EBC

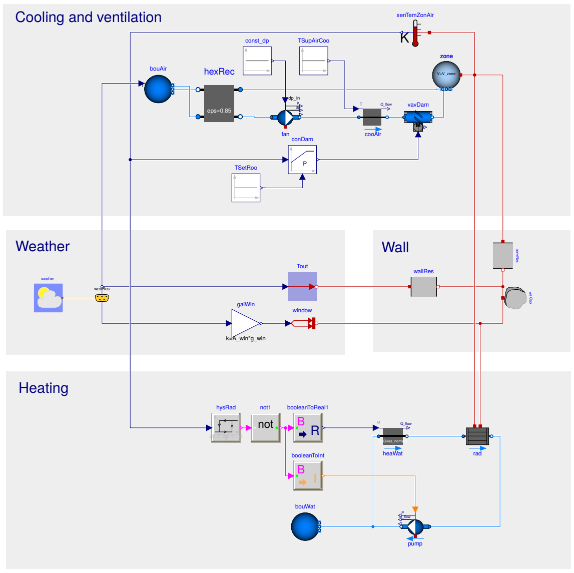
Energy in Buildings and
Communities Programme

International Energy Agency

New Generation Computational Tools for Building & Community Energy Systems

Annex 60 Final Report

September 2017



Michael Wetter and Christoph van Treeck

ISBN 978-0-692-89748-5

EBC is a programme of the International Energy Agency (IEA)

iea Energy Technology Network

EBC

Energy in Buildings and
Communities Programme

International Energy Agency

New Generation Computational Tools for Building & Community Energy Systems

Annex 60 Final Report

September 2017

Edited by

Michael Wetter
Building Technology and Urban Systems Department
Energy Technologies Area
Lawrence Berkeley National Laboratory, USA

Christoph van Treeck
Chair in Energy Efficiency and Sustainable Building (E3D)
RWTH Aachen University, Germany

iea Energy Technology Network

Published by The Regents of the University of California (through Lawrence Berkeley National Laboratory) and RWTH Aachen University, Germany.

ISBN: 978-0-692-89748-5

Participating countries in EBC: Australia, Austria, Belgium, Canada, P.R. China, Czech Republic, Denmark, France, Germany, Ireland, Italy, Japan, Republic of Korea, the Netherlands, New Zealand, Norway, Portugal, Spain, Sweden, Switzerland, United Kingdom and the United States of America.

Additional copies of this report may be obtained from:

EBC Bookshop
C/o AECOM Ltd
The Colmore Building
Colmore Circus Queensway
Birmingham B4 6AT
United Kingdom
Web: www.iea-ebc.org
Email: essu@iea-ebc.org

Contents

Chapter 1

Preface

1.1 The International Energy Agency

The International Energy Agency (IEA) was established in 1974 within the framework of the Organisation for Economic Co-operation and Development (OECD) to implement an international energy programme. A basic aim of the IEA is to foster international co-operation among the 29 IEA participating countries and to increase energy security through energy research, development and demonstration in the fields of technologies for energy efficiency and renewable energy sources.

1.2 The IEA Energy in Buildings and Communities Programme

The IEA co-ordinates international energy research and development (R&D) activities through a comprehensive portfolio of Technology Collaboration Programmes. The mission of the Energy in Buildings and Communities (EBC) Programme is to develop and facilitate the integration of technologies and processes for energy efficiency and conservation into healthy, low emission, and sustainable buildings and communities, through innovation and research. (Until March 2013, the IEA-EBC Programme was known as the Energy in Buildings and Community Systems Programme, ECBCS.)

The research and development strategies of the IEA-EBC Programme are derived from research drivers, national programmes within IEA countries, and the IEA Future Buildings Forum Think Tank Workshops. The research and development (R&D) strategies of IEA-EBC aim to exploit technological opportunities to save energy in the buildings sector, and to remove technical obstacles to market penetration of new energy efficient technologies. The R&D strategies apply to residential, commercial, office buildings and community systems, and will impact the building industry in five focus areas for R&D activities:

- Integrated planning and building design
- Building energy systems
- Building envelope
- Community scale methods
- Real building energy use

1.3 The Executive Committee

Overall control of the IEA-EBC Programme is maintained by an Executive Committee, which not only monitors existing projects, but also identifies new strategic areas in which collaborative efforts may be beneficial. As the Programme is based on a contract with the IEA, the projects are legally established as Annexes to the IEA-EBC Implementing Agreement. At the present time, the following projects have been initiated by the IEA-EBC Executive Committee, with completed projects identified by (*):

Annex 1	Load Energy Determination of Buildings (*)
Annex 2	Ekistics and Advanced Community Energy Systems (*)
Annex 3	Energy Conservation in Residential Buildings (*)
Annex 4	Glasgow Commercial Building Monitoring (*)
Annex 5	Air Infiltration and Ventilation Centre
Annex 6	Energy Systems and Design of Communities (*)

Continued on next page

Table 1.1 – continued from previous page

Annex 7	Local Government Energy Planning (*)
Annex 8	Inhabitants Behaviour with Regard to Ventilation (*)
Annex 9	Minimum Ventilation Rates (*)
Annex 10	Building HVAC System Simulation (*)
Annex 11	Energy Auditing (*)
Annex 12	Windows and Fenestration (*)
Annex 13	Energy Management in Hospitals (*)
Annex 14	Condensation and Energy (*)
Annex 15	Energy Efficiency in Schools (*)
Annex 16	BEMS 1- User Interfaces and System Integration (*)
Annex 17	BEMS 2- Evaluation and Emulation Techniques (*)
Annex 18	Demand Controlled Ventilation Systems (*)
Annex 19	Low Slope Roof Systems (*)
Annex 20	Air Flow Patterns within Buildings (*)
Annex 21	Thermal Modelling (*)
Annex 22	Energy Efficient Communities (*)
Annex 23	Multi Zone Air Flow Modelling (COMIS) (*)
Annex 24	Heat, Air and Moisture Transfer in Envelopes (*)
Annex 25	Real time HVAC Simulation (*)
Annex 26	Energy Efficient Ventilation of Large Enclosures (*)
Annex 27	Evaluation and Demonstration of Domestic Ventilation Systems (*)
Annex 28	Low Energy Cooling Systems (*)
Annex 29	Daylight in Buildings (*)
Annex 30	Bringing Simulation to Application (*)
Annex 31	Energy-Related Environmental Impact of Buildings (*)
Annex 32	Integral Building Envelope Performance Assessment (*)
Annex 33	Advanced Local Energy Planning (*)
Annex 34	Computer-Aided Evaluation of HVAC System Performance (*)
Annex 35	Design of Energy Efficient Hybrid Ventilation (HYBVENT) (*)

