of energy-efficient buildings (Wetter 2009; Treado, Delgoshaei, and Windham 2011; Kim et al. 2013; Kontes et al. 2018). The importance of integrated simulation of buildings, HVAC systems and controls is elaborated in demonstration cases by Kramer, van Schijndel, and Schellen (2017) and Horn et al. (2019).

2.3.1 Simulation technology

With respect to the ability to model and simulate HVAC systems including their controls, traditional building simulation, and equation-based simulation programs can be distinguished (Wetter 2009).

Traditional building simulation

Traditional building simulation programs have been developed to predict the thermal comfort of rooms taking into account the thermal mass, ventilation, internal loads, and climate. Most of the tools used for this purpose consider HVAC systems and controls in a very simplified way (Sahlin, Bring, and Eriksson 2009; Wetter 2009; Roth et al. 2022). For example, in the case of heating, they calculate the added heat required to maintain a given temperature threshold, whereas in a real control loop, the manipulated variable determines the added heat by changing a valve position. In such programs, expressions for physical behavior, data management, and numerical solution methods are mixed in algorithmic, imperative step-by-step instructions, making it difficult to add new or individual components, such as custom controls (Wetter 2009).

These aspects limit the ability to realistically simulate controls in traditional building simulation programs. As a result, it is difficult to use traditional building simulation as a basis for programming real controls. For a control description, a simulation engineer would have to functionally describe the underlying controller behavior as a basis for programming on a real controller by an automation engineer (Roth et al. 2022).

Equation-based simulation environments

Based on advances in computer science, equation-based simulation programs have evolved after the traditional building simulation programs described above (Sahlin 2000; Sahlin, Eriksson, and Vuolle 2003). In these tools, mathematical modeling and solver algorithms are separated. Declarative expressions in algebraic equations, discrete equations, and differential equations are used to model physical processes. The resulting hybrid Differential Algebraic Equation (DAE) systems are solved with variable time steps based on a defined tolerance. The encapsulation of the models with standardized interfaces allows an object-oriented modeling and a flexible integration of new and individual components (Wetter 2009). Due to these computational methods, equation-based BPS are considered more suitable for the simulation of building energy systems and controls than traditional building simulation programs (Wetter 2009; Sahlin, Bring, and Eriksson 2009).

An example of an equation-based, object-oriented modeling language is Modelica (Mattson and Elmqvist 1997). Examples of integrated building, system, and control simulation in Modelica are presented by Jorissen, Wetter, and Helsen (2015), Zuo et al. (2016), and Jorissen, Boydens, and Helsen (2019). However, the simulation time is still too long for practical applications (Sahlin and Lebedev 2018) and the modeling effort for whole building models remains high (Maier et al. 2023). The only equation-based simulation software available as a commercial application that allows coupled simulation of buildings, systems, and controls that has a 3-D modeler and that allows whole-building simulations in a reasonable time is currently IDA ICE. Figure 2.3 gives an overview over the assembly of models for thermal zones, supply systems, and AHUs including controls. IDA ICE uses precompiled components written in Neutral Model Format (NMF) or the Modelica language (Sahlin and Sowell 1989).

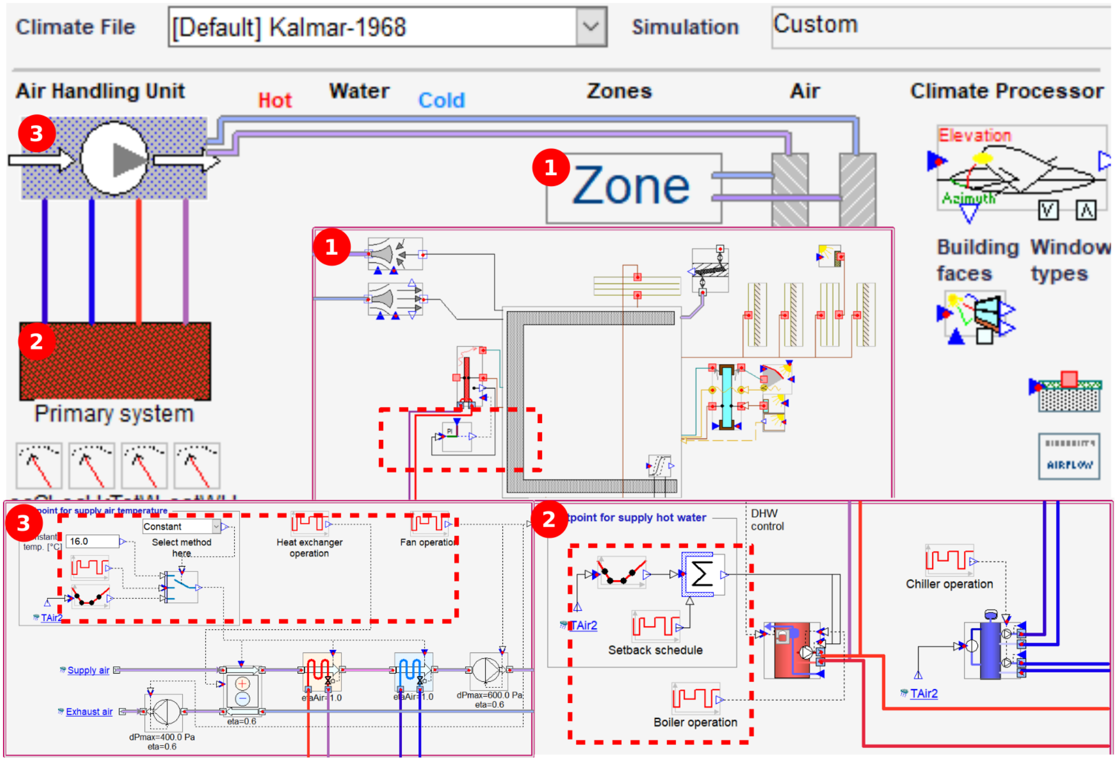


Figure 2.3: Overview over simulation models of thermal zones (1), supply systems (2), AHUs (3), and respective controls (dashed boxes) in IDA ICE

Co-simulation

Developing simulation tools that cover all aspects of building energy modeling is very challenging. One way to exploit the strengths of different tools is the so-called co-simulation (Trčka, Hensen, and Wetter 2009; Wetter 2011; Nicolai and Paepcke 2017; Schweiger et al. 2019). The general approach is that different simulation tools run parallel and exchange signals. For this purpose, the Building Controls Virtual Test Bed (BCVTB) has been developed (Wetter 2011). It allows co-simulation of control-oriented software with building simulation tools such as Energy Plus. In recent years, software coupling using the Functional Mock-up Interface (FMI) standard has gained importance (Fathollahzadeh and Tabares-Velasco 2020; Huang et al. 2023). Based on the FMI standard, the aim of the Spawn project is to combine the advantages of building simulation in Energy Plus and HVAC and control simulation in Modelica (Wetter 2020). Co-simulation can also be used to couple simulation environments and BAS. The co-simulation of building simulation models with BAS is related to the SIL and HIL approaches presented in Section 2.2.3. Béguery et al. (2021) describes the challenge of testing a building management system (BMS) in virtual commissioning on a building simulation model from the perspective of a large automation manufacturer. The limitation to run the BMS in real time is reported to be the main obstacle to an efficient application.

2.3.2 Computational methods in simulation and real controllers

When comparing controls simulated in equation-based simulation environments with the execution of controls on real controllers, fundamental differences in computational methods must be considered (Table 2.2). These differences are briefly outlined below. Detailed descriptions can be found in Sahlin, Bring, and Eriksson (2009) and Wetter et al. (2023).

In equation-based simulation environments, control functions such as PI controllers are typically used in continuous-time implementations together with discrete-time equations, for example for control mode switches. The time step for the resulting DAE system is determined by the solver based on a given tolerance. Contrary, control functions on real controllers are expressed using mainly imperative programming languages and are executed in discrete time with and fixed time steps, typically on the order of milliseconds (Section 2.2.2). Leva et al. (2008) exemplify for a PID controller that the control signal differs between the continuous-time and discrete-time implementations.

Real, discrete-time controllers can be integrated into simulation environments but this results in much longer simulation times because the number of steps increases (see Figure 2.4). As a solution, Leva et al. (2008) suggest the use of controllers in continuous form to check the control strategy and the use of discrete form for more detailed analysis.

Table 2.2: Comparison of control functions in equation-based BPS and real controllers

	Equation-based simulation	**Execution on real controller**
Time representation	Continuous-time (e.g. PI controller) or discrete time (e.g. hysteresis)	Discrete-time
Programming paradigm	Mainly declarative	Mainly imperative
Time steps	Variable steps	Fixed steps
Typical step size	Seconds, minutes, hours	Milliseconds, seconds

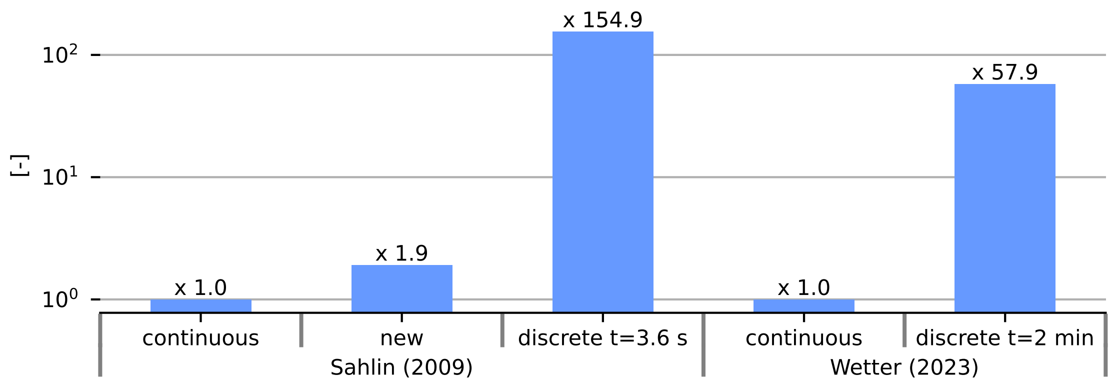
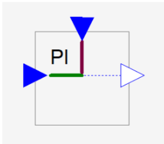


Figure 2.4: Comparison of simulation time for exemplary continuous-time and discrete-time control functions in IDA ICE (Sahlin, Bring, and Eriksson 2009) and Modelica (Wetter et al. 2023)

For this purpose, control libraries must contain both control blocks in continuous and discrete form (Bonvini and Leva 2012). Sahlin, Bring, and Eriksson (2009) presented a multi-rate method to efficiently simulate discrete-time controllers and continuous-time models in IDA ICE (see “new” Figure 2.4).

2.3.3 Features of control functions

Another difference between simulation tools and real controllers are the features provided by the control functions. While control functions in BPS environments are often simplified textbook implementations, control functions on real controllers typically offer more features and corresponding inputs and outputs. As an example, the left column in Figure 2.5 shows the graphical function block of a PI controller from the IDA ICE library.



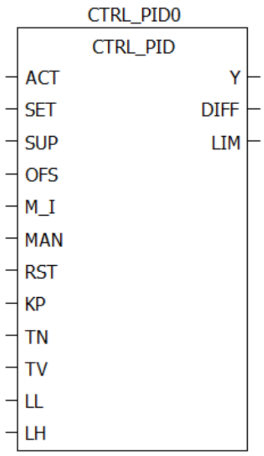


Figure 2.5: Left: PI controller from the IDA ICE library. Right: PID controller from the open-source OSCAT library

The function block takes a measured signal and a setpoint signal as variables. The proportional and integral parameters can be defined, but are fixed. The right column in Figure 2.5 shows the PID controller from the open source OSCAT library. It offers additional features like noise suppression, reset, offset, or manual mode. In addition, all inputs can be changed online, which may be necessary for gain scheduling. Proprietary controllers from vendor libraries (Section 2.2.3) may offer even more features. These differences must be taken into account when comparing measured and simulated control performance. To fill this gap, Bonvini and Leva (2012) have developed a Modelica control library that reflects these peculiarities of commercial control functions.

2.3.4 Transfer of control logic from BPS environments to building controllers

The exchange of control logic between simulation and automation environments is a major challenge due to the different programming and computation methods described above and the heterogeneity in the BPS and BAS domains. Two approaches have been presented in recent years. Sahlin, Skogqvist, and Högberg (2018) developed a toolchain to simulate control code according to IEC 61131-3 together with building and system models in the BPS environment IDA ICE. The core technology in this approach is the previously mentioned multi-rate simulation method (see Section 2.3.2), which efficiently simulates time-discrete control models with small fixed time steps together with longer and variable time steps for building models (Sahlin, Bring, and Eriksson 2009). Alternatively, in the frame of the OBC project, a CDL has been developed as a new exchange format. It allows the exchange of control logic between Modelica-based simulation environments and different proprietary building controllers (Wetter et al. 2018; Wetter et al. 2022). Both

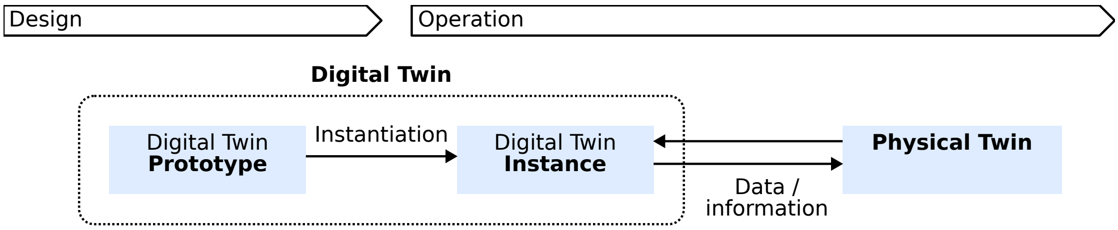


Figure 2.6: Digital Twin concept (Grieves 2014; Grieves and Vickers 2017)

approaches are key developments to enable interoperability between BPS and BAS and are discussed and compared as part of the methodological part of this thesis (chapter 4).

2.4 Digital Twins

2.4.1 Concept

The origins of Digital Twins are generally associated in the literature with the space industry (Tuegel et al. 2011; Shafto et al. 2012) and Product Lifecycle Management (PLM) (Grieves 2014). Depending on the point of view for specific applications, different definitions of what a Digital Twin is can be found in the literature (Brilakis et al. 2019; Souza, Brilakis et al. 2019). Regardless of specific definitions, three core elements can be identified (Grieves 2014; VanDerHorn and Mahadevan 2021; Boje et al. 2020):

1. A physical object
2. A virtual model representation of the physical object, the Digital Twin
3. A bidirectional connection through which data and information between the physical object and the Digital Twin is exchanged

The added value of Digital Twins is generated through different use cases over the life cycle. Grieves and Vickers (2017) introduced the Digital Twin Prototype and the Digital Twin Instance to describe these use cases as follows (see Figure 2.6):

1. *Digital Twin Prototype*

   The Digital Twin Prototype is created in the design phase and usually before the Physical Twin is present. The purpose of the Digital Twin Prototype is to predict the future behavior of the physical object to support conceptualization and decision making in the design phase. The Digital Twin Prototype can be optimized to meet defined requirements.

2. *Digital Twin Instance*

   The Digital Twin Instance represents the Digital Twin of an existing object in operation. The Digital Twin Instance is linked to the physical object for the bidirectional exchange of data and information. This link allows for continuous comparison between Digital Twin Instance and physical object.

The Digital Twin concept has been extensively reviewed in the literature (Jones et al. 2020; Rasheed, San, and Kvamsdal 2020; Zhou, Zhang, and Gu 2022; Semeraro et al. 2021; Singh et al. 2021; Kritzinger et al. 2018). Applications are documented in various industrial sectors (Tao et al. 2019; Cañas et al. 2021), including process industry (Perno, Hvam, and Haug 2022), manufacturing (Roy et al. 2020), or smart farming (Verdouw et al. 2021). The application of Digital Twins from product development to operation was described by Viola and Chen (2020) for industrial applications, for example.

In the context of buildings, Digital Twins are an approach to stimulate the digitalization of design, construction, and operation. Building-specific aspects are examined in several studies (Sacks et al. 2020; Boje et al. 2020; Khajavi et al. 2019; Davila Delgado and Oyedele 2021). The conditions for the use of Digital Twins in buildings are basically given, as modeling and simulation methods have been developed (Section 2.3) and measured data are increasingly available. Several applications in buildings are documented in the literature, which are structured in Section 2.4.4. However, applications are often still in the context of research projects and comprehensive approaches are still being developed.

2.4.2 Model types

Different types of models are used for Digital Twins in design and operation. A general distinction can be made between models that reflect the dynamic behavior of buildings and those that do not, as follows (Ruepp et al. 2022):

- *Information models*

  Information models describe the structure of systems, for example, by characterizing the relationships, semantics, and functions of elements within systems. An example for such models are BIM models. They can be enriched with measured data and used for visualization and information, for example (Spudys et al. 2023).

- *Behavioral models*

  Behavioral models reflect the behavior of a building or system, either by incorporating equations of the underlying physics (white-box models) or by data-driven approaches (black-box model). Behavioral models are required for specific use cases such as MPC. While white-box models must be created for new buildings, data-driven black-box models can be created for existing buildings if sufficient measured data is available. A critical review by Korenhof, Blok, and Kloppenburg (2021) points out that the model representation of the Digital Twin still contains simplifications.

2.4.3 Connection between Digital Twin and Physical Twin

The bidirectional exchange of data and information between the Physical Twin and the Digital Twin is necessary for two reasons. First, the Digital Twin needs to be fed with operational data from the Physical Twin to make a reasonable comparison. Second, some sort of feedback from the Digital Twin to the Physical Twin is required to apply the information gained from the Digital Twin. Several distinctions can be made regarding the nature and configuration of this connection:

- *Manual and automatic exchange*

  Kritzinger et al. (2018) distinguished between manual and automatic exchange. If the exchange is manual in both directions, the virtual representation is a so-called "digital model". This case applies in building related use cases, when a certain period of measured values on Digital Twins is manually applied for performance analysis of fault detection. In the case of a "Digital Shadow", data is automatically imported into the digital object, but manually applied in the other direction. Only if the data exchange is automatic in both directions, there is a "digital twin".

- *Open and closed loop*

  According to Ruepp et al. (2022), open and closed loop use cases can be distinguished. In the open loop case, the model is connected to live measurements, but the model does not send information back to the building. This is similar to the "digital shadow" definition of Kritzinger et al. (2018). In the closed loop case, signals from the model are used, for example, for control purposes, which corresponds to the "Digital Twin" definition of Kritzinger et al. (2018).

- *Online and offline exchange*

  While for certain use cases, such as MPC, an online connection is mandatory, "what-if" analysis can be performed offline using historical data (Ruepp et al. 2022).

For some building-related use cases, as discussed in the next section, manual import of data into the Digital Twin Instance is common and sufficient. For the automatic, online, and bidirectional exchange of data and information, OPC UA is expected to become a widely accepted standard (OPC Foundation 2023).

2.4.4 Building-related use cases

With respect to building-related use cases, a distinction can be made between a design perspective and an operational perspective as follows:

1. *Design*

   In the development perspective, models are used without being linked to live data from the building. According to the definition in Section 2.4.1, the Digital Twin is called Digital Twin Prototype in this case. The following use cases can be distinguished:

   - *Development and decision-making*

     Digital Twin Prototypes of buildings can be used to support design development and decision-making by communicating performance indicators and comparing variants in “what-if” scenarios. Several applications at component level (Tariq et al. 2022), building level (Nytsch-Geusen et al. 2019; Marchione and Ruperto 2022), and district level (Simonsson et al. 2021) are documented in the literature.

   - *Education and learning*

     Digital Twin Prototypes can also support education, especially e-learning, as described by Johra et al. (2021).

   If models are created for these purposes only and with no further use in operations, the question remains as to why this is called Digital Twin (Wright and Davidson 2020). As pointed out by VanDerHorn and Mahadevan (2021) and Wilde (2023), only the continued use of models within the Digital Twin Prototype in a Digital Twin Instance with a bidirectional link to the Physical Twin distinguishes Digital Twins from existing modeling and simulation methods.

2. *Operation*

   In operation, the Digital Twin is connected to the Physical Twin as a Digital Twin Instance. The applications reported in the literature can be grouped as follows:

   - *Performance analysis*

     The Digital Twin Instance can be used for reporting (Spudys et al. 2023) or as a reference for monitoring and Performance Gap Analysiss (Nytsch-Geusen et al. 2019; Ward et al. 2023). Manual or automatic data import is possible for this purpose.

- *Decision-making*

  The Digital Twin Instance enables "what-if" analyses to inform stakeholders and support operational decision-making (Bjørnskov and Jradi 2023). A prerequisite for reliable decision-making is that the control logic is correctly represented in the Digital Twin.

- *Fault Detection and Diagnosis*

  A more specific use case is FDD, which, in the context of control logic, enables the identification and correction of errors at the building and system level. In the literature, FDD is a common application for Digital Twins (Hosamo et al. 2022; Xie et al. 2023; Bjørnskov and Jradi 2023; Cai, Khayatian, and Heer 2021; Jafari et al. 2021; Lu et al. 2020). For FDD, model-free or model-based techniques can be used, but in order to be able to react quickly to errors, a comparison of model variables and measured values should be done in real time if possible. In buildings, error correction typically requires manual intervention on site.

- *Predictive Maintenance*

  Closely related to FDD are Predictive Maintenance (PM) applications. Both applications are commonly integrated (Hosamo et al. 2022; Lu et al. 2020; Peng et al. 2020). For PM, models that reflect system degradation are required. Control logic is relevant in the context of PM when frequent on-off or oscillations affect system lifetime.

- *Model Predictive Control*

  In operation, predictions of the future behavior enable MPC (Clarke et al. 2002; Drgoňa et al. 2020). MPC is reported to enable significant energy savings (Serale et al. 2018; Blum et al. 2022) and plays an important role especially for flexibility in the interaction with higher level power and heat grids. For MPC, behavioral models are mandatory. MPC applications in the context of Digital Twins are reported by Clausen et al. (2021), Berger et al. (2022) or Agouzoul, Simeu, and Tabaa (2023).

- *Virtual measurements*

  Depending on the level of detail of the building model, variables that are not measured in Physical Twin can be used as virtual measurements. A general framework is given by Yoon (2022). Ruepp et al. (2022) describes a prototypical implementation for the BPS environment IDA ICE.

Table 2.3 structures selected studies related to different building Digital Twin use cases, some of which have been cited above. The table distinguishes between design use cases (Digital Twin Prototype) and operational use cases (Digital Twin Instance). The two right columns indicate the model type and the characteristic of the information exchange between Digital Twin and Physical Twin according to Kritzinger et al. (2018) (Section 2.4.3). It can be observed that Digital Twin applications focus either on the design phase (Digital Twin Prototype) or on the operation phase (Digital Twin Instance). Exceptions are two studies by Lydon et al. (2019) and Nytsch-Geusen et al. (2019) and MPC applications (Agouzoul, Simeu, and Tabaa 2023; Cai, Khayatian, and Heer 2021; Berger et al. 2022; Clausen et al. 2021).

Lydon et al. (2019) intended, but did not demonstrate, the use of a thermally activated building system (TABS) component model for operational use cases, such as FDD or controls. Nytsch-Geusen et al. (2019) coupled a building model in Modelica with the openHAB platform. OpenHAB is an open-source platform for home automation and allows for the development of control strategies based on rules and logical calculations (openHAB Foundation 2023). The control strategy defined in openHAB was first tested on the Modelica model and the used for operation in a demonstration building.

MPC applications include by nature the development of building and system models and of a control algorithm in the design phase and their application in the operation phase. Accordingly, the Digital Twin MPC applications use both a Digital Twin Prototype and a Digital Twin Instance[3].

The studies which focus on the operation phase mostly focus on a single use case. An exception is the work of Hosamo et al. (2022), Lu et al. (2020), and Peng et al. (2020), which combined the related use cases FDD and PM. The only study that aims at using Digital Twin for different applications is that of Berger et al. (2022), who intended to use a chiller model developed for MPC also for FDD and PM.

2.4.5 Relation to BIM

In recent years, BIM methods have been developed to make information available between different technical domains and different life cycle stages. The relationship between BIM and Digital Twins was analyzed in several studies (Khajavi et al. 2019; Davila Delgado and Oyedele 2021; Sacks et al. 2020; Boje et al. 2020). According to these studies, the fact that BIM tools are not designed for the previously described model-based use cases

3. In Agouzoul, Simeu, and Tabaa (2023) no practical implementation of the developed MPC is carried out.

Table 2.3: Literature review about Digital Twin use cases, model types, and connection from Digital Twin Instance to Physical Twin (Dec: decision-making, Edu: education, PA: performance analysis, VM: virtual measurements, (x) = described but not executed. Mod: Model type (b: behavioral (wbm/gbm/bbm: white-/grey-/black-box-model, ts/fore: time-series forecasting), i: information (data / rule model)), Conn: connection Digital Twin Instance to Physical Twin (man: manuell, auto: automatic))

	DTP		DTI						Mod	Conn
Source	Dec	Edu	PA	Dec	FDD	PM	MPC	VM		
Buckley et al.	x	-	-	-	-	-	-	-	b:wbm	n/a
Marchione et al.	x	-	-	-	-	-	-	-	b:wbm	n/a
Simonsson et al.	x	-	-	-	-	-	-	-	b:wbm	n/a
Tariq et al.	x	-	-	-	-	-	-	-	b:bbm	n/a
Lydon et al.	x	-	(x)	-	-	-	-	-	b:wbm	n/a
Nytsch-Geusen et al.	x	-	x	-	-	-	-	-	b:wbm	n/a
Agouzoul et al.	x (MPC)	-	-	-	-	-	-	-	b:w/bbm	n/a
Cai et al.	x (MPC)	-	-	-	-	-	(x)	-	b:wbm	man
Berger et al.	x (MPC)	-	-	-	(x)	(x)	x	-	b:gbm	auto
Clausen et al.	x (MPC)	-	-	-	-	-	x	-	b:gbm	auto
Johra et al.	-	x	-	-	-	-	-	-	b:wbm	n/a
Spudys et al.	-	-	x	-	-	-	-	-	i:data	man
Ward et al.	-	-	x	x	-	-	-	-	b:ts/fore	n/a
Bjørnskov et al.	-	-	-	(x)	(x)	-	-	-	b:gbm	man
Plesser	-	-	-	-	x	-	-	-	i:rule	man
Xie et al.	-	-	-	-	x	-	-	-	i:data	man
Peng et al.	- -	- -	- -	- -	x -	- x	- -	- -	i:data/b:ML b:bbm	man man
Jafari et al.	-	-	-	-	x	-	-	-	b:w/bbm	man
Hosamo et al.	- -	- -	- -	- -	x -	- x	- -	- -	i:data/b:ML b:bbm	man man
Lu et al.	- -	- -	- -	- -	x -	- x	- -	- -	i:data b:bbm	man man
Ruepp et al.	-	-	-	-	-	-	-	x	b:wbm	n/a
Yoon	-	-	-	-	-	-	-	x	b:w/g/bbm	n/a

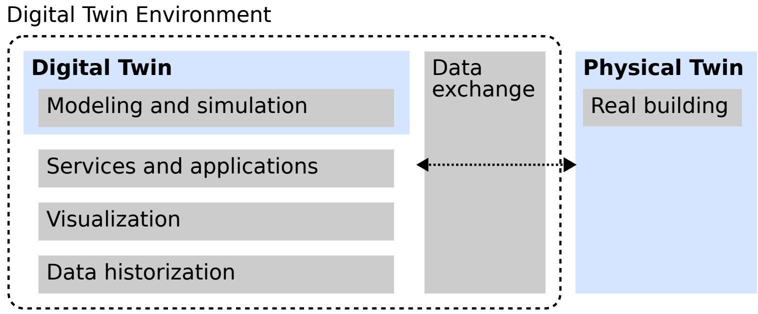


Figure 2.7: Components and features of a Digital Twin Environment

in operation is a main difference to Digital Twins. However, the ability of BIM to provide information can be used to support the creation of Digital Twins, as described by Jradi and Bjørnskov (2023) and Yoon (2023).

2.4.6 Architecture and Digital Twin Environment

For the actual application and operation of Digital Twins in the use cases described above, a software environment is required. This software environment is commonly referred to as Digital Twin Environment (Grieves and Vickers 2017). The components and architecture of a Digital Twin Environment are described by Red Hat (2023) from a software developer's perspective and by Jafari et al. (2021) for a case study. In general, Digital Twin Environments provide the following features (see Figure 2.7):

- *Modeling and simulation*

  The core of a Digital Twin Environment are models of the Physical Twin (Section 2.4.2) and associated simulation engines.

- *Services and applications*

  The service layer contains applications for use cases such as FDD or MPC (Section 2.4.4). These applications usually access simulation models.

- *Visualization*

  Visualization is needed to communicate with engineers, users, and operators. Visualization includes, first, the display of Physical Twin and Digital Twin in operation and, second, the results of Digital Twin applications. While BMS usually include graphical user interfaces (GUIs) to display live and historical data, simulation tools include visualizations for model representations and results. Integrated solutions are still under development (Lin and Low 2021).

- *Data and storage*

  Live data, historical data, and data generated by Digital Twin applications need to be managed in an integrated manner, for example in databases. Aspects of data access, storage, quality, and utility in the case of live data were discussed in detail by Ward et al. (2023).

- *Communication*

  Communication between Physical Twin and Digital Twin uses exchange methods and formats as described in Section 2.4.3.

In today's construction practice, Digital Twin Environments must be created project-specifically. Software solutions that cover all aspects are not yet available.

2.5 Summary

Finding 1. *The development of controls in the design phase is not performance-based.* The way controls should be planned and documented is described in great detail in available standards (Section 2.2.1). However, simulations that allow testing under dynamic conditions are not used. Therefore, the resulting performance of HVAC systems and buildings cannot be verified. For performance at the building level, the two different legislative approaches in Germany and Sweden have been compared (Section 2.1). While in the Swedish case the performance of controls and HVAC systems is implicitly considered, in the German case the controls are standardized and largely simplified.

Finding 2. *The development of detailed controls for HVAC systems and their simulation on coupled building models is possible with today's BPS tools.* Different modeling methods and simulation technologies are available today for different applications in buildings. BPS environments allow for the integrated design and evaluation of buildings, HVAC systems, and controls. However, fully integrated software environments are still rare. In addition, the modeling effort is still high and often requires knowledge from various domains. The fact that BPS is not a standard tool in design is closely related to regulatory requirements. The description of performance requirements at the building level in Germany and Sweden has shown different approaches to requirements for the building energy demand, which do or do not promote the use of BPS tools (Section 2.1).

Finding 3. *Digital formats for transferring controls from the design phase to the implementation phase have been developed or are under development, but they are not reflected in building standards, they are facing a heterogeneous automation sector, and experiences with practical implementations do not exist.* With the PLCopen XML, a digital format has been developed to exchange control logic between design and execution for industrial applications (Section 2.2.4). In the BPS software IDA ICE, a toolchain for the development and simulation of control sequences for PLCs and their communication via the PLCopen XML exists (Section 2.3.4). Another option to digitally transfer control logic from BPS to building controllers is the CDL, which is still in the standardization process. In the building sector, however, the conditions for the use of such digital exchange formats are currently not given, because planning offices do not develop controls at the code level (Section 2.2.1). Instead, they use standardized graphical automation schemas, function lists, and textual descriptions to design and communicate controls from design offices to executing companies. These tools and formats are descriptive rather than deterministic and known to be error-prone.

Finding 4. *Digital Twins are a widely used method, but applications that focus on controls and integrate both design and operations are rare.* The Digital Twin approach as a key concept of Industry 4.0 is widely applied in various industrial sectors. In the context of buildings, Digital Twins are a promising approach to stimulate the digitalization of design, construction, and operation. However, applications still mainly remain in the context of research projects and comprehensive approaches are still being developed. The analysis in Section 2.4 shows that, apart from a few examples, Digital Twins are used for applications either in the design or in the operation phase. However, based on initial definitions, using Digital Twins in both phases seems to be the most promising approach. In the context of building-related control design, only a few concrete examples are reported in the literature.

3 Evaluation of demonstration buildings

For the empirical analysis within this thesis, AHUs in demonstration buildings from research projects are used . These demonstration buildings are described in Section 3.1 AHUs are studied because they are HVAC systems that are frequently used in different types of buildings and which are often a source of malfunction (Gunay, Shen, and Yang 2017).

The analysis is carried out from two points of view:

1. For building Digital Twins in operation, information about control sequences are required. In Section 3.2, different approaches to gathering information about the designed or implemented controls are compared.

2. The operational performance of AHUs in one of the demonstration buildings is evaluated in the context of today's typical design and commissioning processes in Section 3.3.

3.1 Presentation of demonstration buildings

The selected demonstration buildings are taken from research projects in Germany and Sweden (see Table 3.1). They represent a wide range of uses, including a factory building, a university building with seminar and study rooms, and a building with student apartments. All buildings have high user requirements for thermal comfort and energy efficiency.

3.1.1 Factory building

Since 2020, the pump manufacturer Wilo has been operating a 50,000 m^2 factory building on the Wilopark in Dortmund (Germany), which includes production, logistics and office areas (Figure 3.1). This factory building is part of the VEProB (Connected Energy Flows of Production and Office Buildings) research project.

Within the framework of this analysis, two coupled AHUs are examined, which heat, cool and supply fresh air to two industrial production areas (Figure 3.2). The two AHUs studied are identical in size and have a heating coil and a cooling coil, mixed air dampers,

Table 3.1: Comparison of demonstration buildings

	Factory building 	Testbed AH 	Testbed KTH 
Country	Germany	Sweden	Sweden
Area [m^2]	50,000	3,500	300
Year of commissioning	2020	2019	2019
Use	Production and logistics	Seminar and learning rooms	Residential building for students
Number of AHUs	37 (24 central, 13 decentral)	2	1
Components of analyzed AHU	• Liquid heat recovery • Mixed air damper • Heating coil • Cooling coil • Adiabatic extract air humidification	• Rotary heat exchanger • Heating coil • Cooling coil	• Pre-heating / cooling coil • Rotary heat exchanger • Heating coil
Controlled variable	Extract air temperature	Supply air temperature	Extract air temperature
Controller / programming	PLC according to IEC 61131-3	PLC according to IEC 61131-3	Proprietary controller
Field level communication	Modbus	Modbus	BACnet

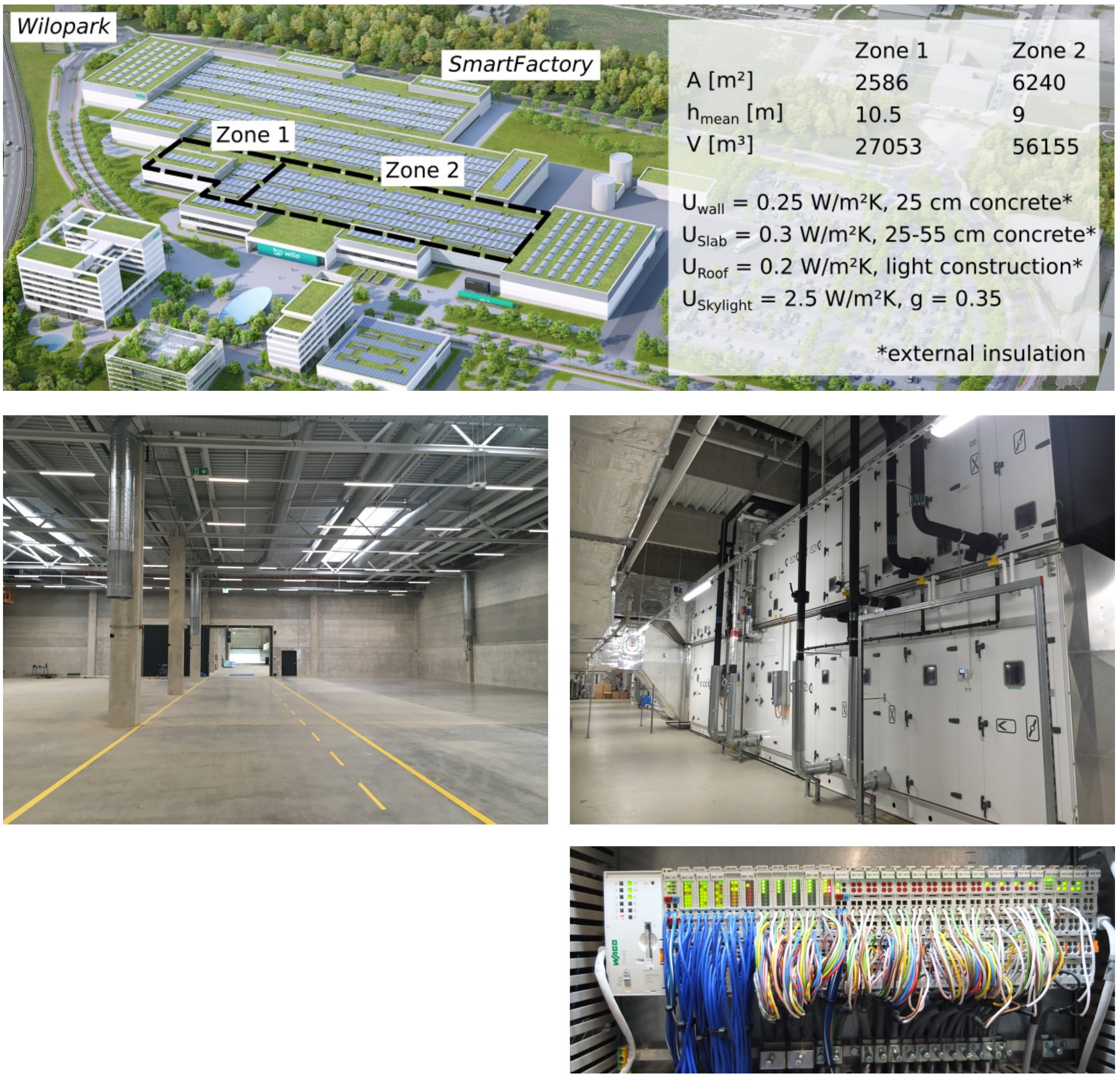


Figure 3.1: Wilo factory building. Top: Aerial view (source: Wilo SE) and investigated zones. Middle left: typical production area without machines. Middle right: air handling unit. Bottom: PLC.

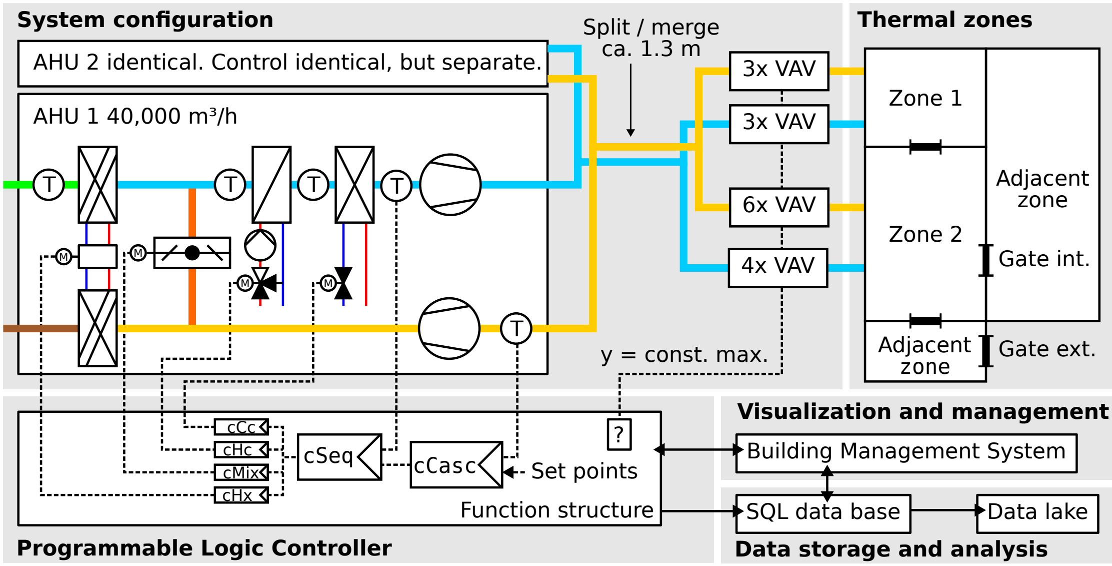


Figure 3.2: AHUs (without outdoor air flaps, filters and sound absorbers), connected zones, sensors, and assumed and simplified initial control logic

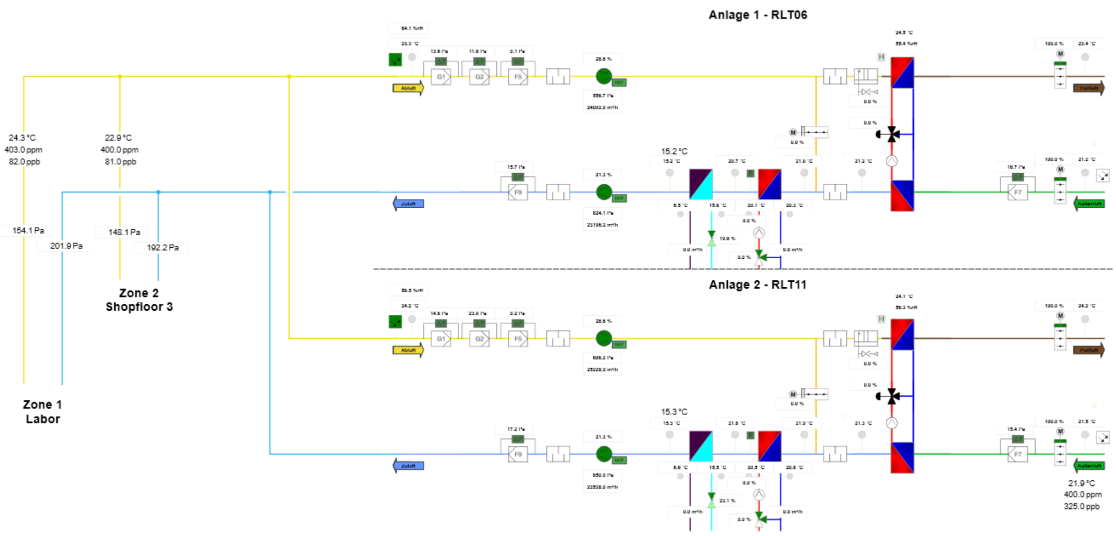


Figure 3.3: User interface of the BMS (scheme mirrored horizontally compared to Figure 3.2)

liquid heat recovery and adiabatic extract air humidifications. The AHUs are designed in parallel, i.e. the supply air flows are first merged and then split into two zones. The air volume can be varied zone by zone using VAV boxes. The selection and sizing of the AHUs and individual components, as well as the design of the initial control logic, were carried out by professional designers outside the research project.

The automation and information and communications technology (ICT) infrastructure is classically divided into field level, automation level and management level (see Figure 3.2 and Section 2.2.2). Temperatures are measured after some, but not all, air treatment stages. In addition, supply and return water temperatures are measured at the heating and cooling coils. Energy flows are recorded via calibrated heat meters at the heating coil and cooling coil. In addition, modern wet-rotor pumps also record heat flows. Air volume flows are recorded by measuring the differential pressure at the nozzle gap of the fans and at the VAV boxes.

The control of the actuators assigned to the heating, cooling, heat recovery and mixed air functions, such as valves and pumps, can be freely programmed, i.e. no proprietary controllers of the AHU manufacturer are used. The control program is implemented on a PLC (bottom left in Figure 3.2), which is programmed in the IDE e!cockpit (WAGO). The software allows programming in the five languages according to IEC 61131-3 (Section 2.2.3). The IDE supports import and export of the PLCopen XML according to IEC 61131-10 (Section 2.2.4). The communication between the PLCs and the actuators is based on the Modbus standard. A BMS allows the building operator to view current measured values and change setpoints (see Figure 3.3). The measurement data is stored in a SQL database and transferred from there to a data lake for data analysis (bottom right in Figure 3.2).

In addition to these AHUs controlled by PLCs, decentralized AHUs with a proprietary control from the manufacturer are installed in selected zones. These systems receive an activation signal and a setpoint from the superior BMS. They are not included in this analysis.

3.1.2 Testbed Akadmiska Hus

The Testbed Akademiska Hus is part of the KTH Live-In Lab on the campus of KTH Royal Institute of Technology (KTH) in Stockholm (Sweden). It is used as a lecture hall with student workplaces. The building has two identical AHUs, which are operated in parallel (see Figure 3.4), similar to those described in Section 3.1.1. The AHUs have a rotary heat exchanger for heat recovery, a heating coil and a cooling coil. Temperatures are measured before and after each air treatment step. The AHUs are controlled by a PLC

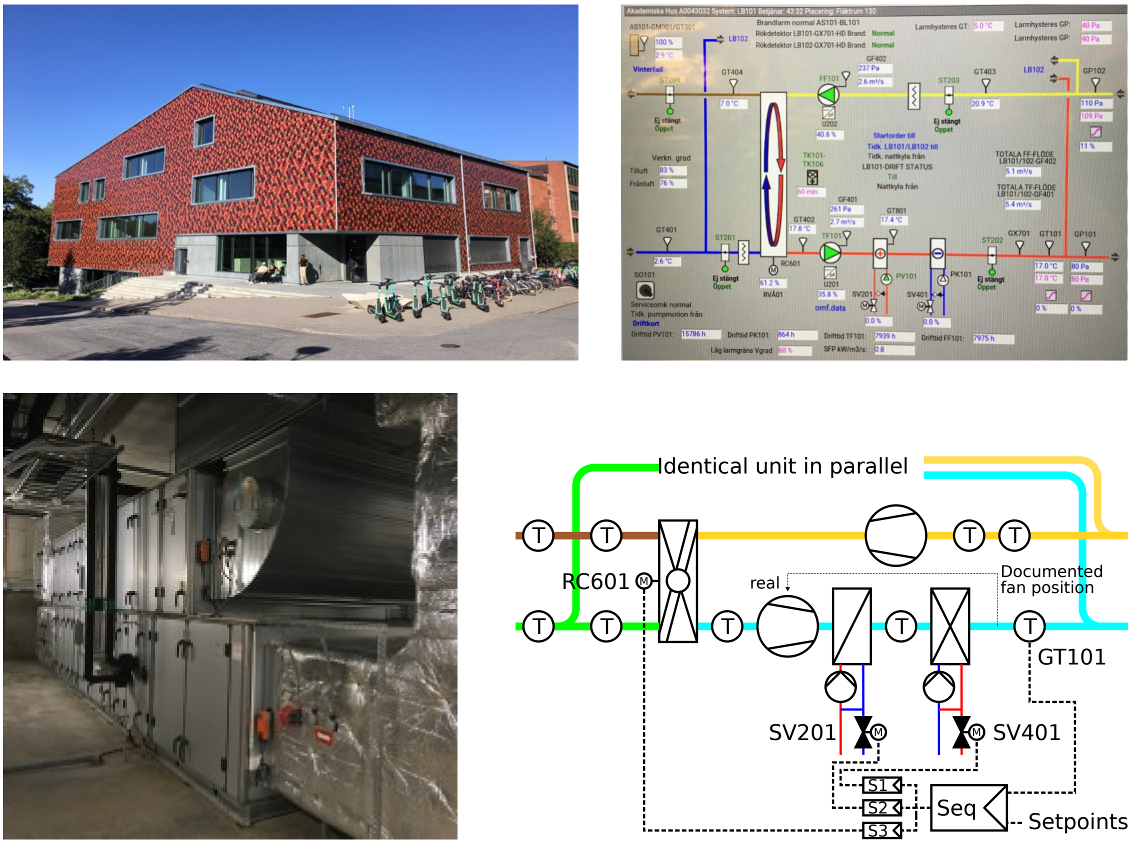


Figure 3.4: Testbed Akademiska Hus. Top left: Building view. Bottom left: AHU. Top right: user interface of the PLC. Bottom right: Simplified scheme of the AHU and controls deduced from textual descriptions from the design phase.

programmed according to IEC 61131-3 in the FX-Editor (Fidelix). The communication protocol used is Modbus.

3.1.3 Testbed KTH

The Testbed KTH is integrated into a student housing complex and is also part of the KTH Live-In Lab. The buildings spaces are used for student apartments and can be varied according to the needs of research projects. The fresh air supply is provided by a AHUs with a preheating and precooling coil connected to a borehole field, a rotary heat exchanger and a heating coil (see Figure 3.5). The control of the AHU is programmed on a building controller in an IDE using a proprietary standard. The communication protocol used is BACnet.

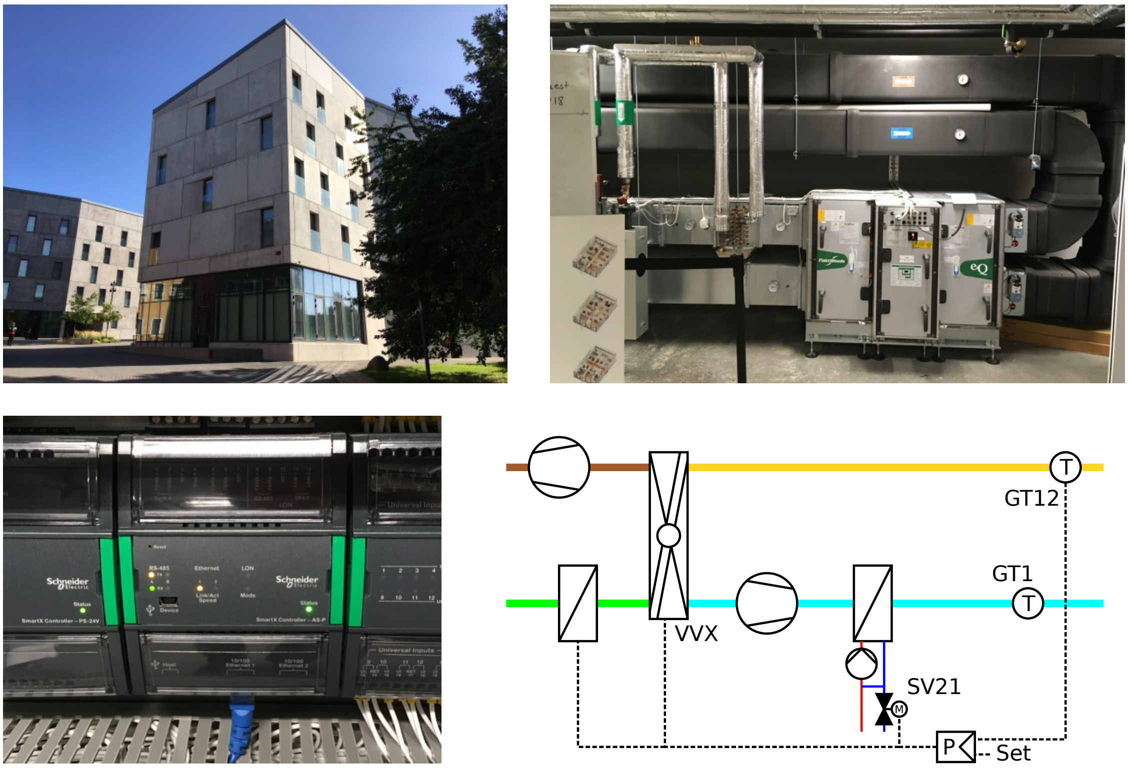


Figure 3.5: Testbed KTH. Top left: Building view. Top right: AHU. Bottom left: Building controller. Bottom right: Simplified scheme of AHU and controls deduced from the functional description and from BPS from the design phase.

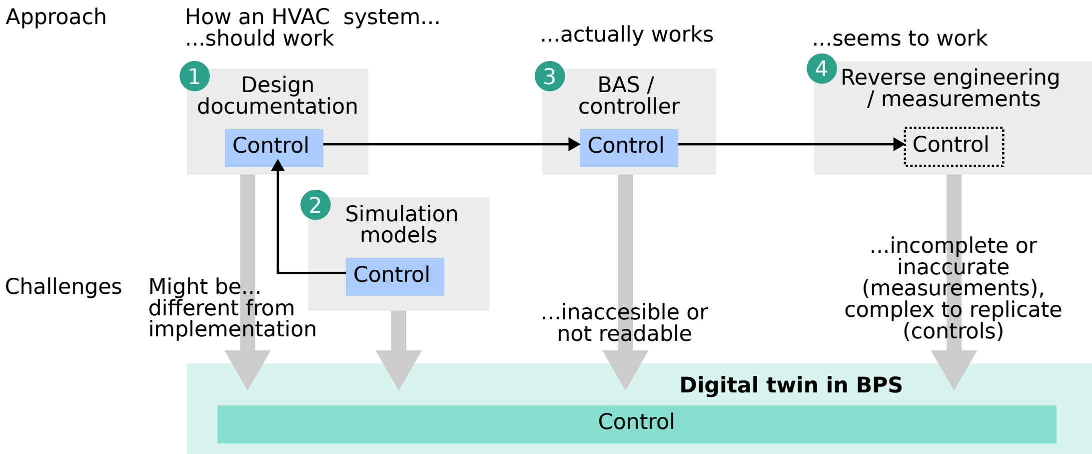


Figure 3.6: Methods and challenges for the information collection about controls for building Digital Twins in operation

3.2 Sources of information about controls

To gather information about the underlying control logic of these AHUs for building Digital Twins, four options are distinguished (see Figure 3.6). Each approach has its technological and organizational barriers which are outlined in the following and analyzed for the demonstration buildings in the later sections.