Continued on next page

Table 1.1 – continued from previous page

Annex 36	Retrofitting of Educational Buildings (*)
Annex 37	Low Exergy Systems for Heating and Cooling of Buildings (LowEx) (*)
Annex 38	Solar Sustainable Housing (*)
Annex 39	High Performance Insulation Systems (*)
Annex 40	Building Commissioning to Improve Energy Performance (*)
Annex 41	Whole Building Heat, Air and Moisture Response (MOIST-ENG) (*)
Annex 42	The Simulation of Building-Integrated Fuel Cell and Other Cogeneration Systems (FC+COGEN-SIM) (*)
Annex 43	Testing and Validation of Building Energy Simulation Tools (*)
Annex 44	Integrating Environmentally Responsive Elements in Buildings (*)
Annex 45	Energy Efficient Electric Lighting for Buildings (*)
Annex 46	Holistic Assessment Tool-kit on Energy Efficient Retrofit Measures for Government Buildings (EnERGo) (*)
Annex 47	Cost-Effective Commissioning for Existing and Low Energy Buildings (*)
Annex 48	Heat Pumping and Reversible Air Conditioning (*)
Annex 49	Low Exergy Systems for High Performance Buildings and Communities (*)
Annex 50	Prefabricated Systems for Low Energy Renovation of Residential Buildings (*)
Annex 51	Energy Efficient Communities (*)
Annex 52	Towards Net Zero Energy Solar Buildings (*)
Annex 53	Total Energy Use in Buildings: Analysis & Evaluation Methods (*)
Annex 54	Integration of Micro-Generation & Related Energy Technologies in Buildings (*)

Continued on next page

Table 1.1 – continued from previous page

Annex 55	Reliability of Energy Efficient Building Retrofitting - Probability Assessment of Performance & Cost (RAP-RETRO) (*)
Annex 56	Cost Effective Energy & CO2 Emissions Optimization in Building Renovation
Annex 57	Evaluation of Embodied Energy & CO2 Equivalent Emissions for Building Construction
Annex 58	Reliable Building Energy Performance Characterisation Based on Full Scale Dynamic Measurements (*)
Annex 59	High Temperature Cooling & Low Temperature Heating in Buildings
Annex 60	New Generation Computational Tools for Building & Community Energy Systems
Annex 61	Business and Technical Concepts for Deep Energy Retrofit of Public Buildings
Annex 62	Ventilative Cooling
Annex 63	Implementation of Energy Strategies in Communities
Annex 64	LowEx Communities - Optimised Performance of Energy Supply Systems with Exergy Principles
Annex 65	Long-Term Performance of Super-Insulating Materials in Building Components and Systems
Annex 66	Definition and Simulation of Occupant Behavior in Buildings
Annex 67	Energy Flexible Buildings
Annex 68	Indoor Air Quality Design and Control in Low Energy Residential Buildings
Annex 69	Strategy and Practice of Adaptive Thermal Comfort in Low Energy Buildings
Annex 70	Energy Epidemiology: Analysis of Real Building Energy Use at Scale
Annex 71	Building Energy Performance Assessment Based on In-situ Measurements
Working Group	Energy Efficiency in Educational Buildings (*)

Continued on next page

Table 1.1 – continued from previous page

Working Group	Indicators of Energy Efficiency in Cold Climate Buildings (*)
Working Group	Annex 36 Extension: The Energy Concept Adviser (*)
Working Group	Survey on HVAC Energy Calculation Methodologies for Non-residential Buildings

Chapter 2

Foreword

This publication is the official report of IEA EBC Annex 60, which was conducted from 2012 to 2017 through a collaboration among 42 institutes from 16 countries. Annex 60 developed and demonstrated new generation computational tools for the design and operation of building and community energy systems. The report is aimed at users of simulation, HVAC and urban energy system designers as well as researchers in the field of energy systems for the built environment.

A key driver for this work are the trends towards zero energy and electrification of the energy infrastructure that demands that buildings and district energy systems become increasingly integrated to reduce energy use, power density and to shift load. Typical measures include high-performance facades, energy storage, waste heat utilization within and among buildings through near ambient-temperature networks, and heat pumps that boost waste heat and renewable sources to usable temperatures. Advanced controls need to orchestrate this operation while providing electrical load shifting and load shedding capabilities, and bidding these capabilities into a dynamic electricity market. What are the implications on building simulation and associated computing and the digitalization of the entire planning process including BIM tools?

Clearly, building simulation programs face new challenges to support such systems throughout the building life cycle. They must become a modular service that integrates seamlessly with other tools, sometimes at small time-steps and below the level of a whole building, during design and operation. This represents a radical structural shift from conventional building simulation programs, which provide little workflow automation for design, analysis and optimization and no facilities for runtime integration. This situation leads to new functional requirements which are not addressed by existing building simulation programs, which are often load-based, assume ideal and non-integrated steady-state control of each individual subsystem, and are hard to extend from design to operation, and from buildings to districts.

In the meantime, other engineering sectors have been making large investments and substantial progress in next generation computing tools for the design and operation of complex, dynamic, engineered systems based on the open standards Modelica, a modeling language, and Functional Mockup Interface (FMI), a standard for exchanging models.

Annex 60 transfers and adapts these technologies to the buildings industries through the collaborative development of Modelica libraries, FMI-technologies, and translators from Building Information Models (BIM) to Modelica.

Due to this large ecosystem of technology that can be adapted for the buildings industry, due to the evolving requirements that demand new approaches for modeling, simulation and optimization that are a shift from todays practice in building performance simulation, and to have a means to collaboratively develop software, all work in Annex 60 was based on the following three open standards:

- The equation-based, object-oriented Modelica language which allows graphical composition of models for simulation, operation and optimization. The models may be multi-physics system models, such as energy systems that couple thermodynamics, heat transfer, fluid dynamics, and electrical systems. These physics-based models can be combined with data-driven models and with models of continuous-time, discrete-time, and event-driven feedback control systems.
- The Functional Mockup Interface (FMI) Standard, a specification that describes how to encapsulate and exchange models or simulators, independent of the authoring tool or application domain. The FMI standard is currently supported by more than 80 tools.
- The Industry Foundation Classes (IFC), the only life cycle model of build-

ings that is an open international standard, governed by ISO 16739. BIM models described by IFC may be HVAC components or entire building energy systems. A building services-related BIM model originating from the digital planning process cannot serve as basis for simulation without further knowledge-based transformation.

Using these standards, the core research problem that has been solved within Annex 60 was the coordinated development, application and demonstration of new generation computational tools for building and community energy systems that are based on open standards and that allow buildings and energy grids to be designed and operated as integrated, robust, and performance based systems.

In hindsight, embracing these standards was instrumental for collaborative research and development, as there was a clear specification of the technology, formalized through standards, that served as the basis of the collaborative development. Probably most important, working through these standards allows the building simulation community to collaborate with experts from other fields, such as experts in multi-physics modeling, hybrid systems, numerical methods, computer algebra, compiler technology or language design, that are important for the new requirements that the building simulation community faces, but that are generally not present in our community.

Software development is an investment in foundational tools that encapsulates sophisticated, complex methods to make them accessible to non-experts through easy-to-use interfaces. By committing to standards rather than a particular tool provider's implementation, the software developed in Annex 60 that does not rely on, and hence locks customers into the use of tools provided by any single vendor.

2.1 Operating Agents and Task Leaders

The operating agents were

Michael Wetter
Building Technology and Urban Systems Department
Energy Technologies Area
Lawrence Berkeley National Laboratory, USA

and

Christoph van Treeck
Chair in Energy Efficiency and Sustainable Building (E3D)
RWTH Aachen University, Germany

Annex 60 was structured into the following subtasks and activities.

Subtask 1: Technology Development

Led by Michael Wetter, LBNL, Berkeley, CA

Activity 1.1: Modelica model libraries

Led by Michael Wetter, LBNL, Berkeley, CA

Activity 1.2: Co-simulation and model exchange through Functional Mockup Units

Led by Frederic Wurtz, Grenoble University, Grenoble, France

Activity 1.3: Building Information Model

Led by Christoph van Treeck, RWTH Aachen University, Germany

Activity 1.4: Workflow automation tools

Led by Sebastian Stratbuecker, Fraunhofer IBP, Holzkirchen, Germany

Subtask 2: Validation and Demonstration

Led by Lieve Helsen, KU Leuven, Leuven, Belgium

Activity 2.1: Design of building systems

Led by Christoph Nytsch-Geusen, Berlin University of the Arts, Berlin, Germany

Activity 2.2: Design of district energy systems

Led by Dirk Saelens, KU Leuven, Leuven, Belgium

Activity 2.3: Model use during operation

Led by Ignacio Torrens, Eindhoven University of Technology, The Netherlands

Subtask 3: Dissemination

Led by Christoph van Treeck and Michael Wetter

2.2 Authors of the Final Report

The final report was co-authored by the participants listed below, and edited by the operating agents Michael Wetter and Christoph van Treeck.

Name	Affiliation
Baetens, Ruben	KU Leuven, Belgium
Bazjanac, Vladimir	Standford University, CA, USA
Blum, David	Lawrence Berkeley National Laboratory, Berkeley, CA, USA
Cao, Jun	RWTH Aachen University, Aachen, Germany
Fuchs, Marcus	RWTH Aachen University, Aachen, Germany
Jorissen, Filip	KU Leuven, Leuven, Belgium
Keane, Marcus M.	National University of Ireland, Galway, Ireland
Lauster, Moritz	RWTH Aachen University, Aachen, Germany
Maile, Tobias	Maile Consulting, Fellbach, Germany
Mitterhofer, Matthias	Fraunhofer Institute for Building Physics IBP, Holzkirchen, Germany
Nouidui, Thierry S.	Lawrence Berkeley National Laboratory, Berkeley, CA, USA
Nytsch-Geusen, Christoph	Berlin University of the Arts, Berlin, Germany
O'Donnel, James	University College Dublin, Dublin
Picard, Damien	KU Leuven, Leuven, Belgium
Pinheiro, Sergio	University College Dublin, Dublin
Protopapadaki, Christina	KU Leuven/EnergyVille, Belgium
Reinbold, Vincent	KU Leuven/EnergyVille, Belgium
Saelens, Dirk	KU Leuven/EnergyVille, Belgium
Sterling, Raymond	National University of Ireland, Galway, Ireland
Stratbuecker, Sebastian	Fraunhofer Institute for Building Physics IBP, Holzkirchen, Germany
Thorade, Matthis	Berlin University of the Arts, Berlin, Germany
Tugores, Carles Ribas	Berlin University of the Arts, Berlin, Germany
van der Heijde, Bram	KU Leuven/EnergyVille, Belgium
van Treeck, Christoph	RWTH Aachen University, Aachen, Germany
Wetter, Michael	Lawrence Berkeley National Laboratory, Berkeley, CA, USA
Wimmer, Reinhard	RWTH Aachen University, Aachen, Germany

2.3 Project Participants

The following 42 institutes from 16 countries participate in Annex 60:

Institute	Country
Lawrence Berkeley National Laboratory	USA
Massachusetts Institute of Technology	USA
Purdue University	USA
Stanford University	USA
Texas A&M University	USA
UCI Engineering, Inc.	USA
University of Alabama	USA
University of Miami	USA
University of Texas, San Antonio	USA
RWTH Aachen University	Germany
Maile Consulting	Germany
TU Dresden	Germany
KIT Karlsruher Institut of Technology	Germany
Fraunhofer ISE	Germany
Berlin University of the Arts (UDK)	Germany
Fraunhofer IBP	Germany
AEC3	Germany
Austrian Institute of Technology	Austria
AEE-Intec	Austria
KU Leuven	Belgium
Cenaero	Belgium
University of Liège	Belgium
Pontifícia Universidade Católica do Paraná	Brazil
Chongqing University	China
Aalborg University	Denmark
University of Southern Denmark	Denmark
LGCgE, Université d'Artois	France
Grenoble University	France
CEA INES	France

Continued on next page

Table 2.1 – continued from previous page

Institute	Country
I2M University of Bordeaux	France
CSTB	France
EDF	France
National University of Ireland, Galway	Ireland
University College Dublin	Ireland
Università Politecnica delle Marche	Italy
Eindhoven University of Technology	The Netherlands
Exergy Studios	Slovakia
IK4-TEKNIKER	Spain
Swegon AB	Sweden
EQUA Simulation AB	Sweden
EMPA	Switzerland
Masdar Institute	United Arab Emirates

2.4 Acknowledgements

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.

This research was supported by the European Union through the Seventh Framework Programme (FP7/2007-2013) under grant agreement nº 285408.

This research was supported by the European Union through the Seventh Framework Programme (FP7/2007-2013) under grant agreement nº 284920.

This research was supported by the International Energy Research Centre and Enterprise Ireland under project n. CC-2011-4005B and by the Irish Research Council - D'Appolonia enterprise partnership scheme.

This research was supported by the German Federal Ministry of Economic Affairs and Energy (BMWi), national research project EnTool:EnEff-BIM (EnOB), promotional references 03ET1177A, 03ET1177B, 03ET1177C, 03ET1177D, 03ET1177E.

This research was supported by the German Federal Ministry of Economic Affairs and Energy (BMWi), national research project EnTool:CoSim (EnOB), promotional references 03ET1215A, 03ET1215B, 03ET1215C, 03ET1215D.

This research was supported by a Marie Curie FP7 Integration Grant within the 7th European Union Framework Programme project title SuPerB, project nº 631617 and Conselho Nacional de Desenvolvimento Científico and Tecnológico (CNPq) under the Programme Science without Borders in Brazil.

Research participants collaboratively worked together under the umbrella of the IEA EBC framework. Duplications of work were avoided and participants benefitted from collaboration and synergies which became possible due to the strong organizational Annex 60 framework.