1. *Design documentation*

   The first approach is to extract information from design documents. Controls of HVAC systems should be defined by HVAC and automation engineers in the design phase ("1" in Figure 3.6). Design documents should clearly describe the intended control for implementation (see Section 2.2.1). Typical challenges are adaptations during the construction and operation phases that cause the actual operation to deviate from the intended operation.

2. *BPS models from design phase*

   Information about the intended control can also be obtained from BPS models ("2" in Figure 3.6) if simulations were performed in the design phase. In practice, BPS are usually used to support the design process rather than as a direct template for programming. However, simulation models, including controls, could be updated and reused as a Digital Twin in operation.

3. *Control code implemented on building controllers*

   The actual operation of HVAC systems is defined by the code implemented on BAS ("3" in Figure 3.6). In order to reproduce the control logic in a Digital Twin, first, the control code must be accessible, and second, engineers capable of reading and understanding the implemented code are needed. Another challenge is that the control program may have changed over time. If these changes are not tracked, this is a problem for performance analysis based on historical data, for example.

4. *Reverse engineering from measured data*

   If no information about the implemented control strategy is available from the design phase or the automation systems, controls can be reverse engineered from measured data ("4" in Figure 3.6). The level of detail that can be derived depends on the measurement infrastructure and the complexity of the AHU. Together with typical control approaches from the literature and expert knowledge, a possible control can then be estimated.

3.2.1 Design documentation

Factory building An automation scheme and a textual functional description are available from the design phase. However, the automation schematic lacks the functional structure and the characteristic curves, and therefore does not meet the normative requirements. In addition, instead of the built configuration shown in Figure 3.2, the automation scheme shows a sequence with heating coil, supply air fan, cooling coil and steam evaporator in the supply air duct.

The textual functional description describes the intended operation as follows (agn Niederberghaus & Partner and siganet 2019):

The supply air temperature is controlled depending on the room temperature. The continuous controller compares the room temperature measured by the sensor with the set point. If there is a deviation, the controller causes the heating or cooling valve to be adjusted. [...] A supply air temperature minimum limitation ensures that the supply air cannot be blown in so cold that unpleasant drafts occur [...]. The supply air may be blown in at a maximum of 2 K (freely parameterizable) below the room temperature. The room ventilation must be controlled depending on the humidity in the room. If this value is exceeded, the supply air volume flow is increased up to the maximum value."

The description mentions the heating and cooling coils, but omits the heat recovery, the mixed air dampers and the adiabatic extract air humidification. In general, the description is rather functional and qualitative. A much more detailed description would have been needed to compensate for the lack of information in the automation schemes.

Testbed Akademiska Hus The automation schemes from the design phase include the sensors and the actuators, but they do not include the characteristic curves nor the function structure. Additionally, the supply air fan in the built system is in a different position than in the drawings (see Figure 3.4).

The concept for heating and cooling of the controller is described textually as follows (incoord 2015):

"The temperature sensor GT101 controls the speed control device RC601 and the control valves SV401 and SV201 in sequence so that the calculated value is obtained. Control sequence from maximum cooling: Control valve SV401 is controlled for cooling 100-0%. The speed control device RC601 is controlled for heat recovery 0-100%. Control valve SV201 is controlled for heating 0-100%."

The supply air temperature set point is given separately as a function of the ambient temperature. Compared to the factory building case, the description refers explicitly to the sensors and actuators that are specified and the sequence of the individual actuators is explicitly defined. According to the building operator, the design documents are frequently used and helpful in understanding the control behavior in operation.

Testbed KTH As in the case of Testbed Akademiska Hus, the automation schemes neither contain characteristic curves nor automation schemes. In the functional description (Bengt Dahlgren 2018), the control strategy is defined as follows:

"The exhaust air temperature at GT12 is regulated to keep the set point constant. GT12 regulates when the temperature drops in the following sequence sequence: VVX is regulated to maximum speed, SV21 opens for heating, so that the current setpoint for GT12 is obtained. When the temperature rises, the sequence is reversed. Supply air sensor GT1 min- and max-limits the supply air temperature."

Also in this case, the description explicitly refers to indicated sensors and actors and determines the control sequence of single actuators.

3.2.2 BPS models from design phase

Factory building During the design phase, BPS were performed by an external service provider using the software TAS (Environmental Design Solutions 2023). The simulation focused on the calculation of load profiles for heat, cooling and electricity. System simulations were performed for the heat and cooling supply system using the TAS systems software component. Since TAS uses an hourly time step and does not allow coupled simulation of buildings, HVAC systems and their controls, these simulations are generally of limited reliability when it comes to control design (see Section 2.3.1). These simulations have not been used to make statements about the intended controls for the AHUs.

Testbed Akademiska Hus BPS were performed in the design phase using IDA ICE, but were not available for this investigation.

Testbed KTH During the design phase, BPS were performed by an external service provider using IDA ICE. The AHUs and their controls were built up specifically for the project as opposed to being merely selected from the software-integrated AHU library. The control was created using graphical function blocks. In the simulation, the supply air temperature is calculated using a P controller with a 1-K proportional band. The controlled variable is the extract temperature. It could not be clarified whether this control scheme was the basis for the implementation.

3.2.3 Control code implemented on building controllers

Factory building The controls were implemented by an executing company in Structured Text according to IEC 61131-3 using the software e!cockpit. The control code was made available to the operator as part of the VEProB project, but with regard to a replication in a Digital Twin several problems arose:

1. The whole control program has about 500 lines of code. Even for experienced engineers it may be difficult to identify the relevant parts and correctly reproduce the control program in another environment.
2. The main obstacle is that the control program contains proprietary function blocks from the developer of the IDE (WAGO 2022). These proprietary function blocks are used not only for PI controllers, but also for cascade and sequence control functions, which are explained in the next paragraph. Their functionality and inputs and outputs are described in a manual (WAGO 2022), but the underlying code is not available.

Based on the available control code and the explanations of the executing engineer, the implemented control logic can be described as follows (see simplified functional diagram in Figure 3.2): The extract air temperature is the controlled variable. A so-called cascade control is used, which first compares the extract air temperature with the setpoint to determine a target supply air temperature (see controller `cCasc` in Figure 3.2). Upper and lower supply air temperature limits are considered. The controller sequence controller `cSeq` compares this target supply air temperature with the measured supply air temperature. The control signal of `cSeq` takes values between 0 and 100 %, where values below 50 mean heating, values above 50 mean cooling. This output signal is the input for the function blocks `cHc` (heating coil), `cCc` (cooling coil), `cMix` (mixing box) and `cHx` (heat recovery).

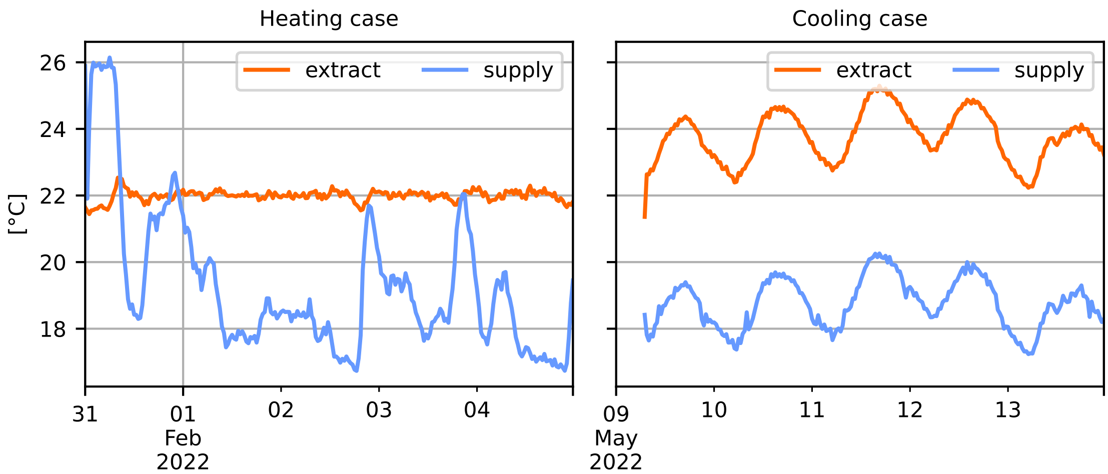


Figure 3.7: Times series of the extract and supply air temperatures of the AHUs in the factory building for a winter and a summer week.

Control inputs, such as setpoints, were often not available as time series in the monitoring database, which is problematic for performance analysis. In addition, the control program was revised and modified several times during commissioning and operation, with no way to track these changes.

Testbed Akademiska Hus and Testbed KTH The control code has not been accessible for this thesis.

3.2.4 Reverse engineering

Factory building As part of the research project, the buildings and the AHUs are to a large extent equipped with sensors, but measurements between the heat recovery and the mixed air damper are missing (see Figure 3.2). Assuming that no design documents or control code were available, it can be observed from the measured data that the extract air temperature is the controlled value (see Figure 3.7). In the heating case, an extract air temperature of 22 °C is maintained by varying the supply air temperature between approximately 17 °C and 26 °C. In the cooling case, a maximum undertemperature of about 5 K is kept. With appropriate expert knowledge, a scheme as shown in Figure 3.2 could be deduced. However, estimating the implemented control for all components (see Table 3.1) in different operating modes based only on observations would require long-term observations for pattern recognition.

Testbed Akademiska Hus Since all relevant setpoints and measurements are available, and the AHU is relatively simple with a heat recovery, a heating coil and a cooling coil, the implemented control logic is relatively easy to reverse engineer. Measured data was not available for this thesis.

Testbed KTH The basic configuration of the AHU with a rotary heat exchanger and a heat exchanger is relatively simple. However, the integration of the preheating/cooling coil coupled to a borehole field is project specific. Therefore, reverse engineering a possible control is not as straightforward as for Testbed Akademiska Hus. Measured data was not available for this thesis.

3.3 Performance analysis of air handling units

In the factory building, the operation of the AHUs could be evaluated over 26 months.

Figure 3.8 a) shows the daily heat consumption over the daily average ambient temperature. Significant amounts of heat are consumed even above ambient temperatures of 15 °C. This clearly indicates inefficient operation. The cause can be found in Figure 3.8 b): The extract air temperature setpoint has a fixed value of 24 °C. If the measured extract air temperature falls below this threshold, the supply air temperature rises up to 28 °C, even if the ambient temperature is in the range of 24 °C. Since the heat recovery cannot provide these high temperatures, the heating coil is activated. This behavior could be avoided, for example, by using a temperature range with a lower limit for heating and an upper limit for cooling instead of a fixed value.

Figure 3.8 c) and d) show the duration curves of the mixed air dampers and VAV boxes. The analysis shows that the position of the mixed air dampers was usually 0 %, which corresponds to 100 % fresh air. For a period of time, fixed mixed air damper positions were set. The VAV boxes were in the 100 % position (nominal volume flow) for most of the operating period. Thus, both AHUs were usually operated exclusively with fresh air at full airflow. This mode of operation was intended for use during the corona pandemic. But even before that, maximum airflow rates were set.

3.4 Summary

Table 3.2 compares the different options to gather information about controls of the analyzed AHUs. Together with the performance analysis, the following findings can be drawn:

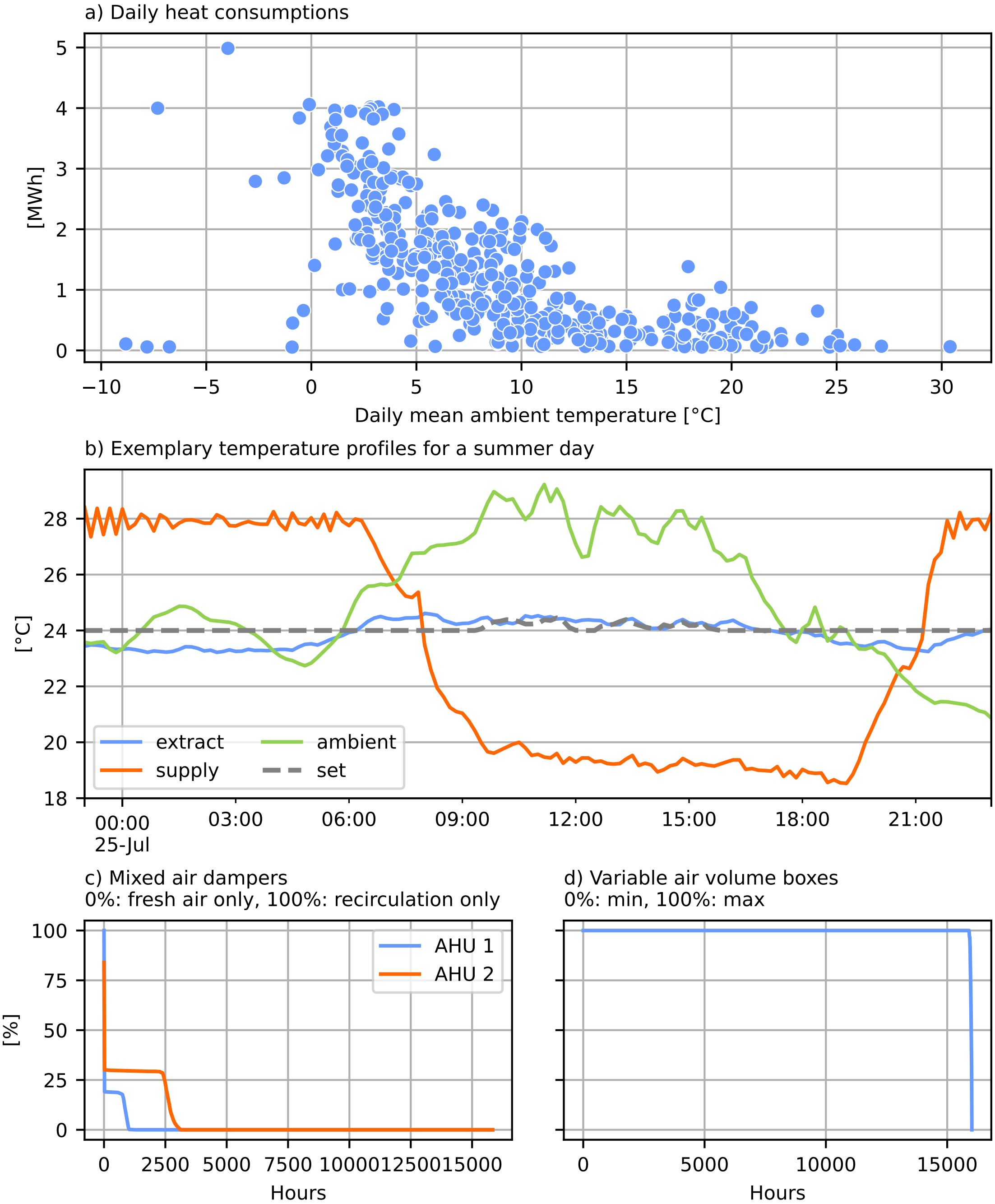


Figure 3.8: Performance of AHU in factory building

Table 3.2: Evaluation of the information sources for controls (Dia: Control diagrams, Sc: Control schemes, Str: Control structure, Td: Textual description, Done: Simulations carried out, Acc: Accessible, Ctrl: Including control simulation, Av: Available, Lib: Open libraries, Meas: Enough measurements available, Std: Standard AHU)

	Design documents				BPS			Code		RevEng	
	Dia	Sc	Str	Td	Done	Acc	Ctrl	Av	Lib	Meas	Std
Factory	-	-	-	-	+	-	-	+	-	o	-
TB AH	-	+	-	+	+	-	n/a	-	n/a	o	+
TB KTH	-	+	-	+	+	+	+	-	n/a	o	o

Finding 5. *Design documents are not an appropriate source of information about controls for building Digital Twins in operation.* None of the available automation schemes for the investigated demonstration buildings contain functional diagrams and characteristic curves (column "Design documents" in Table 3.2). This observation is also reported by Ihlenburg, Benndorf, and Réhault (2022) and is consistent with the generally low quality in the design phase (Fütterer, Schild, and Müller 2017; Fisch et al. 2017). Hardware configuration and setpoints differ between automation schemes and functional descriptions on the one hand, and the built and operated system on the other. Despite the gaps in the automation schemes, the textual description for Testbed Akademiska Hus is comprehensive and frequently consulted by the building operator. This suggests that for rather simple and common HVAC systems, such as the AHU in Testbed Akademiska Hus, where controls are straightforward to engineer, textual descriptions may be sufficient. The error-prone nature of analog formats, and the need for manual reproduction, is a clear barrier to the efficient creation of Digital Twins.

Finding 6. *Replicating control code implemented on BAS for building Digital Twins faces major hurdles due to the use of proprietary solutions.* The control code for the operation of the AHUs was only accessible in the factory building case study (column "Code" in Table 3.2). Proprietary libraries of function blocks where the underlying code is not accessible are a barrier to replicating controls in BPS environments.

Finding 7. *Reverse engineering controls from measured data to build Digital Twins is only possible in very simple cases and associated with large uncertainties.* Reverse engineering of controls from measured data appears to be cumbersome and complicated. First, sensors must be present at the relevant positions, second, the data must be available (no data shortage), and measurement tolerances must be taken into account. In addition, the naming of data points in databases is often ambiguous and unclear. Finally, setpoints must be recorded in a database, otherwise critical information is missing. With some exceptions, the studied AHUs are generally equipped with enough measurements to allow a general understanding of the system behavior. However, only in the case of Testbed Akademiska Hus a relatively simple standard configuration is used, which would allow a replication of the controls (column "RevEng" in Table 3.2).

Finding 8. *The continued use of simulation models and controls from the design phase should be sought for the efficient creation of Digital Twins.* In all projects studied, simulations were performed in the design phase (column "BPS" in Table 3.2). In the factory building, the simulations did not reflect the control of the AHUs and there was no link between the simulations and the control programming. Only the simulation model for Testbed KTH contains the detailed modeling of the controls and is available for further use. However, it could not be proven that the modeled controls were actually implemented. This example suggests that continuing to use models from the design phase is the simplest approach to using Digital Twins in operations. This assumes, of course, that either the same simulation software is used, or that models can be exchanged between BPS tools and building automation (BA) systems.

Finding 9. *The performance of an exemplary AHU is largely determined by the level of detail of the control developed in the design phase.* The example of the factory building shows that poor performance is a direct consequence of a low level of detail in the controls control development during the design phase (see Section 3.2.1 and Section 3.3). Components for efficient operation, such as mixed air dampers and VAV boxes, are built in but not actively used (Section 3.3). This practice is also due to the regulatory situation in Germany, where the design fee is derived from the construction amount as defined in the HOAI (Section 1.2.1). This encourages HVAC designers to combine efficient components into a complex system without having to develop and prove explicit controls.

4 Methodology

This chapter describes methods and tools that address the findings and deficits identified in chapter 2 and chapter 3. Section 4.1 provides an overview of the individual steps. In Section 4.2 the requirements for control development and building energy modeling are defined. Methods for connecting BPS tools and BAS are discussed in Section 4.3. The embedding in a Digital Twin framework is explained in Section 4.4.

4.1 Overview

To address the performance gap related to the control of HVAC systems and to realize the potential of Digital Twins over the building life cycle, the following three-step approach is applied within this thesis (see Figure 4.1):

1. *Design phase (prior to awarding)*

   Development and optimization of controls for HVAC systems on coupled building and system models in BPS environments as a Digital Twin Prototype

2. *Execution and commissioning phase*

   Implementation of the control logic developed in the Digital Twin Prototype in the Physical Twin for the operation of the building and HVAC systems

3. *Operational phase*

   Continued use of building and system models including controls in the Digital Twin Instance through a bidirectional connection with the Physical Twin for model-based applications

The development of detailed controls in the design phase, before the contract is awarded (step 1), is the main technical and organizational difference from current building practice (Section 1.2.2). This increases the level of detail in the design phase (see Figure 1.4) and enables integrated design of HVAC systems and controls (Section 1.2.1). Executing companies then adopt and import these controls into their software environment.

This approach differs from the typical HIL or SIL (see Section 2.2.3) methods in two ways: First, the development and testing of controls takes place in the design phase rather

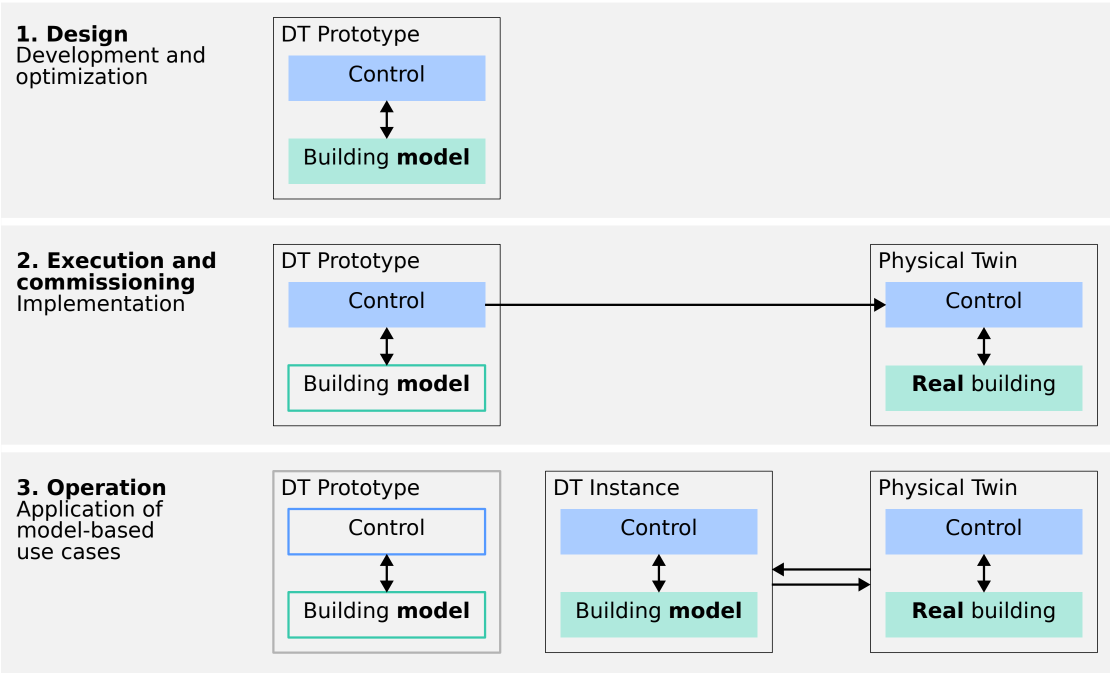


Figure 4.1: Three-step approach for control development and implementation in the Digital Twin and the real building

than just prior to commissioning (see description of today's practice in Section 2.2.3). Second, the controls are integrated directly into BPS environments and do not have any restrictions on execution in real time (see for example Béguery et al. (2021)).

With regard to the Digital Twin methodology, the combination of model-based planning (Digital Twin Prototype) and model-based applications in operation (Digital Twin Instance) is explicitly aimed at (Grieves and Vickers 2017). During commissioning and operation, additional model calibrations must be carried out (Viola and Chen 2020). One such approach for buildings is the "building tracker" described by Ruepp et al. (2022).

Implementing controls previously tested on simulation models follows the approach of Nytsch-Geusen et al. (2019) using Modelica and openHAB. However, openHAB is aimed more at the integration of various home automation systems and the controls in openHAB are limited to rules with triggers, conditions and actions. In contrast, this thesis proposes, for example, to test executable code according to IEC 61131-3 on Digital Twins, which is directly suitable for controlling complex HVAC systems. Moreover, the coupling of Modelica and openHAB was also done real time.

The proposed approach is not limited to HVAC systems and equally applicable to lighting and shading applications. Theoretically, controls of lighting and shading systems for visual comfort, or controls of room-side HVAC systems for acoustic comfort, could

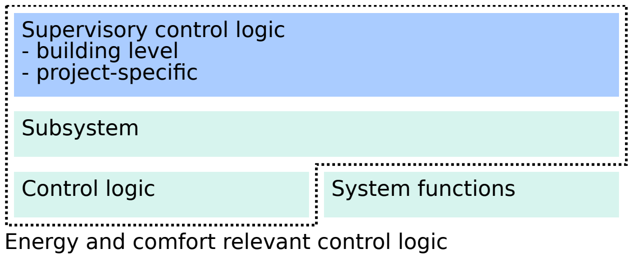


Figure 4.2: Scope of control logic in the proposed workflow

also be considered if the relevant quantities are part of the modeling. However, as stated in Section 1.2.1, this thesis focuses on HVAC systems and their performance with respect to energy and thermal comfort.

Project-specific building energy modeling and control development pays off when respective savings can be realized. This is especially the case for custom HVAC systems in commercial and larger residential buildings. In contrast, single-family houses are often equipped with smaller, recurring HVAC systems and are more relevant when considered at the cluster level.

4.2 Control development and building energy modeling

4.2.1 Spatial and temporal scope

The modeling is usually done at the building scale, but depending on the design task and the purpose of the investigation, subareas or individual systems can be considered. Modeling includes HVAC systems to evaluate energy performance and thermal zones to evaluate thermal comfort. Coupled building and HVAC system models are required to account for the interaction between the systems and the building.

Controls are typically tested over a full year to cover a wide range of possible operating modes and to detect and eliminate undesirable system behavior. Through further parameter variations, system behavior which is not covered by typical assumptions, has to be detected and excluded.

4.2.2 Scope of control development

In the context of this approach, two sub-areas are distinguished in terms of controls (see Figure 4.2).

- *Energy and comfort control*

  Control design in BPS typically focuses on aspects that have a relevant impact on building performance. This is typically the logic by which actors are turned on and off, or their operating behavior is changed discretely or continuously to maintain, for example, a certain room temperature. These aspects are referred to as "energy and comfort control" in this thesis. The energy and comfort control includes both supervisory and subsystem levels (see Section 2.2.2).

- *System functions*

  Subsystem level control programs provide additional functionality to ensure proper operation, but typically do not have a significant impact on overall building energy and comfort performance. This includes fan start sequences, frost protection for hydronic systems, system response to smoke and fire alarms, and manual operating modes necessary for practical and safe operation. These elements are referred to as "system functions". System functions also include pre- and post-processing used for smoothing and filtering measured values, monitoring functions of e.g. filters of AHUs, warnings and error messages, connection to the BMS and historization of measured data in databases. Since system functions are not considered in BPS during the design phase, they must be added and integrated in a separate step.

This separation is necessary to take into account the typical scope of modeling and the level of detail in BPS tools. Control functions are also grouped in VDI 3814-3.1:2013 and VDI 3814-4.3:2022 into input/output, application, functions, operating, monitoring, management, and service/diagnostic functions. However, this structure does not match the above described separation between "Energy and comfort control" and "System functions" exactly. For example, the "application functions" according to the two standards essentially correspond to energy and comfort control. However, they also include frost protection.

4.2.3 Performance requirements

The development of the control scheme is performance-oriented and is subject to verification by means of verifiable indicators. In general, the requirements for the operation of HVAC systems can be structured into safety, energy and comfort relevant aspects and comprehensibility (see Figure 4.3). Safe operation, e.g. in the event of fire or manual operation of devices, is of paramount importance. These aspects are typically part of the system functions (Section 4.2.2) and are therefore not considered here. In addition, the understandability and comprehensibility of the control behavior in operation is important to enable operators to actively manage buildings. Aspects of comprehensibility are also not evaluated in this thesis. As part of the proposed methodology, the following four areas are distinguished to assess operational performance in terms of energy and thermal comfort:

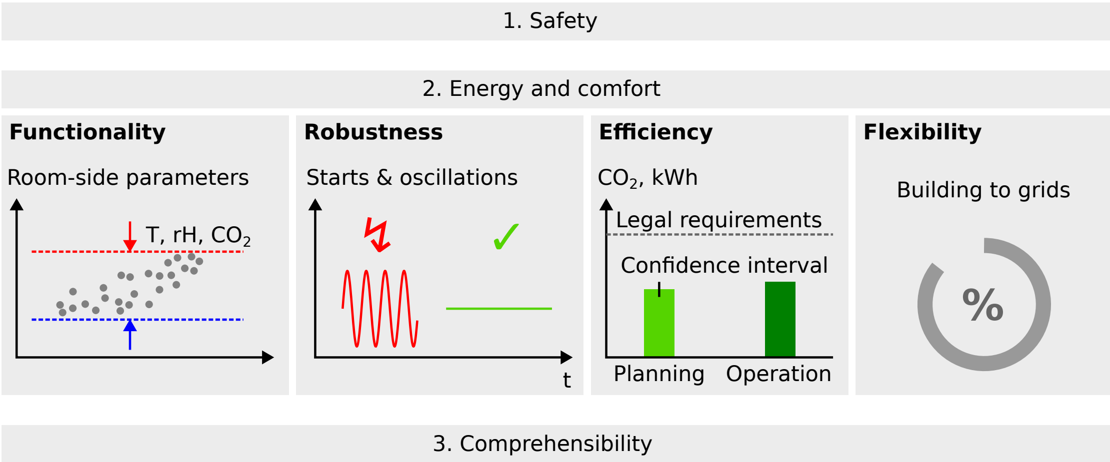


Figure 4.3: Global performance requirements for simulation and real operation

- *Functionality*

  First and foremost is the achievement of functional requirements defined by operators and users. They typically include indoor thermal comfort parameters such as temperature, humidity, and air quality. In the context of indoor comfort requirements, both measurement inaccuracies and the fuzziness of translating expectations into measurable quantities must be considered.

- *Robustness*

  The operation of HVAC systems must be robust at different operating points and under dynamic conditions. This includes, firstly, that the systems are not switched on and off permanently, and secondly, that the controller outputs do not oscillate and consequently wear out actuators such as valves under continuous control. The first step for an assessment is to identify the relevant components and associated control loops, then select performance indicators and limits. Such limits, for example, for the number of pump starts, are often not available. Only for certain components, such as CHP units, are they usually expressed in terms of runtime-per-start ratio.

- *Efficiency*

  At the building level, efficiency criteria in terms of energy or emissions must be demonstrated. By definition, the demonstration of efficiency requires the comparison of two variants. The definition of the requirements should be set by legal requirements at the building level (Section 2.1). The target value from the design must fall below these requirements and provide a confidence interval based on typical variations for climate, use, etc. During operation, it must then be demonstrated that the measured efficiency indicator is below the requirements and within the confidence interval of the simulation.

- *Grid flexibility*

  Buildings have to take part in demand side management (DSM) and react to the volitality of renewable energies. The ability to interact with superordinate power or heat grids is a relatively new requirement of buildings. In the EU, the Smart Readiness Indicator (SRI) has been part of the EPBD (European Parliament and Council of the European Union 2018) since 2018. Concerning the grid flexibility of buildings suitable indicators are developed e.g. in the framework of International Energy Agency (IEA) EBC Annexes 67 and Annex 82 (Knotzer, Pernetti, and Jensen 2019).

These areas cover the full range from building level indicators such as total energy consumption, room temperatures, down to the performance of individual control loops, and must be applicable in both design and operation. For each of these four areas, appropriate indicators must be selected that are equally applicable in both design and operation.

Performance orientation in control design is of great importance because the use of BPS environments does not automatically lead to an optimized control design. In practice, it can be assumed that no more effort is invested than the time and cost budget allows. In terms of performance orientation, it is important that the models are reliable and reflect reality correctly.

4.2.4 Modeling methodology, requirements and simplifications

For modeling buildings, systems, and controls, white-box models are used by default. White-box models are used, firstly, for detailed modeling of the underlying physics and, secondly, because the data required for data-driven approaches are often not available at the design stage. For existing buildings, on the other hand, data-driven modeling methods can be used.

Simplifying building energy models involves a trade-off between the necessary level of detail and the effort required for modeling and computation. Practical applicability demands reasonable simulation time. The modeling of buildings, HVAC systems, and controls must meet the minimum requirements outlined below:

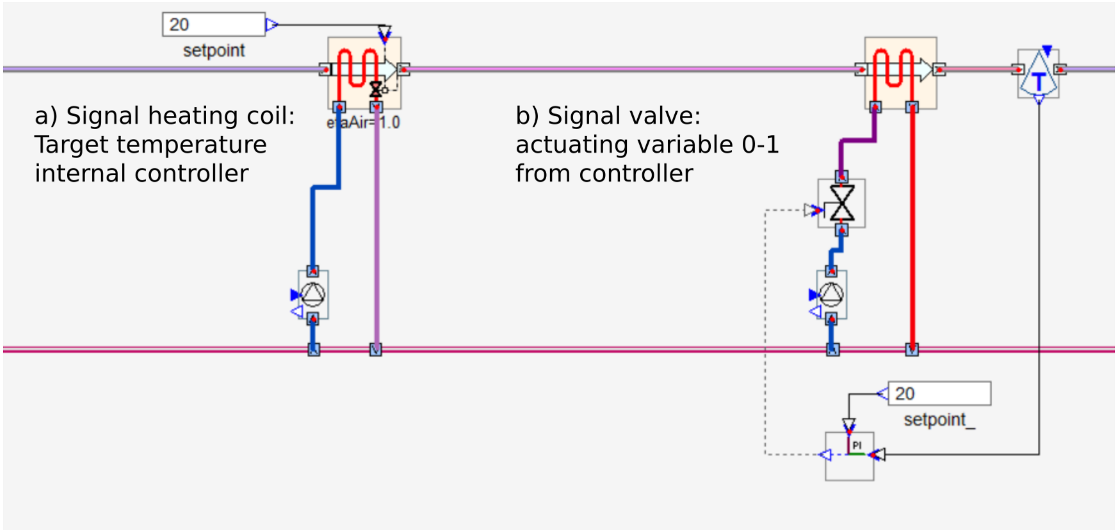


Figure 4.4: Different control representations for a heating coil in IDA ICE

- *Thermal zones*

  The room-side models have to allow for an appropriate assessment of the thermal comfort, for which typically 1-node zone models with ideal mixing of room air are used.

- *HVAC systems*

  HVAC systems are typically modeled as a network of nodes, where each node represents a component such as a heat exchanger, a valve, or a section of pipe. The modeling must allow free, object-oriented composition according to the real systems. In particular, the input signals of the simulation components must match the input signals in the real systems. As an example, Figure 4.4 a) shows a model component for a heating coil from the IDA ICE library, which receives a temperature setpoint for the supply air temperature as an input signal. The component has an internal controller that determines the mass flow to achieve a defined temperature drop. Such a simplified component cannot receive dimensionless actuating variables, which are required in certain situations, such as a sequence control in AHUs (see Section 3.2.3). Components with integrated controls are therefore not suitable for all modeling tasks. In contrast, in the model component shown in Figure 4.4 b) the mass flow is determined externally by a valve controlled by dimensionless actuating variables.

- *Controls*

  The level of detail of the controls in BPS must be such that the control functions are suitable for use in the real building. Whether controller parameters, e.g. of PI controllers, can be taken over from the simulation is analyzed as part of the validation and discussion (chapter 5).

With respect to the availability of building models and model parameters, increased efficiency is expected from the deployment of BIM methods (see Section 1.5).

4.2.5 Standardization and proprietary models and controls

The development of controls in BPS environments in a design phase is associated with increased effort compared to current practice. In order to minimize the modeling effort, the use of standardized models and control macros of recurring subsystems as well as the integration of product-specific controllers and models must be aimed for. The overall goal is that HVAC engineers can concentrate on the optimal project-specific composition of HVAC systems. Although this is not the focus of this thesis, the following recommendations and comments can be made:

- *Modularization of HVAC subsystems*

  Standardized configurations for subsystems such as AHUs and heating or cooling systems should be used to keep the modelling effort low. For AHUs, Kümpel et al. (2022) presented modular models of frequently used hydronic circuits in the Modelica environment. For example, heat exchangers, pumps, valves and sensors for heating coils are modularized, which enables efficient configuration of project-specific AHUs. Further standardization of entire AHUs or heat and cooling generator systems is possible. An example of this could be the extensive library of systems contained in the Polysun software (Vela Solaris 2018).

- *Standardization of controls*

  The system-specific development of a suitable controller is associated with high effort. The heating and cooling plant in Figure 1.3 and the AHU in the factory building described in Section 3.3 are examples for particularly challenging control development tasks. Therefore, standardized control sequences, such as ASHRAE Guideline 36-2021 (Section 2.2.3), should be used for standardized subsystems. It should be noted that according to Wetter et al. (2022), the optimized control sequences in ASHRAE Guideline 36-2021 contain six to seven times more lines of code than base case controls. This underscores the importance of reusing existing and optimized control macros. The development of standardized controller macros and the corresponding standardized HVAC systems must be coordinated to ensure that both fit together (see comment in Section 2.2.3).

- *Proprietary controller and models*

  HVAC subsystems are increasingly equipped with embedded controls, often developed using subsystem plant models (see Section 2.2.3). These proprietary controls should be used because they have been developed specifically for the product. Accordingly, the associated system models should also be used in building energy modeling. In contrast to today's HIL or SIL applications, proprietary models and controls should be integrated into the simulation already in the design phase to avoid developing custom controls that will be replaced later. However, this requires that HVAC system manufacturers provide early access to these product-specific models and controls, for example through the Functional Mock-up Interface for embedded systems (eFMI) standard, to enable future integration with BPS tools.

The described approaches for modularization and standardization are not very common in the building industry today, at least for larger commercial buildings. It is unlikely that this will change in the near future. However, the need to standardize subsystems to enable modular design is generally recognized.

4.3 Connection of BPS tools and building controllers

Two approaches can be distinguished of operating actuators of HVAC systems according to control logic developed in BPS environments (see Figure 4.5): In the first case (Figure 4.5 a), the control logic is transferred from a BPS tool to an automation IDE for implementation and execution on a building controller. This transfer is a one-time action, and there is no permanent link between the BPS environment and the controller. In the second case (Figure 4.5 b), the control logic is executed in real time within the simulation environment to operate HVAC systems. This requires the simulation environment being permanently connected to the building controller via a communication protocol to read and write signals. These two options are described in detail and compared in the following sections.

4.3.1 Transfer of controls

The simplest example of control logic transfer is the use of functional diagrams according to the normative standard (Section 2.2.1). In Section 2.3.4, two options were introduced to digitally transfer controls developed in BPS to building controls: the toolchain with

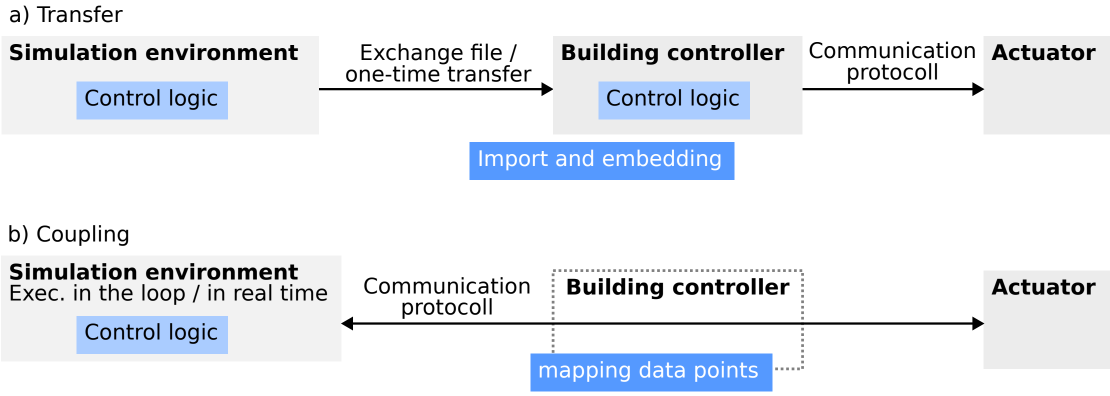


Figure 4.5: Options for the operation of HVAC systems according to control logic developed in BPS.

IDA ICE / Beremiz and the PLCopen XML (Sahlin, Skogqvist, and Högberg 2018) and the Modelica-based CDL (Wetter et al. 2022). These two options will be presented and compared in the following sections.

4.3.1.1 Modelica CDL

Within the framework of the OBC project, a toolchain for control development in Modelica-based simulation environments and an exchange format for the automatic implementation of control logic on building controllers was developed at the Lawrence Berkeley National Laboratory (LBNL) (Wetter et al. 2022). The major challenge in developing vendor-independent standards in the US market is that building controllers use vendor-specific programming standards. Standardized controllers, such as PLCs according to IEC 61131, are not common in building automation, as is the case in the EU market, for example.

The central element of the OBC project is the CDL (top in Figure 4.6). The CDL is a subset of the Modelica language and contains elementary function blocks (Wetter, Grahovac, and Hu 2018). These function blocks have a defined behavior, inputs and outputs, but their definition is language independent. This enables the implementation in various simulation and automation environments. The CDL function blocks are available as a Modelica library and can be used for simulation in Modelica-based environments (left in Figure 4.6). Control development is performed by assembling graphical function blocks. Using graphical function blocks can be a limitation, especially for large control programs or iterations, where textual programming may be more practical (Sahlin, Bring, and Eriksson 2009).

The control logic developed in Modelica can be replicated on CDL certified controllers (right in Figure 4.6). This requires the translation of the control specification in CDL into

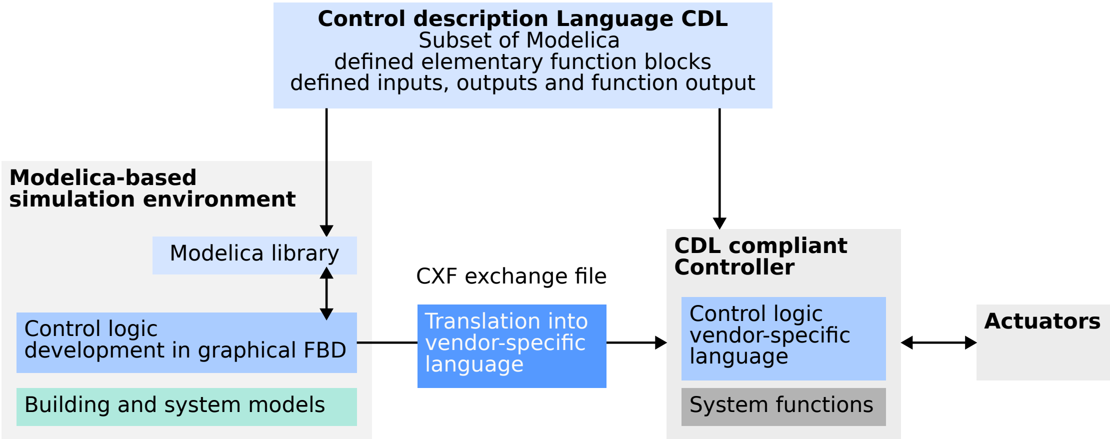


Figure 4.6: Transfer of control logic using CDL according to the OBC project.

vendor-specific formats (Wetter et al. 2021). Standardization of this process is ongoing work in ASHRAE 231P (Wetter et al. 2021; Wetter et al. 2022).

A limitation of this approach may be that the function blocks defined in CDL represent only a selection agreed upon by several control manufacturers (Wetter et al. 2022). Standardization of the function block library is also part of ASHRAE 231P (Wetter et al. 2022).

4.3.1.2 IDA ICE, Beremiz and PLCopen XML

Sahlin, Skogqvist, and Högberg (2018) have developed a toolchain using the open source IDE Beremiz for simulating PLC code according to IEC 61131-3 within the BPS tool IDA ICE. Beremiz (Beremiz 2016) is a complete open source IDE for PLCs based on the PLCopen editor, the MatIEC IEC to C compiler and its own runtime. Beremiz supports all five languages defined in IEC 61131-3. The PLCopen XML enables interoperability with other PLC IDEs.

In this toolchain, Beremiz and IDA ICE are linked as follows (see Figure 4.7): First, the control is developed in Beremiz using one of the five languages according to IEC 61131-3 (left in Figure 4.7, here using Function Block Diagrams). The control program is then compiled into C code using the MatPLC compiler (Sousa 2002) and a DLL (bottom left in Figure 4.7). Finally, this DLL is integrated in IDA ICE (right in Figure 4.7). For the simulation, the control component is connected to a sampling clock with the selected cycle time. The solver uses a multi-rate approach for efficiently simulating time-discrete, algorithmic code at sampling rates of real controllers in a continuous-time simulation environment (Sahlin, Bring, and Eriksson 2009). The whole workflow is currently not implemented as a productive feature in the IDA ICE application.

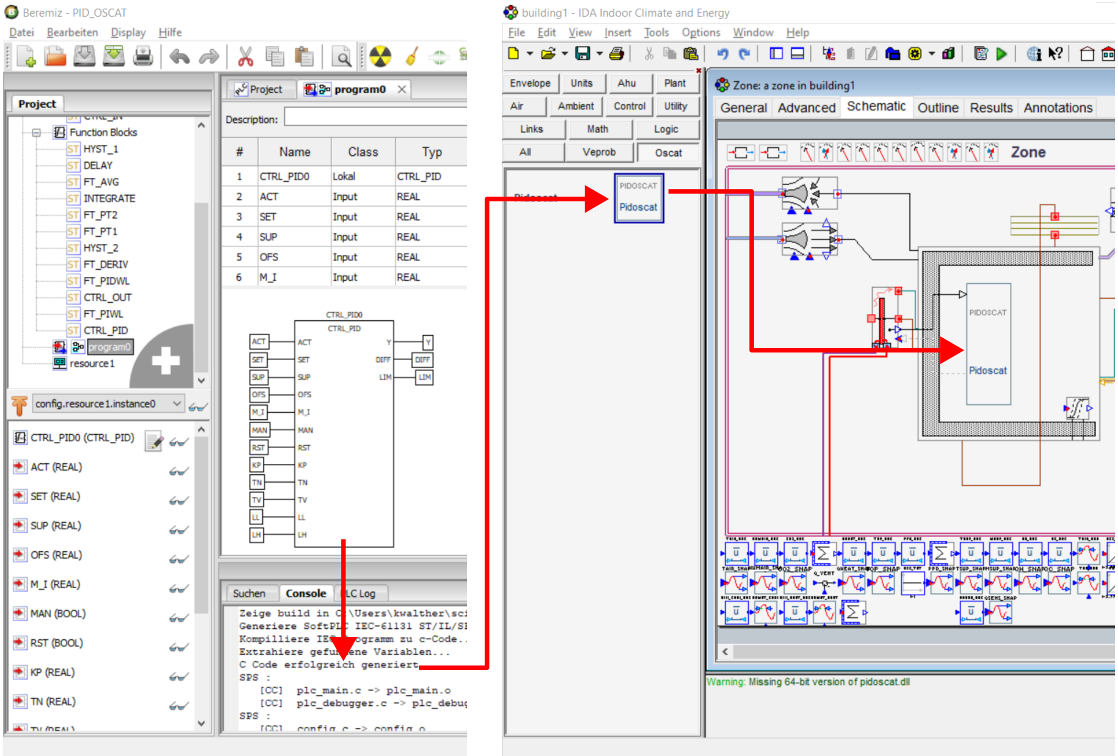


Figure 4.7: Integration of control blocks from Beremiz in IDA ICE

Beremiz projects are saved in a PLCopen XML (Section 2.2.4). This allows interoperability between Beremiz and other commercial IDEs that support the PLCopen standard. It is not possible to use Beremiz to directly program any commercial PLCs, since automation vendors typically bind their controllers to their own proprietary IDE.

Two options are generally conceivable to develop controls with IDA ICE / Beremiz and implement them on PLCs (Sahlin, Skogqvist, and Högberg 2018):

1. *Control development in IDA ICE (top in Figure 4.8)*

   A library of precompiled, time-discrete function blocks is integrated into IDA ICE, as shown for a PID controller in Figure 4.7. A simulation engineer assembles these graphical function blocks to develop the control logic. When the development process is complete, IDA ICE outputs a PLCopen XML that can be imported into a PLC IDE. Such a direct export of a PLCopen XML file from IDA ICE is not yet available.

2. *Control development in Beremiz (bottom in Figure 4.8)*

   The control program is completely developed in Beremiz using the programming features and the languages of IEC 61131-3. The control program is embedded as

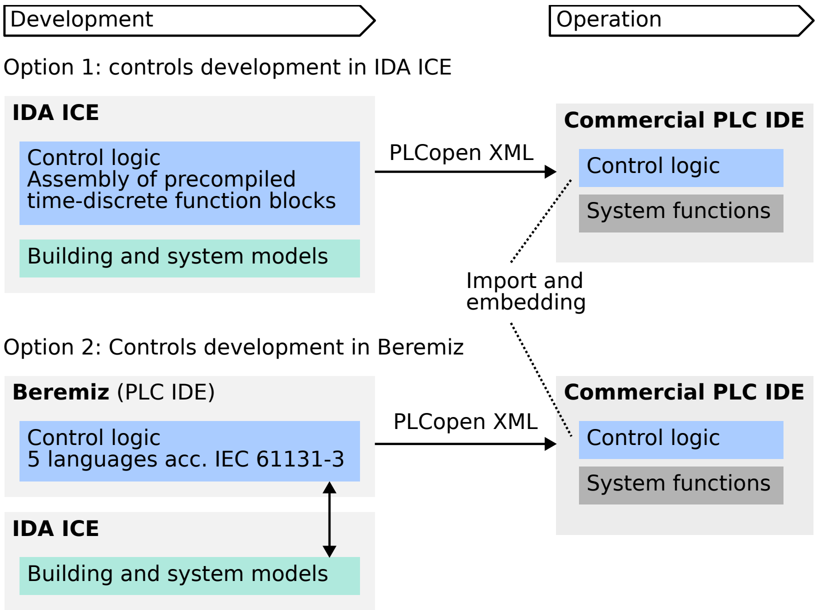


Figure 4.8: Options for controls development using Beremiz and IDA ICE.

one block for testing in IDA ICE. The PLCopen XML saved by Beremiz is used for implementation in commercial IDEs. This workflow is used for a validation case described in Section 5.1.5.2.

4.3.1.3 Comparison

The Modelica CDL and the IDA ICE / PLC approaches differ in several respects (Table 4.1): The main difference is the type and the nature of the exchange file. In the IDA ICE / PLC case, on the one side, executable control code according to IEC 61131-3 is exchanged via the PLCopen XML. On the other side, the CDL does not define executable code, but the behavior of control functions, their relation and parameterization. This aspect is closely related to different technological constraints: The PLCopen XML is an existing industrial automation standard (Section 2.2.4). In the EU market, PLCs programmed according to the IEC 61131-3 standard are commonly used for BA, although not all vendors follow this standard. In contrast, there was no such standardization for the US. Therefore, a new standard had to be developed with the CDL. Since automation vendors in the US use proprietary control languages, the CDL cannot of course contain executable code.

Table 4.1: Comparison of methods to transfer controls from BPS to BAS

	IDA ICE / PLC	**Modelica CDL**
Exchange format content	Executable code according to IEC 61131-3	Control specification, no executable code
Paradigm	Use of existing industry automation standard	Development of new standard based on simulation-oriented modeling language
Standard	IEC 61131-10	ASHRAE 231P
Origin / market	EU	US
Simulation environment	IDA ICE	Modelica-based
Main challenges	• Embedding IEC 61131-3 code in IDA ICE • Efficient simulation of discrete-time controls in continuous-time simulation environment	• Agreement on a common control library • BPS-BAS toolchain facing a heterogeneous automation market

4.3.2 Execution of controls in BPS environments in the loop

The previous section discussed approaches transferring control logic from a BPS environment to a building controller. Alternatively, the controller could be executed in the loop in a BPS environment during operation. From a technical point of view, it has to be noted that BPS tools are developed as tools for the design phase and not for control execution in operation. In particular, simulation solvers are designed to efficiently solve large systems of DAEs. The are “oversized” for the execution of comparatively simple control programs. Moreover, when using simulation solvers in the loop during operation, special care has to be taken in case of faulty or atypical measured values which can lead to numerical errors and aborted execution. Contrary, building controllers, such as PLCs, use robust hardware and software designed for the use in a building environment. Despite these constraints, this option is included here, first, because currently the PLCopen XML has a low prevalence and acceptance as an exchange format between PLC IDEs and thus the continued use of the same environment is an approach to bypass exchange formats and prevent issues resulting from the associated import and export (see experiences within the validation described in chapter 5 and Section 5.3.5.), and second, in the context of Digital Twins (see Section 4.4), this approach is an example for the use of one

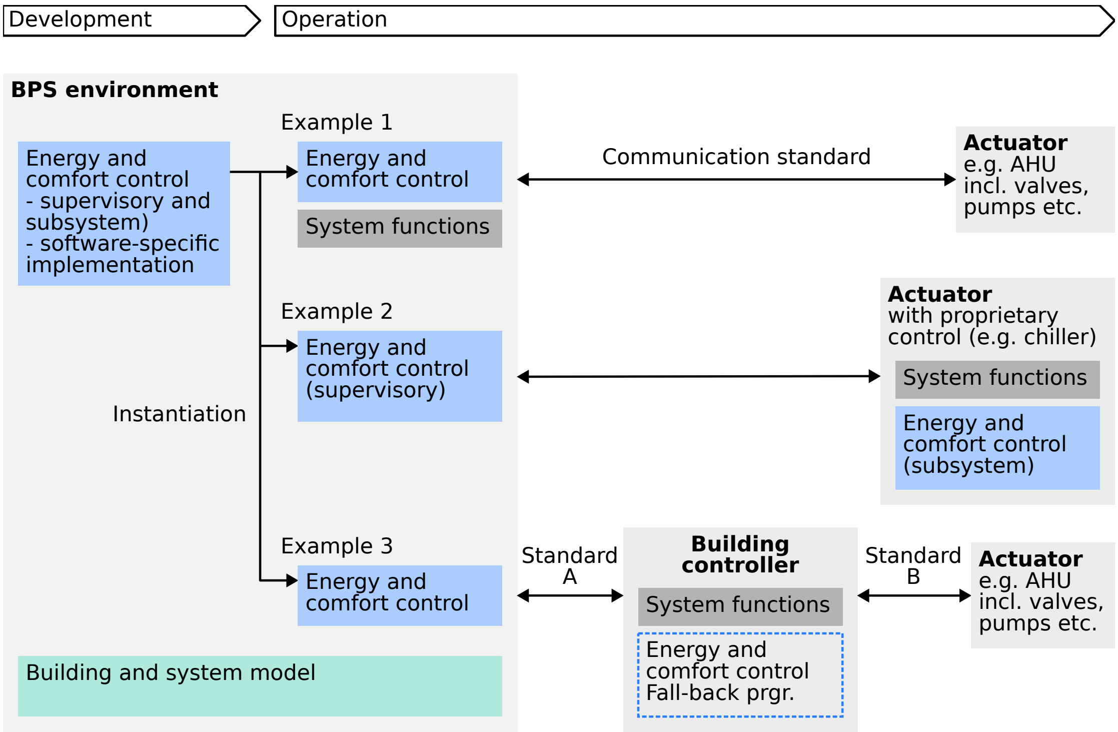


Figure 4.9: Connection of BPS environments and control actuators

environment for design and operation. Using BPS tools in the loop for the operation of HVAC systems generally requires:

1. A simulation environment that can read and write external signals and is capable of real-time simulation.

2. A hardware device on which the BPS tool can be operated, typically a server within the building IT network.

3. A communication standard supported by both the simulation environment and the BA device.

One example of such an approach is using the OPC UA interface of IDA ICE. The original purpose of this interface was to allow HIL testing of controls implemented on PLCs (Sahlin, Skogqvist, and Högberg 2018). However, the OPC UA interface can also be used to operate real HVAC systems as follows: The controller is first designed in a BPS environment (left part in 'BPS environment' in Figure 4.9). For operation, the controller is then instantiated, i.e. decoupled from the simulation models and connected to inputs and outputs of the real building (right part in 'BPS environment' in Figure 4.9). The controller in the simulation is no longer used to control the building and plant models, but to control the real buildings and plants. Several scenarios are possible for the connection between the BPS environment and control actuators (see right in Figure 4.9):

Example 1 The BPS environment sends control signals directly to field devices such as valves and pumps in AHUs. An additional building controller is not required. The main obstacle for this option is the heterogeneity of communication standards at the field level. Due to the high development effort, it is unlikely that a multiple of communication interfaces will be available in individual BPS environments. In addition, required system functions must be added to the controller in the BPS environment. This presents further challenges because the energy and comfort controls and system functions may be developed by different engineers.

Example 2 Similar to example 1, the BPS environment is connected to an HVAC subsystem with an embedded controller, such as chillers and heatpumps (Section 2.2.3). In this case, only the supervisory control would be executed in the BPS environment, while all subsystem control would be covered by the proprietary control.

Example 3 An intermediate building controller is used to coordinate communication between the BPS environment and the field level. This setup reflects the classic hierarchical BAS architectures commonly used today. Protocols such as Modbus or BACnet can be used between the building controller and the field level, and the OPC UA standard can be used for communication between the BPS tool and the building controllers. This option allows a clear separation between energy and comfort control and system functions: The energy and comfort control developed by a simulation engineer is run in the BPS environment. The system functions are added by an automation engineer. A simple fallback program for the energy and comfort control is implemented on the building controller in case of communication failure.

From a methodological point of view, the continued use of the BPS tool in operation has the following advantages:

- *Continuity of software environments and tools*

  The same software tool is used for both development and operations. This facilitates debugging and updates when development and operations are performed within the same engineering service (see Section 5.3.5).

- *Avoidance of exchange formats*

  An intermediate file exchange format, such as the PLCopen XML or CDL file is not required. This is an opportunity for building controllers which support communication standards, such as OPC UA, but whose associated IDE does not support the import of exchange formats for control sequences, such as the PLCopen XML.

- *Integration of existing building models for applications in operation*

  The building and system models can be simulated directly in parallel. This allows the models to be used as Digital Twin Instances, for example for model-based FDD (see Section 2.4.4 and Section 4.4).

- *Visualization*

  The visualization of building and plant operation as well as user interfaces (UIs) of BMS are important features to inform engineers and operators and enable the efficient management in operation. The continued use of visualizations, which have already been developed in simulation environments during the design phase, is an approach to reduce the overall effort.

4.4 Controls in Digital Twins during design and operation

The approaches described in Section 4.3 allow HVAC systems to operate according to controls developed and tested in BPS environments. These options are described below in the context of Digital Twins to illustrate the link between control development and model-based operational use cases.

4.4.1 Controls vs. building and plant models

The different nature of controls in contrast to buildings and systems has to be considered first in order to describe the role of controls in Digital Twins. On the one hand, real buildings and systems are modeled as so-called inexact models (Reddy 2011). The accuracy of these models is limited by the available knowledge about the underlying physical processes and the modeling details. Hence, a model will to a certain degree behave differently that the real object. On the other hand, control code can theoretically be applied identically to simulation models and real buildings. However, differences between the application in simulation environments and on real controllers can occur, when different computational methods are applied, e.g. resulting in different time steps (see Section 2.3.2).

Internal and external loads differ in the Digital Twin Prototype and the Digital Twin Instance as follows: In the Digital Twin Prototype (top left in Figure 4.10), internal and external loads are based on assumptions. Contrarily, in the Digital Twin Instance, internal and external loads are usually imposed as measured values from the Physical Twin (top right in Figure 4.10).

4.4.2 Controls in Digital Twins

The following describes and discusses the role of controllers in Digital Twins in design and operation, first for the transfer of control logic from BPS to building controllers, and second for the execution of controllers in BPS environments. In both cases, the control logic is developed on coupled building and system models within the Digital Twin Prototype (left in Figure 4.10).

Transfer of control logic from BPS on building controller

After the controller development is completed on the Digital Twin Prototype, the control logic is transferred to a building controller, for example via the PLCopen XML or CDL. As a result, the same control logic is implemented in the Digital Twin and the Physical Twin. Accordingly, there should no longer be any deviations between Digital Twin and Physical Twin caused by different controls. This procedure corresponds to the original Digital Twin paradigm (Grieves and Vickers 2017): First the development takes place on a Digital Twin, then the implementation in the Physical Twin. This also solves any problems associated with a reverse order: In Section 3.2, various methods of gathering information in the building were investigated in order to create Digital Twins. All of them had significant hurdles (Section 3.4). In the next step, the simulation models are connected to measured time series to create the Digital Twin Instance. Depending on the use case, this requires manual or automatic data import (see Section 2.4.3).

When comparing Digital Twin Instance and Physical Twin, a global level, a building and system level, and a control level are distinguished ("1" in Figure 4.10, Digital Twin Prototype and internal and external loads are omitted for the representation for the building/system and control levels). These levels help to illustrate different Digital Twin Instance use cases. A detailed assignment to specific use cases such as FDD or MPC is omitted here, but they are indicated by "comparison" and "feedback" in Figure 4.10 These levels differ as follows:

- *Global level*

  At the global level, the control logic and the building and system model are included in the Digital Twin Instance. This allows an integrated comparison of building,

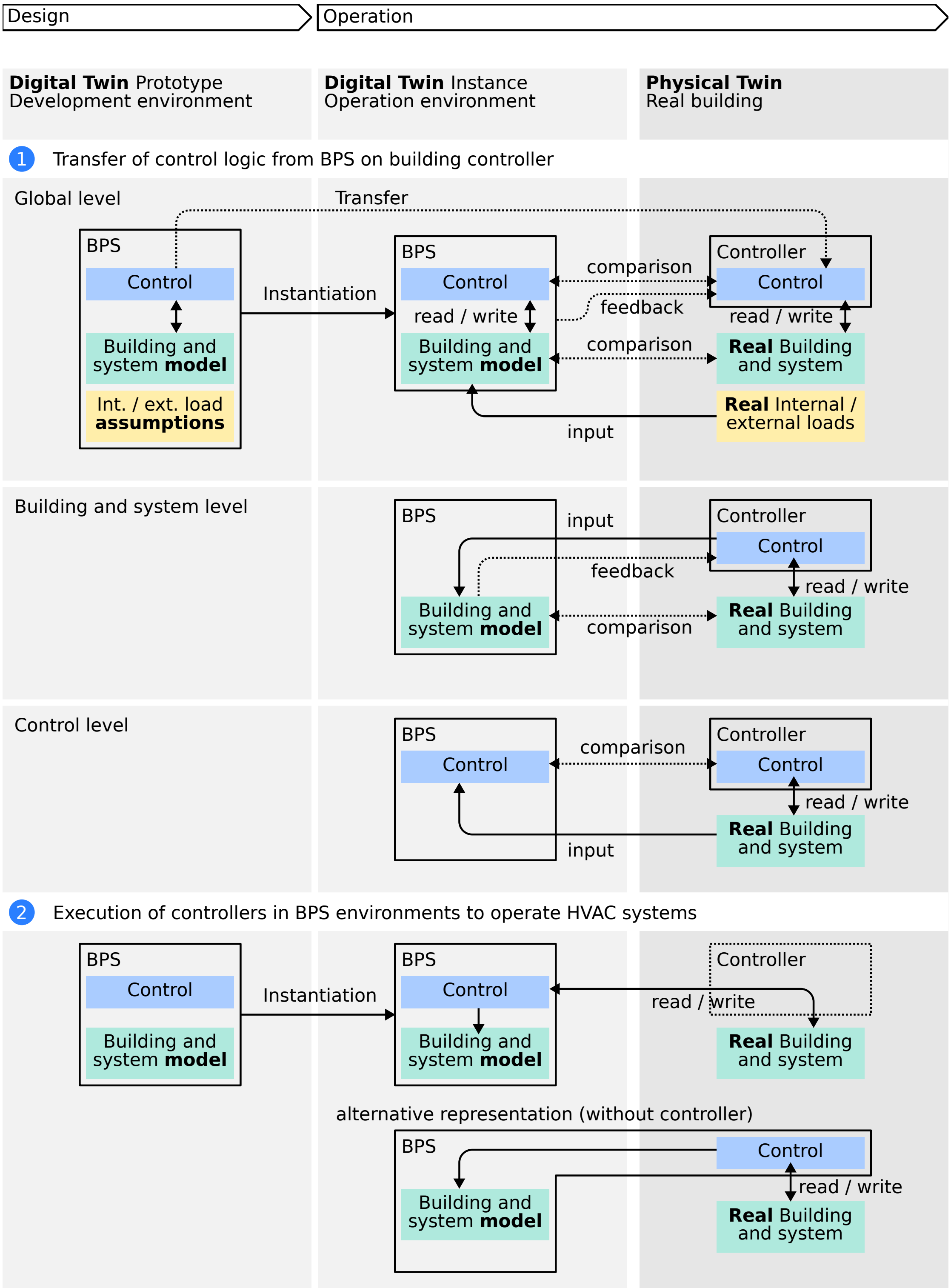


Figure 4.10: Controls in the Digital Twin Prototype, the Digital Twin Instance, and the Physical Twin

system, and control behavior. However, since the control is coupled both with the building and system models , effects on the building or system side cannot be evaluated separately.

- *Building and system level*

  When the control signals of the building controller are used as inputs for building and system models in the Digital Twin Instance, the effects of different control behavior are eliminated. This makes it possible to compare the behavior of the real building and systems with the Digital Twin Instance.

- *Control level*

  Depending on the type of the exchange file , i.e. for example the PLCopen XML containing executable code vs. the CDL CXF with a necessary conversion from CDL to a vendor specific language, and the calculation method (continuous or discrete time), the controller behaves differently in Digital Twin Instance and Physical Twin. The difference can be evaluated by imposing measured controller inputs on the controller models.

Discrepancies between Digital Twin Instance and physical building can also be caused by measurement errors and inaccuracies.

Execution of controllers in BPS environments

When the BPS environment is used to run the controller (Section 4.3.2), the Digital Twin Instance directly controls the Physical Twin (“2” in Figure 4.10). However, one could also argue that the BPS tool acts like the physical building controller, and thus is part of the Physical Twin (bottom in Figure 4.10). This ambiguity is related to the nature of the control logic described above. The analysis on different levels described above can be performed accordingly.

4.5 Summary

In this chapter, a three-step approach is proposed to bridge the gap between control development in the design phase and model-based use cases in operations. In contrast to current practice, the development and definition of energy and comfort-related controls take place in the design phase before the contract is awarded. The control logic is defined in such detail that the control only needs to be imported and supplemented with system functions during implementation by an executing company. Several aspects of control development and building energy modeling, such as modeling scope and modeling requirements, are described. The central aspect is the implementation of controls developed in BPS environments in buildings. Possible options were identified and compared. Finally, control development, implementation, and reuse of building, system, and control models are discussed in the context of Digital Twins.

5 Validation

In this chapter, the approach proposed in chapter 4 is applied to AHUs in a demonstration building. Three options are examined for how controls developed in a BPS environment can be implemented for the operation of the AHUs. For each option, the technical details and organizational approach are first explained in Section 5.1. Each of the three implementations is embedded in a Digital Twin framework to illustrate the opportunities for further use of models from the design phase in operations. In Section 5.2, the simulated and measured performance is compared in detail . Finally, the results and experiences are discussed in Section 5.3.

5.1 Methodology

5.1.1 Demonstration objects and systems

The validation is performed using the existing AHUs in the factory building at the Wilopark (Dortmund, Germany) described in Section 3.1.1. No changes were made to the physical system components as part of the validation. These particular AHUs were chosen for the following reasons:

1. The control of all internal components, such as heat exchangers and valves, was freely programmable (Section 3.1.1). No embedded manufacturer-specific proprietary control, which might not have been replicable in simulation, had to be considered.

2. The PLC on which the control is programmed follows the IEC 61131-3 standard and the IDE used for programming supports the import of the PLCopen XML. This enables the application of the toolchain with IDA ICE and Beremiz described in Section 4.3.1.2. In addition, the PLC has an OPC UA interface that allows coupling with IDA ICE (Section 4.3.2).

The demonstration cases are carried out in industrial zones, therefore aspects of user behavior, user influence and thermal comfort are different from, for example, office zones.

5.1.2 Demonstration design and implementation process

As part of the validation, the control logic developed in a BPS environment was implemented in three different ways. The original control logic (Section 3.2.3) was replaced for this revision. The validation was performed under real-life conditions, which is important for verifying the practical applicability of the proposed methodology. The development of the control and the implementation were carried out with separate responsibilities and services: while the development and testing of the control was part of the work for this thesis, the programming of the building controls and the management of the BAS were performed by an external contractor. This separation reflects the division of roles in both new construction and existing practical building projects (Section 1.2). The experiences and lessons learned from this organization are discussed in Section 5.3.

The three implementations were carried out in time sequence (Table 5.1) on the same AHUs as follows:

Implementation 1 *Transfer of control logic via graphical functional diagrams (Section 5.1.5.1)*

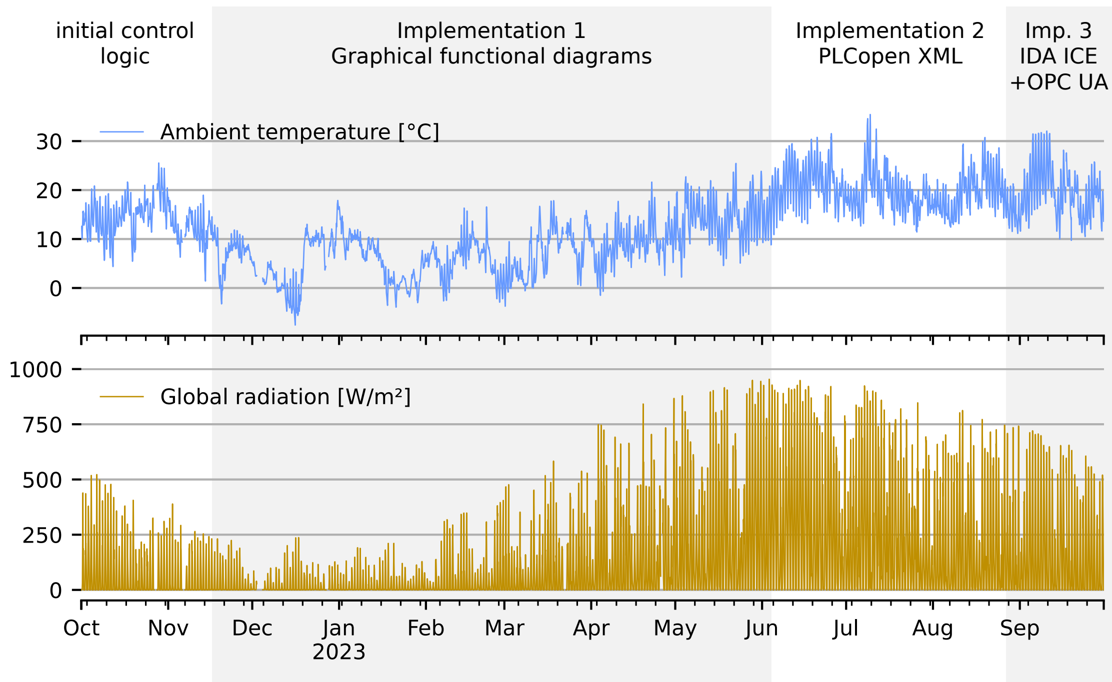
The goal of this implementation is to demonstrate the general usability of control logic developed in BPS environments. The implementation focuses on the transfer of the control strategy and the structure of the control. The control logic is developed in IDA ICE and implemented by the external contractor in PLC using the IDE e!cockpit. Functional diagrams as a graphical means of communication were used to communicate the control logic to the external contractor. Functional diagrams comply with the normative standard according to DIN EN ISO 16484-3:2005 (DIN 2005). Problems related to a non-digital format must be considered, but the usefulness of the control logic developed in BPS can be demonstrated even with imperfect communication. Control parameters were deliberately not communicated, and their appropriate determination was left to the external contractor. The implementation covers winter and spring conditions.

Implementation 2 *Transfer of control logic via the PLCopen XML (Section 5.1.5.2)*

Using the PLCopen XML as the digital exchange format eliminates all causes of performance gaps associated with faulty control logic communication. In this thesis, the Beremiz / IDA ICE tool chain with the PLCopen XML as exchange format is validated for the first time in a quasi-productive building application. The focus of the performance analysis is on the different control and system behavior. Unlike implementation 1, the control parameters were defined in the BPS environment and adopted by the executing company. The AHUs are operated during a summer period. Accordingly, the AHUs are in different operating modes compared to implementation 1.

Table 5.1: Overview of the implementations

	Implementation		
	1	2	3
Description	Section 5.1.5.1	Section 5.1.5.2	Section 5.1.5.3
Goal / motivation	Communication of control structure	Use of a digital exchange format	Continued use of BPS environment in operations
Environment for development	IDA ICE	Beremiz + IDA ICE	Beremiz + IDA ICE
...implementation	Wago e!cockpit	Wago e!cockpit	n/a
Operation software	Wago runtime	Wago runtime	IDA ICE (Windows)
...hardware	Wago PLC	Wago edge device	Building Server
Communication format design to implementation	Graphical functional diagrams	PLCopen XML	Direct coupling from IDA ICE using OPC UA
Setting of control parameters	Executing company	Simulation engineer (design)	Simulation engineer (design)
Start	November 15 2022	June 6 2023	August 24 2023
End	May 31 2023	August 13 2023	September 30 2023



Implementation 3 *Plant operation through IDA ICE and OPC UA (Section 5.1.5.3)*

Unlike implementation 1 and 2, the controller is not operated on a dedicated automation device, but directly through IDA ICE running on a server within the building IT network. The OPC UA standard is used for the communication between IDA ICE and the BAS. This means that both controller development and execution take place in a BPS environment. This workflow is also being validated for the first time in a quasi-productive building application as part of this thesis. The focus in this implementation is on embedding the workflow in a Digital Twin concept (see RQ 2.1 and RQ 2.2). Like implementation 2, this implementation operates under summer conditions.

The validation does not include the control transfer via CDL as a digital specification (Section 4.3.1.1). The CDL is still under development and not available for the PLCs implemented in the demonstration project.

The implementations are examined under the following aspects:

- General suitability of the controls developed and defined in BPS for the operation of AHUs
- Deviations between simulated (expected) and measured performance (see RQ 1.2)
- Analysis of the processes and the organization (see RQ 1.3)

5.1.3 Building energy and control modeling

5.1.3.1 Simulation environment

The BPS software IDA ICE (Section 2.3) was used for modeling and simulation within the validation. IDA ICE was chosen for the following reasons:

1. The software allows coupled simulation of thermal zones, systems and controls integrated in one software environment. Co-simulation with other software is not required.
2. Due to the underlying computational methods, IDA ICE has advantages in simulation speed compared to other equation-based simulation environments such as Modelica (Sahlin and Lebedev 2018). This is important because the control logic and associated HVAC systems and thermal zones must be simulated for a full year (Section 4.2.1).
3. Prototypical features have been developed to link controls in IDA ICE with BAS. This includes the toolchain with Beremiz, which allows the simulation control code according to IEC 61131-3 and to transfer controls via the PLCopen XML as well as the OPC UA interface (Sahlin, Skogqvist, and Högberg 2018).

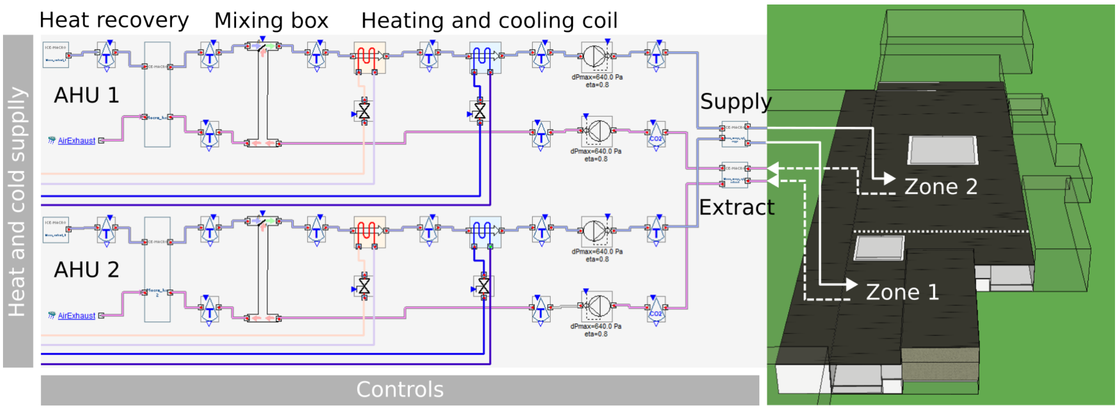


Figure 5.1: Model of AHUs and thermal zones in IDA ICE

5.1.3.2 Thermal zones

The two zones supplied by the two AHUs as well as adjacent zones were modeled (Figure 5.1). The zone model of IDA ICE is validated in several studies (Achermann 2000; Achermann and Zweifel 2003; Moosberger 2007; Loutzenhiser, Manz, and Maxwell 2007; Kropf and Zweifel, EQUA Simulation 2010a, 2010b). Measured time series for climate, TABS operation, and internal loads were imposed on the model with a resolution of 1 hour.

5.1.3.3 HVAC systems

The modeling of the two parallel AHUs is done using graphical blocks from the IDA ICE library. Each block represents an HVAC component, such as a valve, a pump, or a heat exchanger (Figure 5.1), which allows a 1:1 replication of the AHUs. The underlying equations can be examined. Several simplifications are applied:

1. Hydronic and ventilation systems are modeled as node models. 2-D or 3-D effects within ducts, such as temperature stratification, are neglected. The effects of sensor locations within ducts and pipes are not captured.

2. Hydronic and ventilation systems are modeled as massless. This means that the inertia of water in heat exchangers or pipes and air ducts is neglected.

3. Pressure drops due to pipe and duct friction and pipe and duct lengths are neglected.

4. Heat losses through pipes and ducts are neglected.

5. The behavior of components is simplified. For example, mass flow in pipes is determined by a component that calculates the flow based on the equation $\dot{m} =$

$\dot{m}_{max} \cdot y_{ctrl} + \dot{m}_{min} \cdot (1 - y_{ctrl})$, where $\dot{m}_{min}$ and $\dot{m}_{max}$ are the minimum and maximum flows, and y_{ctrl} is the actuating variable. This is a simplification, firstly because flow does not depend on pressure drop but on fixed flow limits, and secondly because the linear relationship between actuating variable and flow does not reflect typical valve characteristics.

6. The behavior of motors in actuators such as valves is omitted and thus response times are neglected. Between the actuating variables and the actuator inputs, first-order blocks with time constants on the order of 60 seconds are typically placed, however, the main goal is to ensure numerical stability.

7. All sensors are considered ideal and measurement uncertainties and noise are neglected.

8. Processing times of automation systems and ICT infrastructure are neglected.

These simplifications reflect the limited availability of information in the pre-award design phase, when details of the system layout may not be specified or available. More detailed modeling would have been possible, but would have required more effort and longer simulation times. The effect of these simplifications (RQ 1.2) is part of the performance gap analysis in Section 5.2.

5.1.3.4 Controls

For the controller models, function blocks from the IDA ICE library (Sahlin, Bring, and Eriksson 2009) were used in implementation 1. In implementations 2 and 3, control functions from the OSCAT library (Mühlbauer 2015a, 2015b) were used in Structured Text according to IEC 61131-3.

5.1.4 Control approach

To operate the AHUs, a rule-based control logic was developed. As in the original implementation (Section 3.2.3), the extract air temperature is the controlled variable. From this control signal, the target supply air temperature is calculated with other quantities, such as the current ambient temperature (Figure 5.2). The control approach differs from the initial one as follows:

- Instead of a fixed target value, a permitted extract air temperature range with a lower limit and an upper limit is defined.

- To increase efficiency, mixed air dampers and VAV boxes are used for demand controlled ventilation.

- The room-side extract air temperatures were used as controlled variables instead of the AHU-side extract air temperatures.
- Both AHUs are controlled by a common target supply air temperature. In the original implementation, their respective extract-side inlet temperatures were controlled independently (see Figure 5.2).
- In addition to the zonal extract air temperatures, the air quality was also taken into account as a controlled variable. For this purpose, temperature and air quality sensors for CO_2 and VOC (Volatile Organic Compounds) were retrofitted in the room-side extract air ducts.

The AHUs and VAV boxes are sequenced as follows (bottom in Figure 5.2): first, at minimum air volume, an attempt is made to meet the comfort criteria by varying the supply air temperatures and the proportion of fresh air within the AHUs. Only when this is no longer sufficient, the air flows are increased independently for each zone via the VAV boxes.

The general control approach remained unchanged in all implementations, but the control programs differed as follows:

Implementation 1 Based on the target supply air temperature and the actual mean supply air temperature, the actuators of each AHU receive identical actuating variables. In the sequence controller (`cSeq` in Figure 5.2) the mixed air damper is activated first, then the heat recovery and finally the heating coil or cooling coil. The control program is accessible in Section B.1.

Implementation 2 Compared to implementation 1, heat recovery is prioritized over mixed air dampers in the sequence controller (`cSeq` in Figure 5.2). The control program is accessible in Section B.2.

Implementation 3 Compared to implementation 1 and 2, separate actuating variables are used for each AHU to ensure similar supply air temperatures. However, the cascade controller `cCasc` still outputs a common target supply air temperature for both AHUs. The control program is accessible in Section B.3.

In general, a pragmatic approach to the development of controllers has been of paramount importance. In particular, the determination of the parameters of the PID controllers leaves room for optimization: The external contractor determined the parameters based on experience in implementation 1. The parameters were determined after variation and testing, but without systematic optimization, in implementations 2

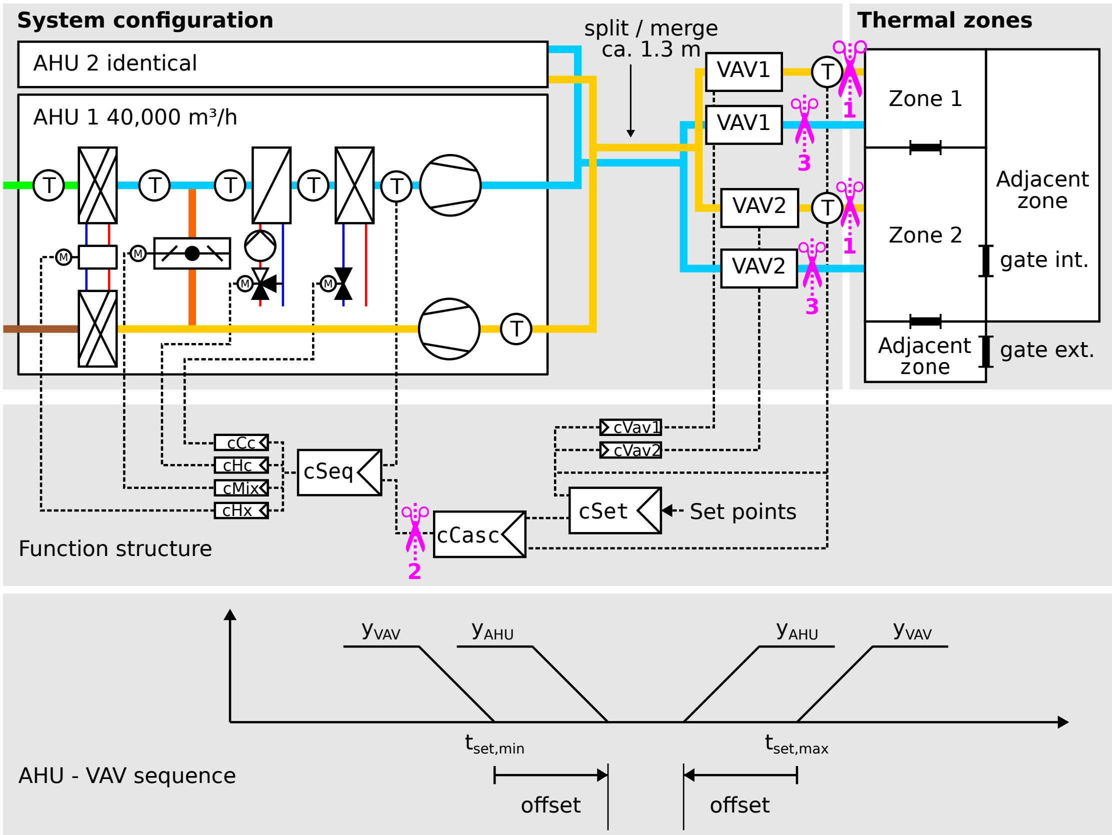


Figure 5.2: Top: Schema of AHUs and zones. Center: functional diagram of the control logic. Bottom: sequence between AHUs and VAV boxes for the heating case. Analog behavior for cooling.

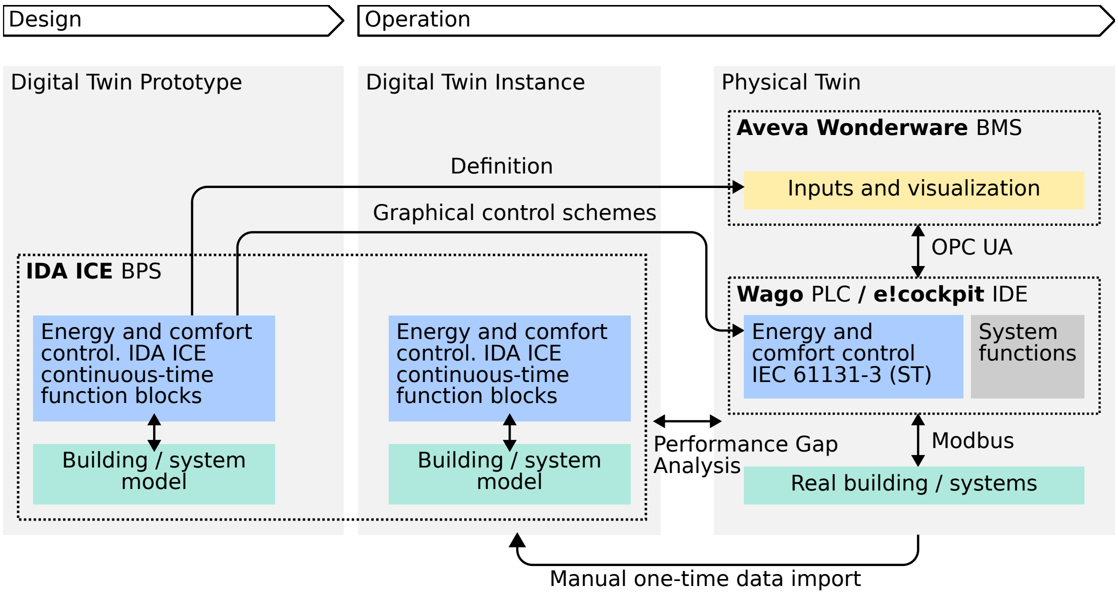


Figure 5.3: Implementation 1: Workflow in a Digital Twins framework

and 3. A further possible enhancement to the control program, which is not implemented, would be shutting down a unit during base load conditions.

The functional requirements were defined by the building operator in terms of minimum and maximum indoor temperatures and maximum indoor CO_2 concentration, as well as minimum and maximum supply air temperatures.

5.1.5 Workflow and Digital Twins in design and operation

In the following, the technical and methodological details of each implementation will be presented. The general approach is that Digital Twins of buildings, systems, and controls are used both during design and operation (Section 2.4.4). In the design phase, the control development is performed on coupled building and system models in the Digital Twin Prototype. In the operation phase, the behavior of the Digital Twin Instance is compared with the Physical Twin.

5.1.5.1 Transferring control logic via functional diagrams

In implementation 1, the control logic was developed by assembling graphical function blocks from the native IDA ICE library (Digital Twin Prototype, left in Figure 5.3). The final control has a total of 77 function blocks.

The control logic was handed over to an external contractor for implementation in the PLC in the form of screenshots of the function structure and a report including a textual description (arrows in the middle of Figure 5.3). The level of detail corresponds to the functional diagrams according to DIN EN ISO 16484-3:2005 and VDI 3814-4.3:2022. PI controller parameters were deliberately not specified, but left to the automation contractor.

The executing contractor translated these functional diagrams into Structured Text according to IEC 61131-3 using the PLC IDE e!cockpit. In the process, the 77 function block were translated into 725 lines of code (Physical Twin, right in Figure 5.3). Since only the energy and comfort controls were included in the development, the system functions were added by the contractor.

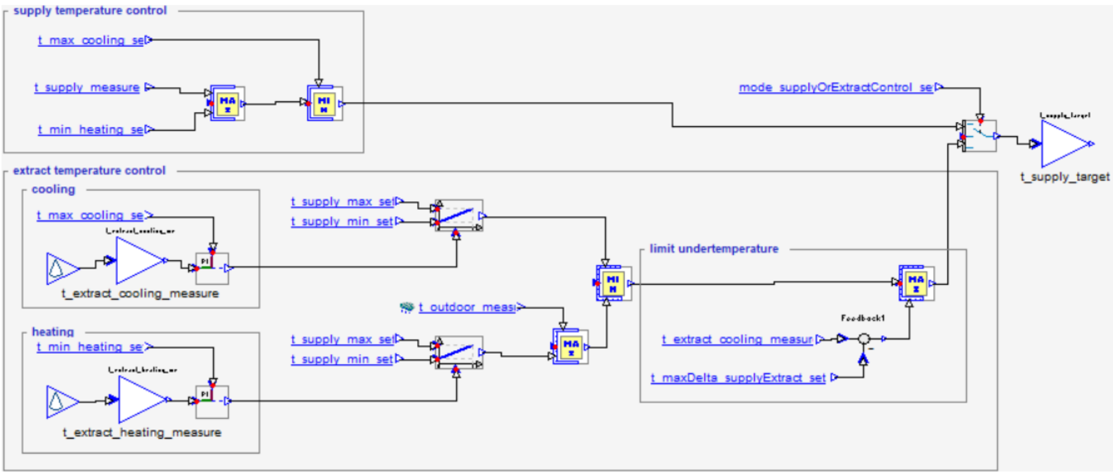
As an example, Figure 5.4 shows the translation of a cascade controller for determining the target supply air temperature (`cCasc` in Figure 5.2). For this simple scheme, direct traceability between functional diagrams and control code is still possible, for example for a review. However, comparing Function Block Diagrams and Structured Text for the entire control program is extremely time-consuming. Another problem for the traceability of the implementation is that the behavior of the function blocks in the automation environment is only documented textually (WAGO 2022). The programmatic implementation and the formulas are not visible or not available.

For the performance gap analysis, measured time series from the Physical Twin for climate, internal loads and TABS operation were manually imported into the Digital Twin Instance and imposed on the models. The comparison is done at building, system and control level (Section 5.2).

5.1.5.2 Transferring control logic via PLCopen XML

Implementation 2 used the toolchain described in Section 4.3.1.2 with IDA ICE as BPS software and Beremiz as IDE for PLC code. In this toolchain, the control program was developed in Beremiz (top left in Figure 5.5). In the Beremiz editor, the whole control program is written in Structured Text according to IEC 61131-3 (option 2 in Figure 4.8). For programming the open source library OSCAT BASIC (Mühlbauer 2015a) was imported into Beremiz. The function blocks available in the OSCAT library proved to be sufficient for the development of the control program. In an iterative process between Beremiz and IDA ICE, the control logic was developed and tested in the Digital Twin Prototype (left part in Figure 5.5 and Figure 5.6). The parameters of the PID controllers were determined pragmatically, but not systematically optimized.

According to the sampling rate of the real PLC, the simulation in IDA ICE was done with a cycle time of 100 milliseconds. During the development process, numerical instabilities were encountered in certain combinations of controller parameters and cycle




```
PIDcooling.rActualValue :=parameters.rT_ExtractCooling_Measure;
PIDcooling.rReferenceValue :=rFB_T_max_cooling_set;

FBPID_t_supply_target_cooling(
        rReferenceValue := PIDcooling.rReferenceValue,
        rActualValue := PIDcooling.rActualValue,
        typConfigParameters := PIDcooling.typConfigParameters,
        rY => PIDcooling.rY);

PIDheating.rActualValue :=parameters.rT_ExtractHeating_Measure;
PIDheating.rReferenceValue :=rFB_T_min_heating_set;

FBPID_t_supply_target_heating(
        rReferenceValue := PIDheating.rReferenceValue,
        rActualValue := PIDheating.rActualValue,
        typConfigParameters := PIDheating.typConfigParameters,
        rY => PIDheating.rY);

IF parameters.iMode_SupplyOrExtractControl_Set = 1 THEN
        parameters.rT_Supply_Target := MIN(MAX(parameters.
            ↪ rT_Supply_Measure,rFB_T_min_heating_set),
            ↪ rFB_T_max_cooling_set);
ELSIF parameters.iMode_SupplyOrExtractControl_Set = 2 THEN
        aFBLin_Trafo[1](
                IN := PIDcooling.rY,
                OUT_MIN := parameters.rT_supply_min_set,
                OUT_MAX := parameters.rT_supply_max_set);
        aFBLin_Trafo[2](
                IN := PIDheating.rY,
                OUT_MIN := parameters.rT_supply_min_set,
                OUT_MAX := parameters.rT_supply_max_set);
        parameters.rT_Supply_Target := MIN(
        MAX(aFBLin_Trafo[1].OUT,parameters.rT_ExtractCooling_Measure-
            ↪ parameters.rT_maxDelta_supplyExtract_set),
        MAX(aFBLin_Trafo[2].OUT,parameters.rT_Outdoor_Measure));
END_IF
```

Figure 5.4: Top: Functional diagram of a cascade controller in IDA ICE (design). Bottom: ST according to IEC 61131-3 (implementation).

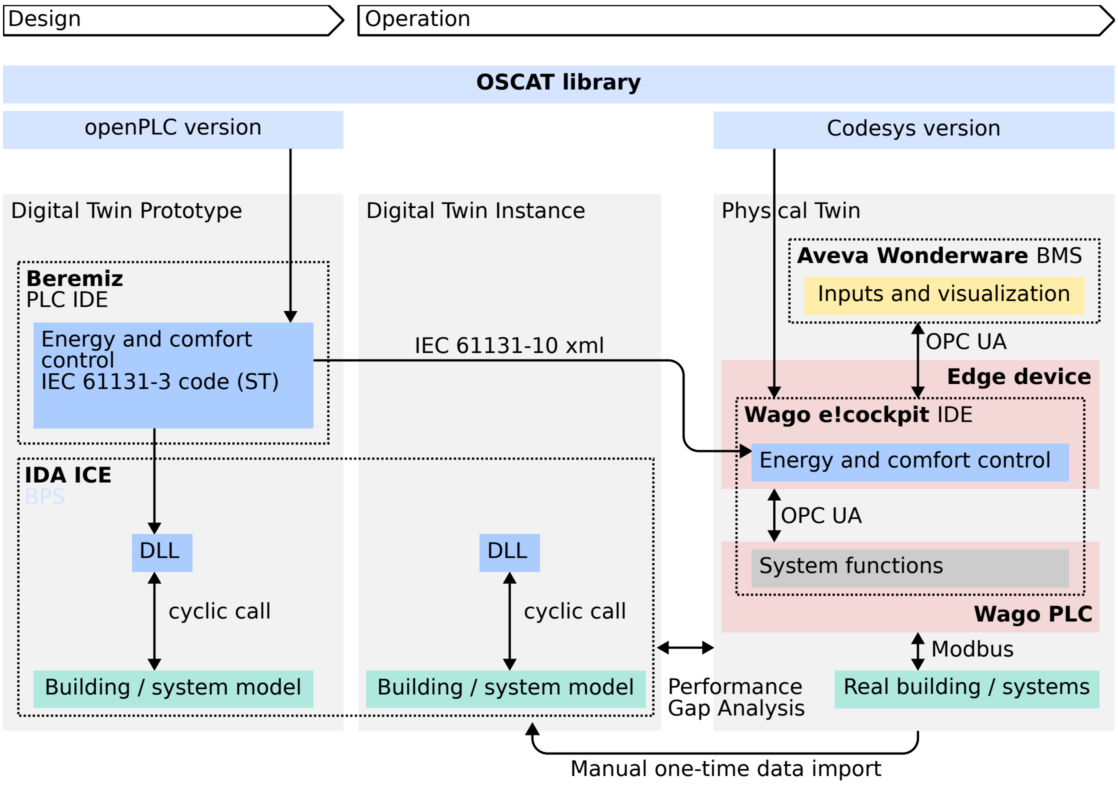


Figure 5.5: Implementation 2: Workflow in a Digital Twins framework

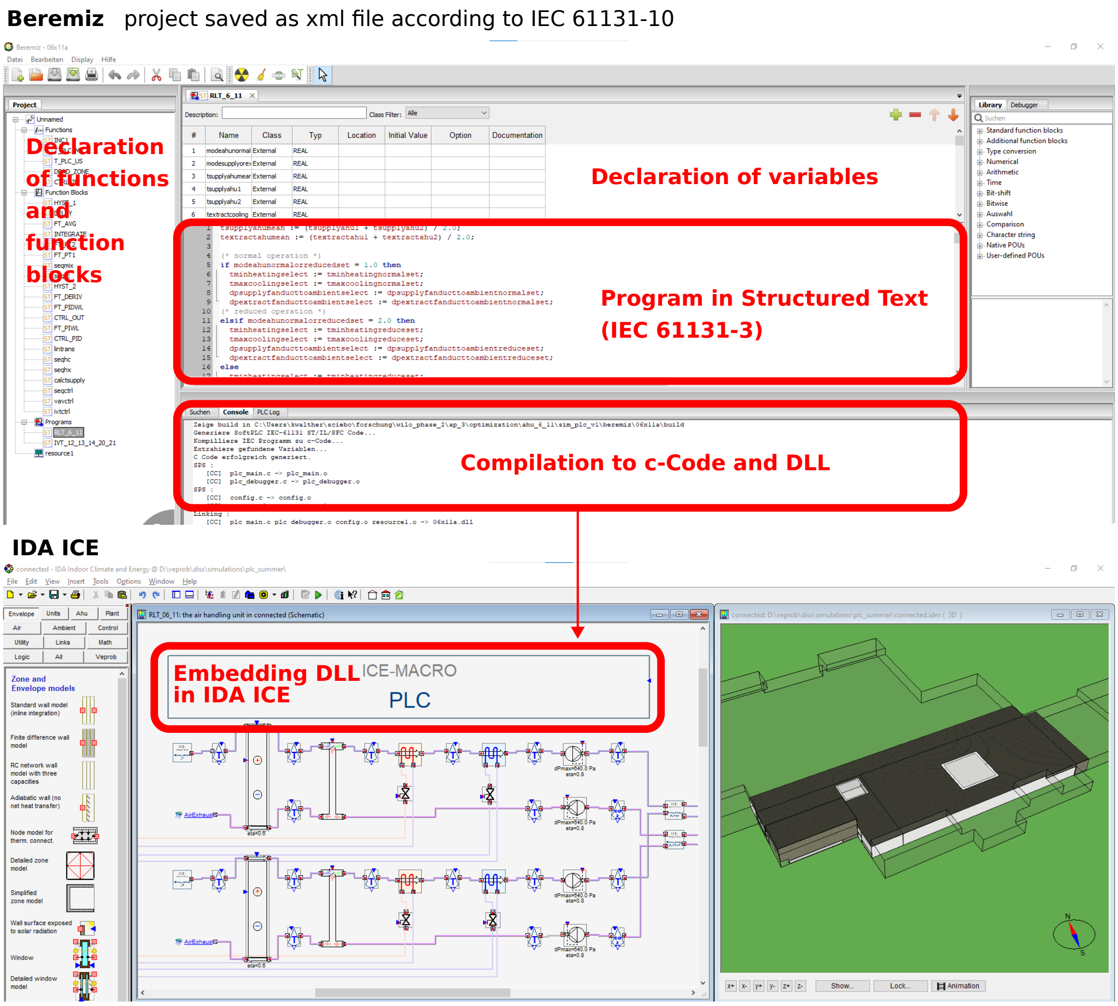


Figure 5.6: Control development in Beremiz and integration in IDA ICE

times. It is assumed that these numerical problems are related to the novel underlying numerical method.

As a result of the development process, the PLCopen XML created by Beremiz was handed over to the external contractor for integration in the e!cockpit IDE (right in Figure 5.5). System functions were added by the external contractor. In contrast to implementation 1, where the control parameters were intentionally not communicated, they were implemented according to the specifications from the simulation. With the following two exceptions the import into e!cockpit worked smoothly:

- All parameters, including PI controller parameters, had to be set manually because they were omitted during the import. Incorrectly entered parameters had to be identified and corrected manually.
- The function to determine the PLC runtime had to be adapted manually. This function is required for the calculation of the integral part in PID controllers and differs between the PLCopen implementation and the Codesys implementation of the OSCAT library (top in Figure 5.5).

During commissioning, individual errors, such as missing bindings of control variables to actuators, were quickly corrected. In addition, the fan control had to be adapted to the dynamics of the newly implemented variable volume flows.

In preparation for implementation 3, the imported energy and comfort control program was not executed on the PLC, but on a separate edge device. Both program instances were programmed in the same IDE (e!cockpit). The OPC UA standard is used for communication between the edge device and the PLC. In general, this setup allows to separate the energy and comfort control engineering from the system functions when programming and also in operation.

As in implementation 1, measured time series from the Physical Twin for climate, internal loads, and TABS operation were manually imported into the Digital Twin Instance and imposed on the models for performance gap analyses.

5.1.5.3 Control execution with IDA ICE and OPC UA

In implementation 3, the same toolchain as in implementation 2 was used to develop the control logic, including IDA ICE and Beremiz (Digital Twin Prototype, left in Figure 5.7). For the operation, IDA ICE including the local project file was first installed on a server within the building IT network (center in Figure 5.7). The inputs and outputs of the control program were then connected to the PLC with import and export modules in IDA ICE (left scheme in Figure 5.8). For the communication between IDA ICE and

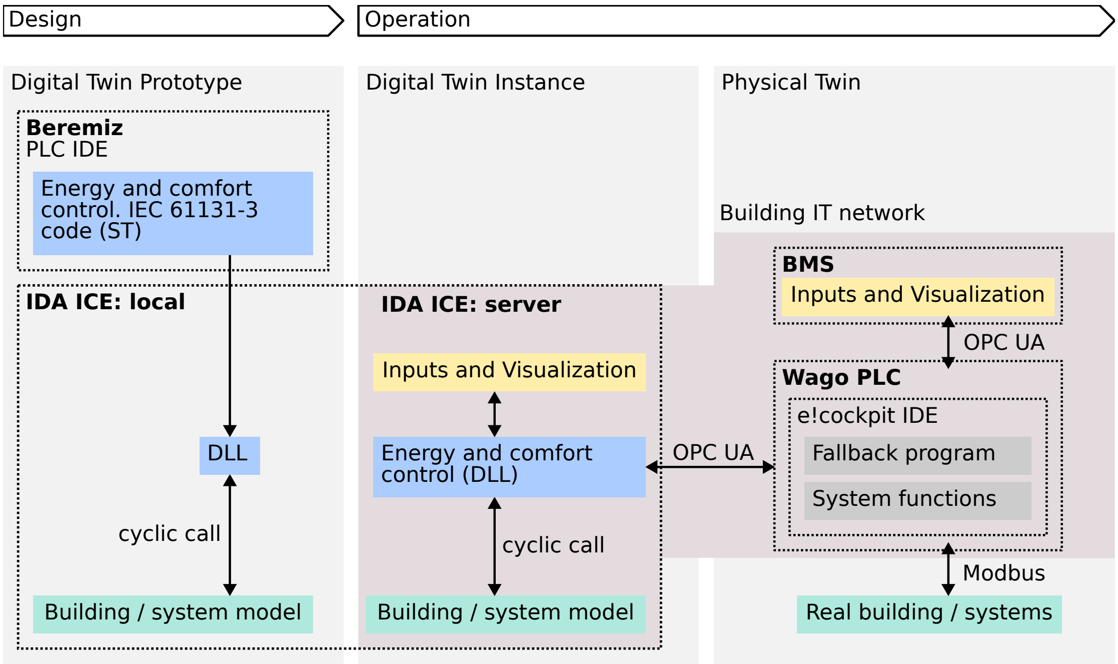


Figure 5.7: Implementation 3: Workflow in a Digital Twins framework

PLC the OPC UA standard was used. It would also have been possible to use the native function blocks from IDA ICE directly instead of the IDA ICE / Beremiz toolchain. However, as mentioned in Section 4.3.1.2, especially for larger programs, programming in Structured Text is more practical than using graphical Function Block Diagrams.

The control execution startup is as follows: First, the simulation is started and a connection to the OPC UA server is established. A function within PLC detects if signals are received from IDA ICE. If so, the input signals from IDA ICE are mapped to the appropriate actuators. The controller within IDA ICE sends control signals for the liquid heat recovery, the mixed air damper, the heating coil and the cooling coil of each AHU, actuating variables for the VAV boxes. The input and output values have no applied filters nor limiters. However, pre- and postprocessing might be needed in a productive application to avoid numerical issues caused by erroneous input values. The control program also outputs setpoints for the fan pressure differential based on a time program, but the fan control is part of the system functions programmed separately on the PLC and handled by the external contractor. The PLC automatically detects if no values are received via a pulse signal. As soon as the connection is detected as interrupted, a fallback program implemented on the PLC is executed. The monitor in Figure 5.8 (right) shows calculated target and measured supply air temperatures during live operation.

The building and system models within the Digital Twin Instance (center in Figure 5.7) were computed at each time step, allowing live visualization of the building and

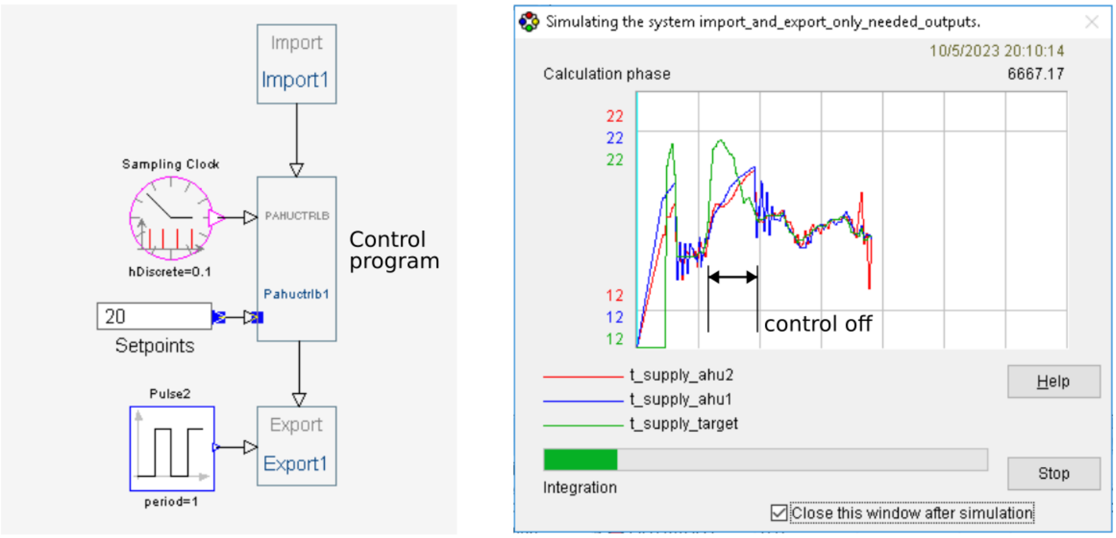


Figure 5.8: Implementation 3: Simulation monitor in IDA ICE during operation in real time

system models. For further development of a Digital Twin Environment, the separate visualizations in IDA ICE and the existing BMS could be merged.