Chapter 3

Executive Summary

The IEA EBC project *Annex 60: New Generation Computational Tools for Building & Community Energy Systems* led to open-source, freely available, documented, validated and verified new generation computational tools. These tools allow buildings and community energy grids to be designed and operated as integrated, robust, performance based systems with low energy use and low peak power demand. The developed tools are all based on three non-proprietary, open standards:

- The Modelica modeling language for implemeting models (https://www.modelica.org/),
- the Functional Mockup Interface (FMI) standards to couple simulators (https://www.fmi-standard.org/), and
- the Industry Foundation Classes (IFC) for building information modeling (http://www.buildingsmart-tech.org/) as well as other BIM-related standards such as Information Delivery Manual (IDM) and Model View Definitions (MVD).

Thus, Annex 60 committed to, leveraged and contributed to open standards that can be used with a variety of tools, rather than developed software technology that depends on the implementation of a single tool provider. This avoids vendor lock-in and provides to industry a stable basis, governed by standards, to invest in.

The target audience of Annex 60 is the building energy research community, design firms and energy service companies, equipment and tool manufacturers, as well as students in building energy-related sciences. Through Annex 60, fragmented duplicative activities in modeling, simulation and optimization of building and community energy systems that are based on the Modelica and FMI standards were coordinated. Tool-chains were created, often by adapting and extending technologies from other industry sectors, to link *Building Information Models* (BIM) to energy modeling, building simulation to controls design tools, and design tools to operational tools. These tools were demonstrated for building design, district energy system design, and for use of models during operation to support fault detection and diagnostics algorithms, model predictive control, and hardware-in-the-loop experimentation.

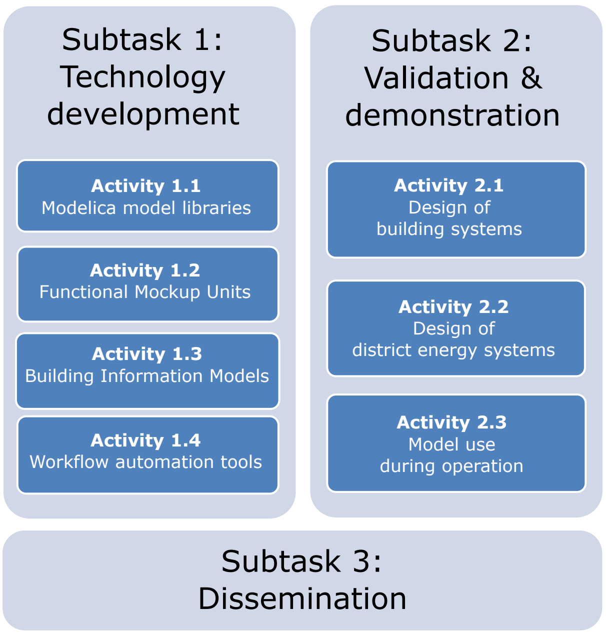

Subtask 1: Technology development
Activity 1.1 Modelica model libraries
Activity 1.2 Functional Mockup Units
Activity 1.3 Building Information Models
Activity 1.4 Workflow automation tools
Subtask 2: Validation & demonstration
Activity 2.1 Design of building systems
Activity 2.2 Design of district energy systems
Activity 2.3 Model use during operation
Subtask 3: Dissemination

Fig. 3.1: Structure and organization of Annex 60.

Annex 60 was organized in the subtasks and activities shown in Fig. 3.1. Subtasks 1 developed and implemented the software technology required by the applications. Subtask 2 was focused on validation, verification and demonstration of the developed software technology for building and community energy system design and operation. Subtask 3 was focused on dissemination of the results through special tracks at professional conferences, through training workshops and through publications in international scientific journals.

Subtasks 1 and 2 consisted of the following activities:

Subtask 1 - Technology Development

Activity 1.1 Modelica model libraries, developed a free open-source library with more than 300 Modelica models for building and community energy systems, available at https://github.com/iea-annex60/modelica-annex60/releases. This library became the core of the four Modelica libraries `AixLib`, developed by RWTH Aachen, Germany, `BuildingsSystems`, developed by UdK Berlin, Germany, `Buildings`, developed by Lawrence Berkeley National Laboratory, Berkeley, CA, USA, and `IDEAS`, developed by KU Leuven, Belgium. Prior to the Annex, theses libraries had limited scope, were mutually incompatible, and in some cases not available to the public.

Activity 1.2 Co-simulation and model exchange through Functional Mockup Units, developed *co-simulation* and model-exchange interfaces in legacy building energy simulation programs and further developed middle-ware for *co-simulation* and model exchange. All work was based on the non-proprietary *Functional Mockup Interface* standard.

Activity 1.3 Building Information Models, developed BIM to Modelica translators. This was accomplished through the use and extension of the Open BIM data formats defined by the Industry Foundation Classes (IFC) and through the use of other BIM-standards such as the Information Delivery Manual (IDM) and through Model View Definitions (MVD).

Activity 1.4 Workflow automation tools, developed free open-source Python packages to automate the workflow of developing and using Modelica models.

Subtask 2 - Validation and Demonstration

Activity 2.1 Design of building systems, demonstrated how to design energy and control systems for buildings and how to size systems under consideration of diurnal weather patterns, energy storage and time-varying electricity prices of a smart grid.

Activity 2.2 Design of district energy systems, validated and demonstrated the tools from Subtask 1, applied to district energy systems and smart grid integration at the scale of the district energy system.

Activity 2.3 Model use during operation, used control models from Activity 1.1 and FMI export programs from Activity 1.2 during the operation of building energy systems, and during hardware-in-the-loop experimentation.

Subtask 3 - Dissemination

Subtask 3, which is not described in this report, focused on the dissemination of the results through special sessions at scientific conferences and through workshops that trained users in the technology developed in Annex 60. Results have further been disseminated through publication in international scientific journals.

Annex 60 was conducted from June 2012 to June 2017. The core of the team will continue key developments and disseminations of Annex 60 under the umbrella of the International Building Performance Simulation Association (IBPSA). This will be the first research project formally conducted under the umbrella of IBPSA, executed as “Project 1: BIM/GIS and Modelica Framework for building and community energy system design and operation.”

Chapter 4

Introduction

To meet increasingly stringent energy performance targets and challenges posed by distributed renewable energy generation on the electrical and thermal distribution grid, recent attention has been given to system-level integration, part-load operation and operational optimization of buildings. The intent is to design and operate a building or a neighborhood optimally as a performance-based, robust system. This requires taking into account system-level interactions between building storage, HVAC systems and electrical and thermal grid. Such a system-level analysis requires multi-physics simulation and optimization using coupled thermal, electrical and control models. Optimal operation also requires closing the gap between designed and actual performance through commissioning, energy monitoring and fault detection and diagnostics. All of these activities can benefit from using models that represent the design intent. These models can then be used to verify responses of installed equipment and control sequences, and to compute optimal control sequences in a Model Predictive Controller (MPC), the latter of which possibly requiring simplified models.