5.2 Simulated and measured performance

The comparison of simulated and measured performance is done on three levels:

1. A separate analysis of AHUs and control to evaluate in particular the interaction between control and system behavior (Section 5.2.1).
2. A separate analysis of the thermal zones to evaluate the accuracy of the zone models (Section 5.2.2).
3. An integrated analysis of thermal zones, AHUs, and controls (Section 5.2.3). The goal is to evaluate the remaining performance gap when the same control is used in the simulation and on the real controller.

For the separate analyses, the simulation model is divided into submodels (see scissors symbols in Figure 5.2) to which measured variables are applied as input time series. Inaccuracies, uncertainties, and installation effects must be taken into account in the measurements, however, these aspects are not part of this analysis. For heat flows, a separate study was conducted to compare calibrated heat flow meters and pumps with heat flow measurement functionality (Walther and Voss 2021).

In the diagrams, the terms "SIM" (simulated) and "MEAS" (measured) are used to differentiate one from the other. The term "MEAS" is also used to denote control signals and actuating variables, which are calculated outputs of the respective control programs executed for operation, and are therefore recorded rather than measured.

5.2.1 Air handling units and control

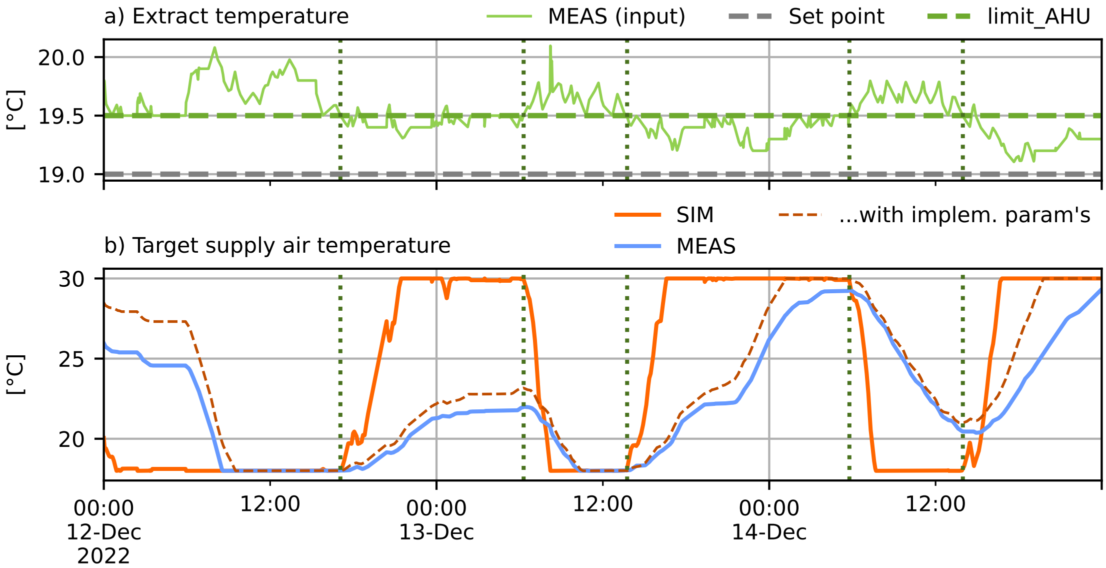
To compare the simulated and implemented controller and the behavior of the AHUs without including the effects of the thermal zones, the zone models are decoupled and measured values are imposed on the model instead (cut 1 in Figure 5.2).

Implementation 1 and implementation 2 are compared in detail to highlight differences related to the nature of the information transfer. In implementations 2 and 3, the identical IEC 61131-3 control code is implemented in the simulation and used for real operation respectively. Only the execution environment differs: While the control code is executed on a PLC in implementation 2, it is executed by the IDA ICE solver in implementation 3. For this reason, only the measured control performance in implementation 3 is briefly presented to demonstrate the successful use of IDA ICE for control purposes in real operation.

Implementation 1

The first step is to compare the target supply air temperature as output of the cascade controller `cCasc` (Figure 5.2) in the simulation and the building controller. Figure 5.9 a) shows the extract air temperature setpoint defined by the operator and the measured extract air temperature time series. Figure 5.9 b) exhibits that the simulated target supply air temperature rises and falls much faster than the measured one. The reason is a more aggressive parameterization of the controller in the simulation, which can be found in the table below Figure 5.9: The proportional gain (simulation: 0.3, implementation: 1.0) and the integration time (simulation: 300 s, implementation: 60 s) differ significantly with respect to the maximum output signal (simulation: 1, implementation: 100). Correspondingly, the difference in the target supply air temperature would result in significantly higher heat consumption in the simulation. If the controller in the simulation is provided with the implemented parameters, a much higher agreement (dashed line) is obtained.

In a second step, the different behavior of the AHUs is evaluated by comparing the simulated and measured supply air temperature. In order to exclude the previously described discrepancies in the calculation of the target supply air temperature, the target supply air temperature of the real controller is imposed on the simulation controller (cut 2 in Figure 5.2).



	Simulation	Implementation PLC
Minimum output	0	0
Maximum output	1	100
Proportional gain	0.3	1
Integration / reset time	300 s	60 s

Figure 5.9: Implementation 1: Comparison of the target supply air temperature as output of the real and simulated controller and parameters in simulation and implementation.

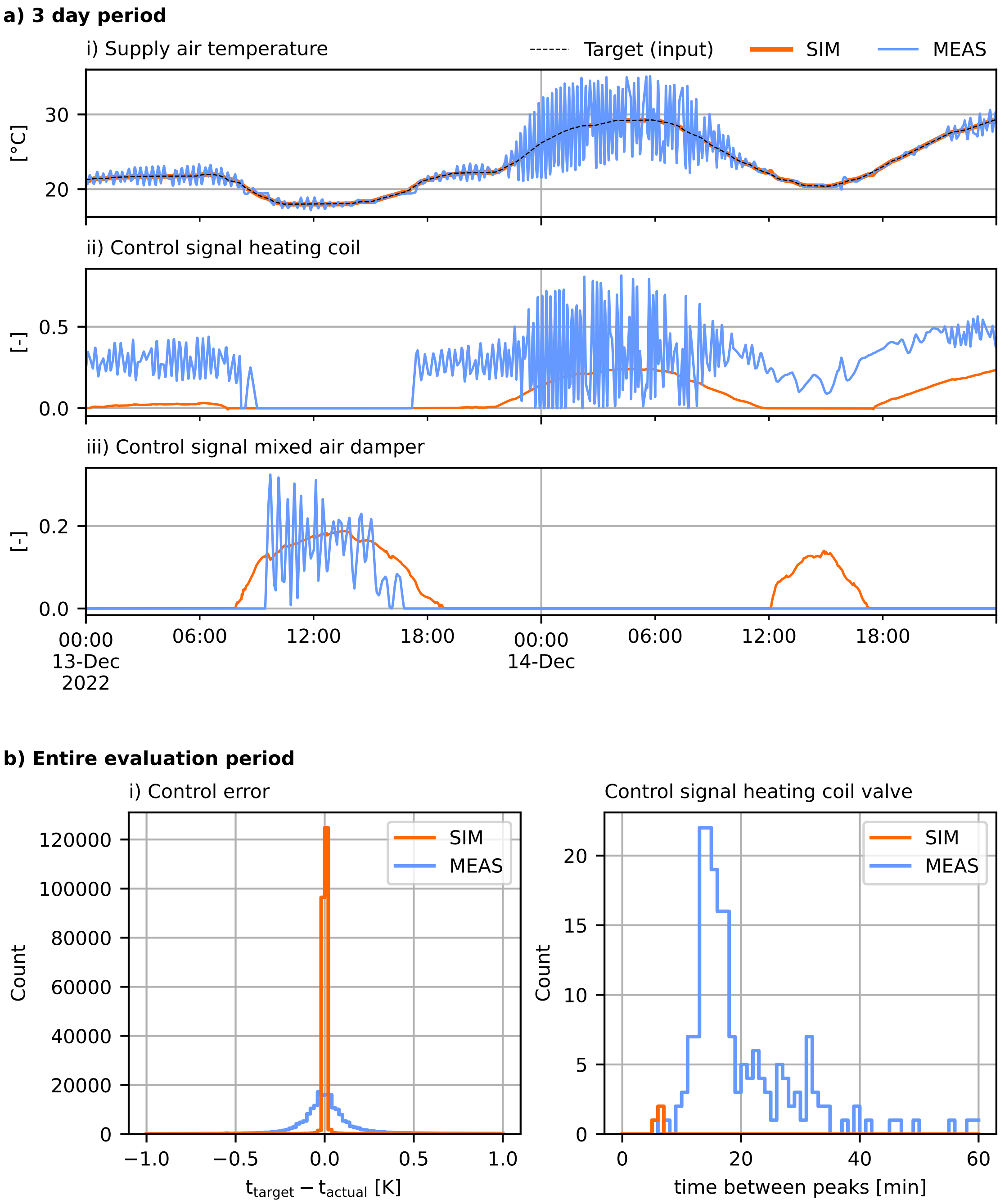


Figure 5.10: Implementation 1: Comparison of controller behavior

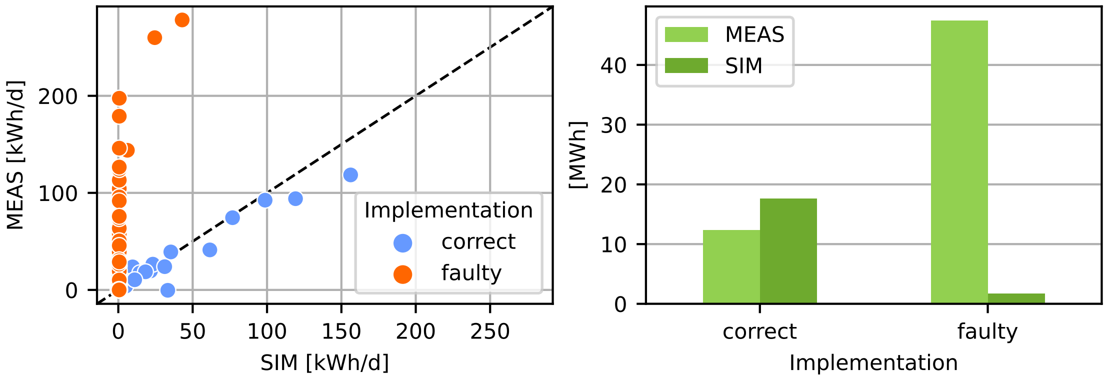


Figure 5.11: Implementation 1: Comparison of daily sums of simulated and measured heat for the heating coil.

Figure 5.10 a i) shows that the simulated supply air temperature exactly follows the target ("target (input)"), while the measured supply air temperature oscillates. The evaluation of the control error for the whole period in Figure 5.10 b) reveals that the control error $t_{target} - t_{actual}$ is almost 0 in the simulated operation, while it shows a much wider distribution in the measured operation. The cause of the oscillations is the unstable control of the heating coil valves and the mixed air dampers as shown in Figure 5.10 a ii). These observations illustrate that the control parameters have not been properly adapted to the system behavior by the external contractor.

A simplified approach to detecting and quantify oscillatory behavior is through peak detection. The number of peaks within a defined observation period and the time between peaks allow to characterize the robustness (Maghnie et al. 2022). The `find_peaks` function from the `scipy` library is used for this purpose[1]. The histogram in Figure 5.10 b) ii) indicates that most of the peaks in the control signal time series for the heating coil valve are between 10 and 20 minutes apart. The peak count over the entire observation period is relatively small compared to the 3 day period shown in Figure 5.10 a) because the heating coils are rarely activated.

Due to an implementation error, the heat recovery was not activated as specified in all operating modes. As a result, the heating demand increases significantly (Figure 5.11). This example illustrates that the use of function graphs as a graphical means of communication remains error-prone. However, during the periods of correct operation, significant savings were achieved compared to the initial operation (see discussion in Section 5.3.6).

By specifying controller parameters and performing a detailed review of the program code in the implementation, deviations between simulation and measurement could be reduced and the quality of the implementation improved. However, this would also

1. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

increase the effort, especially for code verification, which was already very high in the described control documentation process. In essence, the results and experiences underline the general usefulness and applicability of the control logic developed and tested in BPS environments for the operation of the AHUs. However, the limitation and error-proneness of graphical representations of control logic is a major obstacle to efficient and successful implementation.

Implementation 2

By using the PLCopen XML, errors due to incorrect implementation, as observed in implementation 1, are eliminated. For the performance comparison in implementation 2, special attention was paid to the format of the time series used as input to the simulation model. Time series with the raw time steps or even as discrete values in the change-of-value format were used, to allow an unbiased comparison at the controller and system level. Using averaged hourly inputs would have biased the simulated controller outputs.

Figure 5.12 illustrates the general controller behavior for a 24 h summer period. Figure 5.12 a) shows the maximum extract air temperature of both zones ("extract"), which is the controlled variable in the cooling case. The maximum extract air temperature is set by the operator to 24 °C which corresponds to the threshold for the VAV boxes ("limit VAV", see Figure 5.2). The control program defines the setpoint for the AHUs 1 K lower at 23 °C ("limit AHU").

A PI controller, which is part of the cascade controller (`cCasc` in Figure 5.2), compares the measured extract air temperature with the threshold ("limit AHU"). Figure 5.12 b) illustrates that the trajectories are very similar, but the simulated controller signal decays a bit faster. The differences are most likely due to small computational differences, such as step size processing. The simulated target supply air temperature decreases accordingly and reaches the minimum value of 16 °C a little earlier than the measured values. Both simulated and measured supply air temperatures generally follow the target supply air temperature (Figure 5.12 d). Details are evaluated further below.

The control signal for the mixed air dampers toggles between minimum and maximum position (Figure 5.12 e). If the ambient temperature is below the extract temperature (Figure 5.12 a), 100 % fresh air rate is used. In the opposite case, recirculated air is preferred with a minimum position of 10 % fresh air. The control signal for the cooling coil valve in Figure 5.12 f) has a similar behavior, although the valve model is highly simplified (Section 5.1.3.3). The air volume is increased by the VAV boxes when the limit of the extract air temperature is exceeded (see Figure 5.12 a). As illustrated in Figure 5.12 g), the trajectories of the actuating variable for the VAV boxes are also similar. The steps at 12:30 and 22:30 are likely due to numerical effects.

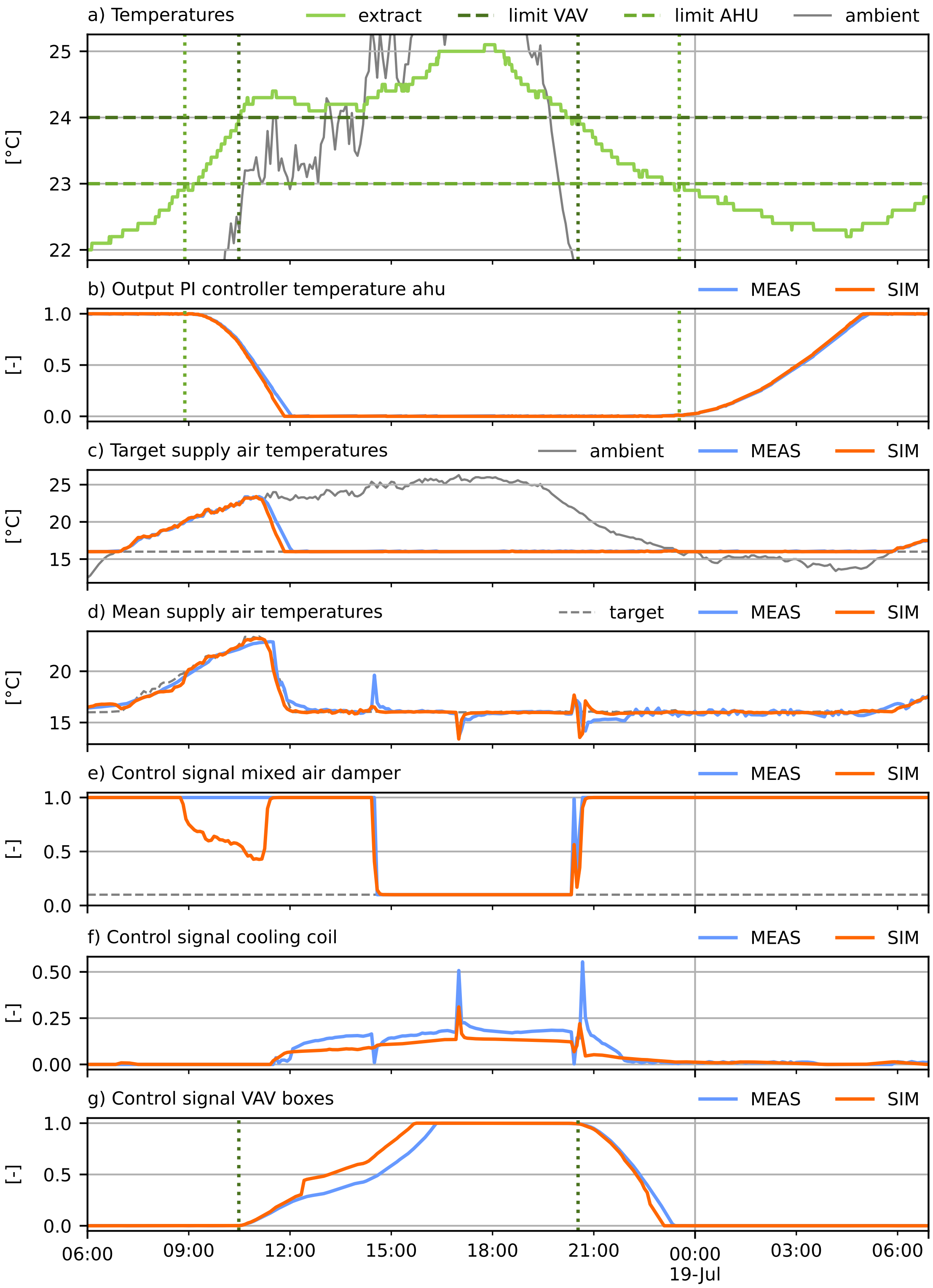


Figure 5.12: Implementation 2: Time series of temperatures and controller signals for a 24 h summer period

Table 5.2: Implementation 2/3: Parameters of PID controllers. Gain scheduling was used for the AHU sequence. The two parameter sets are indicated by "low" and "high".

Controller	Cascade	AHU sequence		VAV
		low	high	
Proportional gain	0.05	0.7	2.0	1.2
Reset time	500	300	100	100
Derivative time	0	0	0	0

In summary, Figure 5.12 demonstrates that the PLCopen XML is implemented correctly, resulting in generally similar control and system behavior. However, the evaluation of the example period also reveals that the limit of the extract air temperature is not respected, which is due to a non-optimal controller parameterization. The parameters of the PID controllers are collected in Table 5.2. In an optimized sequence, the airflow should be increased faster to avoid overheating which could be achieved by higher proportional gain of the VAV-PID controller.

The previous description is completed by an analysis of operating modes with low cooling loads for a 6 hour summer period in Figure 5.13 a). Subfigures i) and ii) present the time series of measured and simulated target and actual supply air temperature. The simulated supply air temperature follows the target trajectory relatively closely, but some deviations can be observed around 9:00. In contrast, the measured supply air temperature oscillates considerably between 8:00 and 10:00. This is due to the behavior of the cooling coil valve, which can be traced in subfigure iii). The control signal of the real controller has a sawtooth profile at very small values of around 2 to 3 %, while it is smooth at higher values. The causes of this behavior are the valve characteristics at small opening positions and the interaction of the valve motor and the operating demand of the upstream feeder pump. These components are outside the modeling scope of the simulation.

The histogram of the control error $t_{target} - t_{actual}$ over the entire period in Figure 5.13 b) i) reveals that the simulated supply air temperature follows the target much closer than the measured one. However, the error in the simulation is more widely distributed than in implementation 1 (Figure 5.10), reflecting the different controller implementation: In implementation 1, the control was based on continuous-time textbook PI controllers optimized for use in IDA ICE. Meanwhile, the control in implementation 2 was based on discrete-time PI controllers designed for real operation on industrial and building control systems, and the control program was solved using the multi-rate method

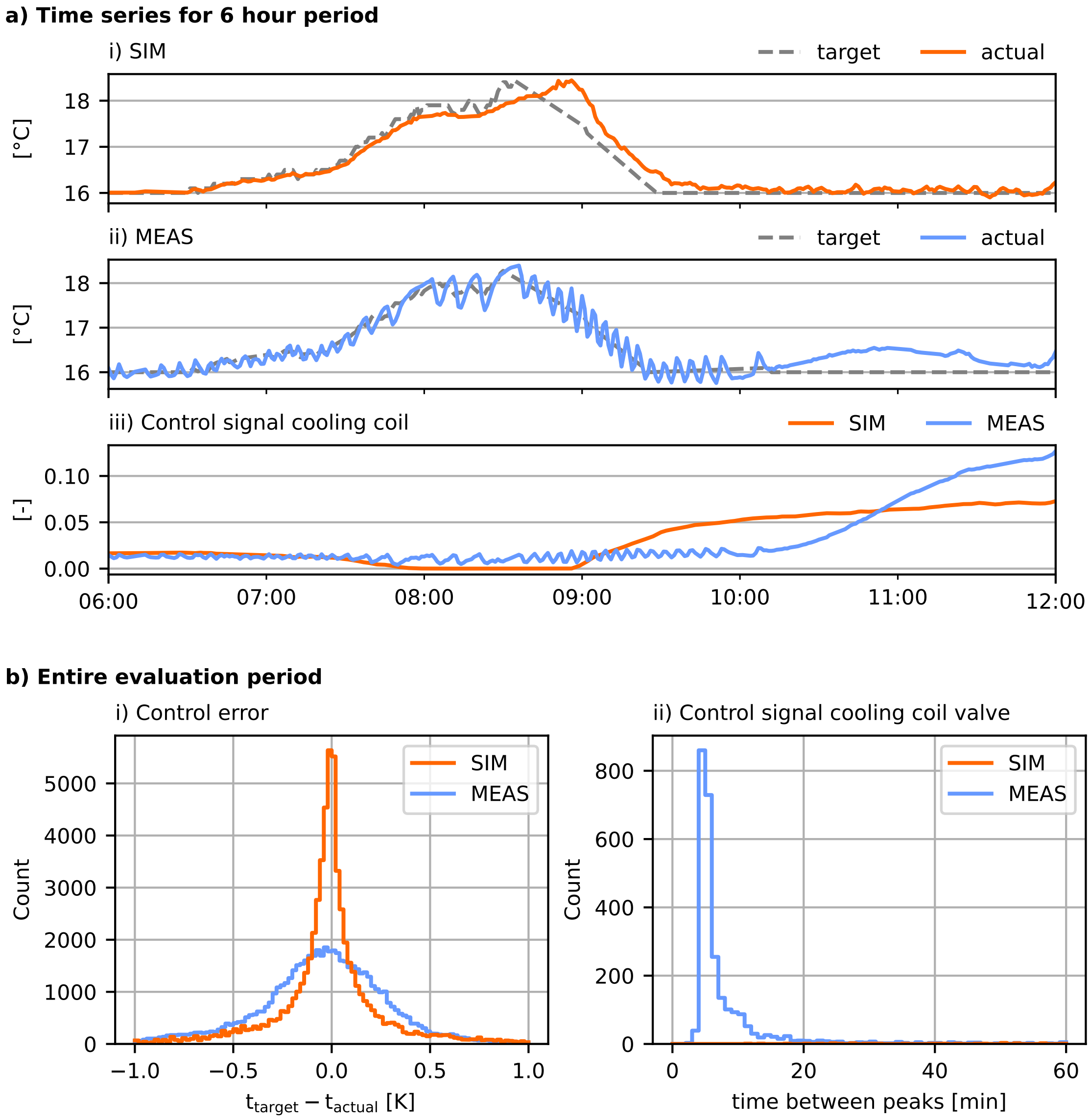


Figure 5.13: Implementation 2: Comparison of controller behavior

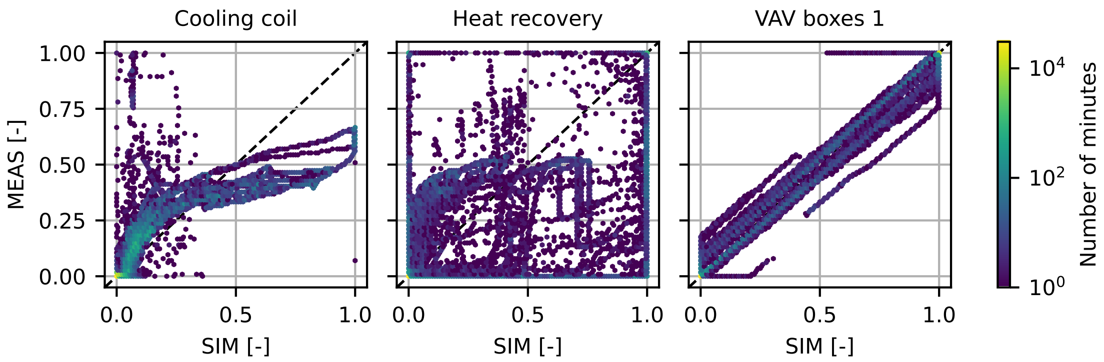


Figure 5.14: Implementation 2: Hexagonal binning plots based on 1 minute measured and simulated control signals for cooling coil valves, heat recovery, and VAV boxes for Zone 1

cite in Section 4.3.1.2. The oscillatory behavior of the cooling coil valves is clearly visible in the histogram of the time between peaks in Figure 5.13 b) ii). Most of the measured peaks are about 5 minutes apart.

Simulated and measured control signals are compared in hexagonal binning plots in Figure 5.14 for selected components[2]. For the cooling coil, the trajectory of the characteristic curve of the physical valve can be clearly traced. However, the simulation reaches the maximum value of 1, while the measurements do not. The causes are likely to lie in different water mass flows and temperatures in the model and real operation. However, the parameters for design conditions were taken from manufacturers' data sheets. Unfortunately, the corresponding water mass flow was not recorded and therefore cannot be compared. A characteristic curve of the liquid heat recovery can also be observed, but it is much less distinct. This is most likely due to simplifications in the simulation, where a simple heat exchanger is used instead of modeling the individual coils of the liquid heat recovery. The control signals of the VAV boxes show a linear relationship. This is due to the behavior that can be traced in Figure 5.12: Because of the sluggish control parameters, it continuously increases from 0 to 1 instead of adjusting the extract temperature.

Implementation 3

In implementation 3, the controller in IDA ICE is directly used for operation. The findings from implementation 2 are therefore transferable and a detailed analysis is omitted here (see also explanation in the introduction to Section 5.2.1). The time series for a summer week is presented in Figure A.2.

2. The comparison for all components is included in Figure A.1.

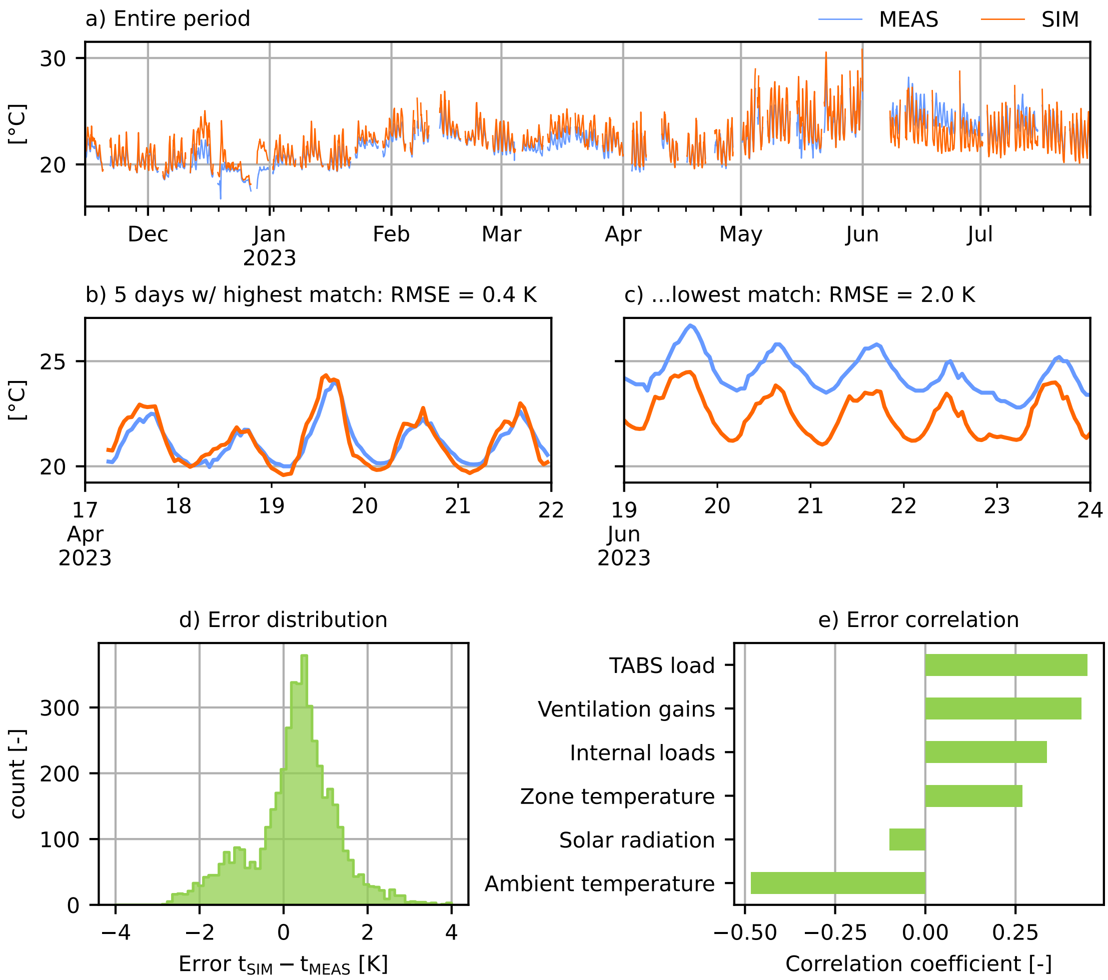


Figure 5.15: Zone model evaluation (Zone 1)

5.2.2 Thermal zones

For a better understanding of the integrated analysis on the coupled building and system models in Section 5.2.3, differences in the room-side behavior are evaluated first. For this purpose, measured values of supply air, climate and internal heat loads from production and lighting are imposed on the room model (cut 3 in Figure 5.2). This decoupled zone model is compared in the following paragraph to the measured extract air temperatures.

Figure 5.15 depicts the time series of measured and simulated extract air temperatures for zone 1 for the whole period (a) as well as for 5 day periods with the highest (b) and lowest (c) agreement. In general, the pattern of daily temperature swing and the relationship between peaks and valleys are correctly reflected. The 5 day period with the highest agreement reaches a RMSE (Root Mean Square Error) of ±0.4 K and the 5 day period with the lowest agreement reaches an RMSE of ±2.0 K.

The histogram of the error $t_{SIM} - t_{MEAS}$ in Figure 5.15 d) indicates that the error distribution is slightly shifted to the right, which means that the temperatures in the simulation are generally slightly higher than the measured ones. Maximum differences are in the range of ±3 K. The error correlation in Figure 5.15 e) shows that the correlation coefficient takes positive values for all internal heat gains and the zone temperature[3]. In simple terms, this means that the temperatures are overestimated in the winter case, while they are rather underestimated in the summer case and at higher internal gains. The results in zone 2 are comparable and can be found in Figure A.4.

In summary, the analysis suggests that the room model generally reproduces the measured pattern correctly, but the differences of up to ±3 K are significant. It is important to note that in the case of this factory building, modeling the complex room-side processes is particularly challenging. This includes the inability to accurately capture cross-zone air flow through open gates, even to the outside, and the correct allocation of internal loads. In addition, infiltration through internal partitions is known to occur, and some ventilation systems within the building operate at excess pressure.

5.2.3 Integrated assessment

In this section, the previously separated submodels are now coupled, simulated, and compared with measurements. The integrated assessment is based on a comparison of the thermal comfort, peaks of control signals, and the energy consumption. These aspects reflect the functionality, robustness, and efficiency requirements defined in Section 4.2.3. The comparison distinguishes between the graphical and digital information transfer in implementation 1 and 2. The results from implementation 2 also apply to implementation 3 for the reasons stated in Section 5.2.1. Implementation 1 and 2 differ as follows in terms of thermal comfort, robustness, and energy efficiency:

- *Thermal comfort (left column in Figure 5.16)*

  The thermal comfort is evaluated on the basis of room-side extract air temperatures. As an indicator, the Kelvin-hour criteria according to DIN EN ISO 52016-1:2018-04 (DIN 2018b) is used. The temperature thresholds defined by the operator are the reference for evaluating undertemperatures in the heating case and overtemperatures in the cooling case.

  In implementation 1, undertemperatures of about 150 Kh are measured in the heating case. Conversely, the simulated overtemperatures (ca. 230 Kh) significantly

3. The related distribution plots can be found in Figure A.3

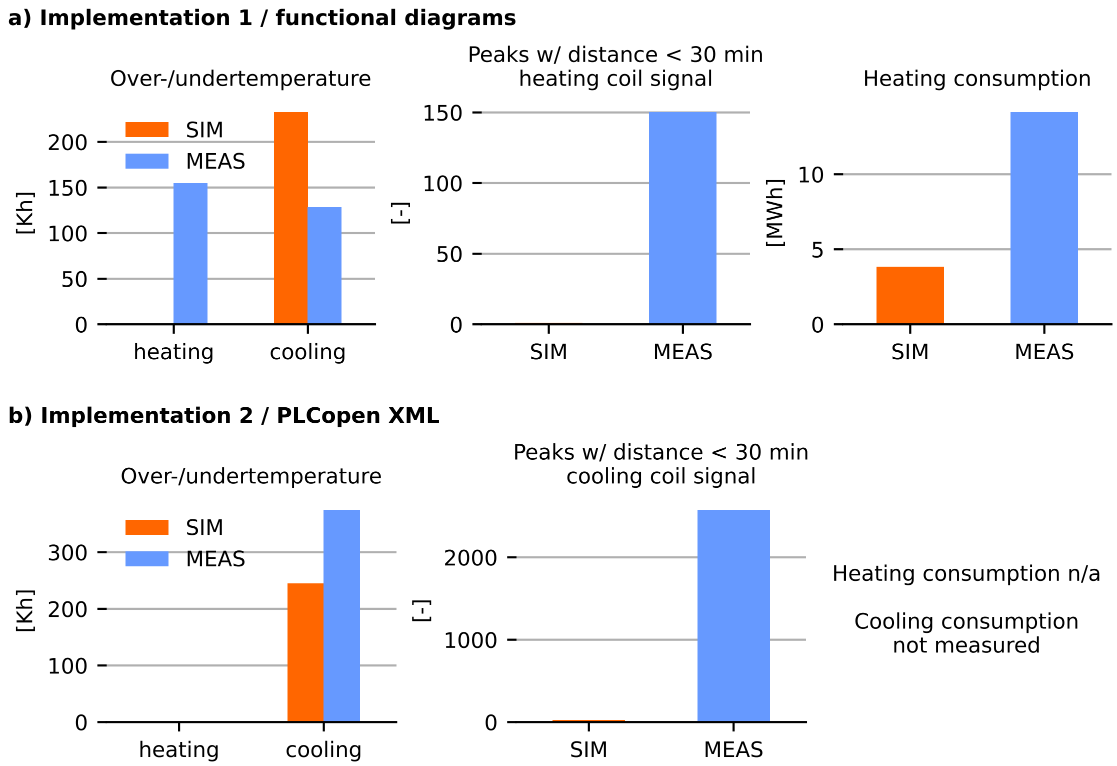


Figure 5.16: Comparison of comfort, robustness, and energy indicators

exceed the measurements in the cooling case (ca. 130 Kh). In implementation 2, the overtemperatures in the cooling case are underestimated in the simulation (ca. 250 Kh) compared to the measurements (ca. 370 Kh). The heating case is not applicable during summer operation. These deviations are related to differences in the room-side behavior (Section 5.2.2).

- *Robustness / peaks (middle column in Figure 5.16)*

  Robustness is evaluated by the number of peaks with a distance below 30 minutes for the heating coil (implementation 1) and the cooling coil time series (implementation 2), respectively (center column in Figure 5.16). According to the observations in Section 5.2.1, the simulation underestimates the oscillatory behavior of the real systems[4]. The fact that the peak count of the measured performance in implementation 2 (more than 2,000) is much higher than in implementation 1 (around 150) does not mean that the control parameters in implementation 2 were set worse than in implementation 1. The reason is that small cooling loads, which result in unstable valve operation at small opening positions (Figure 5.13), are a dominant operation mode during summer operation (implementation 2). Conversely, active heating is rarely required during winter operation (implementation 1).

- *Energy consumption (right column in Figure 5.16)*

  In implementation 1, the measured heat consumption is significantly higher than the simulation[5]. The main reason for this is that room temperatures are overestimated in the simulation and therefore less heating is required.

  No meaningful comparison can be made for implementation 2, as measured cooling loads are not available .

5.3 Discussion

In the following sections, the results, findings, and experiences from the validation cases are discussed. First, the comparison of measured and operated performance is summarized in Section 5.3.1. Then, for each of the three implementations, the control development effort (Section 5.3.2), the building automation architecture and interoperability aspects (Section 5.3.3), services and responsibilities (Section 5.3.4), and practical implementation aspects (Section 5.3.5) are compared. Finally, the opportunities (Section 5.3.6) and barriers (Section 5.3.7) of the proposed workflow are presented.

4. The underlying histogram can be found in Figure A.6.
5. Only periods with correct operation (Figure 5.11) are taken into account.

5.3.1 Control performance and performance gap

The accuracy and reliability of building, system, and control models is paramount from both a design and operational perspective. From a design perspective, reliable models are required because they are the basis for decisions on control approaches and parameterization. If the real objects do not behave as predicted, different design decisions may have been made. During operation, model accuracy is a prerequisite for model-based use cases.

The results in Section 5.2 show that performance gaps remain at several levels:

- *Controls*

  In implementation 1, the controller behavior is significantly different. This is not surprising given the completely different control implementations and parameters of PI controllers. However, transferring the structure of the controls from BPS to a building controller is a first step toward as-designed programming. In implementation 2 and 3, the control code is identical in simulation and operation. Remaining differences are probably related to the different underlying computational methods.

- *Systems in interaction with controls*

  At the system level, several differences related to controls were observed. In this context, the system simplifications in the modeling (Section 5.1.3.3) have to be taken into account. In implementation 1, oscillations in the supply air temperature in active heating mode are caused by inappropriate controller parameters defined by the external contractor. In implementation 2 and 3, unstable behavior is observed at very small opening positions of the cooling coil valves. Such effects are not captured by the simulation models. However, they might be reproducible if the modeling were more detailed. As a result, the simulations significantly overestimate the robustness of the system.

- *Building level*

  At the building level, the effects described above overlap with the room-side behavior (Section 5.2.3). While errors in the control programming have to be considered in implementation 1, significant differences remain also in implementation 2. This illustrates that even when controls are implemented identically in simulation and in a building controller (implementation 2), performance gaps remain. The main reason for the temperature deviations and the associated energy demand is related to the room-side modeling (Section 5.2.2). In summary, this emphasizes that accurate modeling of buildings and their HVAC systems remains a major challenge.

From a design perspective in general, inaccuracies in room-side modeling, or different climates and occupancies, can also result in real systems operating in different modes than anticipated. Automated parameter variations should be performed in the design phase to reflect a wider range of possible operation modes than with default assumptions.

In terms of control performance, the PI controllers are not optimally parameterized in all three implementations. In implementation 1, the parameters were set by the external contractor. In commissioning practice, default parameters are often used, as reported by Fütterer, Schild, and Müller (2017). In implementation 2 and 3, the controllers were not set aggressively enough to avoid overheating in summer (Table 5.2).

5.3.2 Effort for the control development

The effort to develop the control logic is an important aspect for the applicability and acceptance of the proposed workflow in practical applications. The following issues have been identified as problematic in the demonstration cases:

1. *Control function libraries*

   In general, the functions and function blocks of the OSCAT library were found to be sufficient for control development. However, certain HVAC-specific function blocks, such as a cascade and sequence controller, which are typically part of commercial libraries, had to be created manually.

2. *Project-specific development*

   The control logic was developed on a project-specific basis for a project-specific designed AHU. Although the control system allowed for efficient operation, custom development for recurring systems such as AHUs seems problematic given the numerous challenges in the overall building stock. As described in Section 4.2.5, the use of standardized HVAC configurations with standardized controls should be sought to enable efficient processes. In addition, ready-made product-specific controls developed by HVAC manufacturers should be preferred over project-specific developments (Section 4.2.5).

3. *Separation of control logic from system functions*

   The allocation of energy and comfort control and system functions to two separate services allowed a clear separation of responsibilities in the validation cases (see Section 5.3.4). However, the separation of control programs is disadvantageous because of the inevitable interaction. For example, if the actuating variables for the VAV boxes (energy and comfort control) vary too fast, an error in the pressure control of the fans (system functions) may interrupt the fan operation (Section 5.1.5.2). Particularly the startup of a control program can cause steps in the actuating variables that lead to alarms and faults (Section 5.1.5.3). This underscores the need for integrated engineering for energy and comfort control and system function.

The use of proprietary controls for HVAC subsystems (Section 4.2.5), which should integrate both energy and comfort control and system functions in an optimized way, seems to be advantageous. However, these proprietary controls must then be integrated into BPS environments, which is not established today and requires further development efforts towards automated workflows.

5.3.3 Building automation architecture and interoperability

The demonstration building for the practical implementations was equipped with PLCs standardized according to IEC 61131. In addition, the PLC IDE supported the import of the PLCopen XML, which allows interoperability with the Beremiz / IDA ICE toolchain. For the usefulness and success of the PLCopen XML it is necessary that PLC vendors and IDE developers support the IEC 61131-10 standard, which is not always the case today. It remains to be seen whether the PLCopen standard will become more widespread given the lack of use cases today. In addition, the PLC also supports the OPC UA standard, which allows direct coupling with IDA ICE. Without the PLCopen import feature and the OPC UA support, the seamless application of tested and optimized controls from a BPS tool would not have been possible. These aspects underscore the importance of BAS components supporting open standards for innovative and interconnected control approaches.

It is important to note that no vendor-specific libraries were needed to program the energy and comfort control. This is important because today, closed vendor-specific libraries promise user-friendly control programming, but they also lock automation engineers and operators to specific automation vendors. Separating control hardware from control programming is not only possible, it seems to be advantageous: Building control design experts and automation hardware experts can then focus on their specific domains. The advantage of platform-neutral control sequences is also outlined by Roth et al. (2022) in the context of CDL.

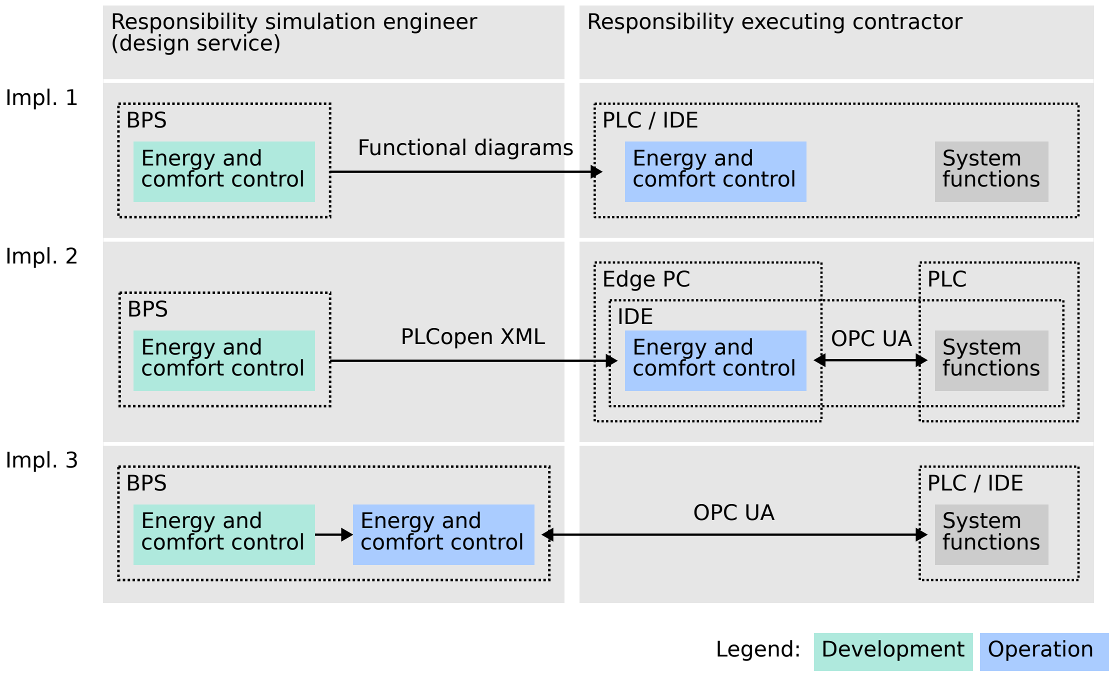


Figure 5.17: Comparison of the responsibilities of the simulation engineer and the executing contractor for the development and the execution of controls

5.3.4 Services and responsibilities

In all three implementations, the control logic was developed by a simulation engineer. In today's construction practice, this engineering service would be assigned to design offices (Section 1.2.2). With respect to the separation of development and operations responsibilities, the implementations differ as follows (Figure 5.17):

- *Implementation 1 and 2*

  The control logic developed by a simulation engineer (green box in Figure 5.17) is handed over to an external contractor either through functional diagrams or the PLCopen XML. Energy and comfort control and system functions are integrated by the external contractor into a control program, which is then used for operation (blue box in Figure 5.17).

  The problem with this structure is that every time the design office needs to make a change to the energy and comfort control, a contractor is needed (see Section 5.3.5).

- *Implementation 3*

  Control logic developed in a BPS tool is instantiated from a BPS development environment to a BPS operational environment, both under the responsibility of the simulation engineer. The BPS operational environment communicates with the PLC under the responsibility of an automation contractor who adds system functions.

The advantage of this structure is that the energy and comfort control is not only designed, but also operated by the simulation engineer. Only one engineering service is needed to manage the energy and comfort control during commissioning and operation.

In general, the engineering effort has been shifted from a later commissioning and operation phase to an earlier design and development phase compared to today's practice. However, this collides with the current practice and legal architecture according to which the risk is transferred from a design office to an executing contractor. One advantage of the proposed approach would certainly be that design offices could be held responsible for the performance of the building in operation directly. Today, this is often difficult because of the low level of detail in planning and the use of ambiguous communication formats that make it difficult to accurately attribute errors.

From a technical point of view, the interaction between energy and comfort control and system functions needs to be considered (Section 5.3.2). Separation requires close communication between the respective engineers. In addition, the separation of responsibilities has implications for errors in the implementation process, as well as debugging and update options, which are discussed in the following section.

5.3.5 Implementation errors, debugging, and updates

Since the three implementations were done under practical conditions, several practical aspects could be observed which are analyzed in the following. These include mainly errors at the interface of the control development and the implementation on a building controller by the external contractor. Debugging and updates may be required during the commissioning phase to ensure correct operation. The ability to perform debugging and updates efficiently is critical to successful commissioning and optimization. These aspects are closely related to the separation of responsibilities (Section 5.3.4). The lessons learned can be summarized as follows:

- *Implementation 1*

  Graphical functional diagrams are obviously the most error-prone communication format. During programming, parts of the given control logic have been implemented incorrectly or even forgotten. The results in Section 5.1.5.1 give an example of the effect of such errors on system performance. Errors in the control program were difficult to locate because the code was written by an external engineer and contained proprietary control functions.

- *Implementation 2*

  Using the PLCopen XML allows for completely error-free information transfer. However, the implementation is not fully automated and the following problems were observed:

  The control parameters had to be set manually by the automation contractor because they were not included in the PLCopen import in the e!cockpit IDE. Some parameters were set incorrectly by the external contractor and had to be identified and corrected manually. This problem with the PLCopen XML is probably related to the low penetration of this exchange format, and consequently perhaps to the low attention paid to this interface by software developers. Further tests verifying the import and export functionality of different IDEs would be required to provide a more detailed market overview.

  Control signals to actuators were not bound correctly, resulting in actuators not being controlled or being controlled in the wrong way. This problem is related to the missing semantic information in the PLCopen XML file about the binding of inputs and outputs to measurements and actuators. Semantic modeling methods which address these issues have gained importance in research recently (Schneider 2019; Ihlenburg, Benndorf, and Réhault 2022; Roth et al. 2022).

  This missing semantic information for variable binding, the missing PLC time implementation in PLCopen (see Section 5.1.5.2), and the missing import of control parameters are a barrier for smooth control program updates. These issues required many manual steps during the update process. As a result, changes to the control code were made manually by the external contractor for the relevant parts of the code, rather than re-importing the entire program. Obviously, this is only feasible for minor changes.

  For debugging, the control code with live values was only visible and accessible in the IDE of the external contractor on request. This is clearly a disadvantage when a design office is responsible for the correct operation, but cannot follow the controller operation in live.

- *Implementation 3*

  The least error-prone approach was for the simulation engineer to directly bind inputs and outputs in the BPS tool to the variables on the OPC UA server. This underscores the advantage of integrating development and deployment in a single service. However, this binding was still a manual process. For the two AHUs in the demonstration cases, 9 variables were imported and 15 variables were exported. This is still manageable, but will inevitably become a challenge for all 37 AHUs and all building energy systems.

Updates could be managed independently by the simulation engineer. For an update, the simulation in IDA ICE was first interrupted. Then the fallback program on PLC was activated (Figure 5.7). After the new control was implemented in IDA ICE, the simulation was restarted and the fallback program was deactivated.

5.3.6 Opportunities in a Digital Twin framework

The development of controls on coupled building and system models, the seamless digital transfer of these controls in operation, and the continued use of the models in operation are the key aspects of the proposed methodology (Figure 4.1). On the one hand, the creation of models for buildings and systems in white-box models, the collection of the required boundary conditions and the detailed development of controls increase the effort in the design phase compared to today's processes. On the other hand, numerous opportunities open up:

1. *Developing and implementing energy-efficient control sequences*

   The savings potential of optimized control sequences is well documented in the literature (Pang, Piette, and Zhou 2017; Zhang et al. 2022). The results in chapter 5 underline that the use of coupled building, system and control simulation in the design phase support the development of optimized project-specific control sequences. Current design methods and tools, such as monthly energy balancing for energy performance and control logic development with static considerations, are not suitable for this task (see summaries in Section 2.5 and Section 3.4). Figure 5.18 demonstrates the success of the revisions to the original control and the savings in both heat and fan power consumption.

2. *Optimized integrated design for control and HVAC systems in the design phase*

   The development of controls on coupled building and system models also supports the selection and sizing of HVAC systems. Both aspects interact and need to be considered in an integrated manner. This is particularly promising because decisions made in the design phase have a large impact on both investment and operating costs, as well as on the use of resources during construction and operation.

   The AHUs in the demonstration cases were designed and built outside of this work and were used "as is" for implementation. However, if the implementations would

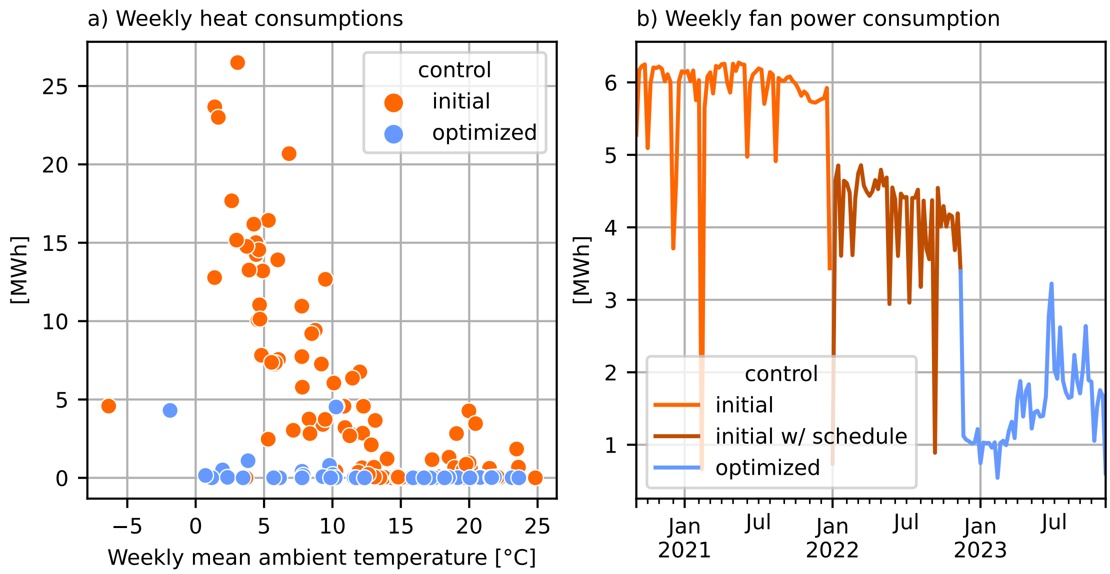


Figure 5.18: Measured heat[1] and fan power savings for the initial and the optimized control program (all three implementations)

[1] Periods without heat recovery due to faulty programming in implementation 1 (see Figure 5.11) are omitted.

have taken place in a real design situation, the AHUs would have been configured differently. The example of small cooling loads as the dominant mode of operation, causing stability problems at small valve openings, suggests that the cooling coils may be oversized.