Furthermore, in the AEC domain the processes of designing, constructing and commissioning buildings and engergy systems are rapidly changing toward digitalization. *Building Information Modeling (BIM)* is an enabler as collaborative method and tool to consistently gather, manage and exchange building-related data on a digital basis over the entire life cycle of a facility. BIM is not a specific software, it is rather a method as part of, but not limited to, the integral design. A truly added value is expected for the near future when design and commissioning in the sense of computer aided facility management comes together. The above mentioned issues of commissioning, energy monitoring and fault detection and diagnostics can therefore highly benefit from a thorough digital planning when location and function of technical systems are together referenced in a digital model, when the as-built state is harmonized with and well documented in a model and when home and building automation becomes integrally linked with BIM.

This shift in focus will require an increased use of models throughout the building delivery stages and continuing into the operational phase. Consider, for example, the development and use of an HVAC system model:

1. During design, a mechanical engineer will construct a model that represents the design intent, such as system layout, equipment selection, and control sequences. The basis of such a model could be from a BIM in the case of a building, or from a GIS in the case of a district.
2. During construction, to reduce cost for implementation of the control sequence, and to ensure that the control intent is properly implemented, a control model could be used to generate code that can be uploaded to supervisory building automation systems, thereby executing the same sequence as was used during design *[NW14]*.
3. During commissioning, the design model will be used to verify proper installation.
4. During operation, the model will be used for comparing actual with expected energy use *[PWBH11]*, and for fault detection and diagnostics *[BSG+14]*. Furthermore, the model may be converted to a form that allows its use during operation as part of an MPC algorithm.

In addition to the focus on closing the performance gap between design and operation, another recent focus is on system integration. Here, the challenge lies in the co-design and operation of building dynamics, HVAC, thermal and electrical storage, renewable energy generation, and grid responsive control in order to maintain the power quality of the electrical grid. Commonly, to support system integration, models from different engineering domains need to be coupled during run-time. For example, for active facade control, it may be necessary to couple a ray-tracing tool such as Radiance with a building energy simulation tool to asses the impact of daylighting controls on reducing glare, energy, and peak cooling demand. Similarly, for building to electrical grid integration, building HVAC and domestic hot water control can be designed such that buildings present themselves as a flexible load to the electrical grid, which can increase the amount of renewable energy integrated into the grid. Such coupling of domain-specific models may be done within Modelica, an equation-based, object-oriented modeling language, or through tool coupling that involves co-simulation, a technique in which simulators exchange data as the simulation-time advances. See *[Wet11a][BDCVR+12][BWN14][WBN16][CH15]* for example applications.

For a larger discussion of functionalities that future building modeling tools will need to provide to address the needs for low energy building and community energy grid design and operation, we refer to *[Wet11b]* and *[Cla15]*.

The aforementioned new foci give rise to new requirements for building simulation tools, including the following:

1. Mechanical engineers should be able to design, assess the performance and verify the correctness of local and, in particular, supervisory control sequences in simulation. They should then use such a verified, non-ambiguous specification to communicate their design intent to the control provider. Moreover, the specification should be used during commissioning to verify that the control contractor implemented the design intent.
2. Controls engineers should be able to extract subsystem models from models used during the building design in order to use them within building control systems for commissioning, model-based controls, fault detection and diagnostics.
3. Urban planners and researchers should be able to combine models of buildings, electrical grids and controls in order to improve the design and operation of such systems to ensure low greenhouse gas emissions or costs, and high quality power delivery *[BDCVR+12][WBN16][BWN14]*.
4. Mechanical engineers should be able to convert design models to a form that allows the efficient and robust solution of optimal control problems as part of MPC *[SOCP11]*. Such models may then be combined with state estimation techniques that adapt the model to the actual building *[BSG+14]*.

The first item requires modeling and simulation of actual control sequences, including proper handling of hybrid systems, i.e., systems in which the state evolves in time based on continuous time semantics that arises from physics, and discrete time and discrete event semantics that arises from digital control *[Wet09][WZNP14]*. This poses computing challenges for the deterministic synchronization of these domains *[BGL+15]*. The second item requires extraction of a subsystem model and exporting this model in a self-contained form that can readily be executed as part of a building automation system as shown in *[NW14]*. The third item requires models of different physical domains and models of control systems to be combined for a dynamic, multi-physics simulation that involves electrical systems, thermal systems, controls and possibly communication systems, which may evolve at vastly different time scales. The fourth item greatly benefits if model equations are accessible to perform model order reduction and to solve optimal control problems.

4.1 Comparison to State-of-the-art in Building Energy Modeling and Simulation

Today's whole-building simulation programs formulate models using imperative programming languages. Imperative programming languages assign values to functions, declare the sequence of execution of these functions and change the state of the program, as is done for example in C/C++, Fortran or MATLAB/Simulink. In such programs, model equations are tightly intertwined with numerical solution methods, often by making the numerical solution procedure part of the actual model equations. This approach has its origin in the seventies when neither modular software approaches were implemented nor powerful computer algebra tools were available. These programs have been developed for the use case of building energy performance assessment to support building design and energy policy development. Other use cases such as control design and verification, model use in support of operation, and multi-physics dynamic analysis that combines building, HVAC, electrical and control models were not priorities, nor even considered *[CLW+96]*. However, the position paper of IBPSA shows that they recently gained importance *[Cla15]*.

Tight coupling of numerical solution methods with model equations and input/output routines makes it difficult to extend these programs to support new use cases. The reason is that this coupling imposes rules that determine for example where inputs to functions that compute HVAC, building or control equipment are received from the internal data structure of the program, when these inputs are updated, when these functions are evaluated to produce new output, and what output values may be lagged in time to avoid algebraic loops. Such rules have made it increasingly difficult for developers to add new functionalities to software without inadvertently introducing an error in other parts of the program. They also make it difficult for users to understand how component models interact with other parts of the system model, in particular their interaction with, and assumptions of, control sequences. Furthermore, they also have shown to make it difficult to use such tools for optimization *[WW04]*.

The tight coupling of numerical solution methods with model equations also makes it difficult to efficiently simulate models for the various use cases. Numerical methods in today's building energy simulation programs are tailored to the use case of energy analysis during design. However, other use cases, such as controls design and verification, coupled modeling of thermal and electrical systems, and model use during operation require different numerical methods. To see why different numerical methods are required, consider these applications:

- Stiff systems: The simulation of feedback control with time constants of seconds coupled to building energy models with time constants of hours leads to stiff ordinary differential equations. Their efficient numerical solution requires implicit solvers *[HW96]*.
- Non-stiff systems: In EnergyPlus *[CLP+99]* and in many TRNSYS *[KDB76]* component models, the dynamics of HVAC equipment and controllers, which is fast compared to the dynamics of the building heat transfer, is generally approximated using steady-state models. Hence, the resulting system model is not stiff as the only dynamics is from the building model. In this situation, explicit time integration algorithms are generally more efficient. Such an approximation of the fast dynamics can also be done with dynamic Modelica HVAC models, see *[JWH15]*.
- Hybrid systems: Hybrid systems require proper simulation of coupled continuous time, discrete time and discrete event dynamics. This in turn requires solution methods with variable time steps and event handling.