3. *Transparency of system complexity at the design stage*

   Another aspect of integrated system and control design is that system complexity becomes fully transparent already in the design phase, before the contract is awarded. When engineers in design offices are forced to develop an explicit control program, they become aware of the actual system complexity and cannot postpone this fundamental development task to later phases. If the project-specific control development for sophisticated systems proves to be too challenging, or if the operating time for certain systems proves to be too short, components can be omitted and systems simplified. In the demonstration cases for example, adiabatic exhaust humidification was omitted to simplify the control development.

4. *Reducing the performance gap and targeted analysis*

   Digital transmission of control logic helps to reduce the performance gap between expected and actual performance. Today, it is often not clear how HVAC systems operate because the intended control logic is described at a low level of detail or the

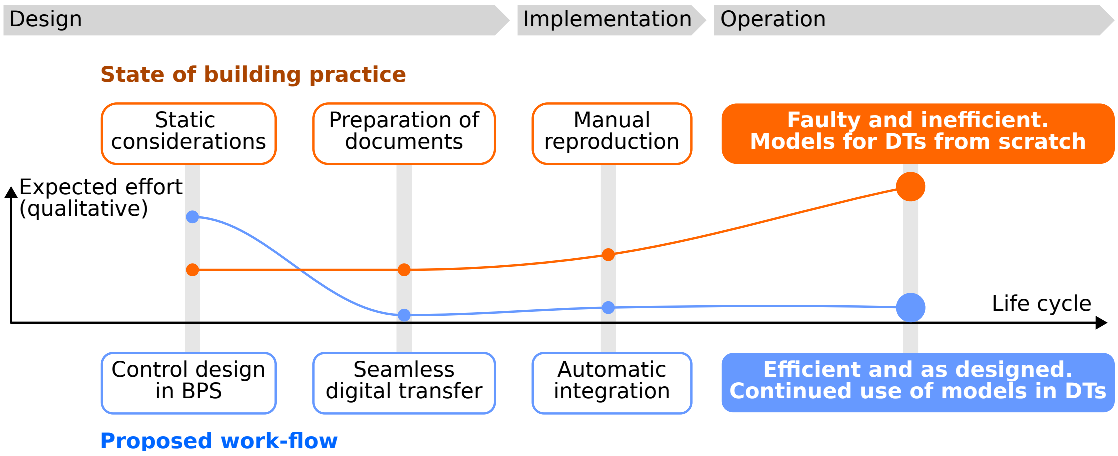


Figure 5.19: Comparison of expected qualitative effort in control design, implementation, and operation

implemented code is not accessible (Section 3.4). The digital transfer of controls between the development tool (BPS) and the building controllers eliminates these uncertainties. This enables targeted performance analysis of, for example, HVAC system characteristics or user behavior.

5. *Efficient processes*

Compared to current practice, modeling buildings and systems in BPS increases the level of detail in the design phase, but also the associated effort (left in Figure 5.19). In return, a number of tasks from HVAC and BA engineering, such as static heating load calculations and monthly balancing procedures to prove energy requirements, can be replaced or merged. The commissioning processes in the validation cases were highly efficient compared to today's practice (center in Figure 5.19). Even in implementation 1, the detailed graphical schemes were translated into PLC code relatively quickly. In the future, as in implementation 2 and 3, the control programs will only be taken over during commissioning, while currently the actual programming takes place in this phase. However, adjustments such as tuning of control parameters may still be required. During operation, energy consumption and emissions are reduced by optimized control sequences (right in Figure 5.19). The error-free implementation of tested and verified controls reduces the monitoring effort and enables targeted fault detection for technical errors. Finally, building and system models can be used continuously for model-based use cases in operations.

6. *Clear allocation of responsibilities*

   Experience in the three implementations has shown that control logic for HVAC systems can be developed by simulation engineers down to local loop controls. The key success factor is the use of coupled building and system models as a Digital Twin Prototype. Although the demonstration cases were in operation, this emphasizes that detailed control development is possible in the design phase. Since the energy and comfort control must be fully defined before the contract is awarded, the developing designer is then responsible for the operation. This is a clear advantage over current practice, as explained in Section 5.3.4.

7. *Promoting model-based use cases in operations*

   The availability of models from the design phase, the Digital Twin Prototypes, and the bidirectional connection between BPS environments and the building promote model-based use cases in operation, such as performance gap analyses, FDD, MPC, and PM. These applications have been developed, tested, and implemented over the past few years with promising results, but market penetration is still low. A common problem today is, that models often have to be newly created in operation.

   In the three implementations, the Digital Twin was used not only as a Digital Twin Prototype for control development, but also as a Digital Twin Instance for performance gap analyses during operation. For these analyses, measured time series were manually imported and applied to the Digital Twin. In addition to this manual connection, IDA ICE was connected to BAS via OPC UA in implementation 3. This connection was used to control the AHUs through IDA ICE running on a server inside the building IT network. This is an example of the continued use of a BPS software for both development and operation as a Digital Twin Environment. The bidirectional connection between simulation models as a Digital Twin Instance and the real building enables future online model-based use cases in operation.

5.3.7 Barriers

The opportunities are contrasted by several obstacles which stand in the way of widespread application of the proposed workflow:

1. *Software tools and toolchains*

   The most obvious obstacle is that the software toolchains needed to connect BPS tools and BAS are not yet available for a productive workflow in commercial applications. The coupling of IDA ICE and Beremiz, for example, is not available as a standard feature of the commercial software. Several manual intermediate steps were required to integrate the DLL from Beremiz in implementation 2 and 3. Additionally, Beremiz as open source software does not offer the same usability and features as a commercial IDE. The OPC UA interfaces for IDA ICE are included in the software by default, but are still under development and require project support for use.

With respect to the OBC workflow and the CDL, the technology readiness level appears to be similar. It remains to be seen how far and how fast CDL and the ASHRAE 231P standard will be accepted and adopted by the building automation industry.

Regarding BPS tools in general, the options for suitable software are very limited. Efforts to develop existing tools for integrated building, system, and control simulation and for linking to BAS need to be increased.

2. *Modeling effort*

   A common drawback of BPS applications is the modeling effort required, especially for geometry import and parameterization. Increased exchange of geometry and parameters through the deployment of BIM methods is necessary in order to reduce this effort. However, it should also be noted that the building and system models in the validation examples were relatively simple and the AHU models used only standard components. Nevertheless, the control logic developed on these models achieved large savings compared to the initial operation. This observation suggests, that using a model to develop control logic at all, is beneficial even with a limited model accuracy.

3. *Legal requirements, incentives, and building practice*

   Legislation, incentives, and building practices are certainly the biggest obstacles. Even if BPS-to-BAS software toolchains were available and the modeling effort were minimized, the following aspects need to be considered:

   - *BPS as a standard tool for design and verification*

     Widespread use of BPS tools is essential for the application of the proposed methodology. The use of simulation tools in practice is significantly influenced by the regulatory framework, as illustrated by the description of the situation in Germany and Sweden (Section 2.1). Construction practice in Sweden shows that a wide application of BPS as a standard method is possible. In this context, it is important to note that the latest draft of the EPBD (European Commission 2021) proposes the use of dynamic simulations with a time step not greater than 1 hour. However, smaller time steps in the order of seconds to minutes are required for system simulation and control development.

Another incentive to use more detailed and realistic simulation methods is the requirement that the calculated target energy demand in the design phase is verified against the actual measured energy consumption in operation, as is the case in Sweden (Section 2.1). In this context, absolute floor area limits seem to be more appropriate than the reference building criterion, as they are more understandable for owners, users and operators.

- *Separation of design and execution and resulting services*

  The detailed development of controls by designers prior to contract award, adopted by contractors and used directly for operation, is a core element of the proposed methodology. However, such an approach differs fundamentally from the current separation of services between pre-award design offices and post-award contractors (Section 1.2.2) and collides with related legal requirements, at least in Germany. In Germany, planning offices simply do not have the necessary insurance to provide detailed engineering services for the implementation of control logic (NAMUR 2020). Therefore, the regulatory and normative design needs to be adapted so that the designer can take responsibility for the implementation and operation of the control logic in the future.

4. *Separation of engineering domains and education*

   The development of control logic on coupled building and system models is a complex task that requires specific knowledge from various domains (Section 1.2.1). However, the knowledge required to develop optimized integrated solutions is typically dispersed and not concentrated (Section 1.2.1). Building physics, HVAC, and control engineering are often performed by separate services, either provided by separate companies or by separate divisions within the same company.

   With respect to the use of BPS tools and controls, the skills are often far apart: For example, building physics engineers are rather familiar with BPS tools, but have little knowledge of control technology. Conversely, control engineers typically do not use BPS tools at all. The required cross-domain knowledge is not reflected in typical engineering programs such as mechanical engineering, civil engineering, control engineering, or architecture. As a result, engineers capable of providing integrated services are rare.

5. *Information for stakeholders and building professionals*

   Many building owners, investors, and operators but also project managers, general planners, architects, and sometimes even HVAC planners are still unaware of the opportunities that exist through the consistent use of simulation in design and the continued use of models in operations. As a result, opportunities for the successful use of simulations, especially of HVAC systems and their control, often remain unused.

6 Conclusion and Outlook

6.1 Conclusion

Aspects of operational management and control of HVAC systems play an increasing role in overall building performance. The control logic implemented on BAS largely determines the operational performance of HVAC systems. Accordingly, the methods and tools used to develop controls are of particular importance. It is well known that the design phase is critical in building processes because decisions made at this stage have a high impact on the operational performance. This is especially true for HVAC systems and their control. Therefore, developing and testing control logic on building and system models during the design phase and the automatic deployment of controls in BAS has great potential, as described in chapter 1. The complexity of buildings and HVAC systems, as well as the methods, processes, and tools used for control design and implementation today, have been identified as major challenges and problems. Accordingly, the overarching goal of this thesis was to describe and validate methods which enable model-based control development in order to overcome the deficits of today's practice.

The description of the state of the art in chapter 2 gives a brief overview of the performance requirements at the building level in the context of HVAC controls. The fundamentals of building automation systems are presented in the context of control logic design and implementation. BPS methods and tools are analyzed with a focus on coupled building, system, and control simulation. With respect to Digital Twins, the methodological approach, model types, and applications in design and operation are described. These fundamentals are complemented by analyses in demonstration buildings in chapter 3. The focus is on how controls are developed, communicated, and implemented in practice and how these aspects affect the operation of AHUs.

The main work related to the two research objectives is carried out in the methodological part in chapter 4 and the validation in chapter 5. With respect to the two research objectives and the related research questions formulated in Section 1.4 for this thesis, the findings and proposals are summarized in the following paragraphs.

Objective 1 *Description, analysis and practical implementation of options to deploy controls developed in BPS environments for the operation of HVAC systems*

The first part of objective 1 is achieved by providing a comprehensive overview of how to link BPS and BAS and the associated engineering services in design and implementation in a Digital Twin framework described in chapter 4. The fundamental difference to today's practice is that BPS tools are used for control development in the design phase before the contract is awarded. Today, monitoring and fault detection of measured data in operations are common approaches to achieving improved operational performance through control changes and corrections. However, these approaches are inefficient when multiple manual iterations are required to develop satisfactory performance. In contrast, control development in BPS environments emphasizes the role of the design phase to minimize effort during commissioning, enabling a more efficient overall process.

In general, using BPS tools for control development is not a new idea. However, techniques for actually deploying controls from BPS tools in BAS have only been gaining importance in recent years. Against this background, the second part of goal two is achieved through three real implementations of control logic for large-scale AHUs in an industrial factory building. Particularly the deployment of IEC code developed in a BPS environment (implementation 2) and the use of IDA ICE for control development and execution (implementation 3) were firsts of their kind.

The implementations were successful and allowed large savings compared to the initial operation that represents the usual, non-optimized operation. The experiences emphasize that the operation of AHUs with control logic developed and tested with BPS is generally possible. However, the comparison of measured and simulated performance shows that performance gaps remain (Section 5.2). Finding the necessary level of detail for the complex physical behavior in buildings and HVAC systems remains a challenging task. In summary, the methods and technologies for the digital deployment of controls from BPS environments (implementation 2 and 3) have the potential to overcome today's practice in the design phase, which has remained almost unchanged over the past decades (Section 2.2). The link between BPS tools and BAS accelerates the much needed digitization in HVAC design and operation.

RQ 1.1 *What options exist to implement control logic developed in BPS environments in buildings for the operation of HVAC systems and how do they differ methodologically?*

In implementation 1, control logic developed in IDA ICE was communicated according to the normative standard via functional diagrams and implemented by an external contractor (Section 5.1.5.1). However, the functional diagrams, as an analog format, did not contain all the information about the control logic, and manual reproduction in control code is labor-intensive and still error-prone. The CDL and PLCopen XML allow the transfer of controls in a digital format. The CDL enables the transfer of a digital control specification between Modelica-based simulation environments and certified building controllers. However, CDL is still in the standardization process. The PLCopen XML is an existing industry standard for exchanging executable code according to IEC 61131 between PLC IDEs. The toolchain with IDA ICE and Beremiz was successfully applied for the first time in a large-scale and quasi-productive building application (Section 5.1.5.2). Further development is needed for both the CDL and IDA ICE-Beremiz toolchains for fully automated processes.

Interoperability between BPS tools and BAS is paramount for the widespread application of the proposed methodology. With respect to the CDL, it remains to be seen whether this new standard will be adopted by the automation industry. The PLCopen XML as an existing standard has little penetration in building automation practice today. The manufacturer of the PLCs in the demonstration building supported the PLCopen XML, but many other vendors do not. In implementation 3, the energy and comfort control was executed in the loop in IDA ICE and connected to the downstream BAS via OPC UA, while the building controller was used only to map data points to field level devices and provide system functions. Although BPS environments are not designed for control execution in operation, this is an example for the continued use of one software environment both for controls development and operation which bypasses otherwise necessary exchange formats.

All three implementations make it clear that the development of the control logic is not dependent on specific hardware or libraries from controller manufacturers. This is important because automation vendors often create closed ecosystems with proprietary IDEs for control programming that lock in customers and owners. Practical implementation experience underscores the importance of interoperability and automated processes to avoid errors when importing controls to building controllers, for debugging during commissioning, and for updates (Section 5.3.5, Section 5.3.3).

RQ 1.2 *Which discrepancies between simulated and measured performance on building, system and control level can be observed when control logic developed in BPS environments is used for the operation of HVAC systems against the background of model simplifications?*

Considering the interaction of building, systems, users, climate, and grids makes control design a challenging task. BPS environments allow to analyze the interaction of these aspects and to develop custom controls on coupled building and system models. In three practical implementations, it has been demonstrated that the implementation of control logic developed on coupled building and system models enables energy efficient operation of AHUs.

The digital transfer of control logic from BPS environments to building controllers closes the performance gap related to communication between design and implementation. However, performance gaps remain at several levels due to model simplifications and assumptions (Section 5.2). In general, analysis of these discrepancies can help to support model improvements, understand user behavior, or refine physical parameters without being biased by different control implementations.

Regarding controls, particularly the unstable behavior of the real systems in certain operating modes is not correctly reflected in the simulation due to modeling simplifications and uncertainties in the characteristics of physical components (Section 5.1.3). More detailed modeling would be possible, but would increase the modeling and simulation effort. However, using building and system models enabled developing optimized controls and the real controller generally behaved as predicted. This emphasize the benefit of using building and system models for control development at all, even if the real system may behave differently. In terms of robust operation, it is likely that the real system will perform even worse if the simulation shows undesirable behavior.

RQ 1.3 *How does the development of controls in BPS tools during a design phase change currently established processes and the effort at the interface of design and execution services?*

Validation within this thesis was conducted in demonstration buildings and with separate engineering services for control design and implementation. Applied to today's building practice, this reflects the separation between design offices and contractors. Currently, the control logic is first designed by the designers before the contract is awarded, and then detailed and programmed by the contractors (Section 2.2.1, Section 2.2.3). In the proposed workflow, the entire control definition, except for system functions, would be done by design offices in BPS environments prior to award. Shifting the responsibility from the executing companies to the planning offices conflicts with current practices and regulations (Section 1.2.2). However, the implementation experience underlines the importance of explicit control programming on coupled building and system models.

It has been shown that project- and system-specific control development at code level is possible in BPS tools (Section 5.1.5). However, such an approach is labor intensive and requires expertise in many different domains. Standardized control sequences such as ASHRAE Guideline 36-2021 and vendor-specific models and controls should be used in order to reduce the overall effort.

Objective 2 *Embedding of the workflow aimed for in Objective 1 into a Digital Twin framework*

To achieve objective 2, the relationship between controls on the one hand and building and system models on the other is first analyzed in chapter 4. Then the three implementations and their respective approaches to linking BPS and BAS in a Digital Twin framework are discussed and compared in chapter 5.

RQ 2.1 *What is the role of controls in Digital Twins in relation to HVAC systems and buildings?*

Controls are an essential part of building operations, building models, and thus Digital Twins. However, controls have a special role in Digital Twins. Models of building structure, technical systems, user behavior, and climate are often simplified in building energy modeling. In contrast, controls implemented as software on BAS can be equally applied to models in Digital Twins as well as to the real building (Section 4.4.1). This allows a separate comparison of the physical parts of a Digital Twin with their counterparts in the Physical Twin (Section 4.4.2). For example, in the context of the validation (Section 5.2), analyses have been carried out both separately for the control level, the system level, the building level, and in an integrated manner.

RQ 2.2 *How do the options described in RQ 1.1 support the application of Digital Twins?*

Today, building and system models are not used in controls design (Section 2.2.1). When simulation models are needed for Digital Twin applications in operation, the replication of the implemented control is fraught with major hurdles (Section 3.2).

In the proposed methodology, controls are developed on coupled building and system models in the Digital Twin Prototype during the design phase (Section 4.1). These controls are then used to operate real HVAC systems. Various options have been described in chapter 4 and implemented in chapter 5. The building and system models including controls from the Digital Twin Prototype are linked to measurements from the Physical Twin and used as a Digital Twin Instance for model-based use cases in operations. Since the same controls are implemented in the Digital Twin Prototype / Digital Twin Instance and the Physical Twin, operational use cases can be executed much more efficiently. This solves the problem of replicating implemented controls during the operating phase (Section 3.4). Thus, the options described in RQ 1.1 bridge the gap between Digital Twin use cases in the design and operational phases (Section 4.4.2).

Three practical implementations demonstrated the benefits of this workflow (chapter 5). A Digital Twin Prototype was used to develop and optimize controls for AHUs in an industrial plant. The Digital Twin Instance allowed in-depth performance analysis for a detailed understanding of the building, system, and control behavior. The bidirectional connection between IDA ICE and the building allows not only the control of the AHUs through a BPS tool, but also other online model-based use cases in operation in the future. In essence, the options identified in Objective 1 support the use of Digital Twins by bridging the gap between development and operations. This ultimately unlocks the full potential of Digital Twins as a key method in the Industry 4.0 paradigm in the building sector.

6.2 Outlook

The results and findings of this work are the starting point for further research:

The toolchains for transferring control logic from BPS environments to building controllers require further development for the application in commercial workflows. Interfaces for the PLCopen XML in both BPS environments and controller IDEs need further development. The CDL is still in the standardization process and not yet available for controllers following the IEC 61131 standard. Existing works to integrate PLCopen XML and CDL and semantic methods to increase interoperability between different applications need to be extended (Roth et al. 2022; Ihlenburg, Benndorf, and Réhault 2022; Schneider 2019). Furthermore, toolchains need to be integrated with BIM methods to increase the efficiency of model creation.

Although the performance gap related to control communication can be closed, the accuracy of building and system models remains a critical issue. Deviations in room temperature of up to ± 3 K, as in the validation cases, could be critical for model-based use cases such as MPC. Therefore, existing efforts to build realistic and reliable models need to be continued. In addition, methods for calibrating models to measured conditions in operation need to be further developed.

The project-specific individuality of HVAC systems remains a crucial aspect for efficient design and implementation processes. It remains to be seen whether a certain degree of modularization will become established in the building industry. Embedding vendor-specific models and controls into building energy modeling at the design stage is a promising way to reduce modeling effort. However, HVAC system manufacturers must provide such models in a standardized format, such as eFMI, which is typically not the case today.

The results and experiences provide important impetus for the further development of these topics. This applies in particular to simulating PLC code according to IEC 61131-3 in BPS environments. The applications in this thesis are the first large-scale demonstrators in buildings and are currently running in productive operation. Additional applications for other AHUs are planned. The continued use of building and HVAC models developed during the design phase in BPS environments also for live applications in operation is especially promising for Digital Twin applications. The results of this thesis also make a significant contribution to this approach. Applications in other types of demonstration objects and HVAC systems must follow in order to further illustrate the potential for intelligent operation and efficient processes.

Bibliography

Literature

Achermann, Matthias. 2000. *Validation of IDA ICE: Version 2.11.06.* With IEA Task 12 - Envelope BESTEST. Edited by Hochschule Technik+Architektur Luzern HLK Engineering. Accessed September 28, 2023. http://www.equaonline.com/iceuser/validation/old_stuff/BESTEST_Report.pdf.

Achermann, Matthias, and Gerhard Zweifel. 2003. *Praxisnahe Validierung von Simulationsprogrammen: Beitrag zu IEA Solar Task 22.* Accessed September 28, 2023. https://www.aramis.admin.ch/Default?DocumentID=61733&Load=true.

agn Niederberghaus & Partner and siganet, eds. 2019. *Funktionsbeschreibung Gebäudeautomation,* December 17, 2019.

Agouzoul, Abdelali, Emmanuel Simeu, and Mohamed Tabaa. 2023. "Enhancement of Building Energy Consumption Using a Digital Twin based Neural Network Model Predictive Control." In *2023 International Conference on Control, Automation and Diagnosis (ICCAD),* pp. 1–6. 2023 International Conference on Control, Automation and Diagnosis (ICCAD), Rome, Italy, May 10–12, 2023. IEEE. ISBN: 979-8-3503-4707-4, accessed October 16, 2023. https://doi.org/10.1109/ICCAD57653.2023.10152308.

Barwig, Floyd E., John M. House, Curtis J. Klaassen, Morteza M. Ardehali, and Theodore F. Smith. 2002. "The National Building Controls Information Program." In *ACEEE Summer Study on Energy Efficiency in Buildings,* Pacific Grove, CA. Accessed June 10, 2023.

Beausoleil-Morrison, Ian. 2021. *Fundamentals of building performance simulation.* Beausoleil-Morrison, Ian, (author.) New York: Routledge Taylor & Francis Group. Accessed November 4, 2023.

Béguery, Patrick, Andreas Kissavos, and Per Sahlin. 2013. "A building control oriented simulation architecture." In *Proceedings of BS2013: 13th Conference of International Building Performance Simulation Association,* Chambéry, France, August 26–28, 2013. Accessed February 16, 2023. http://www.ibpsa.org/proceedings/BS2013/p_1473.pdf.

Béguery, Patrick, Thomas Murienne, Van Lap Ngo, and Oskar Nilsson. 2021. "Coupling building simulation with virtual building management system for advanced test and validation." In *Proceedings of Building Simulation 2021: 17th Conference of IBPSA.* Building Simulation Conference proceedings. 2021 Building Simulation Conference, Bruges. KU Leuven. Accessed February 18, 2023. https://doi.org/10.26868/25222708.2021.30712.

Bengt Dahlgren, ed. 2018. *KTH Live in Lab: Handlingsförteckning.*

Beremiz. 2016. *Beremiz.* V. 1.2. Beremiz SAS. https://beremiz.org/.

Berger, Michael, Filippo Bernardello, Craig Barry, Pravesh Badjoonauth, Shruthi Balaji, and Michael Lakhdar. 2022. "Real-time Model Predictive Control with Digital Twins and Edge Computing Technologies." CLIMA 2022 conference, 2022: CLIMA 2022 The 14th REHVA HVAC World Congress [in en], accessed August 25, 2023. https://doi.org/10.34641/CLIMA.2022.368.

Bjørnskov, Jakob, and Muhyiddine Jradi. 2023. "An ontology-based innovative energy modeling framework for scalable and adaptable building digital twins." PII: S0378778823003766, *Energy and Buildings* 292:113146. ISSN: 03787788. https://doi.org/10.1016/j.enbuild.2023.113146.

Blum, David, Zhe Wang, Chris Weyandt, Donghun Kim, Michael Wetter, Tianzhen Hong, and Mary Ann Piette. 2022. "Field demonstration and implementation analysis of model predictive control in an office HVAC system." PII: S0306261922004895, *Applied Energy* 318:119104. ISSN: 03062619, accessed October 10, 2023. https://doi.org/10.1016/j.apenergy.2022.119104.

Boje, Calin, Annie Guerriero, Sylvain Kubicki, and Yacine Rezgui. 2020. "Towards a semantic Construction Digital Twin: Directions for future research." PII: S0926580519314785, *Automation in Construction* 114:103179. ISSN: 09265805, accessed February 9, 2023. https://doi.org/10.1016/j.autcon.2020.103179.

Bonvini, Marco, and Alberto Leva. 2012. "A Modelica Library for Industrial Control Systems." In *Proceedings of the 9th International Modelica Conference: September 3 - 5, 2012, Munich, Germany,* edited by Dirk Zimmer and Martin Otter, pp. 477–484. 9th International Modelica Conference. Linköping: Modelica Association and Deutsches Zentrum für Luft- und Raumfahrt, Modelica Assoc. https://doi.org/10.3384/ecp12076477.

Brilakis, Ioannis, Yuandong Pan, André Borrmann, Hermann-Georg Mayer, Fabian Rhein, Catharina Vos, Eva Pettinato, and Sigrid Wagner. 2019. *Built Environment Digital Twinning: Report of the International Workshop on Built Environment Digital Twinning presented by TUM Institute for Advanced Study and Siemens AG.* Accessed August 22, 2023. https://publications.cms.bgu.tum.de/reports/2020_Brilakis_BuiltEnvDT.pdf.

Brümmendorf, Erik, Jan Henrik Ziegeldorf, and Johannes Peter Fütterer. 2019. "IoT platform and infrastructure for data-driven optimization and control of building energy system operation." *Journal of Physics: Conference Series* 1343:012040. ISSN: 1742-6588. https://doi.org/10.1088/1742-6596/1343/1/012040.

Buckley, Niall, Gerald Mills, Samuel Letellier-Duchesne, and Khadija Benis. 2021. "Designing an Energy-Resilient Neighbourhood Using an Urban Building Energy Model." PII: en14154445, *Energies* 14 (15): 4445. ISSN: 1996-1073, accessed August 22, 2023. https://doi.org/10.3390/en14154445.

Bundesregierung der Bundesrepublik Deutschland. 2020. *Verordnung über die Honorare für Architekten- und Ingenieurleistungen (Honorarordnung für Architekten und Ingenieure).* HOAI. Accessed December 22, 2020. https://www.gesetze-im-internet.de/hoai_2013/.

Bundeswehr, ed. 2019. *Bundeswehr Handbuch Gebäudeautomation 4.0: Bw HB GA V4.0.*

Cai, Hanmin, Fazel Khayatian, and Philipp Heer. 2021. "Experiment strategy for evaluating advanced building energy management system." *Journal of Physics: Conference Series* 2042 (1): 012030. ISSN: 1742-6588. https://doi.org/10.1088/1742-6596/2042/1/012030. https://iopscience.iop.org/article/10.1088/1742-6596/2042/1/012030/pdf.

Cañas, Héctor, Josefa Mula, Manuel Díaz-Madroñero, and Francisco Campuzano-Bolarín. 2021. "Implementing Industry 4.0 principles." PII: S0360835221002837, *Computers & Industrial Engineering* 158:107379. ISSN: 03608352. https://doi.org/10.1016/j.cie.2021.107379.

Clarke, J.A, J. Cockroft, S. Conner, J.W Hand, N.J Kelly, R. Moore, T. O'Brien, and P. Strachan. 2002. "Simulation-assisted control in building energy management systems." PII: S0378778802000683, *Energy and Buildings* 34 (9): 933–940. ISSN: 03787788, accessed July 24, 2023. https://doi.org/10.1016/S0378-7788(02)00068-3.

Clausen, Anders, Krzysztof Arendt, Aslak Johansen, Fisayo Caleb Sangogboye, Mikkel Baun Kjærgaard, Christian T. Veje, and Bo Nørregaard Jørgensen. 2021. "A digital twin framework for improving energy efficiency and occupant comfort in public and commercial buildings." PII: 153, *Energy Informatics* 4 (S2). Accessed October 16, 2023. https://doi.org/10.1186/s42162-021-00153-9.

CODESYS. *CODESYS.* https://www.codesys.com/.

Da Silva, Martim Afonso Maia Henriques. 2018. "OPC UA support for Beremiz softPLC." Accessed October 6, 2022. https://repositorio-aberto.up.pt/bitstream/10216/115498/2/284527.pdf.

Davila Delgado, Juan Manuel, and Lukumon Oyedele. 2021. "Digital Twins for the built environment: learning from conceptual and process models in manufacturing." PII: S1474034621000859, *Advanced Engineering Informatics* 49:101332. ISSN: 14740346, accessed August 22, 2023. https://doi.org/10.1016/j.aei.2021.101332.

Deutscher Bundestag. 2020. *Gesetz zur Vereinheitlichung des Energieeinsparrechts für Gebäude und zur Änderung weiterer Gesetze.* GEG. Accessed December 11, 2020.

Domingues, Pedro, Paulo Carreira, Renato Vieira, and Wolfgang Kastner. 2016. "Building automation systems: Concepts and technology review." PII: S0920548915001361, *Computer Standards & Interfaces* 45:1–12. ISSN: 09205489, accessed September 29, 2022. https://doi.org/10.1016/j.csi.2015.11.005.

Drath, Rainer, Christian Mosch, Stefan Hoppe, Andreas Faath, Erich Barnstedt, Bernd Fiebiger, and Wolfgang Schlögl. 2023. "Diskussionspapier – Interoperabilität mit der Verwaltungsschale, OPC UA und AutomationML: Zielbild und Handlungsempfehlungen für industrielle Interoperabilität," https://opcfoundation.org/wp-content/uploads/2023/04/Diskussionspapier-Zielbild-und-Handlungsempfehlungen-fur-industrielle-Interoperabilitat-5.3-protected.pdf.

Drgoňa, Ján, Javier Arroyo, Iago Cupeiro Figueroa, David Blum, Krzysztof Arendt, Donghun Kim, Enric Perarnau Ollé, et al. 2020. "All you need to know about model predictive control for buildings." PII: S1367578820300584, *Annual Reviews in Control* 50:190–232. ISSN: 13675788, accessed August 8, 2023. https://doi.org/10.1016/j.arcontrol.2020.09.001.

Egger, Martin, Kerstin Hausknecht, Thomas Liebich, and Jakob Przbylo. 2013. *BIM-Leitfaden für Deutschland.* Accessed October 15, 2023. https://www.bbsr.bund.de/BBSR/DE/forschung/programme/zb/Auftragsforschung/3Rahmenbedingungen/2013/BIMLeitfaden/Endbericht.pdf?___blob=publicationFile&v=2.

Environmental Design Solutions. 2023. *TAS.* V. 9.5. https://www.edsltas.com/.

EQUA Simulation, ed. 2010a. *Validation of IDA Indoor Climate and Energy 4.0 build 4 with respect to ANSI/ASHRAE Standard 140-2004.* Accessed September 28, 2023. http://www.equaonline.com/iceuser/validation/ASHRAE140-2004.pdf.

EQUA Simulation, ed. 2010b. *Validation of IDA Indoor Climate and Energy 4.0 with respect to CEN Standards EN 15255-2007 and EN 15265-2007.* Accessed September 28, 2023. http://www.equaonline.com/iceuser/validation/CEN_VALIDATION_EN_15255_AND_15265.pdf.

European Commission, ed. 2020. *Energy efficiency in buildings.* https://commission.europa.eu/system/files/2020-03/in_focus_energy_efficiency_in_buildings_en.pdf.

European Parliament and Council of the European Union. 2018. *Directive (EU) 2018/844 of the European Parliament and of the Council of 30 May 2018 amending Directive 2010/31/EU on the energy performance of buildings and Directive 2012/27/EU on energy efficiency.* EPBD. Accessed September 18, 2019.

Fathollahzadeh, Mohammad Hassan, and Paulo Cesar Tabares-Velasco. 2020. "Building control virtual test bed and functional mock-up interface standard: comparison in the context of campus energy modelling and control." *Journal of Building Performance Simulation* 13 (4): 456–471. ISSN: 1940-1493, accessed September 3, 2023. https://doi.org/10.1080/19401493.2020.1769191.

Fernandez, Nicholas E.P., Srinivas Katipamula, Weimin Wang, YuLong Xie, Mingjie Zhao, and Charles D. Corbin. 2017. *Impacts of Commercial Building Controls on Energy Savings and Peak Load Reduction.* Accessed October 28, 2022. https://doi.org/10.2172/1400347. https://www.pnnl.gov/main/publications/external/technical_reports/PNNL-25985.pdf.

Fidelix. *FxEditor.* https://support.fidelix.com/en/knowledge/fxeditor.

Fisch, Norbert, Stefan Plesser, Maik Wussler, and David Sauss. 2017. *Spec&Check Gebäudeautomation: Entwicklung und Erprobung einer Methodik zur Beschreibung, Abnahme und Überwachung von Funktionen der Gebäudeautomation.* Vol. F 3043. Forschungsinitiative Zukunft Bau. Stuttgart: Fraunhofer IRB Verlag. ISBN: 978-3-7388-0052-4, accessed February 20, 2020.

Fütterer, Johannes Peter, Thomas Peter Schild, and Dirk Müller. 2017. *Gebäudeautomationssysteme in der Praxis* [in de]. Accessed January 4, 2021. https://doi.org/10.18154/RWTH-2017-05671.

Granderson, J., G. Lin, R. Singla, E. Mayhorn, P. Ehrlich, D. Vrabie, and S. Frank. 2018. *Commercial Fault Detection and Diagnostics Tools: What They Offer, How They Differ, and What's Still Needed.* Accessed October 28, 2022. https://doi.org/10.20357/B7V88H.

Grieves, Michael. 2014. *Digital Twin: Manufacturing Excellence through Virtual Factory Replication.* A Whitepaper. Accessed March 8, 2023. https://www.researchgate.net/publication/275211047_Digital_Twin_Manufacturing_Excellence_through_Virtual_Factory_Replication.

Grieves, Michael, and John Vickers. 2017. "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems." In *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches,* 1st ed. 2017, edited by Anabela Alves, Shannon Flumerfelt, and Franz-Josef Kahlen. Cham: Springer International Publishing / Imprint: Springer. Accessed July 26, 2023.

Gunay, Burak, Weiming Shen, and Chunsheng Yang. 2017. "Characterization of a building's operation using automation data: A review and case study." PII: S0360132317301336, *Building and Environment* 118:196–210. ISSN: 03601323, accessed September 3, 2023. https://doi.org/10.1016/j.buildenv.2017.03.035.

Hanssen, Dag Håkon. 2015. *Programmable logic controllers: A practical approach to IEC 61131-3 using CODESYS* [in eng]. Hanssen, Dag Håkon, (author.) Hanssen, Dag Håkon (VerfasserIn) Lufkin, Dan (ÜbersetzerIn). Chichester, West Sussex: Wiley. ISBN: 9781118949245, accessed October 3, 2023. https://onlinelibrary.wiley.com/doi/pdfdirect/10.1002/9781118949214?download=true.

Harish, V.S.K.V., and Arun Kumar. 2016. "A review on modeling and simulation of building energy systems." PII: S1364032115014239, *Renewable and Sustainable Energy Reviews* 56:1272–1292. ISSN: 13640321, accessed September 3, 2023. https://doi.org/10.1016/j.rser.2015.12.040.

Hensen, Jan, and Roberto Lamberts, eds. 2019. *Building performance simulation for design and operation* [in eng]. Expanded second edition. Hensen, Jan (HerausgeberIn) Lamberts, Roberto (HerausgeberIn). London and New York: Routledge. ISBN: 978-0-429-40229-6. http://site.ebrary.com/lib/alltitles/Doc?id=10603161.

Hjorth, Hans-Olof Karlsson, Roger Antonsson, Lin Liljefors, Mikael Näslund, Linda Lagnerö, and Emma Svensson. 2021. *Implementat ion of the EPBD Sweden: Status in 2021.* Accessed June 10, 2023. https://epbd-ca.eu/wp-content/uploads/2022/10/Implementation-of-the-EPBD-in-Sweden.pdf.

Horn, P., S. Hauer, A. Bres, K. Eder, I. Lindmeier, and A. Frey. 2019. "Educated commissioning and operation of a complex nearly zero energy office building with the help of dynamic thermal HVAC-simulations - a best practice report from the Austrian postal service headquarter Post am Rochus." *Journal of Physics: Conference Series* 1343:012137. ISSN: 1742-6588, accessed January 10, 2021. https://doi.org/10.1088/1742-6596/1343/1/012137. https://iopscience.iop.org/article/10.1088/1742-6596/1343/1/012137/pdf.

Horward, Ann-Cathrin, and Jonas Rosenberger. 2020. *Implementat ion of the EPBD Germany: Status in 2020.* Accessed June 10, 2023. https://epbd-ca.eu/wp-content/uploads/2022/10/Implementation-of-the-EPBD-in-Germany-2020.pdf.

Hosamo, Haidar Hosamo, Paul Ragnar Svennevig, Kjeld Svidt, Daguang Han, and Henrik Kofoed Nielsen. 2022. "A Digital Twin predictive maintenance framework of air handling units based on automatic fault detection and diagnostics." PII: S0378778822001591, *Energy and Buildings* 261:111988. ISSN: 03787788, accessed October 28, 2022. https://doi.org/10.1016/j.enbuild.2022.111988.

Huang, Sen, Robert Lutes, Cary A. Faulkner, Draguna L. Vrabie, Srinivas Katipamula, and Wangda Zuo. 2023. “An open-source framework for simulation-based testing of buildings control strategies.” *Journal of Building Performance Simulation,* 1–13. ISSN: 1940-1493, accessed July 24, 2023. https://doi.org/10.1080/19401493.2023.2191220.

Hydeman, Mark, Steven T. Taylor, and Brent Eubanks. 2015. “Control Sequences & Controller Programming: RP-1455 and Guideline 36.” *ASHRAE Journal,* accessed October 13, 2023.

Ihlenburg, Moritz, Gesa Angelika Benndorf, and Nicolas Réhault. 2022. “Methode für eine integrale digitale Repräsentation der technischen Gebäudeausrüstung mit Schwerpunkt auf der Beschreibung der Regelung.” In *Proceedings of BauSim Conference 2022: 9th Conference of IBPSA-Germany and Austria.* BauSIM, Weimar. ISBN: 978-3-00-073975-0.

Ihlenburg, Moritz, Tim Rist, Gesa Angelika Benndorf, and Nicolas Réhault. 2020. “A Hybrid Method for an Integral Function Description of Building Services.” In *BauSIM 2020: 8th Conference of IBPSA Germany and Austria,* edited by Michael Monsberger, Christina Hopfe, Markus Krüger, and Alexander Passer, pp. 513–519. BauSIM, Graz, September 23–25, 2020. Technische Universität Graz. ISBN: 978-3-85125-786-1.

incoord, ed. 2015. *KTH Undervisningshuset: Teknisk beskrivning mätinsamlingssystem.*

Jafari, Mohsen A., Ali Ghofrani, Esmat Zaidan, and Ammar Abulibdeh. 2021. “Improving building energy footprint and asset performance using digital twin technology.” *Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction* 174 (2): 57–65. Accessed August 22, 2023. https://doi.org/10.1680/jsmic.21.00001.

Johra, Hicham, Ekaterina Aleksandrova Petrova, Lasse Rohde, and Michal Zbigniew Pomianowski. 2021. “Digital Twins of Building Physics Experimental Laboratory Setups for Effective E-learning.” *Journal of Physics: Conference Series* 2069 (1): 012190. ISSN: 1742-6588, accessed August 22, 2023. https://doi.org/10.1088/1742-6596/2069/1/012190. https://iopscience.iop.org/article/10.1088/1742-6596/2069/1/012190/pdf.

Jones, David, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. 2020. “Characterising the Digital Twin: A systematic literature review.” PII: S1755581720300110, *CIRP Journal of Manufacturing Science and Technology* 29:36–52. ISSN: 17555817, accessed March 8, 2023. https://doi.org/10.1016/j.cirpj.2020.02.002.

Jorissen, Filip, W. Boydens, and Lieve Helsen. 2019. "Model implementation and verification of the envelope, HVAC and controller of an office building in Modelica." *Journal of Building Performance Simulation* 12 (4): 445–464. ISSN: 1940-1493, accessed September 3, 2023. https://doi.org/10.1080/19401493.2018.1544277.

Jorissen, Filip, Michael Wetter, and Lieve Helsen. 2015. "Simulation Speed Analysis and Improvements of Modelica Models for Building Energy Simulation." In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015,* edited by Peter Fritzson and Hilding Elmqvist, pp. 59–69. Linköping Electronic Conference Proceedings 118. The 11th International Modelica Conference. Linköping: Modelica Association.

Jradi, Muhyiddine, and Jakob Bjørnskov. 2023. "A Digital Twin Platform for Energy Efficient and Smart Buildings Applications." In *2023 Fifth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA),* pp. 1–6. 2023 Fifth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), Zouk Mosbeh, Lebanon, July 5–7, 2023. IEEE. ISBN: 979-8-3503-0995-9, accessed June 9, 2023. https://doi.org/10.1109/ACTEA58025.2023.10194071.

Khajavi, Siavash H., Naser Hossein Motlagh, Alireza Jaribion, Liss C. Werner, and Jan Holmstrom. 2019. "Digital Twin: Vision, Benefits, Boundaries, and Creation for Buildings." *IEEE Access* 7:147406–147419. Accessed February 9, 2023. https://doi.org/10.1109/ACCESS.2019.2946515. https://ieeexplore.ieee.org/ielx7/6287639/8600701/08863491.pdf?tp=&arnumber=8863491&isnumber=8600701&ref=.

Kim, Donghun, Wangda Zuo, James E. Braun, and Michael Wetter. 2013. "Comparisons of building system modeling approaches for control system design." In *Proceedings of BS2013: 13th Conference of International Building Performance Simulation Association,* Chambéry, France, August 26–28, 2013. Accessed February 16, 2023. https://publications.ibpsa.org/proceedings/bs/2013/papers/bs2013_1409.pdf.

Klepeis, N. E., W. C. Nelson, W. R. Ott, J. P. Robinson, A. M. Tsang, P. Switzer, J. V. Behar, S. C. Hern, and W. H. Engelmann. 2001. "The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants" [in eng]. Journal Article Research Support, Non-U.S. Gov't Research Support, U.S. Gov't, Non-P.H.S. *Journal of exposure analysis and environmental epidemiology* 11 (3): 231–252. ISSN: 1053-4245, accessed May 7, 2023. https://doi.org/10.1038/sj.jea.7500165. eprint: 11477521.

Knotzer, Armin, Roberta Pernetti, and Soren Ostergaard Jensen, eds. 2019. *Characterization of energy flexibility in buildings: Characterization of Energy Flexibility in Buildings.* Annex 67 Energy flexible buildings. International Energy Agency. Accessed December 21, 2020.

Kontes, Georgios, Georgios Giannakis, Víctor Sánchez, Pablo de Agustin-Camacho, Ander Romero-Amorrortu, Natalia Panagiotidou, Dimitrios Rovas, Simone Steiger, Christopher Mutschler, and Gunnar Gruen. 2018. "Simulation-Based Evaluation and Optimization of Control Strategies in Buildings." PII: en11123376, *Energies* 11 (12): 3376. ISSN: 1996-1073, accessed August 22, 2023. https://doi.org/10.3390/en11123376.

Korenhof, Paulan, Vincent Blok, and Sanneke Kloppenburg. 2021. "Steering Representations—Towards a Critical Understanding of Digital Twins." PII: 484, *Philosophy & Technology* 34 (4): 1751–1773. ISSN: 2210-5433, accessed August 22, 2023. https://doi.org/10.1007/s13347-021-00484-1.

Kramer, R. P., A.W.M. van Schijndel, and H. L. Schellen. 2017. "The importance of integrally simulating the building, HVAC and control systems, and occupants' impact for energy predictions of buildings including temperature and humidity control: validated case study museum Hermitage Amsterdam." *Journal of Building Performance Simulation* 10 (3): 272–293. ISSN: 1940-1493, accessed February 16, 2023. https://doi.org/10.1080/19401493.2016.1221996.

Kritzinger, Werner, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. 2018. "Digital Twin in manufacturing: A categorical literature review and classification." PII: S2405896318316021, *IFAC-PapersOnLine* 51 (11): 1016–1022. ISSN: 24058963, accessed August 23, 2023. https://doi.org/10.1016/j.ifacol.2018.08.474.

Kropf, Sven, and Gerhard Zweifel. *Validation of the Building Simulation Program IDA-ICE According to CEN 13791 „Thermal Performance of Buildings - Calculation of Internal Temperatures of a Room in Summer Without Mechanical Cooling - General Criteria and Validation Procedures“.* Accessed September 28, 2023. http://www.equaonline.com/iceuser/validation/ICE_vs_prEN%2013791.pdf.

Kümpel, Alexander, Jens Teichmann, Paul Mathis, and Dirk Müller. 2022. "Modular hydronic subsystem models for testing and improving control algorithms of air-handling units." PII: S2352710222004521, *Journal of Building Engineering* 53:104439. ISSN: 23527102, accessed September 3, 2023. https://doi.org/10.1016/j.jobe.2022.104439.

Lechner, Raphael, Nicholas O'Connell, Thorsten Meierhofer, and Markus Brautsch. 2018. *Entwicklung, Umsetzung und Bewertung optimierter Monitoring-, Betriebs- und Regelstrategien für Blockheizkraftwerke: [Abschlussbericht. Bearbeitungszeitraum: 11/2014 12/2016 Aktenzeichen: SWD-10.08.18.7-14]* [in ger]. Vol. F 3058. Forschungsinitiative Zukunft Bau. Stuttgart: Fraunhofer IRB Verlag. Accessed January 17, 2020.

Leva, Alberto, Filippo Donida, Marco Bonvini, and Lorenzo Ravelli. 2008. "Modelica library for logic control systems written in the FBD language." In *Proceedings of the 6th International Modelica Conference: Volume 1.* Accessed February 18, 2023. https://modelica.org/events/modelica2008/Proceedings/sessions/session2a3.pdf.

Lin, W. D., and M. Y. H. Low. 2021. "Design and Development of a Digital Twin Dashboards System Under Cyber-physical Digital Twin Environment." In *IEEE IEEM 21 virtual: 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) : 13-16 December 2021,* pp. 1716–1720. 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, Singapore, December 13–16, 2021. Piscataway, NJ: Institute of Electrical and Electronics Engineers, IEEE. ISBN: 978-1-6654-3771-4, accessed August 23, 2023. https://doi.org/10.1109/IEEM50564.2021.9672870.

Loutzenhiser, Peter, Heinrich Manz, and Gregory Maxwell. 2007. *Empirical Validations of Shading/Daylighting/Load Interactions in Building Energy Simulation Tools: A Report for the Internation Energy Agency's SHC Task 34 / ECBCS Annex 43 Project C.* Accessed September 28, 2023. http://www.equaonline.com/iceuser/validation/IEATask34.pdf.

Lu, Qiuchen, Xiang Xie, Ajith Kumar Parlikad, and Jennifer Mary Schooling. 2020. "Digital twin-enabled anomaly detection for built asset monitoring in operation and maintenance." PII: S0926580520303654, *Automation in Construction* 118:103277. ISSN: 09265805, accessed February 9, 2023. https://doi.org/10.1016/j.autcon.2020.103277.

Lydon, G. P., S. Caranovic, I. Hischier, and A. Schlueter. 2019. "Coupled simulation of thermally active building systems to support a digital twin." PII: S0378778819305201, *Energy and Buildings* 202:109298. ISSN: 03787788, accessed July 26, 2023. https://doi.org/10.1016/j.enbuild.2019.07.015.

Maghnie, Marwa, Felix Stegemerten, Alexander Kümpel, and Dirk Muller. 2022. "Evaluating Hardware Building Control in the Cloud." In *Proceedings of BauSim Conference 2022: 9th Conference of IBPSA-Germany and Austria.* BauSIM, Weimar. ISBN: 978-3-00-073975-0.

Maier, Laura, David Jansen, Fabian Wüllhorst, Martin Kremer, Alexander Kümpel, Tobias Blacha, and Dirk Müller. 2023. "AixLib: an open-source Modelica library for compound building energy systems from component to district level with automated quality management." *Journal of Building Performance Simulation,* 1–24. ISSN: 1940-1493, accessed February 10, 2023. https://doi.org/10.1080/19401493.2023.2250521.

Mansson, Lars-Göran, and McIntyre Don. 1997. *Building Energy Management Systems: A Summary of IEA Annexes 16 & 17.* Technical Synthesis Report. Accessed July 10, 2023. https://www.iea-ebc.org/Data/publications/EBC_Annex_16-17_tsr.pdf.

Marchione, Piergiorgio, and Francesco Ruperto. 2022. "Prototyping a digital twin – a case study of a 'U-shaped' military building." *International Journal of Energy Production and Management* 5 (4): 83–94. ISSN: 2056-3272, accessed August 22, 2023. https://doi.org/10.2495/EQ-V7-N1-83-94.

Marcos, Marga, Elisabet Estevez, Federico Perez, and Eelco Der Wal. 2009. "XML exchange of control programs." *IEEE Industrial Electronics Magazine* 3 (4): 32–35. ISSN: 1932-4529, accessed April 10, 2023. https://doi.org/10.1109/MIE.2009.934794.

Mattson, Sven Erik, and Hilding Elmqvist. 1997. "Modelica - An international effort to design the next generation modeling language." In *7th IFAC Symposium on Computer Aided Control Systems Design.* CACSD'97, Gent, Belgium, April 28–30, 1997. Accessed October 15, 2023. https://modelica.org/papers/CACSD97Modelica.pdf.

Mishra, Sandipan, and John T. Wen, eds. 2018. *Intelligent Building Control Systems: A Survey of Modern Building Control and Sensing Strategies.* 1st ed. 2018. Advances in Industrial Control. Mishra, Sandipan (editor.) Wen, John T (editor.) Cham: Springer International Publishing. Accessed February 18, 2023.

Moosberger, Sven. 2007. *IDA ICE CIBSE-Validation: Test of IDA Indoor Climate and Energy version 4.0 according to CIBSE TM33, issue 3.* Accessed September 28, 2023. http://www.equaonline.com/iceuser/validation/ICE-Validation-CIBSE_TM33.pdf.

Mühlbauer, Tobias. 2015a. *OSCAT: BASIC:LIBRARY.* Version 3.33. Accessed November 10, 2022. http://www.oscat.de/component/jdownloads/send/2-oscat-basic/9-oscat-basic333-en.html?Itemid=0.

Mühlbauer, Tobias. 2015b. *OSCAT: Building: LIBRARY.* Version 1.00. Accessed November 10, 2022. http://www.oscat.de/component/jdownloads/send/5-oscat-building/27-oscat-building100-en.html?Itemid=0.

Nicolai, Andreas, and Anne Paepcke. 2017. "Co-Simulation between detailed building energy performance simulation and Modelica HVAC component models." In *Proceedings of the 12th International Modelica Conference.* Linköping Electronic Conference Proceedings. The 12th International Modelica Conference, Prague, Czech Republic, May 15–16, 2017. Linköping University Electronic Press. https://www.bauklimatik-dresden.de/nandrad/doc/Nicolai_Modelica_NANDRAD_CoSim_2017.pdf.

Nytsch-Geusen, Christoph, Werner Kaul, Jörg Rädler, and Lucas Westermann. 2019. "The Digital Twin as a Base for the Design of Building Control Strategies." In *Proceedings of Building Simulation 2019: 16th Conference of IBPSA,* edited by Vincenzo Corrado, Enrico Fabrizio, Andrea Gasparella, and Francesco Patuzzi. Building Simulation Conference proceedings. Building Simulation 2019, Rome, Italy, September 2–4, 2019. IBPSA.

OPC Foundation, ed. 2020. *OPC UA for Field Level Communication OPC UA for Field Level Communications: A Theory of Operation A Theory of Operation.* Accessed October 10, 2023. https://www.hannovermesse.de/apollo/hannover_messe_2023/obs/Binary/A1229156/1229156_03798544.pdf.

OPC Foundation, ed. 2023. *Digital Twin Consortium and OPC Foundation Announce Liaison Agreement.* Orlando, FL, February 6, 2023.

openHAB Foundation. 2023. *openHAB.* Accessed November 5, 2023. https://www.openhab.org/.

Pang, Xiufeng, Mary A. Piette, and Nan Zhou. 2017. "Characterizing variations in variable air volume system controls." PII: S0378778816315754, *Energy and Buildings* 135:166–175. ISSN: 03787788, accessed June 10, 2023. https://doi.org/10.1016/j.enbuild.2016.11.031.

Peng, Yang, Ming Zhang, Fangqiang Yu, Jinglin Xu, and Shang Gao. 2020. "Digital Twin Hospital Buildings: An Exemplary Case Study through Continuous Lifecycle Integration." Rodrigues, Hugo (Editor) PII: 8846667 Rodrigues, Hugo (Editor) PII: 8846667, *Advances in Civil Engineering* 2020:1–13. ISSN: 1687-8086. https://doi.org/10.1155/2020/8846667.

Perno, Matteo, Lars Hvam, and Anders Haug. 2022. "Implementation of digital twins in the process industry: A systematic literature review of enablers and barriers." PII: S0166361521001652, *Computers in Industry* 134:103558. ISSN: 01663615, accessed July 26, 2023. https://doi.org/10.1016/j.compind.2021.103558.

Plesser, Stefan. 2018. *Quality Management for Building Performance: Closing the Gap between design and operation.* Accessed February 9, 2023. https://www.quantum-project.eu/fileadmin/News_Images/1-2_Plesser_QUANTUM_FutureBuild_190305.pdf.

Plesser, Stefan, Norbert Fisch, Claas Pinkernell, Thomas Kurpick, and Bernhard Rumpe. 2010. "The Energy Navigator: A Web Based Platform For Functional Quality Management in Buildings." In *Proceedings of the 10th International Conference for Enhanced Building Operation (ICEBO).* International Conference for Enhanced Building Operation (ICEBO), Kuwait City, October 26–28, 2010. Accessed December 14, 2020.

Rasheed, Adil, Omer San, and Trond Kvamsdal. 2020. "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective." *IEEE Access* 8:21980–22012. Accessed March 8, 2023. https://doi.org/10.1109/ACCESS.2020.2970143. https://ieeexplore.ieee.org/ielx7/6287639/8948470/08972429.pdf?tp=&arnumber=8972429&isnumber=8948470&ref=.

Red Hat, ed. 2023. "Understanding digital twin environments: An overview of architecture and solution implementation." Accessed August 23, 2023. https://www.redhat.com/rhdc/managed-files/co-understanding-digital-twin-environments-detail-f30893-202201-en_0.pdf.

Reddy, T. Agami. 2011. *Applied data analysis and modeling for energy engineers and scientists.* New York: Springer. Accessed May 8, 2023.

Roth, Amir, Michael Wetter, Kyle Benne, David Blum, Yan Chen, Gabriel Fierro, Marco Pritoni, Avijit Saha, and Draguna Vrabie. 2022. *Towards Digital and Performance-Based Supervisory HVAC Control Delivery.* Accessed September 15, 2023. https://doi.org/10.20357/B70G62.

Roy, Rohan Basu, Debasish Mishra, Surjya K. Pal, Tapas Chakravarty, Satanik Panda, M. Girish Chandra, Arpan Pal, Prateep Misra, Debashish Chakravarty, and Sudip Misra. 2020. “Digital twin: current scenario and a case study on a manufacturing process.” PII: 5306, *The International Journal of Advanced Manufacturing Technology* 107 (9-10): 3691–3714. ISSN: 0268-3768, accessed March 8, 2023. https://doi.org/10.1007/s00170-020-05306-w.

Royapoor, Mohammad, Anu Antony, and Tony Roskilly. 2018. “A review of building climate and plant controls, and a survey of industry perspectives.” PII: S0378778817318522, *Energy and Buildings* 158:453–465. ISSN: 03787788, accessed September 29, 2022. https://doi.org/10.1016/j.enbuild.2017.10.022.

Ruepp, Daniel, Simon Ollander, Per Sahlin, Markus Hogberg, Sven Moosberger, and Dagmar Jähnig. 2022. “On the track of optimized building operation.” In *ISEC: 2nd Internations Sustainable Energy Conference 2022,* edited by Werner Weiss. Conference for Renewable Heating and Cooling in Integrated Urban and Industrial Energy Systems, Graz, Austria, April 5–7, 2022. Accessed July 11, 2022.

Sacks, Rafael, Ioannis Brilakis, Ergo Pikas, Haiyan Sally Xie, and Mark Girolami. 2020. “Construction with digital twin information systems.” PII: S2632673620000167, *Data-Centric Engineering* 1. Accessed February 9, 2023. https://doi.org/10.1017/dce.2020.16.

Sahlin, Per. 2000. *The Methods of 2020 for Building Envelope and HVAC Systems Simulation - Will the Present Tools Survive?* Accessed May 22, 2022. https://www.equa.se/dncenter/Dublin2000.pdf.

Sahlin, Per, Axel Bring, and Lars Eriksson. 2009. “Real controllers in the context of full-building, whole-year simulation.” In *Eleventh International IBPSA Conference,* Glasgow, Scotland, July 27–30, 2009. Accessed February 16, 2023. https://www.aivc.org/sites/default/files/BS09_0072_79.pdf.

Sahlin, Per, Lars Eriksson, and Mika Vuolle. 2003. “Will equation-based building simulation make it?-experiences from the introduction of IDA Indoor Climate And Energy.” In *Building Simulation 2003.* https://www.academia.edu/16918862.

Sahlin, Per, and Alexey Lebedev. 2018. *Benchmark building and energy system models.* Accessed November 9, 2023. https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=0CAIQw7AJahcKEwjYmKKdhKKBAxUAAAAAHQAAAAAQAg&url=https%3A%2F%2Fitea4.org%2Fproject%2Fworkpackage%2Fdocument%2Fdownload%2F4998%2FD5.4.%2520OPENCPS%2520-%2520Benchmark%2520Building%2520And%2520Energy%2520System%2520Models%2520M24.pdf&psig=AOvVaw1K6d0UgZXoxwEz-038yRqc&ust=1694503203606659&opi=89978449.

Sahlin, Per, Patrik Skogqvist, and Markus Högberg. 2018. *IEC 61131-3 code simulation for Software-in-the-loop PLC testing with EQUA tools.* Accessed January 20, 2021. https://itea3.org/project/workpackage/document/download/5090/D3.5%20OPENCPS%20-%20IEC61131-3%20Code%20Simulation%20For%20SiL%20PLC%20Testing%20M36.pdf.

Sahlin, Per, and Edward Sowell. 1989. "A Neutral Format for Building Simulation Models." In *IBPSA Building Simulation '89 conference,* Vancouver. Accessed October 24, 2019.

Salsbury, Timothy I. 2005. "A SURVEY OF CONTROL TECHNOLOGIES IN THE BUILDING AUTOMATION INDUSTRY." PII: S1474667016374092, *IFAC Proceedings Volumes* 38 (1): 90–100. ISSN: 14746670, accessed October 14, 2023. https://doi.org/10.3182/20050703-6-CZ-1902.01397.

Schaper, Hauke. 2011. "Translation of Sequential Function Charts to Transition Systems." Accessed September 14, 2022.

Schild, Thomas Peter, Matti Rademacher, Marc Axel Baranski, and Dirk Müller. 2019. *Gebäudeautomationssysteme in der Praxis: Höhere Regelalgorithmen* [in de]. Accessed January 4, 2021. https://doi.org/10.18154/RWTH-2019-00911. http://publications.rwth-aachen.de/record/753892/files/753892.pdf.

Schluck, Thomas, Philipp Kräuchi, and Matthias Sulzer. 2015. "Non-linear thermal networks: How can a meshed network improve energy efficiency?" [In en]. In *Proceedings of International Conference CISBAT 2015: Future Buildings and Districts,* Sustainability from Nano to Urban Scale, edited by Jean-Louis Scartezzini. CISBAT. https://doi.org/10.5075/epfl-cisbat2015-779-784.

Schneider, Georg Ferdinand. 2019. "Semantic Modelling of Control Logic in Automation Systems: Knowledge-Based Support of the Engineering and Operation of Control Logic in Building and Industrial Automation Systems." Dissertation, Karlsruher Institut für Technologie. Accessed October 14, 2020.

Schweiger, G., C. Gomes, G. Engel, I. Hafner, J. Schoeggl, A. Posch, and T. Nouidui. 2019. "An empirical survey on co-simulation: Promising standards, challenges and research needs." PII: S1569190X1930053X, *Simulation Modelling Practice and Theory* 95:148–163. ISSN: 1569190X, accessed October 24, 2023. https://doi.org/10.1016/j.simpat.2019.05.001.

Semeraro, Concetta, Mario Lezoche, Hervé Panetto, and Michele Dassisti. 2021. "Digital twin paradigm: A systematic literature review." PII: S0166361521000762, *Computers in Industry* 130:103469. ISSN: 01663615. https://doi.org/10.1016/j.compind.2021.103469.

Serale, Gianluca, Massimo Fiorentini, Alfonso Capozzoli, Daniele Bernardini, and Alberto Bemporad. 2018. "Model Predictive Control (MPC) for Enhancing Building and HVAC System Energy Efficiency: Problem Formulation, Applications and Opportunities." PII: en11030631, *Energies* 11 (3): 631. ISSN: 1996-1073, accessed August 25, 2023. https://doi.org/10.3390/en11030631.

Shafto, Mike, Mike Conroy, Rich Doyle, Ed Glaessgen, Chris Kemp, Jacqueline LeMoigne, and Lui Wang. 2012. *Modeling, Simulation, Information Technology & Processing Roadmap: Technology Area 11.* Accessed July 26, 2023. https://www.nasa.gov/sites/default/files/501321main_TA11-ID_rev4_NRC-wTASR.pdf.

Simonsson, Johan, Khalid Tourkey Atta, Gerald Schweiger, and Wolfgang Birk. 2021. "Experiences from City-Scale Simulation of Thermal Grids." PII: resources10020010, *Resources* 10 (2): 10. Accessed August 22, 2023. https://doi.org/10.3390/resources10020010.

Simros, Markus, Stefan Theurich, and Martin Wollschlaeger. 2012. "Programming embedded devices in IEC 61131-languages with industrial PLC tools using PLCopen XML." In *10th Portuguese Conference on Automatic Control,* Funchal, July 16–18, 2012.

Singh, Maulshree, Evert Fuenmayor, Eoin Hinchy, Yuansong Qiao, Niall Murray, and Declan Devine. 2021. "Digital Twin: Origin to Future." PII: asi4020036, *Applied System Innovation* 4 (2): 36. Accessed August 22, 2023. https://doi.org/10.3390/asi4020036.

Sommer, Tobias, Matthias Sulzer, Michael Wetter, Artem Sotnikov, Stefan Mennel, and Christoph Stettler. 2020. "The reservoir network: A new network topology for district heating and cooling." PII: S0360544220305259, *Energy* 199:117418. ISSN: 03605442, accessed June 10, 2023. https://doi.org/10.1016/j.energy.2020.117418.

Sousa, M. de. 2002. "MatPLC-the truly open automation controller." In *Proceedings of the 28th annual conference of the IEEE Industrial Electronics Society: Sevilla, Spain, November 5 - 8, 2002,* pp. 2278–2283. 28th Annual Conference of the IEEE Industrial Electronics Society, Sevilla, Spain, November 5–8, 2002. Piscataway, NJ: Institute of Electrical and Electronics Engineers and IEEE Industrial Electronics Society, IEEE Operations Center. ISBN: 0-7803-7474-6, accessed July 31, 2023. https://doi.org/10.1109/IECON.2002.1185327.

Souza, Althea de. *Digital Twin: Looking Behind the Buzzwords.* Accessed July 26, 2023. https://www.nafems.org/downloads/dropbox/resource_center/magazine/bm_apr_18_3.pdf.

Spudys, Paulius, Nicholas Afxentiou, Phoebe-Zoe Georgali, Egle Klumbyte, Andrius Jurelionis, and Paris Fokaides. 2023. "Classifying the operational energy performance of buildings with the use of digital twins." PII: S0378778823003365, *Energy and Buildings* 290:113106. ISSN: 03787788, accessed May 24, 2023. https://doi.org/10.1016/j.enbuild.2023.113106.

Stluka, Petr, Karel Mařík, and Petr Endel. 2014. "Advanced Control Solutions for Building Systems." PII: S1474667016416812, *IFAC Proceedings Volumes* 47 (3): 606–611. ISSN: 14746670, accessed February 16, 2023. https://doi.org/10.3182/20140824-6-ZA-1003.02533.

Storek, T., J. Lohmöller, A. Kümpel, Marc Baranski, and D. Müller. 2019. "Application of the open-source cloud platform FIWARE for future building energy management systems." *Journal of Physics: Conference Series* 1343:012063. ISSN: 1742-6588, accessed January 10, 2021. https://doi.org/10.1088/1742-6596/1343/1/012063. https://iopscience.iop.org/article/10.1088/1742-6596/1343/1/012063/pdf.

Tao, Fei, He Zhang, Ang Liu, and A. Y. C. Nee. 2019. "Digital Twin in Industry: State-of-the-Art." *IEEE Transactions on Industrial Informatics* 15 (4): 2405–2415. ISSN: 1551-3203, accessed August 22, 2023. https://doi.org/10.1109/TII.2018.2873186.

Tariq, Rasikh, C. E. Torres-Aguilar, J. Xamán, I. Zavala-Guillén, A. Bassam, Luis J. Ricalde, and O. Carvente. 2022. "Digital twin models for optimization and global projection of building-integrated solar chimney." PII: S0360132322000555, *Building and Environment* 213:108807. ISSN: 03601323, accessed July 26, 2023. https://doi.org/10.1016/j.buildenv.2022.108807.

Togashi, Eisuke, and Masato Miyata. 2019. "Development of building thermal environment emulator to evaluate the performance of the HVAC system operation." *Journal of Building Performance Simulation* 12 (5): 663–684. ISSN: 1940-1493, accessed February 16, 2023. https://doi.org/10.1080/19401493.2019.1601259.

Trčka, Marija, and Jan L.M. Hensen. 2010. "Overview of HVAC system simulation." PII: S0926580509001897, *Automation in Construction* 19 (2): 93–99. ISSN: 09265805, accessed September 3, 2023. https://doi.org/10.1016/j.autcon.2009.11.019.

Trčka, Marija, Jan L.M. Hensen, and Michael Wetter. 2009. "Co-simulation of innovative integrated HVAC systems in buildings." *Journal of Building Performance Simulation* 2 (3): 209–230. ISSN: 1940-1493, accessed September 3, 2023. https://doi.org/10.1080/19401490903051959.

Treado, Stephen, Payam Delgoshaei, and Andrew Windham. 2011. "Simlation-based approaches for building control system design and integration." In *Proceedings of Building Simulation 2011: 12th Conference of International Building Performance Simulation Association,* Sydney, November 14–16, 2011. Accessed February 16, 2023. http://www.ibpsa.org/proceedings/BS2011/P_1793.pdf.

Tuegel, Eric J., Anthony R. Ingraffea, Thomas G. Eason, and S. Michael Spottswood. 2011. "Reengineering Aircraft Structural Life Prediction Using a Digital Twin." PII: 154798, *International Journal of Aerospace Engineering* 2011:1–14. ISSN: 1687-5966, accessed July 26, 2023. https://doi.org/10.1155/2011/154798. https://downloads.hindawi.com/journals/ijae/2011/154798.pdf?_ga=2.164188159.1187401459.1690364524-2058600187.1690364524.

VanDerHorn, Eric, and Sankaran Mahadevan. 2021. "Digital Twin: Generalization, characterization and implementation." PII: S0167923621000348, *Decision Support Systems* 145:113524. ISSN: 01679236, accessed October 15, 2023. https://doi.org/10.1016/j.dss.2021.113524.

Veichtlbauer, Armin, Martin Ortmayer, and Thomas Heistracher. 2017. “OPC UA integration for field devices.” In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN),* pp. 419–424, Emden, Germany, July 24–26, 2017. Accessed October 10, 2023. https://doi.org/10.1109/INDIN.2017.8104808. https://www.en-trust.at/papers/Veichtlbauer17a.pdf.

Vela Solaris, ed. 2018. *Polysun: Benutzerhandbuch Polysun Software.* Accessed August 21, 2019. https://www.velasolaris.com/wp-content/uploads/2019/02/Tutorial_DE.pdf.

Verdouw, Cor, Bedir Tekinerdogan, Adrie Beulens, and Sjaak Wolfert. 2021. “Digital twins in smart farming.” PII: S0308521X20309070, *Agricultural Systems* 189:103046. ISSN: 0308521X, accessed March 8, 2023. https://doi.org/10.1016/j.agsy.2020.103046.

Viola, Jairo, and YangQuan Chen. 2020. “Digital Twin Enabled Smart Control Engineering as an Industrial AI: A New Framework and Case Study.” In *The 2nd International Conference on Industrial Artificial Intelligence (IAI 2020): 23-25 Oct., 2020, Shenyang, China,* pp. 1–6. 2020 2nd International Conference on Industrial Artificial Intelligence (IAI), Shenyang, China, October 23–25, 2020. Piscataway, NJ: IEEE. ISBN: 978-1-7281-8216-2, accessed July 11, 2023. https://doi.org/10.1109/IAI50351.2020.9262203.

Visier, Jean Christophe, ed. 2004. *Commissioning Tools for Improved Energy Performance: Results of IEA ECBCS ANNEX 40.* Accessed March 7, 2023. https://www.iea-ebc.org/Data/publications/EBC_Annex_40_Commissioning_Tools_for_Improved_Energy_Performance.pdf.

Voss, Karsten, Andreas Wagner, Anton Maas, Sebastian Herkel, Doreen Kalz, and Thomas Lützkendorf, eds. 2016. *Performance von Gebäuden: Kriterien, Konzepte und Erfahrungen* [in ger]. Voss, Karsten (HerausgeberIn) Wagner, Andreas (HerausgeberIn) Maas, Anton (HerausgeberIn) Herkel, Sebastian (HerausgeberIn) Kalz, Doreen (HerausgeberIn) Lützkendorf, Thomas (HerausgeberIn). Stuttgart: Fraunhofer IRB Verlag. ISBN: 978-3-8167-9583-4.

WAGO, ed. 2019. *e!COCKPIT Application note: WAGO-I/O-SYSTEM 750.* HVAC-SYSTEM-MACROs. V. 1.0.1. WAGO GmbH & Co. KG. Accessed November 10, 2022.

WAGO. 2022. *Libraries for Building Automation: Function Block Descriptions for HVAC Functions.* WAGO GmbH & Co. KG. Accessed February 14, 2023.

WAGO. *e!cockpit.* WAGO GmbH & Co. KG. https://www.wago.com/global/automation-technology/discover-software/ecockpit-engineering-software.

Waide, Paul, Jim Ure, Nadia Karagianna, Graham Smith, and Bill Bordass. 2014. *The scope for energy and CO2 savings in the EU through the use of building automation technology: Second edition.* Accessed January 4, 2021. http://neu.eubac.org/fileadmin/eu.bac/BACS_studies_and_reports/2014.06.13_Waide_ECI_-_Energy_and_CO2_savings_BAT.pdf.

Walther, Karl, Isil Kalpkirmaz Rizaoglu, Hale Tugcin Kirant-Mitic, and Ghadeer Derbas. 2021. *Building 2226 on the Test Bench: Simulation study on the relocated 2226 building by Baumschlager Eberle.* https://www.researchgate.net/publication/366976251_Building_2226_on_the_Test_Bench_-_Simulation_study_on_the_relocated_2226_building_by_Baumschlager_Eberle.

Walther, Karl, and Karsten Voss. 2021. "Heat flow metering in building practice - A critical field study for a large industrial building complex." *Journal of Physics: Conference Series* 2042 (1). ISSN: 1742-6588. https://doi.org/10.1088/1742-6596/2042/1/012085.

Wang, Shengwei, and Zhenjun Ma. 2008. "Supervisory and Optimal Control of Building HVAC Systems: A Review." PII: 934120924, *HVAC&R Research* 14 (1): 3–32. ISSN: 1078-9669, accessed July 10, 2023. https://doi.org/10.1080/10789669.2008.10390991.

Ward, Rebecca, Ruchi Choudary, Melanie Jans Singh, Flora Roumpani, Tomas Lazauskas, May Yong, Nicholas Barlow, and Markus Hauru. 2023. "The challenges of using live-streamed data in a predictive digital twin." *Journal of Building Performance Simulation,* 1–22. ISSN: 1940-1493, accessed July 28, 2023. https://doi.org/10.1080/19401493.2023.2187463.

Wetter, Michael. 2009. "Modelica-based modelling and simulation to support research and development in building energy and control systems." *Journal of Building Performance Simulation* 2 (2): 143–161. ISSN: 1940-1493, accessed September 3, 2023. https://doi.org/10.1080/19401490902818259.

Wetter, Michael. 2011. "Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed." *Journal of Building Performance Simulation* 4 (3): 185–203. ISSN: 1940-1493, accessed September 3, 2023. https://doi.org/10.1080/19401493.2010.518631.

Wetter, Michael. 2020. "Lifting the Garage Door on Spawn, an Open-Source BEM-Controls Engine." In *2020 Building Performance Analysis Conference and SimBuild co-organized by ASHRAE and IBPSA-USA,* vol. 9, pp. 518–525. SimBuild Conference. September. Virtual: ASHRAE/IBPSA-USA. https://publications.ibpsa.org/conference/paper/?id=simbuild2020_C063.

Wetter, Michael, Yan Chen, Karthik Devaprasad, Paul Ehrlich, and Jianjun Hu. 2023. "Bridging the gap between building energy modeling and controls implementation – experiences of, and best practice for, coupled HVAC-control modeling." In *The 18th International IBPSA Conference and Exhibition.* Building Simulation, Shanghai, China. Accessed September 14, 2023.

Wetter, Michael, Paul Ehrlich, Antoine Gautier, Milica Grahovac, Philip Haves, Jianjun Hu, Anand Prakash, Dave Robin, and Kun Zhang. 2022. "OpenBuildingControl: Digitizing the control delivery from building energy modeling to specification, implementation and formal verification." PII: S0360544221017497, *Energy* 238:121501. ISSN: 03605442, accessed April 24, 2022. https://doi.org/10.1016/j.energy.2021.121501.

Wetter, Michael, Milica Grahovac, and Jianjun Hu. 2018. "Control Description Language." In *Proceedings of the 1st American Modelica Conference.* Accessed October 15, 2020.

Wetter, Michael, Jianjun Hu, Milica Grahovac, Brent Eubanks, and Philip Haves. 2018. "OpenBuildingControl: Modeling Feedback Control as a Step Towards Formal Design, Specification, Deployment and Verification of Building Control Sequences." In *Proceedings of Building Performance Modeling Conference and SimBuild,* pp. 775–782. 2018 Building Performance Modeling Conference and SimBuild co-organized by ASHRAE and IBPSA-USA, Chicago, IL. Accessed October 15, 2020. https://simulationresearch.lbl.gov/wetter/download/2018-simBuild-OpenBuildingControl.pdf.

Wetter, Michael, Jianjun Hu, Anand Prakash, Paul Ehrlich, Gabe Fierro, Milica Grahovac, Marco Pritoni, Lisa Rivalin, and Dave Robin. 2021. "Modelica-json: Transforming energy models to digitize the control delivery process." In *Proceedings of Building Simulation 2021: 17th Conference of IBPSA.* Building Simulation Conference proceedings. 2021 Building Simulation Conference, Bruges. KU Leuven. Accessed July 10, 2023. https://doi.org/10.26868/25222708.2021.30141. https://simulationresearch.lbl.gov/wetter/download/2021-ibpsa-modelica-json.pdf.

Wilde, Pieter de. 2014. “The gap between predicted and measured energy performance of buildings: A framework for investigation.” *Automation in Construction* 41:40–49. ISSN: 09265805. https://doi.org/10.1016/j.autcon.2014.02.009.

Wilde, Pieter de. 2018. *Building performance analysis.* Hoboken, NJ: Wiley Blackwell.

Wilde, Pieter de. 2023. “Building Performance Simulation in the Brave New World of Artificial Intelligence and Digital Twins: a systematic review.” PII: S0378778823004012, *Energy and Buildings,* 113171. ISSN: 03787788, accessed May 19, 2023. https://doi.org/10.1016/j.enbuild.2023.113171.

Wright, Louise, and Stuart Davidson. 2020. “How to tell the difference between a model and a digital twin.” PII: 147, *Advanced Modeling and Simulation in Engineering Sciences* 7 (1). Accessed July 26, 2023. https://doi.org/10.1186/s40323-020-00147-4.

Xie, Xiang, Jorge Merino, Nicola Moretti, Pieter Pauwels, Janet Yoon Chang, and Ajith Parlikad. 2023. “Digital twin enabled fault detection and diagnosis process for building HVAC systems.” PII: S0926580522005659, *Automation in Construction* 146:104695. ISSN: 09265805, accessed February 16, 2023. https://doi.org/10.1016/j.autcon.2022.104695.

Yoon, Sungmin. 2022. “Virtual sensing in intelligent buildings and digitalization.” PII: S0926580522004484, *Automation in Construction* 143:104578. ISSN: 09265805, accessed October 21, 2023. https://doi.org/10.1016/j.autcon.2022.104578.

Yoon, Sungmin. 2023. “Building digital twinning: Data, information, and models.” PII: S2352710223012007, *Journal of Building Engineering* 76:107021. ISSN: 23527102, accessed October 18, 2023. https://doi.org/10.1016/j.jobe.2023.107021.

Zhang, Kun, David Blum, Hwakong Cheng, Gwelen Paliaga, Michael Wetter, and Jessica Granderson. 2022. “Estimating ASHRAE Guideline 36 energy savings for multizone variable air volume systems using Spawn of EnergyPlus.” *Journal of Building Performance Simulation* 15 (2): 215–236. ISSN: 1940-1493, accessed February 16, 2023. https://doi.org/10.1080/19401493.2021.2021286.

Zhou, Jiehan, Shouhua Zhang, and Mu Gu. 2022. “Revisiting digital twins: Origins, fundamentals, and practices.” PII: 216, *Frontiers of Engineering Management* 9 (4): 668–676. ISSN: 2095-7513, accessed July 26, 2023. https://doi.org/10.1007/s42524-022-0216-2.

Zuo, Wangda, Michael Wetter, Wei Tian, Dan Li, Mingang Jin, and Qingyan Chen. 2016. "Coupling indoor airflow, HVAC, control and building envelope heat transfer in the Modelica Buildings library." *Journal of Building Performance Simulation* 9 (4): 366–381. ISSN: 1940-1493, accessed September 3, 2023. https://doi.org/10.1080/19401493.2015.1062557.

Standards, guidelines and regulations

ASHRAE (American Society of Heating, Refrigeration and Air-Conditioning Engineers). 2021. *High-Performance Sequences of Operation for HVAC Systems.* ASHRAE Guideline 36-2021. Atlanta, GA: American Society of Heating, Refrigeration and Air-Conditioning Engineers.

DIN (Deutsches Institut für Normung). 2004. *Systeme der Gebäudeautomation (GA) - Teil 2: Hardware (ISO 16484-2:2004).* DIN EN ISO 16484-2. Deutsches Institut für Normung, October 1, 2004.

DIN (Deutsches Institut für Normung). 2005. *Systeme der Gebäudeautomation (GA) - Teil 3: Funktionen (ISO 16484-3:2005).* DIN EN ISO 16484-3. Deutsches Institut für Normung, December 1, 2005.

DIN (Deutsches Institut für Normung). 2011. *Systeme der Gebäudeautomation (GA) - Teil 1: Projektplanung und -ausführung (ISO 16484-1:2010).* DIN EN ISO 16484-1. Deutsches Institut für Normung, March 1, 2011.

DIN (Deutsches Institut für Normung). 2014. *Speicherprogrammierbare Steuerungen. Teil 3: Programmiersprachen (IEC 61131-3:2013).* DIN EN 61131-3. Deutsches Institut für Normung, June 1, 2014.

DIN (Deutsches Institut für Normung). 2018a. *Energetische Bewertung von Gebäuden – Berechnung des Nutz-, End- und Primärenergiebedarfs für Heizung, Kühlung, Lüftung, Trinkwarmwasser und Beleuchtung.* 18599-7. DIN V. Deutsches Institut für Normung, September 1, 2018. Accessed May 31, 2021.

DIN (Deutsches Institut für Normung). 2018b. *Energetische Bewertung von Gebäuden – Energiebedarf für Heizung und Kühlung, Innentemperaturen sowie fühlbare und latente Heizlasten – Teil 1: Berechnungsverfahren (ISO 52016-1:2017).* DIN EN ISO 52016-1. Deutsches Institut für Normung, April 1, 2018. Accessed October 26, 2023.

DIN (Deutsches Institut für Normung). 2018c. *Energy efficiency of buildings – Calculation of the net, final and primary energy demand for heating, cooling, ventilation, domestic hot water and lighting – Part 1: General balancing procedures, terms and definitions, zoning and evaluation of energy sources.* DIN V 18599-1. Deutsches Institut für Normung, September 1, 2018.

DIN (Deutsches Institut für Normung). 2019a. *VOB Vergabe- und Vertragsordnung für Bauleistungen - Teil C: Allgemeine Technische Vertragsbedingungen für Bauleistungen (ATV). Gebäudeautomation.* DIN 18386. Deutsches Institut für Normung, September 1, 2019.

DIN (Deutsches Institut für Normung). 2019b. *VOB Vergabe- und Vertragsordnung für Bauleistungen - Teil C: Allgemeine Technische Vertragsbedingungen für Bauleistungen (ATV). Heizungsanlagen und zentrale Wassererwärmungsanlagen.* DIN 18380. Deutsches Institut für Normung, September 1, 2019.

DIN (Deutsches Institut für Normung). 2019c. *VOB Vergabe- und Vertragsordnung für Bauleistungen - Teil C: Allgemeine Technische Vertragsbedingungen für Bauleistungen (ATV). Raumlufttechnische Anlagen.* DIN 18379. Deutsches Institut für Normung, September 1, 2019.

DIN (Deutsches Institut für Normung). 2020. *Verfahren zur Berechnung der Raumheizlast. Teil 1: Nationale Ergänzungen zur DIN EN 12831-1, mit CD-ROM.* DIN/TS 12831-1. Deutsches Institut für Normung, April 1, 2020.

European Commission. 2021. *Proposal for a Directive of the European Parliament and of the Council on the energy performance of buildings (recast).* European Commission. Accessed July 28, 2023. https://eur-lex.europa.eu/resource.html?uri=cellar:c51fe6d1-5da2-11ec-9c6c-01aa75ed71a1.0001.02/DOC_1&format=PDF.

NAMUR (Interessengemeinschaft Automatisierungstechnik der Prozessindustrie e.V.). 2020. *Vergleich Phasenmodell NA 35 und HOAI.* NA 180. Interessengemeinschaft Automatisierungstechnik der Prozessindustrie e.V.

Swedish National Board of Housing, Building and Planning Boverket. 2018. *Boverkets föreskrifter om ändring av verkets föreskrifter och allmänna råd (2016:12) om fastställande av byggnadens energianvändning vid normalt brukande och ett normalår, BFS 2017:6.* Swedish National Board of Housing, Building and Planning Boverket. Accessed November 21, 2022. https://rinfo.boverket.se/BEN/PDF/BFS2017-6-BEN-2.pdf.

VDI (Verein Deutscher Ingenieure). 2008. *Gebäudeautomation (GA). Grafische Darstellung von Steuerungsaufgaben.* VDI 3814-6. Düsseldorf: Verein Deutscher Ingenieure, July 1, 2008.

VDI (Verein Deutscher Ingenieure). 2019a. *Gebäudeautomation (GA) GA-Funktionen Automationsfunktionen.* VDI 3814-3.1:2019. Richtlinie. Verein Deutscher Ingenieure, January 1, 2019.

VDI (Verein Deutscher Ingenieure). 2019b. *Gebäudeautomation (GA). Grundlagen.* VDI 3814-1. Richtlinie. Verein Deutscher Ingenieure, January 1, 2019. Accessed October 22, 2019.

VDI (Verein Deutscher Ingenieure). 2019c. *Gebäudeautomation (GA). Planung Planungsinhalte, Systemintegration und Schnittstellen.* 3814-2.2. Richtlinie. Verein Deutscher Ingenieure, January 1, 2019.

VDI (Verein Deutscher Ingenieure). 2022. *3814-4.3 Gebäudeautomation (GA). Methoden und Arbeitsmittel für Planung, Ausführung und Übergabe. GA-Automationsschema, GA-Funktionsliste, GA-Funktionsbeschreibung.* 3814-4.3. Richtlinie. Verein Deutscher Ingenieure, July 1, 2022.

A Additional figures

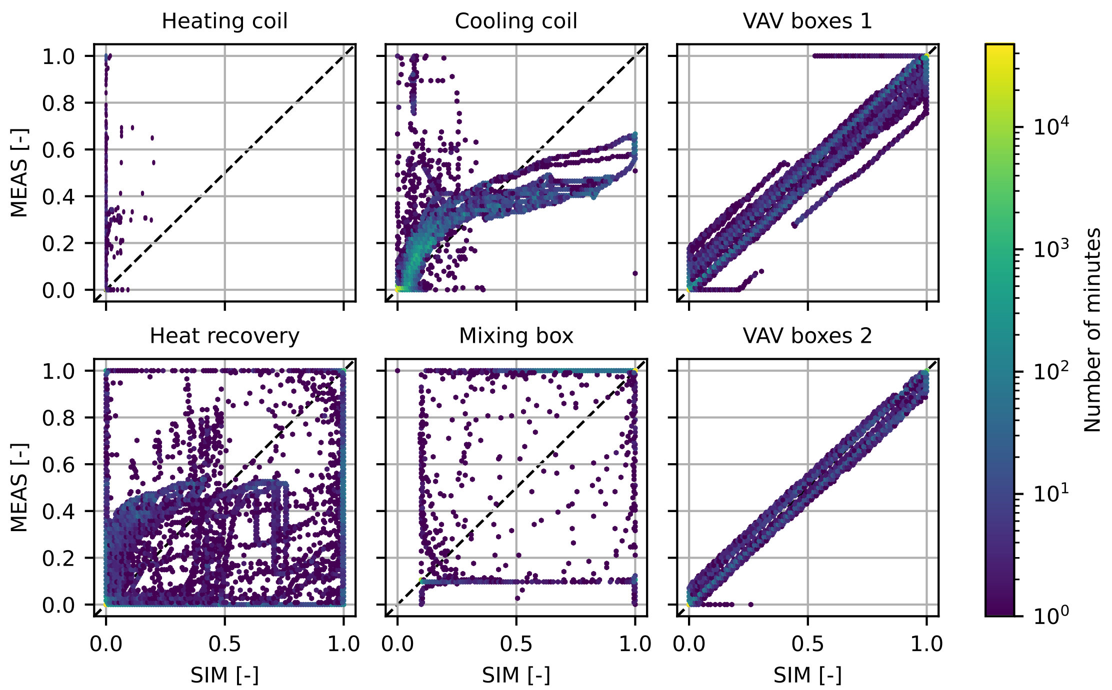


Figure A.1: Implementation 2: Hexagonal binning plots based on 1 minute measured and simulated control signals for heating and cooling coil valves, heat recovery, mixed air dampers, and VAV boxes

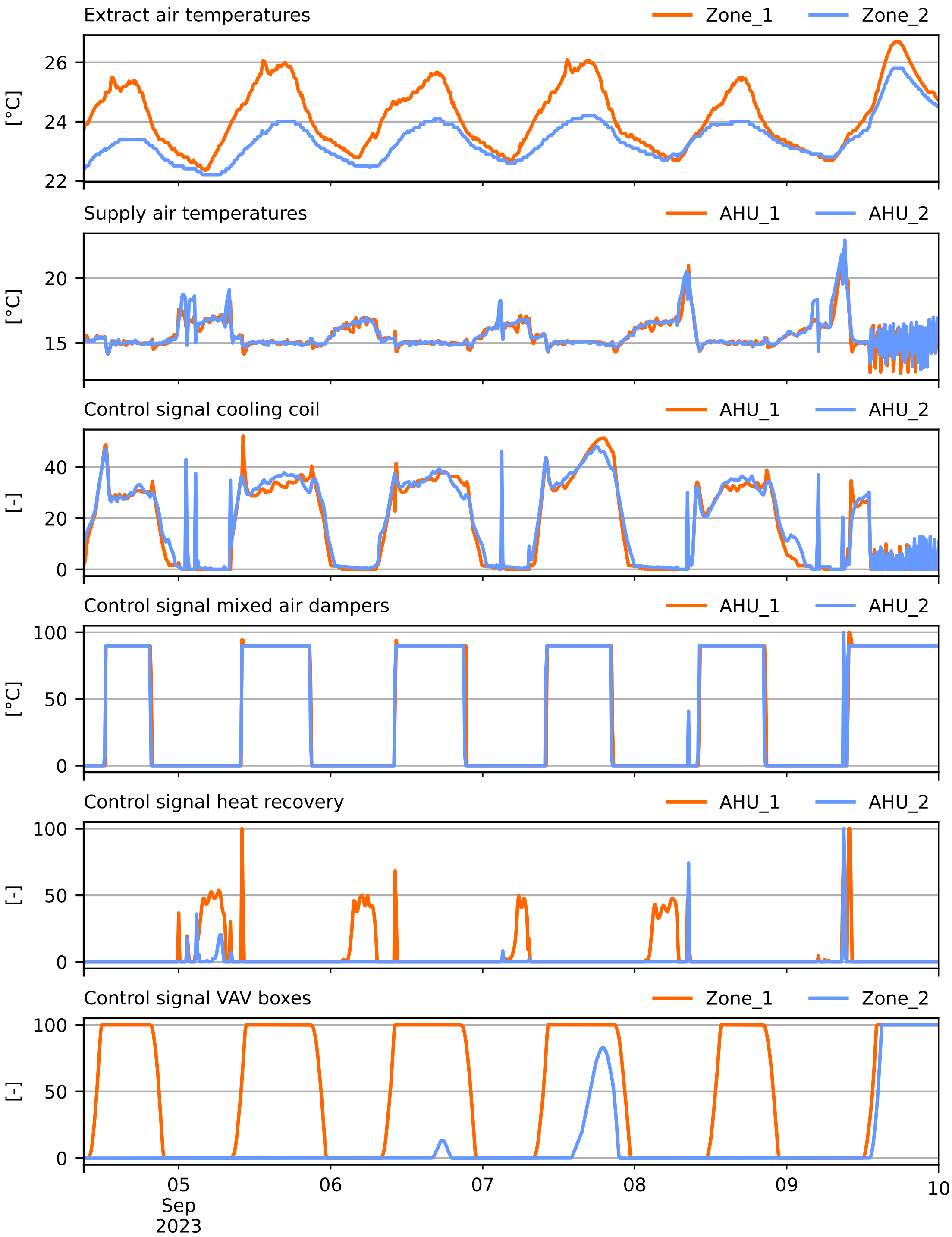


Figure A.2: Implementation 3: Time series of temperatures and control signals for a 7 day period

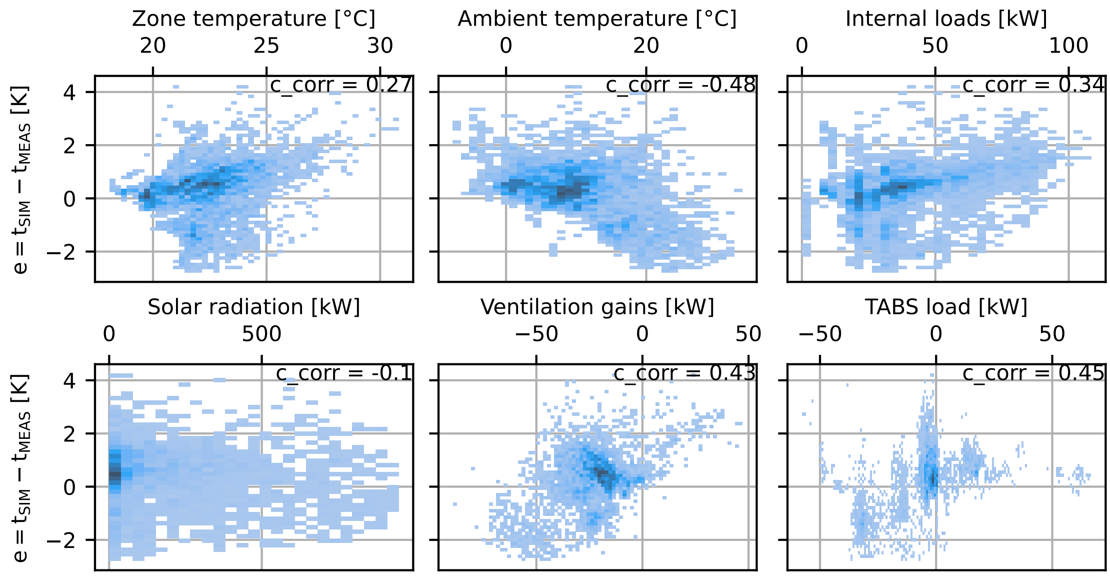


Figure A.3: Zone model comparison for Zone 1: Distribution plots of the error between simulated / measured temperatures and different variables

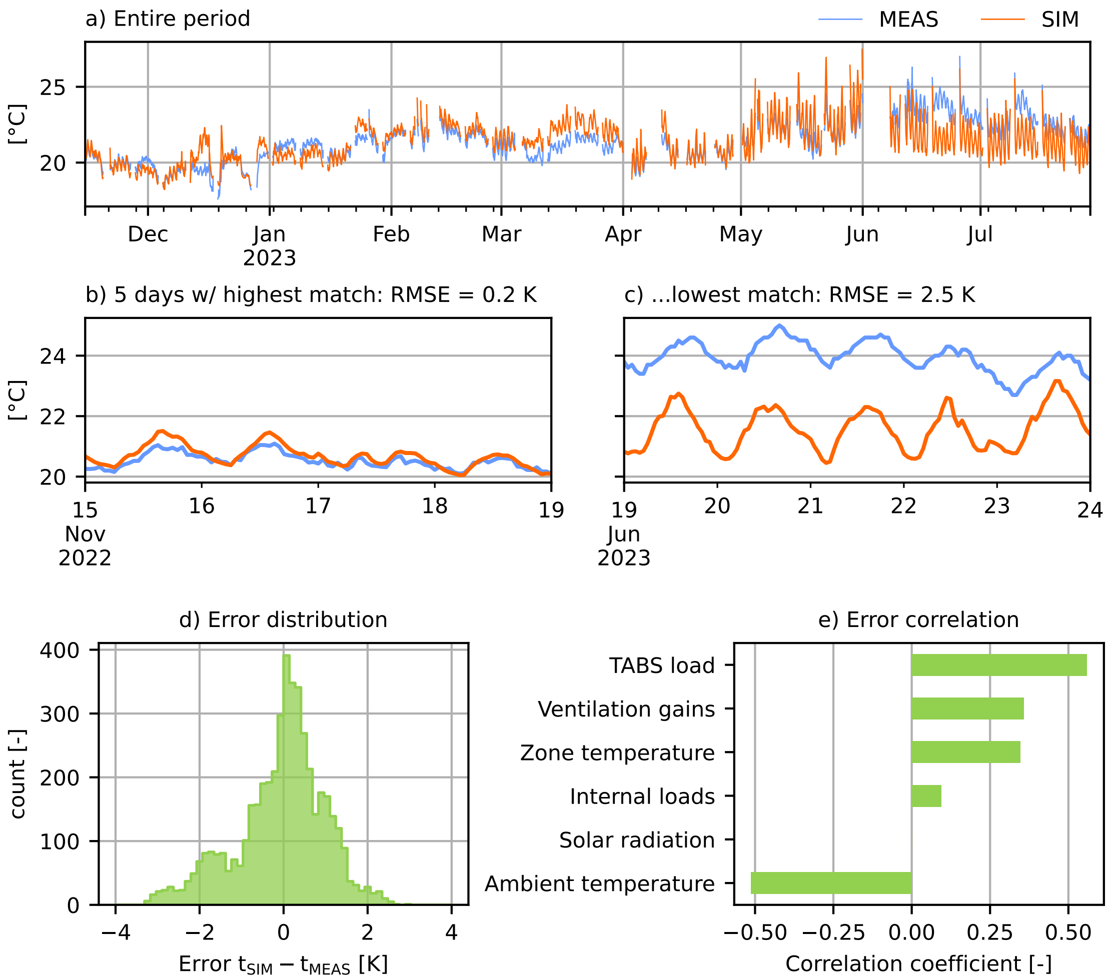


Figure A.4: Zone model evaluation (Zone 2)

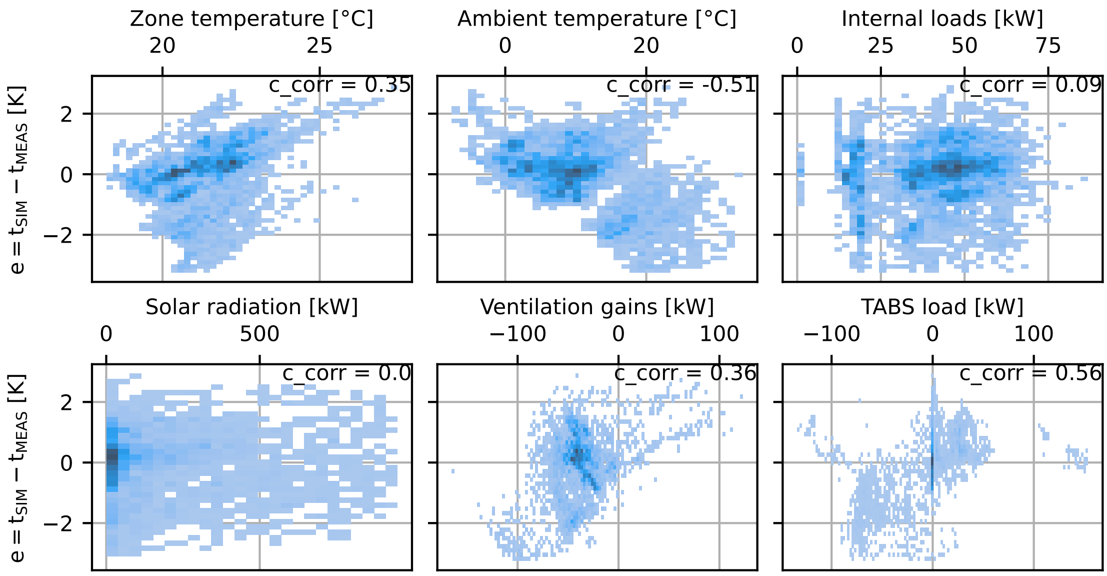


Figure A.5: Zone model comparison for Zone 1: Distribution plots of the error between simulated / measured temperatures and different variables