> For example, when a temperature sensor crosses a setpoint or a battery reaches its state of charge, a state event takes place that may switch a controller, necessitating solving for the time instant when the switch happens and reducing accordingly the integration time step. Standard ordinary differential equation solvers require an iteration in time to solve for the time instant of the event, and reinitializing integrators after the event, which both are computationally expensive. A new class of ordinary differential equation solvers called Quantized State System (QSS) integration *[ZL98][KJ01][CK06][Kof03][MBKC13]* are promising for the efficient simulation of such systems as they do not require iteration for state event detection. However, their efficient use requires knowledge of the dependency graph of the state equations, which generally is not available in legacy building simulators, but readily available in equation-based languages.

It follows from this discussion that for models to be applicable to a wide range of applications, it should be possible to use them with different numerical solvers. Therefore, models for building energy systems and their numerical solution methods should be separated where possible. Exceptions are equations for which special tailored solution methods and parallel programming patterns allow humans to better exploit the structure of the equations than is currently supported by code generators, often arising from partial differential equations or from light distributions. Examples include solvers for computational fluid dynamics, heat transfer in borehole heat exchangers *[PH14]*, and ray-tracing for daylighting. Work, however, is ongoing to remedy this situation *[Cas15][SWF+15][BBCK15]*.

4.2 New Technologies for Building Energy Modeling and Simulation

This section describes new technologies which can be applied to building energy modeling in support of the different use cases.

4.2.1 Equation-based Modeling

As explained above, the use of imperative programming languages limits the applicability and extensibility of models. Furthermore, in building simulation programs, numerical solution algorithms are often tightly integrated into the models and thereby can mandate the use of supervisory control logic that is far removed from how control sequences are implemented in reality. For example, in EnergyPlus, a cooling coil may request from the supervisory control a certain air mass flow rate in order to meet the load computed in the predictor step of the thermal zone heat balance. In actuality, the air mass flow rate would be determined by the position of dampers in combination with the speed of a supply fan, each of which could be controlled by zone temperature and duct static pressure feedback controllers.

A key difference between imperative programming languages and equation-based languages is that the latter do not require a specification of the sequence of computer assignments required to simulate a model. Rather, a model developer can specify the mathematical equations, package them into graphically represented components and store them in a hierarchical library. A model user then assembles these components in a schematic editor to form a system model. A simulation environment analyses these equations, optimally rearranges them using computer algebra, translates them to executable code, typically C, and links them with numerical solvers.

Specifically, the translation of equations to executable code involves determining which variables can be replaced by so-called alias variables, for example, in the case of a mass flow rate that may be the same for all components that are used to compose an air handler unit. It also involves reducing the dimension of coupled linear and nonlinear system of equations through symbolic inverting equations and through Block Lower Triangularization and Tearing *[CK06][EO94]*, which often significantly reduces the dimension of the coupled systems of equations. See also Section 5.3.4. Furthermore, during translation, *zero-crossing functions* are generated, for example, to indicate when a thermostat crosses a set-point, and high-order differential algebraic systems of equations are reduced to index 1 *[MS93]*: Some Modelica translators also generate code for specific solvers. The benefit of this has been demonstrated by Fernandez and Kofman who showed for QSS methods more than an order of magnitude simulation speed improvements when code is generated in a form that is specifically designed for the QSS methods [FK14], as opposes to using QSS methods with a conventional discrete event simulation solver. Symbolic manipulations also allow to partition the model automatically for parallel computing [EMO14].

Loosely speaking, while simulation models implemented using imperative programming languages require numerical solvers to select numerical inputs and compare the function values for these inputs to infer what equations they solve, equation-based modeling languages such as Modelica allow for the understand of equation structure, and making use of this understanding to generate efficient code for computation. Examples of structures include which variables are connected to each other through algebraic constraints or differential equations, which equations can be differentiated, which equations can be inverted, and which equations trigger an event that can instantly change a control signal. For a more detailed discussion, see [Elm78], [CK06], [EO94] and [EOC95]. To make these technologies accessible to a wide range of users in building simulation, research and development is required and ongoing to advance translators and solvers to better handle large models [Wet09][Zim13][WZNP14][JWH15][Cas15][SWF+15][BBCK15].

A promising aspect of Modelica is that it is an open-source language that is supported internationally by various industries. As these industry sectors use the same modeling language, modeling environments, simulation and optimization code generators and solvers, the investement in these technologies can be shared. Consequently, large international projects that further advance Modelica have been conducted, such as

- MODELISAR (https://itea3.org/project/modelisar.html, 29 partners, Euro 26.6M, 2008-2011) which initiated the FMI standard,
- EUROSYSLIB (http://www.eurosyslib.com/, 19 partners, Euro 16M, 2007-2010) which developed Modelica libraries for embedded system modeling and simulation, and
- MODRIO (https://www.modelica.org/external-projects/modrio, Euro 21M, 2012-2015) which extended Modelica and FMI to support property/requirement modelling, state estimation, multi-mode modelling, e.g., systems with multiple operating modes and varying number of states, and nonlinear model predictive control.

4.2.2 Co-Simulation and Model Exchange

In 2008, a European project called MODELISAR started with the objective to facilitate interoperability between simulation models and simulation tools through a standardized application programming interface (API). This project resulted in the Functional Mockup Interface (FMI) standard, which is a tool-independent, open-source standard which supports exporting, exchanging and importing simulation models or simulation tools *[MC14]*.

A simulation model or a complete simulator that is exported in the format specified by the FMI standard is called a Functional Mockup Unit (FMU). The FMI standard defines a set of C-functions (FMI functions) to interact with the model or the simulator. It also defines an xml schema that is used to declare properties of the exported model or simulator. In addition, it standardizes how to package as a zip file the xml file, the C-functions, possibly as compiled binaries, and resources required by the model or simulator, such as files with weather data.

The FMI standard distinguishes between model-exchange and co-simulation. In FMI for model-exchange, a system of differential, algebraic and discrete-time equations can be exported, and the host simulator that executes the FMU needs to provide an algorithm that integrates the equations in time. In contrast, in FMI for co-simulation, the host simulator requests the FMU to integrate its equations in time. See for example *[BBG+13]* for such an algorithm.

Version 2.0 of FMI standard was released in 2014, and it adds features that will facilitate the use of FMU models to support the design and operation of buildings. Some of the important features are as follows:

Saving and restoring the state The complete FMU state can be saved, restored, and serialized to a byte vector. As a result, a simulation can be restarted from a saved FMU state. This is a very important feature for model-based fault detection as the one in *[BSG+14]* or model predictive controls applications as both of them require state initialization.

Input and state dependencies In the xml file, it can be declared which state variables and which output variables have a *direct dependency* on the input variables, and which output variables have a direct dependency on the state variables.