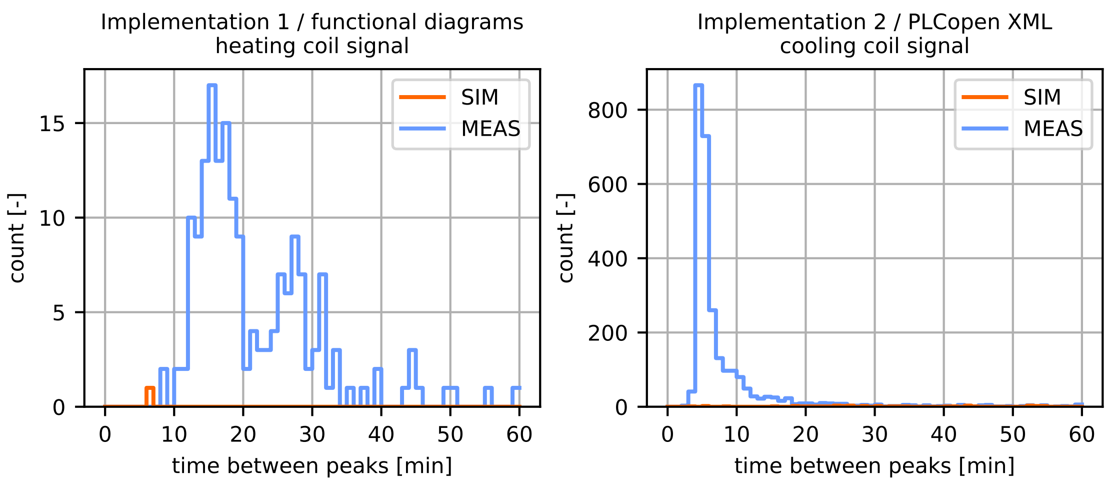


Figure A.6: Histograms corresponding to peaks count in Figure 5.16

B Implemented controls

B.1 Implementation 1

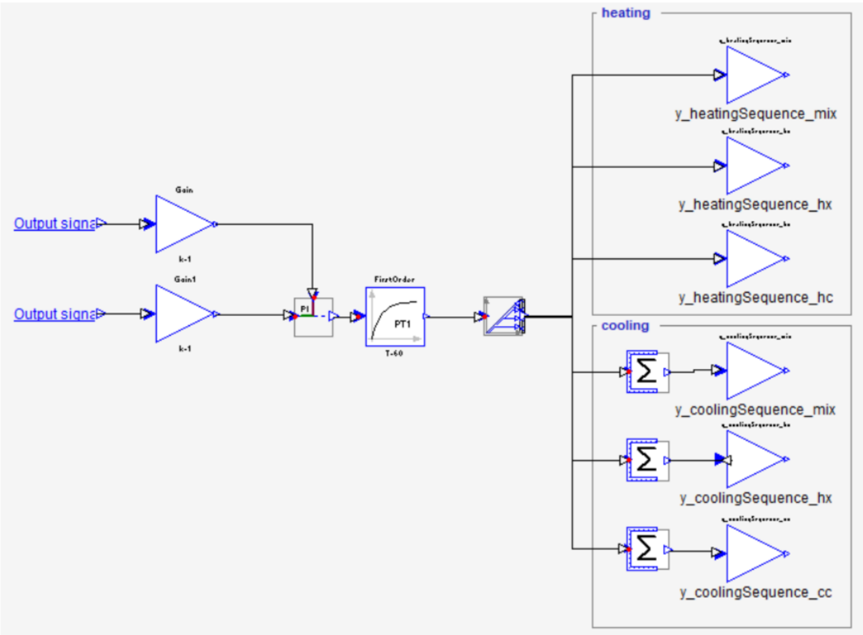


Figure B.1: Sequence controller

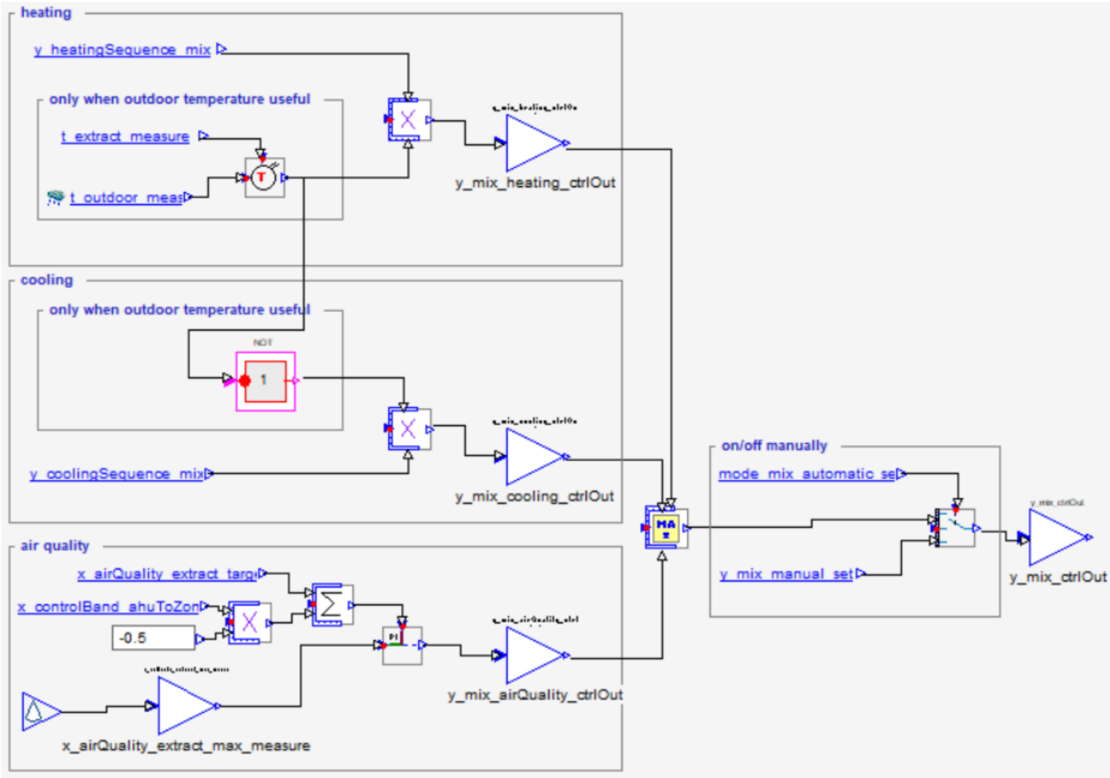


Figure B.2: Control sequence mixed air damper

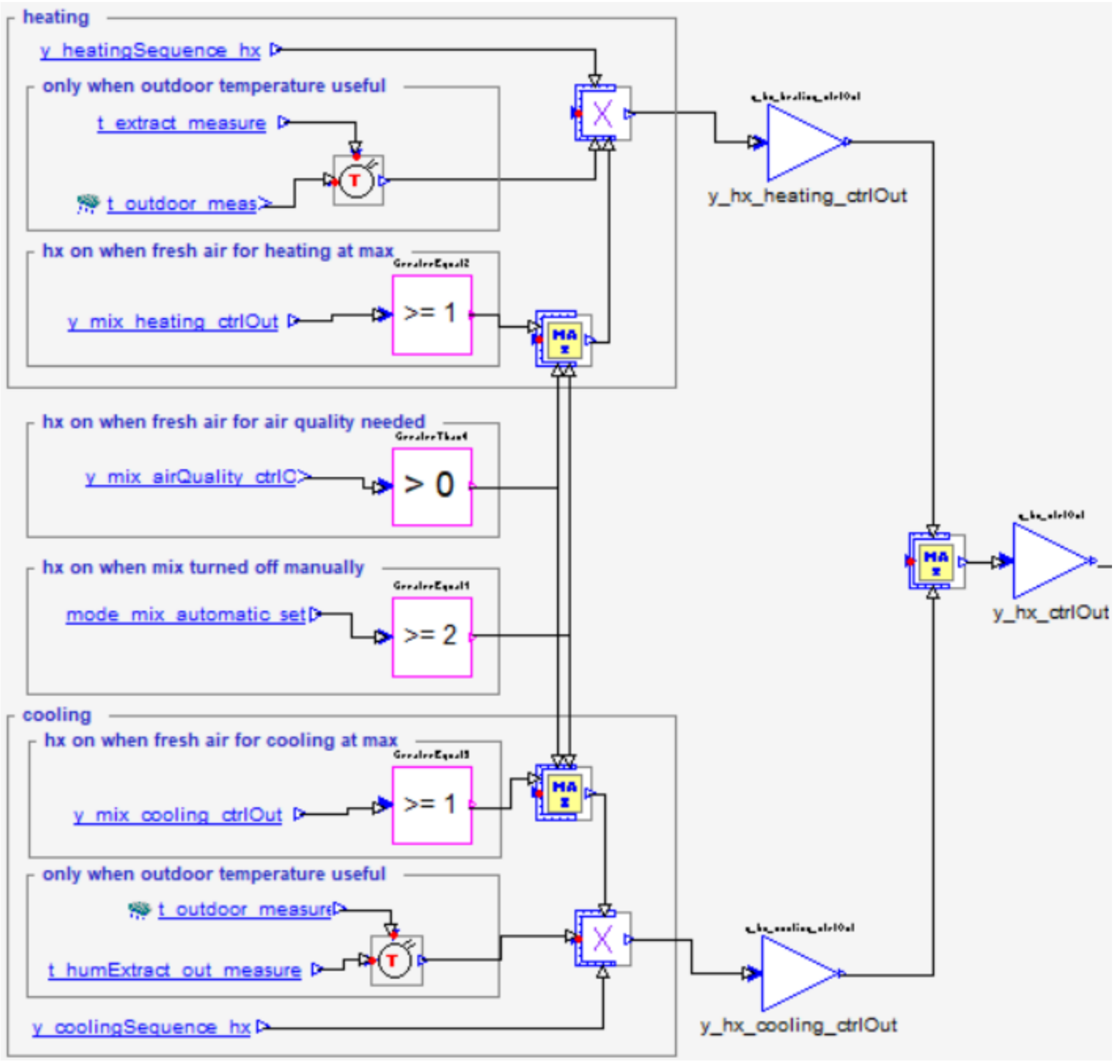


Figure B.3: Control sequence heat recovery

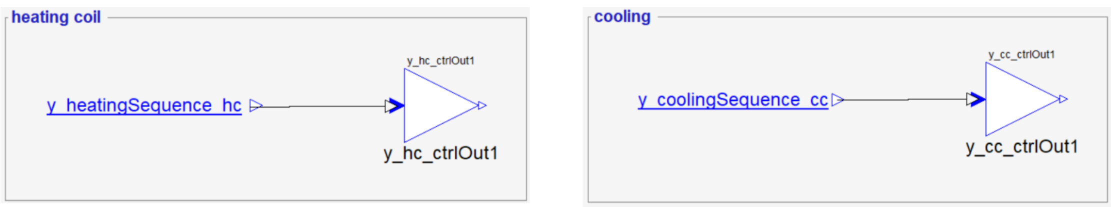


Figure B.4: Control sequence heating coil and cooling coil

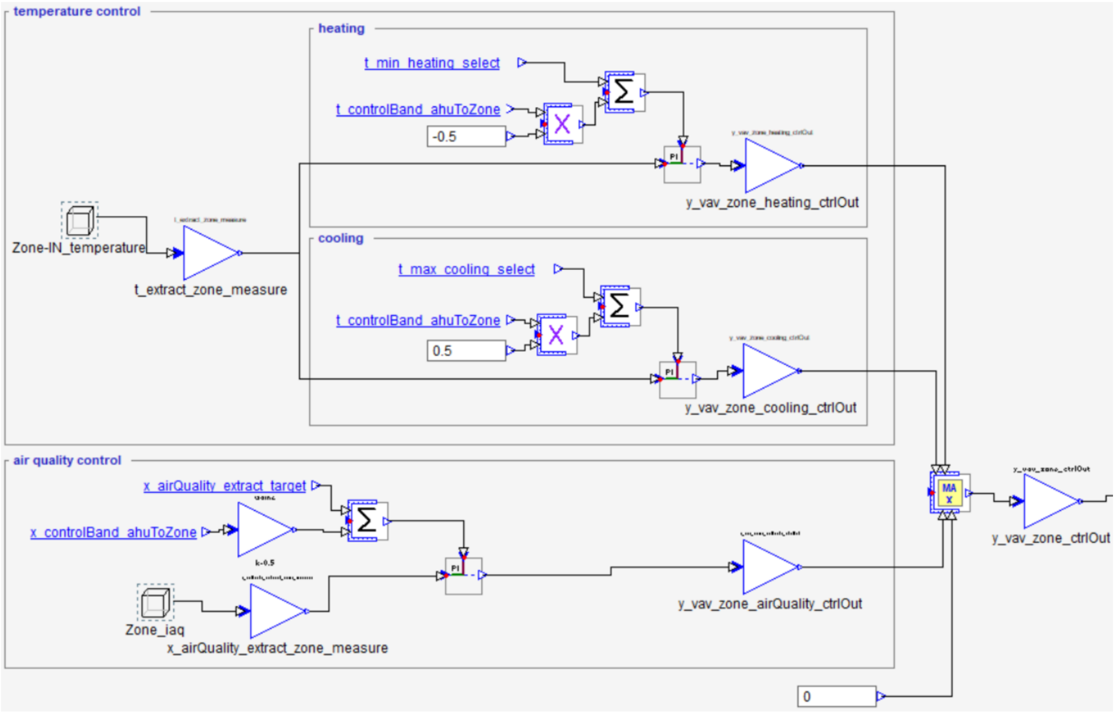


Figure B.5: Control sequence VAV boxes

B.2 Implementation 2

Listing B.1: IEC 61131-3 program implementation 2

```
FUNCTION T_PLC_US : UDINT
  VAR
    tx : UDINT;
  END_VAR
  VAR_INPUT
    debug : BOOL;
  END_VAR
  VAR
    N : INT := 0;
    offset : UDINT := 0;
    temp : DWORD := 1;
  END_VAR

  {extern unsigned long __tick;
  extern unsigned long long common_ticktime__;
  unsigned long long ticktime_ms = (common_ticktime__)/1000000;
  UDINT plc_time = (UDINT)(ticktime_ms * (unsigned long long)__tick);
  TX = plc_time}

  T_PLC_US := tx*1000;
  IF debug THEN
        T_PLC_US := (DWORD_TO_UDINT(SHL(UDINT_TO_DWORD(T_PLC_US),N) OR SHL(temp,N))-1) + OFFSET;
  END_IF;

  (* Original Code:
  tx := TIME();
  T_PLC_US := TIME_TO_DWORD(Tx)*1000;
  IF debug THEN
        T_PLC_US := (SHL(T_PLC_US,N) OR SHL(DWORD#1,N)-1) + OFFSET;
  END_IF;
```

```
  *)

  (* From OSCAT library, www.oscat.de

  this is a temporary T_PLC_US FB until OpenPLC gets its own time() functionality *)

  (* PLC_TIME and Global variables PLC_SCAN_CYCL and PLC_CYCL_TIME required *)
END_FUNCTION

FUNCTION_BLOCK FT_PIWL
  VAR_INPUT
    IN : REAL;
    KP : REAL := 1.0;
    KI : REAL := 1.0;
    LIM_L : REAL := -1.0E38;
    LIM_H : REAL := 1.0E38;
    RST : BOOL;
  END_VAR
  VAR_OUTPUT
    Y : REAL;
    LIM : BOOL;
  END_VAR
  VAR
    init : BOOL;
    tx : UDINT;
    tc : REAL;
    t_last : UDINT;
    in_last : REAL;
    i : REAL;
    p : REAL;
  END_VAR

  IF NOT init OR RST THEN
        init := TRUE;
        in_last := in;
                t_last := T_PLC_US(en:=true);
        i := 0.0;
        tc := 0.0;
  ELSE
        (* read last cycle time in Microseconds *)
                tx := T_PLC_US(en:=true);
        tc := UDINT_TO_REAL(tx - t_last);
        t_last := tx;

        (* calculate proportional part *)
        p := KP * IN;

        (* run integrator *)
        i := (IN + in_last) * 5.0E-7 * KI * tc + i;
        (*i := (IN + in_last) * 0.5 * KI * 1.0 + i;*)
        in_last := IN;

        (* calculate output Y *)
        Y := p + i;

        (* check output for limits *)
        IF Y >= LIM_H THEN
                Y := LIM_H;
                IF ki <> 0.0 THEN
                        i := LIM_H - p;
                ELSE
                        i := 0.0;
                END_IF;
                LIM := TRUE;
        ELSIF Y <= LIM_L THEN
                Y := LIM_L;
                IF ki <> 0.0 THEN
                        i := LIM_L - p;
                ELSE
                        i := 0.0;
                END_IF;
                LIM := TRUE;
        ELSE
                LIM := FALSE;
        END_IF;
  END_IF;
```

```

  (* From OSCAT Library, www.oscat.de *)
  (* T_PLC_US required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK FT_DERIV
  VAR_INPUT
    IN : REAL;
    K : REAL := 1.0;
    RUN : BOOL := TRUE;
  END_VAR
  VAR_OUTPUT
    OUT : REAL;
  END_VAR
  VAR
    old : REAL;
    tx : UDINT;
    last : UDINT;
    init : BOOL;
    tc : REAL;
  END_VAR

  (*tx:= T_PLC_US(en:=true);
  tc := UDINT_TO_REAL(tx - last);*)

  (* init on firsat startup *)
  IF NOT init THEN
        init := TRUE;
        old := in;
  ELSIF run AND tc > 0.0 THEN
        (*out := (in - old) / tc * 1000000.0 * K;*)
        out := (in - old) / 1.0 * K;
        old := in;
  ELSE
        out := 0.0;
  END_IF;

  (*last := tx;*)

  (* From OSCAT Library, www.oscat.de *)
  (* T_PLC_US, required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK FT_PIDWL
  VAR_INPUT
    IN : REAL;
    KP : REAL := 1.0;
    TN : REAL := 1.0;
    TV : REAL := 1.0;
    LIM_L : REAL := -1.0E38;
    LIM_H : REAL := 1.0E38;
    RST : BOOL;
  END_VAR
  VAR_OUTPUT
    Y : REAL;
    LIM : BOOL;
  END_VAR
  VAR
    piwl : FT_PIWL;
    diff : FT_DERIV;
  END_VAR

  IF rst THEN
        piwl(rst := TRUE);
        piwl.RST := FALSE;
  ELSE
        (* run PIWL controller first *)
        (* we need to check if TN = 0 and do alternative calls *)
        IF TN = 0.0 THEN
                piwl(in := IN * KP, KP := 1.0, KI := 0.0, LIM_L := LIM_L, LIM_H := LIM_H);
        ELSE
                piwl(in := IN * KP, KP := 1.0, KI := 1.0 / TN, LIM_L := LIM_L, LIM_H := LIM_H);
        END_IF;

        (* run differentiator and add_to_output *)
```

```
            diff(IN := IN, K := KP * TV);
            Y := piwl.Y + diff.out;

            (* limit the output *)
            IF Y < LIM_L THEN
                    LIM := TRUE;
                    Y := LIM_L;
            ELSIF Y > LIM_H THEN
                    LIM := TRUE;
                    Y := LIM_H;
            ELSE
                    LIM := FALSE;
            END_IF;
    END_IF;



    (* From OSCAT Library, www.oscat.de *)
    (* T_PLC_US, FT_DERIV required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK CTRL_OUT
  VAR_INPUT
    CI : REAL;
    OFFSET : REAL;
    MAN_IN : REAL;
    LIM_L : REAL;
    LIM_H : REAL;
    MANUAL : BOOL;
  END_VAR
  VAR_OUTPUT
    Y : REAL;
    LIM : BOOL;
  END_VAR

  Y := SEL(manual, CI, MAN_IN) + OFFSET;

  (* Limit the output *)
  IF Y >= LIM_H THEN
        Y := LIM_H;
        LIM := TRUE;
  ELSIF Y <= lim_L THEN
        Y := LIM_L;
        LIM := TRUE;
  ELSE
        LIM := FALSE;
  END_IF;

  (* From OSCAT Library, www.oscat.de *)
END_FUNCTION_BLOCK

FUNCTION DEAD_ZONE : REAL
  VAR_INPUT
    X : REAL;
    L : REAL;
  END_VAR

  IF ABS(x) > L THEN
        dead_zone := X;
  ELSE
        DEAD_ZONE := 0.0;
  END_IF;

  (* From OSCAT Library, www.oscat.de *)
END_FUNCTION

FUNCTION CTRL_IN : REAL
  VAR_INPUT
    SET_POINT : REAL;
    ACTUAL : REAL;
    NOISE : REAL;
  END_VAR

  CTRL_IN := DEAD_ZONE(SET_POINT - ACTUAL, NOISE);

  (* From OSCAT Library, www.oscat.de *)
  (* DEAD_ZONE required *)
END_FUNCTION
```

```

FUNCTION_BLOCK CTRL_PID
  VAR_INPUT
    ACT : REAL;
    SET : REAL;
    SUP : REAL;
    OFS : REAL;
    M_I : REAL;
    MAN : BOOL;
    RST : BOOL := FALSE;
    KP : REAL := 1.0;
    TN : REAL := 1.0;
    TV : REAL := 1.0;
    LL : REAL := -1000.0;
    LH : REAL := 1000.0;
  END_VAR
  VAR_OUTPUT
    Y : REAL;
    DIFF : REAL;
    LIM : BOOL;
  END_VAR
  VAR
    _pid : FT_PIDWL;
    co : CTRL_OUT;
  END_VAR

  DIFF := CTRL_IN(SET, ACT, SUP);
  _pid(in := DIFF, kp := KP, tn := TN, tv := TV, lim_l := LL, lim_h := LH , rst := RST);
  co(ci := _pid.Y, OFFSET := OFS, man_in := M_I, lim_l := LL, lim_h := LH , manual := MAN);
  Y := co.Y;
  LIM := co.LIM;


  (* From OSCAT Library, www.oscat.de *)
  (* CTRL_IN, FT_PIDWL, CTRL_out reauired *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK FT_PT1
  VAR_INPUT
    IN : REAL;
    _T : TIME := t#1s;
    K : REAL := 1.0;
  END_VAR
  VAR_OUTPUT
    OUT : REAL;
  END_VAR
  VAR
    last : UDINT;
    tx : UDINT;
    init : BOOL;
  END_VAR

  tx:= T_PLC_US(en:=true);

  (* startup initialisation *)
  IF NOT init OR _T = t#0s THEN
        init := TRUE;
        out := K * in;
  ELSE
        out := out + (in * K - out) * UDINT_TO_REAL(Tx - last) / TIME_TO_REAL(_T) * 1.0E-3;
        IF ABS(out) < 1.0E-20 THEN out := 0.0; END_IF;
  END_IF;
  last := tx;

  (* From OSCAT Library, www.oscat.de *)
  (* T_PLC_US required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK vavctrl
  VAR_INPUT
    tminheatingselectvav : REAL;
    tmaxcoolingselectvav : REAL;
    tmeasure : REAL;
  END_VAR
  VAR_OUTPUT
```

```
      out : REAL;
    END_VAR
    VAR
      pidvavc : CTRL_PID;
      pidvavh : CTRL_PID;
      ypidvavc : REAL;
      ypidvavh : REAL;
    END_VAR
    VAR_INPUT
      kp : REAL;
      tn : REAL;
      tv : REAL;
    END_VAR
    VAR
      pt1vav : FT_PT1;
    END_VAR
    VAR_INPUT
      nmin : REAL;
    END_VAR

    pidvavc(
         ACT := tmaxcoolingselectvav,
         SET := tmeasure,
         SUP := 0.0,
         OFS := 0.0,
         M_I := 0.0,
         MAN := false,
         RST := false,
         KP := kp,
         TN := tn,
         TV := tv,
         LL := 0.0,
         LH := 100.0,
         Y => ypidvavc);

    pidvavh(
         ACT := tmeasure,
         SET := tminheatingselectvav,
         SUP := 0.0,
         OFS := 0.0,
         M_I := 0.0,
         MAN := false,
         RST := false,
         KP := kp,
         TN := tn,
         TV := tv,
         LL := 0.0,
         LH := 100.0,
         Y => ypidvavh);

    out := max(ypidvavc, ypidvavh);

    (* consider defined minimum value *)
    out := max(out, nmin);

    if false then
      pt1vav(
         IN := out,
         _T := t#12h,
         K := 1.0,
         OUT => out);
    end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqctrl
    VAR_INPUT
      sp : REAL;
      pv : REAL;
    END_VAR
    VAR
      manual : BOOL;
    END_VAR
    VAR_INPUT
      modeahuonoroff : REAL;
    END_VAR
    VAR
      pidseq : CTRL_PID;
    END_VAR
```

```
  VAR_INPUT
    kplow : REAL;
    kphigh : REAL;
  END_VAR
  VAR_OUTPUT
    kpcalc : REAL;
  END_VAR
  VAR_INPUT
    tnlow : REAL;
    tnhigh : REAL;
  END_VAR
  VAR_OUTPUT
    tncalc : REAL;
  END_VAR
  VAR_INPUT
    tvlow : REAL;
    tvhigh : REAL;
  END_VAR
  VAR_OUTPUT
    tvcalc : REAL;
    out : REAL := 49.0;
  END_VAR
  VAR
    diff : REAL;
    diffperc : REAL;
  END_VAR
  VAR_INPUT
    seqctrlthreshold : REAL;
    seqctrldeadband : REAL;
  END_VAR

  if modeahuonoroff <> 1.0 then
    manual := true;
  else
    manual := false;
  end_if;

  diff := abs(sp - pv);
  if sp <> 0.0 then
    diffperc := diff ;
  else
    diffperc := diff;
  end_if;

  if diffperc > (seqctrlthreshold + seqctrldeadband * 0.5) then
    kpcalc := kphigh;
    tncalc := tnhigh;
    tvcalc := tvhigh;
  elsif diffperc < (seqctrlthreshold - seqctrldeadband * 0.5) then
    kpcalc := kplow;
    tncalc := tnlow;
    tvcalc := tvlow;
  else
    kpcalc := kplow + (diff - (seqctrlthreshold - seqctrldeadband * 0.5))
        ↪ * (kphigh - kplow) / (seqctrldeadband);
    tncalc := tnlow + (diff - (seqctrlthreshold - seqctrldeadband * 0.5))
        ↪ * (tnhigh - tnlow) / (seqctrldeadband);
    tvcalc := tvlow + (diff - (seqctrlthreshold - seqctrldeadband * 0.5))
        ↪ * (tvhigh - tvlow) / (seqctrldeadband);
  end_if;


  pidseq(
        ACT := pv,
        SET := sp,
        SUP := 0.0,
        OFS := 0.0,
        M_I := 0.5,
        MAN := manual,
        RST := false,
        KP := kpcalc,
        TN := tncalc,
        TV := tvcalc,
        LL := 0.0,
        LH := 100.0,
        Y => out);
END_FUNCTION_BLOCK
```

```
FUNCTION INC1 : INT
  VAR_INPUT
    X : INT;
    N : INT;
  END_VAR

  IF X >= N - 1 THEN
        INC1 := 0;
  ELSE
        INC1 := X + 1;
  END_IF;
  (* from OSCAT library www.oscat.de *)
END_FUNCTION

FUNCTION_BLOCK DELAY
  VAR_INPUT
    IN : REAL;
    N : INT;
    RST : BOOL;
  END_VAR
  VAR_OUTPUT
    OUT : REAL;
  END_VAR
  VAR
    i : INT;
    init : BOOL;
    stop : INT;
    buf : ARRAY [0..31] OF REAL;
  END_VAR

  stop := LIMIT(0,N,32) - 1;
  IF rst OR NOT init THEN
        init := TRUE;
        FOR i := 0 TO stop DO buf[i] := in; END_FOR;
        out := in;
        i := 0;
  ELSIF stop < 0 THEN
        out := in;
  ELSE
        out := buf[i];
        buf[i] := in;
        i := INC1(i, N);
  END_IF;
  (* from OSCAT library www.oscat.de *)
  (* inc1 requiered *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqmix
  VAR_INPUT
    in : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
  END_VAR
  VAR
    m1 : REAL;
    m2 : REAL;
    n1 : REAL;
    n2 : REAL;
  END_VAR
  VAR_INPUT
    extractaboveoutdoor : BOOL;
    recirculationonly : BOOL;
    x0 : REAL;
    x1 : REAL;
    x2 : REAL;
    x3 : REAL;
    x4 : REAL;
  END_VAR
  VAR
    pt1 : FT_PT1;
    pt2 : DINT;
  END_VAR

  (* parameters for heating *)
  m1 := (100.0 - 0.0) / (x4 - x3);
  n1 := -m1 * x3;

```

```

  (* parameters for cooling *)
  m2 := (100.0 - 0.0) / (x1 - x2);
  n2 := -m2 * x2;

  if recirculationonly then
    out := 100.0;
  else
    if in > x3 and extractaboveoutdoor then
      out := min(m1 * in + n1, 100.0);
    elsif in < x2 and not extractaboveoutdoor then
      out := min(m2 * in + n2, 100.0);
    else
      out := 0.0;
    end_if;
  end_if;
END_FUNCTION_BLOCK

FUNCTION T_PLC_MS : UDINT
  VAR
    tx : UDINT;
  END_VAR
  VAR_INPUT
    debug : BOOL;
  END_VAR
  VAR
    N : INT := 0;
    offset : UDINT := 0;
    temp : DWORD := 1;
  END_VAR

  tx := 0;

  {extern unsigned long __tick;
  extern unsigned long long common_ticktime__;
  unsigned long long ticktime_ms = (common_ticktime__)/1000000;
  UDINT plc_time = (UDINT)(ticktime_ms * (unsigned long long)__tick);
  TX = plc_time}

  T_PLC_MS := tx;
  IF debug THEN
        T_PLC_MS := (DWORD_TO_UDINT(SHL(UDINT_TO_DWORD(T_PLC_MS),N) OR SHL(temp,N))-1) + OFFSET;
  END_IF;

  (* Original Code:
  tx := TIME();
  T_PLC_MS := TIME_TO_DWORD(Tx);
  IF debug THEN
        T_PLC_MS := (SHL(T_PLC_MS,N) OR SHL(DWORD#1,N)-1) + OFFSET;
  END_IF;
  *)

  (* From OSCAT library, www.oscat.de

  this is a temporary T_PLC_MS FB until OpenPLC gets its own time() functionality *)

  (* PLC_TIME and Global variables PLC_SCAN_CYCL and PLC_CYCL_TIME required *)
END_FUNCTION

FUNCTION_BLOCK INTEGRATE
  VAR_INPUT
    _E : BOOL := TRUE;
    X : REAL;
    K : REAL := 1.0;
  END_VAR
  VAR_IN_OUT
    Y : REAL;
  END_VAR
  VAR
    x_last : REAL;
    init : BOOL;
    last : UDINT;
    tx : UDINT;
  END_VAR
```

```

  tx:= T_PLC_MS(en:=true);

  IF NOT init THEN
        init := TRUE;
        X_last := X;
  ELSIF _E THEN
        Y := (X + X_LAST) * 0.5E-3 * UDINT_TO_REAL(tx-last) * K + Y;
        X_last := X;
  END_IF;
  last := tx;

  (* From OSCAT Library, www.oscat.de *)
  (* T_PLC_MS required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqcc
  VAR_INPUT
    in : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
    pump : REAL;
  END_VAR
  VAR
    m : REAL;
    n : REAL;
  END_VAR
  VAR_INPUT
    x1 : REAL;
  END_VAR
  VAR
    pt1 : FT_PT1;
  END_VAR

  m := 100.0 / (0.0 - x1);
  n := 100.0;

  if in < x1 then
    out := m * in + n;
    pump := 1.0;
  else
    out := 0.0;
    pump := 0.0;
  end_if;

  if false then
    pt1(
        IN := out,
        _T := t#1h,
        K := 1.0,
        OUT => out);
  end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqhc
  VAR_INPUT
    in : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
    pump : REAL;
  END_VAR
  VAR
    m : REAL;
    n : REAL;
  END_VAR
  VAR_INPUT
    x0 : REAL;
    x4 : REAL;
  END_VAR
  VAR
    pt1 : FT_PT1;
  END_VAR

  m := (100.0 - 0.0) / (100.0 - x4);
  n := -m * x4;

```

```
  if in > x4 then
    out := m * in + n;
    pump := 1.0;
  else
    out := 0.0;
    pump := 0.0;
  end_if;

  if false then
    pt1(
       IN := out,
       _T := t#1h,
       K := 1.0,
       OUT => out);
   end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqhx
  VAR_INPUT
    x0 : REAL;
    x2 : REAL;
    x3 : REAL;
    in : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
  END_VAR
  VAR
    m1 : REAL;
    m2 : REAL;
    n1 : REAL;
    n2 : REAL;
  END_VAR
  VAR_INPUT
    extractaboveoutdoor : BOOL;
    recirculationonly : BOOL;
  END_VAR
  VAR
    pt1 : FT_PT1;
  END_VAR

  (* parameters for heating *)
  m1 := (100.0 - 0.0) / (x3 - x0);
  n1 := -m1 * x0;


  (* parameters for cooling *)
  m2 := (100.0 - 0.0) / (x2 - x0);
  n2 := -m2 * x0;

  if recirculationonly then
    out := 0.0;
  else
    (*heating case*)
    if in > x0 and extractaboveoutdoor then
      out := min(m1 * in + n1, 100.0);
    (*cooling case*)
    elsif in <= x0 and not extractaboveoutdoor then
      out := min(m2 * in + n2, 100.0);
    else
      out := 0.0;
    end_if;
  end_if;

  if false then
    pt1(
       IN := out,
       _T := t#1h,
       K := 1.0,
       OUT => out);
   end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK HYST_2
  VAR_INPUT
    IN : REAL;
    VAL : REAL;
```

```
      HYS : REAL;
    END_VAR
    VAR_OUTPUT
      Q : BOOL;
      WIN : BOOL;
    END_VAR
    VAR
      tmp : REAL;
    END_VAR

    tmp := val - hys * 0.5;
    IF in < tmp THEN
          Q := FALSE;
          win := FALSE;
    ELSIF in > tmp + hys THEN
          Q := TRUE;
          win := FALSE;
    ELSE
          win := TRUE;
    END_IF;

    (* From OSCAT Library, www.oscat.de *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK lintrans
    VAR_INPUT
      InSignal : REAL;
      MinValue : REAL;
      MaxValue : REAL;
    END_VAR
    VAR_OUTPUT
      OutSignal : REAL;
    END_VAR

    OutSignal := MinValue + InSignal / 1.0 * (MaxValue - MinValue);
END_FUNCTION_BLOCK

FUNCTION_BLOCK calctsupply
    VAR
      pidc : CTRL_PID;
      pidh : CTRL_PID;
    END_VAR
    VAR_INPUT
      textractcooling : REAL;
      textractheating : REAL;
      tmax : REAL;
      tmin : REAL;
    END_VAR
    VAR_OUTPUT
      out : REAL;
    END_VAR
    VAR_INPUT
      kp : REAL;
      tn : REAL;
      tv : REAL;
      modesupplyorextractcontrol : REAL;
      tsupplymin : REAL;
      tsupplymax : REAL;
      toutdoor : REAL;
      tmaxdeltasupplyextract : REAL;
    END_VAR
    VAR_OUTPUT
      ypidcascc : REAL;
      ypidcasch : REAL;
    END_VAR
    VAR
      lintransc : lintrans;
      lintransh : lintrans;
    END_VAR
    VAR_OUTPUT
      ylintransc : REAL;
      ylintransh : REAL;
    END_VAR
    VAR
      maxin1 : REAL;
      maxin2 : REAL;
      pt1out : FT_PT1;
    END_VAR

```

```
  pidc(
       ACT := textractcooling,
       SET := tmax,
       SUP := 0.0,
       OFS := 0.0,
       M_I := 0.0,
       MAN := false,
       RST := false,
       KP := kp,
       TN := tn,
       TV := tv,
       LL := 0.0,
       LH := 1.0,
       Y => ypidcascc);

  lintransc(
       InSignal := ypidcascc,
       MinValue := tsupplymin,
       MaxValue := tsupplymax,
       OutSignal => ylintransc);

  (*ylintransc := 20.0;*)

  pidh(
       ACT := textractheating,
       SET := tmin,
       SUP := 0.0,
       OFS := 0.0,
       M_I := 0.0,
       MAN := false,
       RST := false,
       KP := kp,
       TN := tn,
       TV := tv,
       LL := 0.0,
       LH := 1.0,
       Y => ypidcasch);

  lintransh(
       InSignal := ypidcasch,
       MinValue := tsupplymin,
       MaxValue := tsupplymax,
       OutSignal => ylintransh);

  maxin1 := textractcooling - tmaxdeltasupplyextract;
  maxin2 := min(ylintransc, max(toutdoor, ylintransh));

  out := max(maxin1, maxin2);

  if false then
  pt1out(
       IN := out,
       _T := t#2h,
       K := 1.0,
       OUT => out);
  end_if;
END_FUNCTION_BLOCK

PROGRAM RLT_6_11
  VAR_EXTERNAL
    modeahunormalorreducedset : REAL;
    modesupplyorextractcontrolset : REAL;
    tsupplyahumean : REAL;
    tsupplyahu1 : REAL;
    tsupplyahu2 : REAL;
    textractcooling : REAL;
    textractheating : REAL;
    textractzone1 : REAL;
    textractahu1 : REAL;
    textractahu2 : REAL;
    textractzone2 : REAL;
    kpcascset : REAL;
    tncascset : REAL;
    tvcascset : REAL;
    tsupplyminset : REAL;
    tsupplymaxset : REAL;
    toutdoor : REAL;
    textractahumean : REAL;
```

```
    tmaxdeltasupplyextractset : REAL;
    kpseqlowset : REAL;
    kpseqhighset : REAL;
    kpseqcalc : REAL;
    tnseqlowset : REAL;
    tnseqhighset : REAL;
    tnseqcalc : REAL;
    tvseqlowset : REAL;
    tvseqhighset : REAL;
    tvseqcalc : REAL;
    tsupplyahutarget : REAL;
  END_VAR
  VAR
    seq1 : seqctrl;
    sequencec : seqcc;
    sequenceh : seqhc;
  END_VAR
  VAR_EXTERNAL
    ycc : REAL;
    yhc : REAL;
    ymix : REAL;
    yhx : REAL;
    modeahuonoroffset : REAL;
  END_VAR
  VAR
    sequencehx : seqhx;
    sequencemix : seqmix;
    extractaboveoutdoor : BOOL;
  END_VAR
  VAR_EXTERNAL
    extractaboveoutdoorreal : REAL;
    seqypid : REAL;
    seqoutcorrect : REAL;
    seqx0set : REAL;
    seqx1set : REAL;
    seqx2set : REAL;
    seqx3set : REAL;
    seqx4set : REAL;
    tminheatingselect : REAL;
    tmaxcoolingselect : REAL;
    tminheatingnormalset : REAL;
    tmaxcoolingnormalset : REAL;
    tminheatingreduceset : REAL;
    tmaxcoolingreduceset : REAL;
  END_VAR
  VAR
    pt1_1 : FT_PT1;
    pt1_2 : FT_PT1;
    m1 : REAL;
    m2 : REAL;
    n1 : REAL;
    n2 : REAL;
    blockfreeheating : BOOL;
    blockfreecooling : BOOL;
    hystheat : HYST_2;
    hystoutdoor : HYST_2;
    tsuppt1 : REAL;
    outdoorabovelimit : BOOL;
    recirculationonly : BOOL;
  END_VAR
  VAR_EXTERNAL
    nfreshminset : REAL;
  END_VAR
  VAR
    calctsupply1 : calctsupply;
  END_VAR
  VAR_EXTERNAL
    dttoleranceahutovavset : REAL;
  END_VAR
  VAR
    tminheatingselectahu : REAL;
    tminheatingselectvav : REAL;
    tmaxcoolingselectahu : REAL;
    tmaxcoolingselectvav : REAL;
    vavzone1 : vavctrl;
    vavzone2 : vavctrl;
  END_VAR
  VAR_EXTERNAL
    yvavzone1 : REAL;
```

```
  yvavzone2 : REAL;
END_VAR
VAR
  pt1tsupply : FT_PT1;
  pt1textractzone1 : FT_PT1;
  pt1textractzone2 : FT_PT1;
  pt1ymix : FT_PT1;
  pt1yhx : FT_PT1;
  pt1yhc : FT_PT1;
  pt1ycc : FT_PT1;
END_VAR
VAR_EXTERNAL
  tymixset : REAL;
  tyhxset : REAL;
  tyhcset : REAL;
  tyccset : REAL;
  ypidcasch : REAL;
  ypidcascc : REAL;
  ylintransh : REAL;
  ylintransc : REAL;
  kpvavset : REAL;
  tnvavset : REAL;
  tvvavset : REAL;
  seqctrlthresholdset : REAL;
  seqctrldeadbandset : REAL;
  dpsupplyfanducttoambientnormalset : REAL;
  dpextractfanducttoambientnormalset : REAL;
  dpsupplyfanducttoambientreduceset : REAL;
  dpextractfanducttoambientreduceset : REAL;
  dpsupplyfanducttoambientselect : REAL;
  dpextractfanducttoambientselect : REAL;
  nvavminzone1set : REAL;
  nvavminzone2set : REAL;
  modepumphc : REAL;
END_VAR

tsupplyahumean := (tsupplyahu1 + tsupplyahu2) / 2.0;
textractahumean := (textractahu1 + textractahu2) / 2.0;

(* normal operation *)
if modeahunormalorreducedset = 1.0 then
  tminheatingselect := tminheatingnormalset;
  tmaxcoolingselect := tmaxcoolingnormalset;
  dpsupplyfanducttoambientselect := dpsupplyfanducttoambientnormalset;
  dpextractfanducttoambientselect := dpextractfanducttoambientnormalset
    ↪ ;
(* reduced operation *)
elsif modeahunormalorreducedset = 2.0 then
  tminheatingselect := tminheatingreduceset;
  tmaxcoolingselect := tmaxcoolingreduceset;
  dpsupplyfanducttoambientselect := dpsupplyfanducttoambientreduceset;
  dpextractfanducttoambientselect := dpextractfanducttoambientreduceset
    ↪ ;
else
  tminheatingselect := tminheatingreduceset;
  tmaxcoolingselect := tmaxcoolingreduceset;
  dpsupplyfanducttoambientselect := dpsupplyfanducttoambientreduceset;
  dpextractfanducttoambientselect := dpextractfanducttoambientreduceset
    ↪ ;
end_if;

tminheatingselectahu := tminheatingselect + dttoleranceahutovavset;
tmaxcoolingselectahu := tmaxcoolingselect - dttoleranceahutovavset;
tminheatingselectvav := tminheatingselect;
tmaxcoolingselectvav := tmaxcoolingselect;

textractheating := min(textractzone1, textractzone2);
textractcooling := max(textractzone1, textractzone2);

(*check if free heating is possible
true if the extract temperature is above the outdoor temperature
false if the extract temperature is below the outdoor temperature*)
hystheat(
     IN := textractahumean,
     VAL := toutdoor,
     HYS := 4.0,
     Q => extractaboveoutdoor);

```

```
  hystoutdoor(
       IN := toutdoor,
       VAL := tmaxcoolingselect,
       HYS := 4.0,
       Q => outdoorabovelimit);

  extractaboveoutdoorreal := bool_to_real(extractaboveoutdoor);

  calctsupply1(
       modesupplyorextractcontrol := modesupplyorextractcontrolset,
       tmax := tmaxcoolingselectahu,
       tmin := tminheatingselectahu,
       textractcooling := textractcooling,
       textractheating := textractheating,
       kp := kpcascset,
       tn := tncascset,
       tv := tvcascset,
       tsupplymin := tsupplyminset,
       tsupplymax := tsupplymaxset,
       toutdoor := toutdoor,
       tmaxdeltasupplyextract := tmaxdeltasupplyextractset,
       out => tsupplyahutarget,
       ypidcasch => ypidcasch,
       ypidcascc => ypidcascc,
       ylintransh => ylintransh,
       ylintransc => ylintransc);


  seq1(
       modeahuonoroff := modeahuonoroffset,
       seqctrlthreshold := seqctrlthresholdset,
       seqctrldeadband := seqctrldeadbandset,
       sp := tsupplyahutarget,
       pv := tsupplyahumean,
       kplow := kpseqlowset,
       tnlow := tnseqlowset,
       tvlow := tvseqlowset,
       kphigh := kpseqhighset,
       tnhigh := tnseqhighset,
       tvhigh := tvseqhighset,
       kpcalc => kpseqcalc,
       tncalc => tnseqcalc,
       tvcalc => tvseqcalc,
       out => seqypid);

  (* heating case *)
  m1 := (100.0 - seqx4set) / (100.0 - seqx0set);
  n1 := seqx4set - m1 * seqx0set;

  (* cooling case *)
  m2 := (seqx1set - 0.0) / (seqx0set - 0.0);
  n2 := 0.0;

  if outdoorabovelimit then
    recirculationonly := true;
    seqoutcorrect := m2 * seqypid + n2;
  else
    recirculationonly := false;

    (* cooling case *)
    (* if the extract temperature is above the outdoor temperature (can
       ↪ not be used for cooling) + there is cooling demand: deactivate
       ↪ hx and mix *)
    if extractaboveoutdoor and seqypid < seqx0set then
      seqoutcorrect := m2 * seqypid + n2;
    (* heating case *)
    (* if the extract temperature is below the outdoor temperature (can
       ↪ not be used for heating) + there is heating demand: deactivate
       ↪ hx and mix *)
    elsif not extractaboveoutdoor and seqypid > seqx0set then
      seqoutcorrect := m1 * seqypid + n1;
    else
      seqoutcorrect := seqypid;
    end_if;
  end_if;

  sequencec(
```

```
    in := seqoutcorrect,
    x1 := seqx1set,
    out => ycc);

  sequenceh(
    in := seqoutcorrect,
    x0 := seqx0set,
    x4 := seqx4set,
    pump => modepumphc,
    out => yhc);

  sequencehx(
    in := seqoutcorrect,
    extractaboveoutdoor := extractaboveoutdoor,
    recirculationonly := recirculationonly,
    x0 := seqx0set,
    x2 := seqx2set,
    x3 := seqx3set,
    out => yhx);

  sequencemix(
    in := seqoutcorrect,
    extractaboveoutdoor := extractaboveoutdoor,
    recirculationonly := recirculationonly,
    x0 := seqx0set,
    x1 := seqx1set,
    x2 := seqx2set,
    x3 := seqx3set,
    x4 := seqx4set,
    out => ymix);

  if false then
    pt1ymix(
        IN := ymix,
        _T := REAL_TO_TIME(tymixset*1000.0),
        K := 1.0,
        OUT => ymix);
    pt1yhx(
        IN := yhx,
        _T := REAL_TO_TIME(tyhxset*1000.0),
        K := 1.0,
        OUT => yhx);
    pt1yhc(
        IN := yhc,
        _T := REAL_TO_TIME(tyhcset*1000.0),
        K := 1.0,
        OUT => yhc);
    pt1ycc(
        IN := ycc,
        _T := REAL_TO_TIME(tyccset*1000.0),
        K := 1.0,
        OUT => ycc);
  end_if;

  if modeahuonoroffset <> 1.0 then
    yhc := 0.0;
    ycc := 0.0;
    yhx := 0.0;
    ymix := 100.0;
    (*tsupplyahutarget := tsupplyahumean;*)
  end_if;

  ymix := min(ymix, (100.0 - nfreshminset));

  vavzone1(
        tmaxcoolingselectvav := tmaxcoolingselectvav,
        tminheatingselectvav := tminheatingselectvav,
        tmeasure := textractzone1,
        kp := kpvavset,
        tn := tnvavset,
        tv := tvvavset,
        nmin := nvavminzone1set,
        out => yvavzone1);

  vavzone2(
        tmaxcoolingselectvav := tmaxcoolingselectvav,
        tminheatingselectvav := tminheatingselectvav,
```

```
        tmeasure := textractzone2,
        kp := kpvavset,
        tn := tnvavset,
        tv := tvvavset,
        nmin := nvavminzone1set,
        out => yvavzone2);


  if false then
    yhc := 0.0;
    ycc := 0.0;
    yhx := 0.0;
    ymix := 100.0;
    yvavzone1 := 100.0;
    yvavzone2 := 100.0;
  end_if;
END_PROGRAM

FUNCTION_BLOCK FT_PT2
  VAR_INPUT
    IN : REAL;
    _T : TIME;
    _D : REAL;
    K : REAL := 1.0;
  END_VAR
  VAR_OUTPUT
    OUT : REAL;
  END_VAR
  VAR
    init : BOOL;
    int1 : INTEGRATE;
    int2 : INTEGRATE;
    tn : REAL;
    I1 : REAL;
    I2 : REAL;
    tn2 : REAL;
  END_VAR

  (* startup initialisation *)
  IF NOT init OR _T = T#0s THEN
        init := TRUE;
        out := K * in;
        I2 := out;
  ELSE
        TN := TIME_TO_REAL(_T) / 1000.0;
        tn2 := TN * TN;
        int1(X := in * K / tn2 - I1 * 0.5 * _D / TN - I2 / TN2, Y := I1);
        I1 := int1.Y;
        int2(X := I1,Y := I2);
        I2 := int2.Y;
        out := I2;
  END_IF;

  (* From OSCAT Library, www.oscat.de *)
  (* INTEGRATE required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK HYST_1
  VAR_INPUT
    IN : REAL;
    HIGH : REAL;
    LOW : REAL;
  END_VAR
  VAR_OUTPUT
    Q : BOOL;
    WIN : BOOL;
  END_VAR

  IF in < low THEN
        Q := FALSE;
        win := FALSE;
  ELSIF in > high THEN
        Q := TRUE;
        win := FALSE;
  ELSE
        win := TRUE;
  END_IF;

```

```
  (* From OSCAT Library, www.oscat.de *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK FT_AVG
  VAR_INPUT
    IN : REAL;
    _E : BOOL := TRUE;
    RST : BOOL;
    N : INT := 32;
  END_VAR
  VAR_OUTPUT
    AVG : REAL;
  END_VAR
  VAR
    buff : DELAY;
    i : INT;
    init : BOOL;
  END_VAR

  buff.N := LIMIT(0, N, 32);

  IF NOT init OR rst THEN
        FOR i := 1 TO N DO
                buff(in := in);
        END_FOR;
        avg := in;
        init := TRUE;
  ELSIF _E THEN
        buff(in := in);
        avg := avg + (in - buff.out ) / INT_TO_REAL(N);
  END_IF;
  (* from OSCAT library www.oscat.de *)
  (* FB FC delay and inc1 requiered *)
END_FUNCTION_BLOCK


CONFIGURATION config
  VAR_GLOBAL
    seqx0set : REAL := 50.0;
    seqx1set : REAL := 24.0;
    seqx2set : REAL := 26.0;
    seqx3set : REAL := 74.0;
    seqx4set : REAL := 76.0;
    modesupplyorextractcontrolset : REAL := 1.0;
    modeahunormalorreducedset : REAL;
    tsupplyahu1 : REAL;
    tsupplyahu2 : REAL;
    tminheatingnormalset : REAL := 19.0;
    tminheatingreduceset : REAL := 15.0;
    tmaxcoolingnormalset : REAL := 25.0;
    tmaxcoolingreduceset : REAL := 27.0;
    textractzone1 : REAL;
    textractzone2 : REAL;
    kpcascset : REAL := 0.05;
    tncascset : REAL := 500.0;
    tvcascset : REAL := 0.0;
    tsupplyminset : REAL := 18.0;
    tsupplymaxset : REAL := 35.0;
    toutdoor : REAL;
    tmaxdeltasupplyextractset : REAL := 15.0;
    kpseqlowset : REAL := 0.7;
    kpseqhighset : REAL := 2;
    tnseqlowset : REAL := 300;
    tnseqhighset : REAL := 100;
    tvseqlowset : REAL := 0.0;
    tvseqhighset : REAL := 0.0;
    modeahuonoroffset : REAL;
    nfreshminset : REAL := 10.0;
    kpvavset : REAL := 1.2;
    tnvavset : REAL := 100.0;
    tvvavset : REAL := 0.0;
    seqctrlthresholdset : REAL := 1.0;
    seqctrldeadbandset : REAL := 0.5;
    tymixset : REAL := 1.0;
    tyhxset : REAL := 1.0;
    tyhcset : REAL := 1.0;
    tyccset : REAL := 1.0;
    dttoleranceahutovavset : REAL := 1.0;
```

```
    textractahu1 : REAL;
    textractahu2 : REAL;
    dpsupplyfanducttoambientnormalset : REAL := 200.0;
    dpextractfanducttoambientnormalset : REAL := 150.0;
    dpsupplyfanducttoambientreduceset : REAL := 200.0;
    dpextractfanducttoambientreduceset : REAL := 150.0;
    nvavminzone1set : REAL := 0.0;
    nvavminzone2set : REAL := 0.0;
    emptymodelconnection : REAL := 0.0;
    textractahumean : REAL;
    tsupplyahumean : REAL;
    ymix : REAL;
    yhx : REAL;
    yhc : REAL;
    ycc : REAL;
    textractcooling : REAL;
    textractheating : REAL;
    ypidcascc : REAL;
    ypidcasch : REAL;
    tsupplyahutarget : REAL;
    seqypid : REAL := 49.0;
    seqoutcorrect : REAL;
    extractaboveoutdoorreal : REAL;
    yvavzone1 : REAL;
    yvavzone2 : REAL;
    kpseqcalc : REAL;
    tnseqcalc : REAL;
    tvseqcalc : REAL;
    tminheatingselect : REAL;
    tmaxcoolingselect : REAL;
    ylintransh : REAL;
    ylintransc : REAL;
    toutdoorsmooth : REAL;
    dpsupplyfanducttoambientselect : REAL;
    dpextractfanducttoambientselect : REAL;
    modepumphc : REAL;
  END_VAR

  RESOURCE resource1 ON PLC
    TASK task0(INTERVAL := T#100ms,PRIORITY := 0);
    PROGRAM instance0 WITH task0 : RLT_6_11;
  END_RESOURCE
END_CONFIGURATION
```

B.3 Implementation 3

Listing B.2: IEC 61131-3 program implementation 3

```
FUNCTION T_PLC_US : UDINT
  VAR
    tx : UDINT;
  END_VAR
  VAR_INPUT
    debug : BOOL;
  END_VAR
  VAR
    N : INT := 0;
    offset : UDINT := 0;
    temp : DWORD := 1;
  END_VAR

  {extern unsigned long __tick;
  extern unsigned long long common_ticktime__;
  unsigned long long ticktime_ms = (common_ticktime__)/1000000;
  UDINT plc_time = (UDINT)(ticktime_ms * (unsigned long long)__tick);
  TX = plc_time}

  T_PLC_US := tx*1000;
  IF debug THEN
        T_PLC_US := (DWORD_TO_UDINT(SHL(UDINT_TO_DWORD(T_PLC_US),N) OR
            ↪ SHL(temp,N))-1) + OFFSET;
  END_IF;
```

```
  (* Original Code:
  tx := TIME();
  T_PLC_US := TIME_TO_DWORD(Tx)*1000;
  IF debug THEN
        T_PLC_US := (SHL(T_PLC_US,N) OR SHL(DWORD#1,N)-1) + OFFSET;
  END_IF;
  *)

  (* From OSCAT library, www.oscat.de

  this is a temporary T_PLC_US FB until OpenPLC gets its own time()
     ↪ functionality *)

  (* PLC_TIME and Global variables PLC_SCAN_CYCL and PLC_CYCL_TIME
     ↪ required *)
END_FUNCTION

FUNCTION_BLOCK FT_PIWL
  VAR_INPUT
    IN : REAL;
    KP : REAL := 1.0;
    KI : REAL := 1.0;
    LIM_L : REAL := -1.0E38;
    LIM_H : REAL := 1.0E38;
    RST : BOOL;
  END_VAR
  VAR_OUTPUT
    Y : REAL;
    LIM : BOOL;
  END_VAR
  VAR
    init : BOOL;
    tx : UDINT;
    tc : REAL;
    t_last : UDINT;
    in_last : REAL;
    i : REAL;
    p : REAL;
  END_VAR

  IF NOT init OR RST THEN
        init := TRUE;
        in_last := in;
                t_last := T_PLC_US(en:=true);
        i := 0.0;
        tc := 0.0;
  ELSE
        (* read last cycle time in Microseconds *)
                tx := T_PLC_US(en:=true);
        tc := UDINT_TO_REAL(tx - t_last);
        t_last := tx;

        (* calculate proportional part *)
        p := KP * IN;

        (* run integrator *)
        i := (IN + in_last) * 5.0E-7 * KI * tc + i;
        (*i := (IN + in_last) * 0.5 * KI * 1.0 + i;*)
        in_last := IN;

        (* calculate output Y *)
        Y := p + i;

        (* check output for limits *)
        IF Y >= LIM_H THEN
                Y := LIM_H;
                IF ki <> 0.0 THEN
                        i := LIM_H - p;
                ELSE
                        i := 0.0;
                END_IF;
                LIM := TRUE;
        ELSIF Y <= LIM_L THEN
                Y := LIM_L;
                IF ki <> 0.0 THEN
                        i := LIM_L - p;
```

```
                ELSE
                        i := 0.0;
                END_IF;
                LIM := TRUE;
        ELSE
                LIM := FALSE;
        END_IF;
  END_IF;

  (* From OSCAT Library, www.oscat.de *)
  (* T_PLC_US required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK FT_DERIV
  VAR_INPUT
    IN : REAL;
    K : REAL := 1.0;
    RUN : BOOL := TRUE;
  END_VAR
  VAR_OUTPUT
    OUT : REAL;
  END_VAR
  VAR
    old : REAL;
    tx : UDINT;
    last : UDINT;
    init : BOOL;
    tc : REAL;
  END_VAR

  (*tx:= T_PLC_US(en:=true);
  tc := UDINT_TO_REAL(tx - last);*)

  (* init on firsat startup *)
  IF NOT init THEN
        init := TRUE;
        old := in;
  ELSIF run AND tc > 0.0 THEN
        (*out := (in - old) / tc * 1000000.0 * K;*)
        out := (in - old) / 1.0 * K;
        old := in;
  ELSE
        out := 0.0;
  END_IF;

  (*last := tx;*)

  (* From OSCAT Library, www.oscat.de *)
  (* T_PLC_US, required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK FT_PIDWL
  VAR_INPUT
    IN : REAL;
    KP : REAL := 1.0;
    TN : REAL := 1.0;
    TV : REAL := 1.0;
    LIM_L : REAL := -1.0E38;
    LIM_H : REAL := 1.0E38;
    RST : BOOL;
  END_VAR
  VAR_OUTPUT
    Y : REAL;
    LIM : BOOL;
  END_VAR
  VAR
    piwl : FT_PIWL;
    diff : FT_DERIV;
  END_VAR

  IF rst THEN
        piwl(rst := TRUE);
        piwl.RST := FALSE;
  ELSE
        (* run PIWL controller first *)
        (* we need to check if TN = 0 and do alternative calls *)
        IF TN = 0.0 THEN
```

```
                piwl(in := IN * KP, KP := 1.0, KI := 0.0, LIM_L := LIM_L, LIM_H := LIM_H);
        ELSE
                piwl(in := IN * KP, KP := 1.0, KI := 1.0 / TN, LIM_L := LIM_L, LIM_H := LIM_H);
        END_IF;

        (* run differentiator and add_to_output *)
        diff(IN := IN, K := KP * TV);
        Y := piwl.Y + diff.out;

        (* limit the output *)
        IF Y < LIM_L THEN
                LIM := TRUE;
                Y := LIM_L;
        ELSIF Y > LIM_H THEN
                LIM := TRUE;
                Y := LIM_H;
        ELSE
                LIM := FALSE;
        END_IF;
  END_IF;



  (* From OSCAT Library, www.oscat.de *)
  (* T_PLC_US, FT_DERIV required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK CTRL_OUT
  VAR_INPUT
    CI : REAL;
    OFFSET : REAL;
    MAN_IN : REAL;
    LIM_L : REAL;
    LIM_H : REAL;
    MANUAL : BOOL;
  END_VAR
  VAR_OUTPUT
    Y : REAL;
    LIM : BOOL;
  END_VAR

  Y := SEL(manual, CI, MAN_IN) + OFFSET;

  (* Limit the output *)
  IF Y >= LIM_H THEN
        Y := LIM_H;
        LIM := TRUE;
  ELSIF Y <= lim_L THEN
        Y := LIM_L;
        LIM := TRUE;
  ELSE
        LIM := FALSE;
  END_IF;

  (* From OSCAT Library, www.oscat.de *)
END_FUNCTION_BLOCK

FUNCTION DEAD_ZONE : REAL
  VAR_INPUT
    X : REAL;
    L : REAL;
  END_VAR

  IF ABS(x) > L THEN
        dead_zone := X;
  ELSE
        DEAD_ZONE := 0.0;
  END_IF;

  (* From OSCAT Library, www.oscat.de *)
END_FUNCTION

FUNCTION CTRL_IN : REAL
  VAR_INPUT
    SET_POINT : REAL;
    ACTUAL : REAL;
```

```
    NOISE : REAL;
  END_VAR

  CTRL_IN := DEAD_ZONE(SET_POINT - ACTUAL, NOISE);

  (* From OSCAT Library, www.oscat.de *)
  (* DEAD_ZONE required *)
END_FUNCTION

FUNCTION_BLOCK CTRL_PID
  VAR_INPUT
    ACT : REAL;
    SET : REAL;
    SUP : REAL;
    OFS : REAL;
    M_I : REAL;
    MAN : BOOL;
    RST : BOOL := FALSE;
    KP : REAL := 1.0;
    TN : REAL := 1.0;
    TV : REAL := 1.0;
    LL : REAL := -1000.0;
    LH : REAL := 1000.0;
  END_VAR
  VAR_OUTPUT
    Y : REAL;
    DIFF : REAL;
    LIM : BOOL;
  END_VAR
  VAR
    _pid : FT_PIDWL;
    co : CTRL_OUT;
  END_VAR

  DIFF := CTRL_IN(SET, ACT, SUP);
  _pid(in := DIFF, kp := KP, tn := TN, tv := TV, lim_l := LL, lim_h := LH, rst := RST);
  co(ci := _pid.Y, OFFSET := OFS, man_in := M_I, lim_l := LL, lim_h := LH, manual := MAN);
  Y := co.Y;
  LIM := co.LIM;


  (* From OSCAT Library, www.oscat.de *)
  (* CTRL_IN, FT_PIDWL, CTRL_out reauired *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK FT_PT1
  VAR_INPUT
    IN : REAL;
    _T : TIME := t#1s;
    K : REAL := 1.0;
  END_VAR
  VAR_OUTPUT
    OUT : REAL;
  END_VAR
  VAR
    last : UDINT;
    tx : UDINT;
    init : BOOL;
  END_VAR

  tx:= T_PLC_US(en:=true);

  (* startup initialisation *)
  IF NOT init OR _T = t#0s THEN
        init := TRUE;
        out := K * in;
  ELSE
        out := out + (in * K - out) * UDINT_TO_REAL(Tx - last) / TIME_TO_REAL(_T) * 1.0E-3;
        IF ABS(out) < 1.0E-20 THEN out := 0.0; END_IF;
  END_IF;
  last := tx;

  (* From OSCAT Library, www.oscat.de *)
  (* T_PLC_US required *)
```

```
END_FUNCTION_BLOCK

FUNCTION_BLOCK vavctrl
  VAR_INPUT
    tminheatingselectvav : REAL;
    tmaxcoolingselectvav : REAL;
    tmeasure : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
  END_VAR
  VAR
    pidvavc : CTRL_PID;
    pidvavh : CTRL_PID;
  END_VAR
  VAR_OUTPUT
    ypidvavc : REAL;
    ypidvavh : REAL;
  END_VAR
  VAR_INPUT
    kp : REAL;
    tn : REAL;
    tv : REAL;
  END_VAR
  VAR
    pt1vav : FT_PT1;
  END_VAR
  VAR_INPUT
    nmin : REAL;
  END_VAR
  VAR_OUTPUT
    modeOff : REAL;
    modeOn : REAL;
    modeTemperatureOff : REAL;
    modeCoolingOn : REAL;
    modeHeatingOn : REAL;
    modeAirQualityOn : REAL;
    modeAirQualityOff : REAL;
  END_VAR

  pidvavc(
       ACT := tmaxcoolingselectvav,
       SET := tmeasure,
       SUP := 0.0,
       OFS := 0.0,
       M_I := 0.0,
       MAN := false,
       RST := false,
       KP := kp,
       TN := tn,
       TV := tv,
       LL := nmin,
       LH := 100.0,
       Y => ypidvavc);

  pidvavh(
       ACT := tmeasure,
       SET := tminheatingselectvav,
       SUP := 0.0,
       OFS := 0.0,
       M_I := 0.0,
       MAN := false,
       RST := false,
       KP := kp,
       TN := tn,
       TV := tv,
       LL := nmin,
       LH := 100.0,
       Y => ypidvavh);

  (* consider defined minimum value *)
  out := max(ypidvavc, ypidvavh);

  if false then
    pt1vav(
       IN := out,
       _T := t#12h,
       K := 1.0,
```

```
          OUT => out);
    end_if;

    (* operation modes *)
    if (ypidvavh = 0.0) and (ypidvavc = 0.0) then
      modeOff := 1.0;
      modeOn := 0.0;
      modeTemperatureOff := 1.0;
      modeCoolingOn := 0.0;
      modeHeatingOn := 0.0;
      modeAirQualityOn := 0.0;
      modeAirQualityOff := 1.0;
    else
      modeOff := 0.0;
      modeOn := 1.0;
      modeTemperatureOff := 0.0;
      modeAirQualityOn := 0.0;
      modeAirQualityOff := 0.0;
      if ypidvavh > 0.0 then
        modeCoolingOn := 0.0;
        modeHeatingOn := 1.0;
      elsif ypidvavc > 0.0 then
        modeCoolingOn := 1.0;
        modeHeatingOn := 0.0;
      end_if;
    end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqctrl
    VAR_INPUT
      sp : REAL;
      pv : REAL;
    END_VAR
    VAR
      manual : BOOL;
    END_VAR
    VAR_INPUT
      modeahuonoroff : REAL;
    END_VAR
    VAR
      pidseq : CTRL_PID;
    END_VAR
    VAR_INPUT
      kplow : REAL;
      kphigh : REAL;
    END_VAR
    VAR_OUTPUT
      kpcalc : REAL;
    END_VAR
    VAR_INPUT
      tnlow : REAL;
      tnhigh : REAL;
    END_VAR
    VAR_OUTPUT
      tncalc : REAL;
    END_VAR
    VAR_INPUT
      tvlow : REAL;
      tvhigh : REAL;
    END_VAR
    VAR_OUTPUT
      tvcalc : REAL;
      out : REAL;
    END_VAR
    VAR
      diff : REAL;
      diffperc : REAL;
    END_VAR
    VAR_INPUT
      seqctrlthreshold : REAL;
      seqctrldeadband : REAL;
      tsupplytargeteqtfresh : REAL;
    END_VAR

    if (modeahuonoroff <> 1.0) or (tsupplytargeteqtfresh = 1.0) then
      manual := true;
    else
      manual := false;
    end_if;
```

```

  diff := abs(sp - pv);
  if sp <> 0.0 then
    diffperc := diff ;
  else
    diffperc := diff;
  end_if;

  if diffperc > (seqctrlthreshold + seqctrldeadband * 0.5) then
    kpcalc := kphigh;
    tncalc := tnhigh;
    tvcalc := tvhigh;
  elsif diffperc < (seqctrlthreshold - seqctrldeadband * 0.5) then
    kpcalc := kplow;
    tncalc := tnlow;
    tvcalc := tvlow;
  else
    kpcalc := kplow + (diff - (seqctrlthreshold - seqctrldeadband * 0.5))
        ↪ * (kphigh - kplow) / (seqctrldeadband);
    tncalc := tnlow + (diff - (seqctrlthreshold - seqctrldeadband * 0.5))
        ↪ * (tnhigh - tnlow) / (seqctrldeadband);
    tvcalc := tvlow + (diff - (seqctrlthreshold - seqctrldeadband * 0.5))
        ↪ * (tvhigh - tvlow) / (seqctrldeadband);
  end_if;

  pidseq(
        ACT := pv,
        SET := sp,
        SUP := 0.0,
        OFS := 0.0,
        M_I := 50.0,
        MAN := manual,
        RST := false,
        KP := kpcalc,
        TN := tncalc,
        TV := tvcalc,
        LL := 0.0,
        LH := 100.0,
        Y => out);
END_FUNCTION_BLOCK

FUNCTION INC1 : INT
  VAR_INPUT
    X : INT;
    N : INT;
  END_VAR

  IF X >= N - 1 THEN
        INC1 := 0;
  ELSE
        INC1 := X + 1;
  END_IF;
  (* from OSCAT library www.oscat.de *)
END_FUNCTION

FUNCTION_BLOCK DELAY
  VAR_INPUT
    IN : REAL;
    N : INT;
    RST : BOOL;
  END_VAR
  VAR_OUTPUT
    OUT : REAL;
  END_VAR
  VAR
    i : INT;
    init : BOOL;
    stop : INT;
    buf : ARRAY [0..31] OF REAL;
  END_VAR

  stop := LIMIT(0,N,32) - 1;
  IF rst OR NOT init THEN
        init := TRUE;
        FOR i := 0 TO stop DO buf[i] := in; END_FOR;
        out := in;
        i := 0;
  ELSIF stop < 0 THEN
```

```
        out := in;
  ELSE
        out := buf[i];
        buf[i] := in;
        i := INC1(i, N);
  END_IF;
  (* from OSCAT library www.oscat.de *)
  (* inc1 requiered *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqmix
  VAR_INPUT
    in : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
  END_VAR
  VAR
    m1 : REAL;
    m2 : REAL;
    n1 : REAL;
    n2 : REAL;
  END_VAR
  VAR_INPUT
    extractabovefresh : BOOL;
    recirculationonly : BOOL;
    x0 : REAL;
    x1 : REAL;
    x2 : REAL;
    x3 : REAL;
    x4 : REAL;
  END_VAR
  VAR
    pt1 : FT_PT1;
    pt2 : DINT;
  END_VAR
  VAR_OUTPUT
    mode : REAL;
  END_VAR
  VAR_INPUT
    nmin : REAL;
  END_VAR

  (* parameters for heating *)
  m1 := (100.0 - nmin) / (x4 - x3);
  n1 := -m1 * x3;

  (* parameters for cooling *)
  m2 := (100.0 - nmin) / (x1 - x2);
  n2 := -m2 * x2;

  if recirculationonly then
    out := 100.0 - nmin;
    mode := 0.0;
  else
    (* heating case *)
    if in > x3 and extractabovefresh then
      out := min(m1 * in + n1, 100.0);
      mode := 1.0;
    (* cooling case *)
    elsif in < x2 and not extractabovefresh then
      out := min(m2 * in + n2, 100.0);
      mode := 2.0;
    else
      out := 0.0;
      mode := 0.0;
    end_if;
  end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK HYST_2
  VAR_INPUT
    IN : REAL;
    VAL : REAL;
    HYS : REAL;
  END_VAR
  VAR_OUTPUT
    Q : BOOL;
```

```
    WIN : BOOL;
  END_VAR
  VAR
    tmp : REAL;
  END_VAR

  tmp := val - hys * 0.5;
  IF in < tmp THEN
        Q := FALSE;
        win := FALSE;
  ELSIF in > tmp + hys THEN
        Q := TRUE;
        win := FALSE;
  ELSE
        win := TRUE;
  END_IF;

  (* From OSCAT Library, www.oscat.de *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqcc
  VAR_INPUT
    in : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
  END_VAR
  VAR
    m : REAL;
    n : REAL;
  END_VAR
  VAR_INPUT
    x1 : REAL;
  END_VAR
  VAR
    pt1 : FT_PT1;
  END_VAR

  m := 100.0 / (0.0 - x1);
  n := 100.0;

  if in < x1 then
    out := m * in + n;
  else
    out := 0.0;
  end_if;

  if false then
    pt1(
       IN := out,
       _T := t#1h,
       K := 1.0,
       OUT => out);
  end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqhc
  VAR_INPUT
    in : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
    pump : REAL;
  END_VAR
  VAR
    m : REAL;
    n : REAL;
  END_VAR
  VAR_INPUT
    x0 : REAL;
    x4 : REAL;
  END_VAR
  VAR
    pt1 : FT_PT1;
  END_VAR

  m := (100.0 - 0.0) / (100.0 - x4);
  n := -m * x4;

```

```

  if in > x4 then
    out := m * in + n;
    pump := 1.0;
  else
    out := 0.0;
    pump := 0.0;
  end_if;

  if false then
    pt1(
       IN := out,
       _T := t#1h,
       K := 1.0,
       OUT => out);
   end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK seqhx
  VAR_INPUT
    x0 : REAL;
    x2 : REAL;
    x3 : REAL;
    in : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
  END_VAR
  VAR
    m1 : REAL;
    m2 : REAL;
    n1 : REAL;
    n2 : REAL;
  END_VAR
  VAR_INPUT
    extractabovefresh : BOOL;
    recirculationonly : BOOL;
  END_VAR
  VAR
    pt1 : FT_PT1;
  END_VAR
  VAR_OUTPUT
    mode : REAL;
  END_VAR

  (* parameters for heating *)
  m1 := (100.0 - 0.0) / (x3 - x0);
  n1 := -m1 * x0;


  (* parameters for cooling *)
  m2 := (100.0 - 0.0) / (x2 - x0);
  n2 := -m2 * x0;

  if recirculationonly then
    out := 0.0;
    mode := 0.0;
  else
    (*heating case*)
    if in > x0 and extractabovefresh then
      out := min(m1 * in + n1, 100.0);
      mode := 1.0;
    (*cooling case*)
    elsif in <= x0 and not extractabovefresh then
      out := min(m2 * in + n2, 100.0);
      mode := 2.0;
    else
      out := 0.0;
      mode := 0.0;
    end_if;
  end_if;

  if false then
    pt1(
       IN := out,
       _T := t#1h,
       K := 1.0,
       OUT => out);
```

```
    end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK lintrans
  VAR_INPUT
    InSignal : REAL;
    MinValue : REAL;
    MaxValue : REAL;
  END_VAR
  VAR_OUTPUT
    OutSignal : REAL;
  END_VAR

  OutSignal := MinValue + InSignal / 1.0 * (MaxValue - MinValue);
END_FUNCTION_BLOCK

FUNCTION_BLOCK calctsupply
  VAR
    pidc : CTRL_PID;
    pidh : CTRL_PID;
  END_VAR
  VAR_INPUT
    textractcooling : REAL;
    textractheating : REAL;
    tmax : REAL;
    tmin : REAL;
  END_VAR
  VAR_OUTPUT
    out : REAL;
  END_VAR
  VAR_INPUT
    kp : REAL;
    tn : REAL;
    tv : REAL;
    modesupplyorextractcontrol : REAL;
    tsupplymin : REAL;
    tsupplymax : REAL;
    tfresh : REAL;
    tmaxdeltasupplyextract : REAL;
  END_VAR
  VAR_OUTPUT
    ypidcascc : REAL;
    ypidcasch : REAL;
  END_VAR
  VAR
    lintransc : lintrans;
    lintransh : lintrans;
  END_VAR
  VAR_OUTPUT
    ylintransc : REAL;
    ylintransh : REAL;
  END_VAR
  VAR
    maxin1 : REAL;
    maxin2 : REAL;
    pt1out : FT_PT1;
  END_VAR
  VAR_OUTPUT
    tsupplytargeteqtfresh : REAL;
  END_VAR

  pidc(
        ACT := textractcooling,
        SET := tmax,
        SUP := 0.0,
        OFS := 0.0,
        M_I := 0.0,
        MAN := false,
        RST := false,
        KP := kp,
        TN := tn,
        TV := tv,
        LL := 0.0,
        LH := 1.0,
        Y => ypidcascc);

  lintransc(
        InSignal := ypidcascc,
        MinValue := tsupplymin,
```

```
        MaxValue := tsupplymax,
        OutSignal => ylintransc);

  (*ylintransc := 20.0;*)

  pidh(
        ACT := textractheating,
        SET := tmin,
        SUP := 0.0,
        OFS := 0.0,
        M_I := 0.0,
        MAN := false,
        RST := false,
        KP := kp,
        TN := tn,
        TV := tv,
        LL := 0.0,
        LH := 1.0,
        Y => ypidcasch);

  lintransh(
        InSignal := ypidcasch,
        MinValue := tsupplymin,
        MaxValue := tsupplymax,
        OutSignal => ylintransh);

  maxin1 := textractcooling - tmaxdeltasupplyextract;
  maxin2 := min(ylintransc, max(tfresh, ylintransh));

  out := max(maxin1, maxin2);

  if out = tfresh then
    tsupplytargeteqtfresh := 1.0;
  else
    tsupplytargeteqtfresh := 0.0;
  end_if;

  if false then
  pt1out(
        IN := out,
        _T := t#2h,
        K := 1.0,
        OUT => out);
  end_if;
END_FUNCTION_BLOCK

FUNCTION_BLOCK ahu
  VAR_INPUT
    textract : REAL;
    tfresh : REAL;
  END_VAR
  VAR
    seq1 : seqctrl;
    hystheat : HYST_2;
    hystfresh : HYST_2;
    extractabovefresh : BOOL;
  END_VAR
  VAR_OUTPUT
    extractabovefreshreal : REAL;
  END_VAR
  VAR_INPUT
    tmaxcoolingselect : REAL;
    tminheatingselect : REAL;
  END_VAR
  VAR
    freshabovelimit : BOOL;
  END_VAR
  VAR_INPUT
    modeahuonoroffset : REAL;
    seqctrlthreshold : REAL;
    seqctrldeadband : REAL;
    tsupplytarget : REAL;
    tsupplyahu : REAL;
    kpseqlow : REAL;
    tnseqlow : REAL;
    tvseqlow : REAL;
    kpseqhigh : REAL;
    tnseqhigh : REAL;
```

```
    tvseqhigh : REAL;
  END_VAR
  VAR_OUTPUT
    kpseqcalc : REAL;
    tnseqcalc : REAL;
    tvseqcalc : REAL;
    seqypid : REAL;
  END_VAR
  VAR
    m1 : REAL;
    m2 : REAL;
    n1 : REAL;
    n2 : REAL;
  END_VAR
  VAR_INPUT
    seqx0 : REAL;
    seqx1 : REAL;
    seqx2 : REAL;
    seqx3 : REAL;
    seqx4 : REAL;
  END_VAR
  VAR
    recirculationonly : BOOL;
  END_VAR
  VAR_OUTPUT
    seqoutcorrect : REAL;
  END_VAR
  VAR
    sequencec : seqcc;
    sequenceh : seqhc;
    sequencehx : seqhx;
    sequencemix : seqmix;
  END_VAR
  VAR_OUTPUT
    modepumphc : REAL;
    yhc : REAL;
    ycc : REAL;
    yhx : REAL;
    ymix : REAL;
  END_VAR
  VAR_INPUT
    nfreshmin : REAL;
  END_VAR
  VAR
    tsupply : calctsupply;
  END_VAR
  VAR_INPUT
    modesupplyorextractcontrol : REAL;
    tmaxzones : REAL;
    tminzones : REAL;
    kpcasc : REAL;
    tncasc : REAL;
    tvcasc : REAL;
    tsupplymin : REAL;
    tsupplymax : REAL;
    tmaxdeltasupplyextract : REAL;
    tsupplytargeteqtfresh : REAL;
  END_VAR
  VAR_OUTPUT
    modeOff : REAL;
    modeNormalOn : REAL;
    modeOn : REAL;
    modeReduceOn : REAL;
    modeCcOn : REAL;
    modeHcOn : REAL;
    modeHcAndCcOff : REAL;
    modeHxOff : REAL;
    modeHxOn : REAL;
    modeFreshAirAirQualityOff : REAL;
    modeFreshAirTemperatureOff : REAL;
    modeFreshAirOn : REAL;
    modeFreshAirAirQualityOn : REAL;
    modeFreshAirCoolingOn : REAL;
    modeFreshAirHeatingOn : REAL;
    modeRecirculationOnlyOn : REAL;
  END_VAR
  VAR_INPUT
    modeahunormalorreduced : REAL;
  END_VAR
```

```
VAR
  modehx : REAL;
  modemix : REAL;
END_VAR

(*check if free heating is possible
true if the extract temperature is above the fresh air temperature
false if the extract temperature is below the fresh air temperature*)
hystheat(
     IN := textract,
     VAL := tfresh,
     HYS := 4.0,
     Q => extractabovefresh);
extractabovefreshreal := bool_to_real(extractabovefresh);

hystfresh(
     IN := tfresh,
     VAL := tmaxcoolingselect,
     HYS := 4.0,
     Q => freshabovelimit);

(* calculate output of sequence controller *)
seq1(
     modeahuonoroff := modeahuonoroffset,
     seqctrlthreshold := seqctrlthreshold,
     seqctrldeadband := seqctrldeadband,
     sp := tsupplytarget,
     pv := tsupplyahu,
     kplow := kpseqlow,
     tnlow := tnseqlow,
     tvlow := tvseqlow,
     kphigh := kpseqhigh,
     tnhigh := tnseqhigh,
     tvhigh := tvseqhigh,
     tsupplytargeteqtfresh := tsupplytargeteqtfresh,
     kpcalc => kpseqcalc,
     tncalc => tnseqcalc,
     tvcalc => tvseqcalc,
     out => seqypid);

(* heating case *)
m1 := (100.0 - seqx4) / (100.0 - seqx0);
n1 := seqx4 - m1 * seqx0;

(* cooling case *)
m2 := (seqx1 - 0.0) / (seqx0 - 0.0);
n2 := 0.0;

(* calculate corrected output of sequence controller depending on fresh air and extract temperature *)
if freshabovelimit then
  recirculationonly := true;
  seqoutcorrect := m2 * seqypid + n2;
else
  recirculationonly := false;

  (* cooling case *)
  (* if the extract temperature is above the fresh air temperature (can not be used for cooling) + there is cooling demand: deactivate hx and mix *)
  if extractabovefresh and seqypid < seqx0 then
    seqoutcorrect := m2 * seqypid + n2;
  (* heating case *)
  (* if the extract temperature is below the fresh air temperature (can not be used for heating) + there is heating demand: deactivate hx and mix *)
  elsif not extractabovefresh and seqypid > seqx0 then
    seqoutcorrect := m1 * seqypid + n1;
  else
    seqoutcorrect := seqypid;
  end_if;
end_if;

sequencec(
  in := seqoutcorrect,
  x1 := seqx1,
```

```
    out => ycc);

  sequenceh(
    in := seqoutcorrect,
    x0 := seqx0,
    x4 := seqx4,
    pump => modepumphc,
    out => yhc);

  sequencehx(
    in := seqoutcorrect,
    extractabovefresh := extractabovefresh,
    recirculationonly := recirculationonly,
    x0 := seqx0,
    x2 := seqx2,
    x3 := seqx3,
    mode => modehx,
    out => yhx);

  sequencemix(
    in := seqoutcorrect,
    nmin := nfreshmin,
    extractabovefresh := extractabovefresh,
    recirculationonly := recirculationonly,
    x0 := seqx0,
    x1 := seqx1,
    x2 := seqx2,
    x3 := seqx3,
    x4 := seqx4,
    mode => modemix,
    out => ymix);

  (* set positions manually if ahu is off *)
  (**)
  if modeahuonoroffset <> 1.0 then
    yhc := 0.0;
    ycc := 0.0;
    yhx := 0.0;
    ymix := 100.0;
  end_if;
  (**)

  (* operation modes *)
  if modeahuonoroffset = 0.0 then
    modeOff := 1.0;
    modeNormalOn := 0.0;
    modeOn := 0.0;
    modeReduceOn := 0.0;
    modeCcOn := 0.0;
    modeHcOn := 0.0;
    modeHcAndCcOff := 0.0;
    modeHxOff := 0.0;
    modeHxOn := 0.0;
    modeFreshAirTemperatureOff := 0.0;
    modeFreshAirOn := 0.0;
    modeFreshAirCoolingOn := 0.0;
    modeFreshAirHeatingOn := 0.0;
    modeRecirculationOnlyOn := 0.0;
    modeFreshAirAirQualityOn := 0.0;
    modeFreshAirAirQualityOff := 0.0;
  else
    modeOff := 0.0;
    modeOn := 1.0;
    if modeahunormalorreduced = 1.0 then
      modeNormalOn := 1.0;
      modeReduceOn := 0.0;
    elsif modeahunormalorreduced = 2.0 then
      modeNormalOn := 0.0;
      modeReduceOn := 1.0;
    end_if;
    if (yhc > 0.0) or (ycc > 0.0) then
      modeHcAndCcOff := 0.0;
      if yhc > 0.0 then
        modeHcOn := 1.0;
        modeCcOn := 0.0;
      end_if;
      if ycc > 0.0 then
        modeHcOn := 0.0;
```

```
        modeCcOn := 1.0;
      end_if;
    else
      modeHcAndCcOff := 1.0;
      modeCcOn := 0.0;
      modeHcOn := 0.0;
    end_if;
    if yhx > 0.0 then
      modeHxOff := 0.0;
      modeHxOn := 1.0;
    else
      modeHxOff := 1.0;
      modeHxOn := 0.0;
    end_if;
    if ymix > 0.0 then
      modeFreshAirOn := 1.0;
      if modemix = 0.0 then
        modeFreshAirTemperatureOff := 1.0;
        modeFreshAirHeatingOn := 0.0;
        modeFreshAirCoolingOn := 0.0;
      elsif modemix = 1.0 then
        modeFreshAirTemperatureOff := 0.0;
        modeFreshAirHeatingOn := 1.0;
        modeFreshAirCoolingOn := 0.0;
      elsif modemix = 2.0 then
        modeFreshAirTemperatureOff := 0.0;
        modeFreshAirHeatingOn := 0.0;
        modeFreshAirCoolingOn := 1.0;
      end_if;
    else
      modeFreshAirOn := 0.0;
      modeFreshAirTemperatureOff := 0.0;
      modeFreshAirCoolingOn := 0.0;
      modeFreshAirHeatingOn := 0.0;
    end_if;
  end_if;
  modeFreshAirAirQualityOff := 1.0;
  modeFreshAirAirQualityOn := 0.0;
END_FUNCTION_BLOCK

FUNCTION T_PLC_MS : UDINT
  VAR
    tx : UDINT;
  END_VAR
  VAR_INPUT
    debug : BOOL;
  END_VAR
  VAR
    N : INT := 0;
    offset : UDINT := 0;
    temp : DWORD := 1;
  END_VAR

  tx := 0;

  {extern unsigned long __tick;
  extern unsigned long long common_ticktime__;
  unsigned long long ticktime_ms = (common_ticktime__)/1000000;
  UDINT plc_time = (UDINT)(ticktime_ms * (unsigned long long)__tick);
  TX = plc_time}

  T_PLC_MS := tx;
  IF debug THEN
        T_PLC_MS := (DWORD_TO_UDINT(SHL(UDINT_TO_DWORD(T_PLC_MS),N) OR SHL(temp,N))-1) + OFFSET;
  END_IF;

  (* Original Code:
  tx := TIME();
  T_PLC_MS := TIME_TO_DWORD(Tx);
  IF debug THEN
        T_PLC_MS := (SHL(T_PLC_MS,N) OR SHL(DWORD#1,N)-1) + OFFSET;
  END_IF;
  *)

  (* From OSCAT library, www.oscat.de
```

```
  this is a temporary T_PLC_MS FB until OpenPLC gets its own time()
      ↪ functionality *)

  (* PLC_TIME and Global variables PLC_SCAN_CYCL and PLC_CYCL_TIME
      ↪ required *)
END_FUNCTION

FUNCTION_BLOCK INTEGRATE
  VAR_INPUT
    _E : BOOL := TRUE;
    X : REAL;
    K : REAL := 1.0;
  END_VAR
  VAR_IN_OUT
    Y : REAL;
  END_VAR
  VAR
    x_last : REAL;
    init : BOOL;
    last : UDINT;
    tx : UDINT;
  END_VAR

  tx:= T_PLC_MS(en:=true);

  IF NOT init THEN
        init := TRUE;
        X_last := X;
  ELSIF _E THEN
        Y := (X + X_LAST) * 0.5E-3 * UDINT_TO_REAL(tx-last) * K + Y;
        X_last := X;
  END_IF;
  last := tx;

  (* From OSCAT Library, www.oscat.de *)
  (* T_PLC_MS required *)
END_FUNCTION_BLOCK

PROGRAM RLT_6_11
  VAR
    pt1_1 : FT_PT1;
    pt1_2 : FT_PT1;
  END_VAR
  VAR_EXTERNAL
    modeahunormalorreducedset : REAL;
    modesupplyorextractcontrolset : REAL;
    tsupplyahu1 : REAL;
    tsupplyahu2 : REAL;
    tmaxzones : REAL;
    tminzones : REAL;
    textractzone1 : REAL;
    textractzone2 : REAL;
    textractahu1 : REAL;
    textractahu2 : REAL;
    kpcascset : REAL;
    tncascset : REAL;
    tvcascset : REAL;
    tsupplyminwinterset : REAL;
    tsupplyminsommerset : REAL;
    tsupplymaxset : REAL;
    toutdooroffice : REAL;
    tfreshahu1 : REAL;
    tfreshahu2 : REAL;
    tmaxdeltasupplyextractset : REAL;
    kpseqlowset : REAL;
    kpseqhighset : REAL;
    kpseqcalcahu1 : REAL;
    kpseqcalcahu2 : REAL;
    tnseqlowset : REAL;
    tnseqhighset : REAL;
    tnseqcalcahu1 : REAL;
    tnseqcalcahu2 : REAL;
    tvseqlowset : REAL;
    tvseqhighset : REAL;
    tvseqcalcahu1 : REAL;
    tvseqcalcahu2 : REAL;
    tsupplytarget : REAL;
    yccahu1 : REAL;
```

```
    yccahu2 : REAL;
    yhcahu1 : REAL;
    yhcahu2 : REAL;
    ymixahu1 : REAL;
    ymixahu2 : REAL;
    yhxahu1 : REAL;
    yhxahu2 : REAL;
    modeahuonoroffset : REAL;
    extractaboveoutdoorrealahu1 : REAL;
    extractaboveoutdoorrealahu2 : REAL;
    seqypidahu1 : REAL;
    seqypidahu2 : REAL;
    seqoutcorrectahu1 : REAL;
    seqoutcorrectahu2 : REAL;
    seqx0set : REAL;
    seqx1set : REAL;
    seqx2set : REAL;
    seqx3set : REAL;
    seqx4set : REAL;
    tminheatingselect : REAL;
    tmaxcoolingselect : REAL;
    tminheatingnormalset : REAL;
    tmaxcoolingnormalset : REAL;
    tminheatingreduceset : REAL;
    tmaxcoolingreduceset : REAL;
    nfreshminset : REAL;
    nfreshmincalc : REAL;
  END_VAR
  VAR
    calctsupply1 : calctsupply;
  END_VAR
  VAR_EXTERNAL
    dttoleranceahutovavset : REAL;
  END_VAR
  VAR
    tminheatingselectahu : REAL;
    tminheatingselectvav : REAL;
    tmaxcoolingselectahu : REAL;
    tmaxcoolingselectvav : REAL;
    vavzone1 : vavctrl;
    vavzone2 : vavctrl;
  END_VAR
  VAR_EXTERNAL
    yvavzone1 : REAL;
    yvavzone2 : REAL;
  END_VAR
  VAR
    pt1tsupply : FT_PT1;
    pt1textractzone1 : FT_PT1;
    pt1textractzone2 : FT_PT1;
    pt1ymix : FT_PT1;
    pt1yhx : FT_PT1;
    pt1yhc : FT_PT1;
    pt1ycc : FT_PT1;
  END_VAR
  VAR_EXTERNAL
    tymixset : REAL;
    tyhxset : REAL;
    tyhcset : REAL;
    tyccset : REAL;
    ypidcasch : REAL;
    ypidcascc : REAL;
    ylintransh : REAL;
    ylintransc : REAL;
    kpvavset : REAL;
    tnvavset : REAL;
    tvvavset : REAL;
    seqctrlthresholdset : REAL;
    seqctrldeadbandset : REAL;
    dpsupplyfanducttoambientnormalset : REAL;
    dpextractfanducttoambientnormalset : REAL;
    dpsupplyfanducttoambientreduceset : REAL;
    dpextractfanducttoambientreduceset : REAL;
    dpsupplyfanducttoambientselect : REAL;
    dpextractfanducttoambientselect : REAL;
    nvavminzone1set : REAL;
    nvavminzone1calc : REAL;
    nvavminzone2set : REAL;
    nvavminzone2calc : REAL;
```

```
    modepumphcahu1 : REAL;
    modepumphcahu2 : REAL;
    modeOff : REAL;
    modeNormalOn : REAL;
    modeOn : REAL;
    modeReduceOn : REAL;
    modeCcOn : REAL;
    modeHcOn : REAL;
    modeHcAndCcOff : REAL;
    modeHxOff : REAL;
    modeHxOn : REAL;
    modeFreshAirAirQualityOff : REAL;
    modeFreshAirTemperatureOff : REAL;
    modeFreshAirOn : REAL;
    modeFreshAirAirQualityOn : REAL;
    modeFreshAirCoolingOn : REAL;
    modeFreshAirHeatingOn : REAL;
    modeRecirculationOnlyOn : REAL;
    modeVariableAirFlowsZone1Off : REAL;
    modeVariableAirFlowsZone2Off : REAL;
    modeVariableAirFlowsZone1AirQualityOff : REAL;
    modeVariableAirFlowsZone1TemperatureOff : REAL;
    modeVariableAirFlowsZone2AirQualityOff : REAL;
    modeVariableAirFlowsZone2TemperatureOff : REAL;
    modeVariableAirFlowsZone1On : REAL;
    modeVariableAirFlowsZone2On : REAL;
    modeVariableAirFlowsZone1AirQualityOn : REAL;
    modeVariableAirFlowsZone1CoolingOn : REAL;
    modeVariableAirFlowsZone1HeatingOn : REAL;
    modeVariableAirFlowsZone2AirQualityOn : REAL;
    modeVariableAirFlowsZone2CoolingOn : REAL;
    modeVariableAirFlowsZone2HeatingOn : REAL;
  END_VAR
  VAR
    ahu1 : ahu;
    ahu2 : ahu;
  END_VAR
  VAR_EXTERNAL
    dttoleranceahutovavoffset : REAL;
    elload1 : REAL;
    elload2 : REAL;
    nfreshminelload : REAL;
    nvavminelload : REAL;
    pelfreshmaxset : REAL;
    pelvavmaxset : REAL;
    pelsum : REAL;
    useelloadforaq : REAL;
    ypidvavhz1 : REAL;
    ypidvavhz2 : REAL;
    ypidvavcz1 : REAL;
    ypidvavcz2 : REAL;
  END_VAR
  VAR
    elloadfilter : FT_PT1;
  END_VAR
  VAR_EXTERNAL
    tsupplytargeteqtfresh : REAL;
    textractmean : REAL;
    tsupplymean : REAL;
  END_VAR
  VAR
    tsupply : calctsupply;
    tsupplymincalc : REAL;
    tfreshmean : REAL;
  END_VAR

  (* calculate mean values *)
  textractmean := 0.0;
  tsupplymean := 0.0;

  (* use electrical load for calculation of fresh air ratio and vavs *)
  pelsum := max(elload1 + elload2, 0.0);

  elloadfilter(
       IN := pelsum,
       _T := t#2h,
       K := 1.0,
       OUT => pelsum);

```

```
  if pelsum < pelfreshmaxset then
    nfreshminelload := pelsum / pelfreshmaxset * 100.0;
    nvavminelload := 0.0;
  elsif (pelsum > pelfreshmaxset) and (pelsum < pelvavmaxset) then
    nfreshminelload := 100.0;
    nvavminelload := (pelsum - pelfreshmaxset) / (pelvavmaxset -
        ↪ pelfreshmaxset) * 100.0;
  else
    nfreshminelload := 100.0;
    nvavminelload := 100.0;
  end_if;

  if useelloadforaq = 1.0 then
    nfreshmincalc := max(nfreshminset, nfreshminelload);
    nvavminzone1calc := max(nvavminzone1set, nvavminelload);
    nvavminzone2calc := max(nvavminzone2set, nvavminelload);
  else
    nfreshmincalc := nfreshminset;
    nvavminzone1calc := nvavminzone1set;
    nvavminzone2calc := nvavminzone2set;
  end_if;

  (* calculate general inputs for both ahus *)
  (**)

  (*define setpoints depending on normal or reduced operation *)
  (* normal operation *)
  (**)
  if modeahunormalorreducedset = 1.0 then
    tminheatingselect := tminheatingnormalset;
    tmaxcoolingselect := tmaxcoolingnormalset;
    dpsupplyfanducttoambientselect := dpsupplyfanducttoambientnormalset;
    dpextractfanducttoambientselect := dpextractfanducttoambientnormalset
        ↪ ;
  (**)
  (* reduced operation *)
  (**)
  elsif modeahunormalorreducedset = 2.0 then
    tminheatingselect := tminheatingreduceset;
    tmaxcoolingselect := tmaxcoolingreduceset;
    dpsupplyfanducttoambientselect := dpsupplyfanducttoambientreduceset;
    dpextractfanducttoambientselect := dpextractfanducttoambientreduceset
        ↪ ;
  else
    tminheatingselect := tminheatingreduceset;
    tmaxcoolingselect := tmaxcoolingreduceset;
    dpsupplyfanducttoambientselect := dpsupplyfanducttoambientreduceset;
    dpextractfanducttoambientselect := dpextractfanducttoambientreduceset
        ↪ ;
  end_if;
  (**)

  (* define setpoints for ahu and vavs with offset *)
  (**)
  tminheatingselectahu := tminheatingselect + dttoleranceahutovavset;
  tmaxcoolingselectahu := tmaxcoolingselect - dttoleranceahutovavset;
  tminheatingselectvav := tminheatingselect;
  tmaxcoolingselectvav := tmaxcoolingselect;
  (**)

  (* calculate minimal and maximal extract temperatures from zones *)
  (**)
  tminzones := min(textractzone1, textractzone2);
  tmaxzones := max(textractzone1, textractzone2);
  (**)

  tfreshmean := (tfreshahu1 + tfreshahu2) / 2.0;

  (* calculate minimal supply air temperature based on outdoor
      ↪ temperature *)
  if toutdooroffice < 10.0 then
    tsupplymincalc := tsupplyminwinterset;
  elsif toutdooroffice > 20.0 then
    tsupplymincalc := tsupplyminsommerset;
  else
```

```
    tsupplymincalc := tsupplyminwinterset + (tsupplyminwinterset -
        ↪ tsupplyminsommerset) / (10.0 - 20.0) * (toutdooroffice - 10.0);
  end_if;

  (*calculate target supply air temperature for both ahus*)
  (**)
  tsupply(
       modesupplyorextractcontrol := modesupplyorextractcontrolset,
       tmax := tmaxcoolingselectahu,
       tmin := tminheatingselectahu,
       textractcooling := tmaxzones,
       textractheating := tminzones,
       kp := kpcascset,
       tn := tncascset,
       tv := tvcascset,
       tsupplymin := tsupplymincalc,
       tsupplymax := tsupplymaxset,
       tfresh := tfreshmean,
       tmaxdeltasupplyextract := tmaxdeltasupplyextractset,
       out => tsupplytarget,
       ypidcasch => ypidcasch,
       ypidcascc => ypidcascc,
       ylintransh => ylintransh,
       ylintransc => ylintransc,
       tsupplytargeteqtfresh => tsupplytargeteqtfresh);
  (**)

  (* calculate controller outputs per ahu *)
  (**)
  ahu1(
    modeahunormalorreduced := modeahunormalorreducedset,
    textract := textractahu1,
    tfresh := tfreshahu1,
    tmaxcoolingselect := tmaxcoolingselectahu,
    tminheatingselect := tminheatingselectahu,
    modeahuonoroffset := modeahuonoroffset,
    seqctrlthreshold := seqctrlthresholdset,
    seqctrldeadband := seqctrldeadbandset,
    tsupplyahu := tsupplyahu1,
    kpseqlow := kpseqlowset,
    kpseqhigh := kpseqhighset,
    tnseqlow := tnseqlowset,
    tnseqhigh := tnseqhighset,
    tvseqlow := tvseqlowset,
    tvseqhigh := tvseqhighset,
    seqx0 := seqx0set,
    seqx1 := seqx1set,
    seqx2 := seqx2set,
    seqx3 := seqx3set,
    seqx4 := seqx4set,
    nfreshmin := nfreshmincalc,
    modesupplyorextractcontrol := modesupplyorextractcontrolset,
    tmaxzones := tmaxzones,
    tminzones := tminzones,
    kpcasc := kpcascset,
    tncasc := tncascset,
    tvcasc := tvcascset,
    tsupplymin := tsupplyminwinterset,
    tsupplymax := tsupplymaxset,
    tmaxdeltasupplyextract := tmaxdeltasupplyextractset,
    tsupplytarget := tsupplytarget,
    tsupplytargeteqtfresh := tsupplytargeteqtfresh,
    extractabovefreshreal => extractaboveoutdoorrealahu1,
    kpseqcalc => kpseqcalcahu1,
    tnseqcalc => tnseqcalcahu1,
    tvseqcalc => tvseqcalcahu1,
    seqypid => seqypidahu1,
    seqoutcorrect => seqoutcorrectahu1,
    yhc => yhcahu1,
    modepumphc => modepumphcahu1,
    ycc => yccahu1,
    ymix => ymixahu1,
    yhx => yhxahu1);

  ahu2(
    modeahunormalorreduced := modeahunormalorreducedset,
    textract := textractahu2,
    tfresh := tfreshahu2,
```

```
      tmaxcoolingselect := tmaxcoolingselectahu,
      tminheatingselect := tminheatingselectahu,
      modeahuonoroffset := modeahuonoroffset,
      seqctrlthreshold := seqctrlthresholdset,
      seqctrldeadband := seqctrldeadbandset,
      tsupplyahu := tsupplyahu2,
      kpseqlow := kpseqlowset,
      kpseqhigh := kpseqhighset,
      tnseqlow := tnseqlowset,
      tnseqhigh := tnseqhighset,
      tvseqlow := tvseqlowset,
      tvseqhigh := tvseqhighset,
      seqx0 := seqx0set,
      seqx1 := seqx1set,
      seqx2 := seqx2set,
      seqx3 := seqx3set,
      seqx4 := seqx4set,
      nfreshmin := nfreshmincalc,
      modesupplyorextractcontrol := modesupplyorextractcontrolset,
      tmaxzones := tmaxzones,
      tminzones := tminzones,
      kpcasc := kpcascset,
      tncasc := tncascset,
      tvcasc := tvcascset,
      tsupplymin := tsupplyminwinterset,
      tsupplymax := tsupplymaxset,
      tmaxdeltasupplyextract := tmaxdeltasupplyextractset,
      tsupplytarget := tsupplytarget,
      tsupplytargeteqtfresh := tsupplytargeteqtfresh,
      extractabovefreshreal => extractaboveoutdoorrealahu2,
      kpseqcalc => kpseqcalcahu2,
      tnseqcalc => tnseqcalcahu2,
      tvseqcalc => tvseqcalcahu2,
      seqypid => seqypidahu2,
      seqoutcorrect => seqoutcorrectahu2,
      yhc => yhcahu2,
      modepumphc => modepumphcahu2,
      ycc => yccahu2,
      ymix => ymixahu2,
      yhx => yhxahu2);


  vavzone1(
        tmaxcoolingselectvav := tmaxcoolingselectvav,
        tminheatingselectvav := tminheatingselectvav,
        tmeasure := textractzone1,
        kp := kpvavset,
        tn := tnvavset,
        tv := tvvavset,
        nmin := nvavminzone1calc,
        modeOff => modeVariableAirFlowsZone1Off,
        modeOn => modeVariableAirFlowsZone1On,
        modeTemperatureOff => modeVariableAirFlowsZone1TemperatureOff,
        modeCoolingOn => modeVariableAirFlowsZone1CoolingOn,
        modeHeatingOn => modeVariableAirFlowsZone1HeatingOn,
        modeAirQualityOn => modeVariableAirFlowsZone1AirQualityOn,
        modeAirQualityOff => modeVariableAirFlowsZone1AirQualityOff,
        out => yvavzone1,
        ypidvavh => ypidvavhz1,
        ypidvavc => ypidvavcz1);

  vavzone2(
        tmaxcoolingselectvav := tmaxcoolingselectvav,
        tminheatingselectvav := tminheatingselectvav,
        tmeasure := textractzone2,
        kp := kpvavset,
        tn := tnvavset,
        tv := tvvavset,
        nmin := nvavminzone2calc,
        modeOff => modeVariableAirFlowsZone2Off,
        modeOn => modeVariableAirFlowsZone2On,
        modeTemperatureOff => modeVariableAirFlowsZone2TemperatureOff,
        modeCoolingOn => modeVariableAirFlowsZone2CoolingOn,
        modeHeatingOn => modeVariableAirFlowsZone2HeatingOn,
        modeAirQualityOn => modeVariableAirFlowsZone2AirQualityOn,
        modeAirQualityOff => modeVariableAirFlowsZone2AirQualityOff,
        out => yvavzone2,
        ypidvavh => ypidvavhz2,
        ypidvavc => ypidvavcz2);
```

```
END_PROGRAM

FUNCTION_BLOCK FT_PT2
  VAR_INPUT
    IN : REAL;
    _T : TIME;
    _D : REAL;
    K : REAL := 1.0;
  END_VAR
  VAR_OUTPUT
    OUT : REAL;
  END_VAR
  VAR
    init : BOOL;
    int1 : INTEGRATE;
    int2 : INTEGRATE;
    tn : REAL;
    I1 : REAL;
    I2 : REAL;
    tn2 : REAL;
  END_VAR

  (* startup initialisation *)
  IF NOT init OR _T = T#0s THEN
        init := TRUE;
        out := K * in;
        I2 := out;
  ELSE
        TN := TIME_TO_REAL(_T) / 1000.0;
        tn2 := TN * TN;
        int1(X := in * K / tn2 - I1 * 0.5 * _D / TN - I2 / TN2, Y := I1);
        I1 := int1.Y;
        int2(X := I1,Y := I2);
        I2 := int2.Y;
        out := I2;
  END_IF;

  (* From OSCAT Library, www.oscat.de *)
  (* INTEGRATE required *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK HYST_1
  VAR_INPUT
    IN : REAL;
    HIGH : REAL;
    LOW : REAL;
  END_VAR
  VAR_OUTPUT
    Q : BOOL;
    WIN : BOOL;
  END_VAR

  IF in < low THEN
        Q := FALSE;
        win := FALSE;
  ELSIF in > high THEN
        Q := TRUE;
        win := FALSE;
  ELSE
        win := TRUE;
  END_IF;

  (* From OSCAT Library, www.oscat.de *)
END_FUNCTION_BLOCK

FUNCTION_BLOCK FT_AVG
  VAR_INPUT
    IN : REAL;
    _E : BOOL := TRUE;
    RST : BOOL;
    N : INT := 32;
  END_VAR
  VAR_OUTPUT
    AVG : REAL;
  END_VAR
  VAR
    buff : DELAY;
    i : INT;
```

```
      init : BOOL;
    END_VAR

    buff.N := LIMIT(0, N, 32);

    IF NOT init OR rst THEN
          FOR i := 1 TO N DO
                  buff(in := in);
          END_FOR;
          avg := in;
          init := TRUE;
    ELSIF _E THEN
          buff(in := in);
          avg := avg + (in - buff.out ) / INT_TO_REAL(N);
    END_IF;
    (* from OSCAT library www.oscat.de *)
    (* FB FC delay and inc1 requiered *)
END_FUNCTION_BLOCK


CONFIGURATION config
    VAR_GLOBAL
      seqx0set : REAL := 50.0;
      seqx1set : REAL := 22.0;
      seqx2set : REAL := 28.0;
      seqx3set : REAL := 72.0;
      seqx4set : REAL := 78.0;
      modesupplyorextractcontrolset : REAL := 1.0;
      modeahunormalorreducedset : REAL;
      modeahuonoroffset : REAL;
      tsupplyahu1 : REAL;
      tsupplyahu2 : REAL;
      tminheatingnormalset : REAL := 19.0;
      tminheatingreduceset : REAL := 15.0;
      tmaxcoolingnormalset : REAL := 24.0;
      tmaxcoolingreduceset : REAL := 27.0;
      textractzone1 : REAL;
      textractzone2 : REAL;
      kpcascset : REAL := 0.05;
      tncascset : REAL := 500.0;
      tvcascset : REAL := 0.0;
      tsupplyminwinterset : REAL := 18.0;
      tsupplyminsommerset : REAL := 16.0;
      tsupplymaxset : REAL := 35.0;
      toutdooroffice : REAL;
      tfreshahu1 : REAL;
      tfreshahu2 : REAL;
      tmaxdeltasupplyextractset : REAL := 15.0;
      kpseqlowset : REAL := 0.7;
      tnseqlowset : REAL := 300;
      tvseqlowset : REAL := 0.0;
      kpseqhighset : REAL := 2;
      tnseqhighset : REAL := 100;
      tvseqhighset : REAL := 0.0;
      nfreshminset : REAL := 10.0;
      kpvavset : REAL := 1.5;
      tnvavset : REAL := 100.0;
      tvvavset : REAL := 0.0;
      seqctrlthresholdset : REAL := 1.0;
      seqctrldeadbandset : REAL := 0.5;
      tymixset : REAL := 1.0;
      tyhxset : REAL := 1.0;
      tyhcset : REAL := 1.0;
      tyccset : REAL := 1.0;
      dttoleranceahutovavset : REAL := 1.0;
      textractahu1 : REAL;
      textractahu2 : REAL;
      nvavminzone1set : REAL := 0.0;
      nvavminzone2set : REAL := 0.0;
      dpsupplyfanducttoambientnormalset : REAL := 200.0;
      dpextractfanducttoambientnormalset : REAL := 150.0;
      dpsupplyfanducttoambientreduceset : REAL := 200.0;
      dpextractfanducttoambientreduceset : REAL := 150.0;
      dttoleranceahutovavoffset : REAL := 0.0;
      elload1 : REAL;
      elload2 : REAL;
      pelfreshmaxset : REAL := 700;
      pelvavmaxset : REAL := 1400;
```

```
    useelloadforaq : REAL := 0.0;
    emptymodelconnection : REAL := 0.0;
    ymixahu1 : REAL;
    ymixahu2 : REAL;
    yhxahu1 : REAL;
    yhxahu2 : REAL;
    yhcahu1 : REAL;
    yhcahu2 : REAL;
    yccahu1 : REAL;
    yccahu2 : REAL;
    tmaxzones : REAL;
    tminzones : REAL;
    ypidcascc : REAL;
    ypidcasch : REAL;
    tsupplytarget : REAL;
    seqypidahu1 : REAL;
    seqypidahu2 : REAL;
    seqoutcorrectahu1 : REAL;
    seqoutcorrectahu2 : REAL;
    extractaboveoutdoorrealahu1 : REAL;
    extractaboveoutdoorrealahu2 : REAL;
    yvavzone1 : REAL;
    yvavzone2 : REAL;
    kpseqcalcahu1 : REAL;
    kpseqcalcahu2 : REAL;
    tnseqcalcahu1 : REAL;
    tnseqcalcahu2 : REAL;
    tvseqcalcahu1 : REAL;
    tvseqcalcahu2 : REAL;
    tminheatingselect : REAL;
    tmaxcoolingselect : REAL;
    ylintransh : REAL;
    ylintransc : REAL;
    toutdoorsmooth : REAL;
    dpsupplyfanducttoambientselect : REAL;
    dpextractfanducttoambientselect : REAL;
    modepumphcahu1 : REAL;
    modepumphcahu2 : REAL;
    modeOff : REAL;
    modeNormalOn : REAL;
    modeOn : REAL;
    modeReduceOn : REAL;
    modeCcOn : REAL;
    modeHcOn : REAL;
    modeHcAndCcOff : REAL;
    modeHxOff : REAL;
    modeHxOn : REAL;
    modeFreshAirAirQualityOff : REAL;
    modeFreshAirTemperatureOff : REAL;
    modeFreshAirOn : REAL;
    modeFreshAirAirQualityOn : REAL;
    modeFreshAirCoolingOn : REAL;
    modeFreshAirHeatingOn : REAL;
    modeRecirculationOnlyOn : REAL;
    modeVariableAirFlowsZone1Off : REAL;
    modeVariableAirFlowsZone2Off : REAL;
    modeVariableAirFlowsZone1AirQualityOff : REAL;
    modeVariableAirFlowsZone1TemperatureOff : REAL;
    modeVariableAirFlowsZone2AirQualityOff : REAL;
    modeVariableAirFlowsZone2TemperatureOff : REAL;
    modeVariableAirFlowsZone1On : REAL;
    modeVariableAirFlowsZone2On : REAL;
    modeVariableAirFlowsZone1AirQualityOn : REAL;
    modeVariableAirFlowsZone1CoolingOn : REAL;
    modeVariableAirFlowsZone1HeatingOn : REAL;
    modeVariableAirFlowsZone2AirQualityOn : REAL;
    modeVariableAirFlowsZone2CoolingOn : REAL;
    modeVariableAirFlowsZone2HeatingOn : REAL;
    elloadperc : REAL;
    pelsum : REAL;
    nfreshminelload : REAL;
    nvavminelload : REAL;
    nfreshmincalc : REAL;
    nvavminzone1calc : REAL;
    nvavminzone2calc : REAL;
    ypidvavhz1 : REAL;
    ypidvavhz2 : REAL;
    ypidvavcz1 : REAL;
    ypidvavcz2 : REAL;
```

```
    tsupplytargeteqtfresh : REAL;
    textractmean : REAL;
    tsupplymean : REAL;
  END_VAR

  RESOURCE resource1 ON PLC
    TASK task0(INTERVAL := T#100ms,PRIORITY := 0);
    PROGRAM instance0 WITH task0 : RLT_6_11;
  END_RESOURCE
END_CONFIGURATION
```