This allows

1. determining the sparsity pattern for Jacobians, and
2. to use sparse matrix methods in numerical solvers to simulate stiff FMUs.

The information about dependency also opens the door to the implementation of efficient asynchronous numerical time integration algorithms such as QSS.

Furthermore, for FMUs that are connected to form a cyclic graph, the dependency information of outputs on inputs is required for the deterministic execution *[BBG+13]*, and the detection of algebraic loops. Once those algebraic loops are detected, nonlinear equation solvers such as a Newton-Raphson solver can be used to solve them.

The following example, which is borrowed from *[BBG+13]*, illustrates why exposing such dependencies is important. Consider the FMU that comprises the system shown in Fig. 4.1. If this FMU is imported in a simulator and y_1 is connected to u, possibly using an algebraic function $f\colon \mathbb{R} \to \mathbb{R}$, then a master algorithm can output the state y_1, assign $u = f(y_1)$, output $y_2 = -5\,u$ and integrate the state. If however u were connected to $f(y_2)$ rather than $f(y_1)$, then a master algorithm can output y_1, next it needs to solve $u = f(y_2) = -5\,f(u)$, in general using numerical iterations, and then it can integrate the state. This illustrates that input-output dependency is important as it allows a simulator to detect whether cyclic graphs, formed by connecting inputs and outputs among FMUs, lead to an algebraic system of equations that may require an iterative solution. See *[BBG+13]* for a more detailed discussion.

Directional derivatives Directional derivatives can be computed for derivatives of continuous-time states and for outputs. This is useful when connecting FMUs and the partial derivatives of the connected FMU shall be computed, for example by a stiff ordinary differential equation solver, an algebraic loop solver, an extended Kalman filter, or for model linearization. If the exported FMU performs this computation analytically, then all numerical algorithms based on these partial deriva-

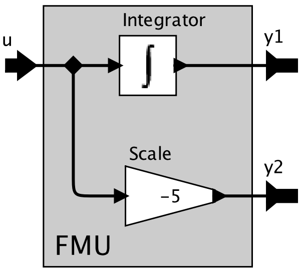


Fig. 4.1: *FMU for which output y_1 does not have a direct dependency on input u but output y_2 does.*

tives are more efficient and more reliable [MC14]. Directional derivatives are also required by second-order QSS algorithms [Kof03]. This is illustrated with the following example. Consider an FMU which implements $dx(t)/dt = f(x,u,t)$ for some differentiable function $f\colon \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. If the FMU provides directional derivatives, then the second time derivative can be computed exactly because

$$\frac{df(x,u,t)}{dt} = \frac{\partial f(x,u,t)}{\partial x}\frac{dx(t)}{dt} + \frac{\partial f(x,u,t)}{\partial u}\frac{du(t)}{dt},$$

where $\partial f(x,u,t)/\partial x$ and $\partial f(x,u,t)/\partial u$ are the directional derivatives with respect to state and input, which are provided by the FMU.

In summary, there are various benefits in using equation-based languages, such as Modelica, for system simulation. First, they allow for sufficient semantics for a code generator to identify the state variables in a model, which supports saving and restoring states for initializing simulations. Second, they allow for the discovery of input-output and input-state dependencies, which supports master algorithm development. Lastly, they allow for automatic differentiation of model equations, which supports in providing directional derivatives to solvers. While these pieces of information could in principle also be specified by a model developer in models that are written using imperative languages, the size of models typically encountered in building simulation would make such a manual declaration a tedious, expensive and error-prone proposition.

The relevance of these properties for the building simulation community has been illustrated in the following examples. Broman et al. [BBG+13] developed a master algorithm for the deterministic composition of FMUs for co-simulation, which is only possible if input/output dependencies are provided for FMUs that are connected in a cyclic graph. Wetter et al. [WNL+15] simulated a building with radiant heating system using a collection of FMUs for model exchange that are asynchronously integrated in time using QSS methods. The input-state dependencies were required to determine which state variables need to be updated. Bonvini et al. [BWS14] have developed and applied an FMU-based state and parameter estimator that has been used as part of a fault detection algorithm which is capable of identifying faults in a valve. This algorithm required saving and restoring states.

The capabilities of FMI and the aforementioned use cases indicate its applicability to support building simulation for design and operation. At the time of writing, there are more than 70 tools which support import or export of simulation models or tools as FMUs. This indicates the adoption of the standard and its relevance for the building simulation community.

4.2.3 Optimization

Equation-based modeling languages allow code generators to convert model equations to a form that is well suited to solve large scale nonlinear optimization problems [AAG+10]. This section describes a state of the art method that converts an infinite dimensional optimal control problem into a finite dimensional approximation that standard nonlinear programming (NLP) solvers can solve. Equation-based modeling languages allow automating this conversion.

Equation-based modeling languages allow to describe systems of differential algebraic equations (DAE) in the general form

$$\begin{aligned} F(t,\, \dot{x}(t),\, x(t),\, u(t),\, y(t),\, \Theta) &= 0, \\ Y(t,\, x(t),\, u(t),\, y(t),\, \Theta) &= 0, \\ F_0(\dot{x}(t_0), x(t_0), u(t_0), y(t_0), \Theta) &= 0, \end{aligned} \tag{4.1}$$

where $F(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ describes the time rate of change, $Y(\cdot,\cdot,\cdot,\cdot,\cdot)$ are algebraic constraints, $F_0(\cdot,\cdot,\cdot,\cdot,\cdot)$ implicitly defines initial conditions, $t \in [t_0,\ t_f]$ is time for some initial and final time t_0 and t_f, $x\colon \mathbb{R} \to \mathbb{R}^{n_x}$ is the state vector, $u\colon \mathbb{R} \to \mathbb{R}^{n_u}$ is the control function, $y\colon \mathbb{R} \to \mathbb{R}^{n_y}$ is the vector of algebraic variables, and $\Theta \in \mathbb{R}^p$ is the vector of parameters. Such a DAE system can be used to model a building, its HVAC systems and controllers. Necessary and sufficient conditions for existence, uniqueness and differentiability of a solution to (4.1) can be found in *[Wet05]*.

Once the model is available, we can add constraints and a cost function to define an optimal control problem that minimizes energy consumption or cost. An example optimal control problem for (4.1) is

$$\begin{aligned}
\underset{u(\cdot)\in\mathcal{U},\, \Theta\in\mathbb{R}^p}{\text{minimize}} \quad & f(x(t),\, u(t),\, y(t),\, \Theta), \\
\text{subject to} \quad & F(t,\, \dot{x}(t),\, x(t),\, u(t),\, y(t),\, \Theta) = 0, \\
& Y(t,\, x(t),\, u(t),\, y(t),\, \Theta) = 0, \\
& F_0(\dot{x}(t_0),\, x(t_0),\, u(t_0),\, y(t_0),\, \Theta) = 0, \\
& H(t,\, \dot{x}(t),\, x(t),\, u(t),\, y(t),\, \Theta) = 0, \\
& G(t,\, \dot{x}(t),\, x(t),\, u(t),\, y(t),\, \Theta) \leq 0,
\end{aligned} \tag{4.2}$$

for all $t \in [t_0,\, t_f]$, where $f(\cdot,\cdot,\cdot,\cdot)$ is the cost function and $\mathcal{U}$ is the set of admissible control functions. The solution to (4.2) is the optimal control function and the optimal design parameter that minimizes $f(\cdot,\cdot,\cdot,\cdot)$ while satisfying the system dynamics $F(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot) = 0$, and $Y(\cdot,\cdot,\cdot,\cdot,\cdot) = 0$, the initial conditions $F_0(\cdot,\cdot,\cdot,\cdot,\cdot) = 0$ and the constraints $H(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot) = 0$ and $G(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot) \leq 0$. For generality, we assume (4.2) to be nonlinear and twice continuously differentiable *[Pol97]*.

The problem (4.2) is infinite dimensional because its solution is a functional that has to be valid for all $t \in [t_0,\, t_f]$. Directly solving an infinite dimensional optimal control problem for a general nonlinear system is not possible and it therefore needs to be converted into a finite dimensional approximation *[Pol97]*. Biegler *[Bie10]* presents multiple methods for such a conversion into the form

$$\begin{aligned}
\underset{z\in\mathbb{R}^{n_z}}{\text{minimize}} \quad & c(z), \\
\text{subject to} \quad & z^l \leq z \leq z^u, \\
& g(z) = 0, \\
& h(z) \leq 0,
\end{aligned}$$

where z is the finite dimensional optimization variable, z^l and z^u are the lower and upper bounds, $c(\cdot)$ is the cost function, and $g(\cdot)$ and $h(\cdot)$ are the equality and inequality constraints.

Among the available techniques, we describe direct collocation methods because they are well suited for equation-based modeling languages [AAG+10]. Direct collocation methods use polynomials to approximate the trajectories of the variables of a DAE system. The polynomials are defined on a finite number of support points that are called collocation points. By optimizing these finite number of control points, they convert the infinite to a finite dimensional optimization problem, which can be solved by a NLP solver such as IPOPT [WB06].

The method starts by dividing the time horizon $[t_0, t_f]$ into n_e elements, each element containing the same number of collocation points n_c. For example, the JModelica software [AGT09] uses the Radau collocation method to place these points. The Radau collocation method places a collocation point at the start and end of each element to ensure continuity of the state trajectories, and places the others to maximize accuracy. In each element, time is normalized as $\tilde{t}_i(\tau) = t_{i-1} + h_i\,(t_f - t_0)\,\tau$, for $\tau \in [0, 1]$ and $i \in \{1, \ldots, n_e\}$, where t_i is the time at the end of element i, $\tau \in [0, 1]$ is the normalized time within the element, and h_i is the length of element i. The time dependent variables $\dot{x}(\cdot)$, $x(\cdot)$, $u(\cdot)$, and $y(\cdot)$ are approximated using collocation polynomials in each element. The collocation polynomials use the Lagrange basis polynomials, and they use the collocation points as the interpolation points. The collocation polynomials are

$$\begin{aligned} x_i(\tau) &= \sum_{k=0}^{n_c} x_{i,k}\,\tilde{l}_k(\tau), \\ u_i(\tau) &= \sum_{k=1}^{n_c} u_{i,k}\,l_k(\tau), \\ y_i(\tau) &= \sum_{k=1}^{n_c} y_{i,k}\,l_k(\tau), \end{aligned} \tag{4.3}$$

where $x_{i,k}$, $u_{i,k}$, and $y_{i,k}$ are the values of the variable $x(\cdot)$, $u(\cdot)$ and $y(\cdot)$ at the collocation point k in element i, $l_k(\cdot)$ is the Lagrange basis polynomial and $\tilde{l}_k(\cdot)$ is the Lagrange basis polynomial that includes the first point to ensure continuity of the state variables. The Lagrange bases are, with $i \in \{1, \ldots, n_e\}$,

$$\begin{aligned}\tilde{l}_k(\tau) &= \prod_{j\in\{0,\ldots,n_c\}\setminus\{k\}} \frac{\tau-\tau_j}{\tau_k-\tau_j},\\ l_k(\tau) &= \prod_{j\in\{1,\ldots,n_c\}\setminus\{k\}} \frac{\tau-\tau_j}{\tau_k-\tau_j}.\end{aligned} \tag{4.4}$$

As τ is normalized, the basis polynomials are the same for all elements. The polynomial approximation of the derivative $\dot{x}_i(\cdot)$ in (4.3) is

$$\dot{x}_i(\tau) = \frac{1}{h_i\,(t_f-t_0)} \sum_{k=0}^{n_c} x_{i,k} \frac{d\tilde{l}_k(\tau)}{d\tau}. \tag{4.5}$$

The collocation method defines the approximations (4.3) and (4.5) of the variables in (4.2). Equation-based modeling languages allow accessing the model equations, thereby allowing to automatically generate the finite dimensional approximations defined by the collocation methods.

JModelica employs a collocation method to transcribe the problem (4.2) into an NLP problem. A local optimum to the finite dimensional approximation of (4.2) will be found by solving the first-order Karush-Kuhn-Tucker (KKT) conditions, using iterative techniques based on Newton's method. This requires first- and second-order derivatives of the cost and constraint functions with respect to the NLP variables. JModelica uses CasADi *[And13]*, a software for automatic differentiation that is tailored for dynamic optimization. Equation-based modeling languages allow for automatically providing the information required by CasADi to build a symbolic representation of the optimization problem. Using the symbolic representation of the NLP problem, CasADi can efficiently compute the required derivatives and exploit the sparsity pattern of the problem. NLP solvers such as IPOPT are then used to find a piecewise polynomial approximation of the solution to the original problem (4.2). The number of variables in the approximated problem is $n_z = (1+n_e\,n_c)(2n_x+n_u+n_y)+(n_e-1)n_x+n_p+2$. For a more detailed overview see *[MA12]*.

In summary, equation-based modeling languages provide three main advantages for optimization: First, they support the automatic conversion of simulation models into optimization problems, reducing engineering costs and time. Second, they can provide analytic expressions for gradients to be used by NLP solvers. Third, they allow for automatic generation of the finite dimensional approximations defined by the collocation methods.

Section 9.4.3 shows how this improves computing performance relative to simulation-based optimization.

4.2.4 Building Information Modeling

Building Information Modeling (BIM) provides methods, interfaces and tools for the integral design, construction, commissioning and operation of buildings. It is furthermore an enabler for quality assurance and digital documentation of the as-built state and to manage other building life cycle-relevant data *[ETSL11]*. Managing projects with BIM promises major improvements in the adherence of schedules, in transparency and in cost control *[VIB15]*, if a BIM project is properly set-up and run. Digital planning methods are a key element for the design, commissioning and operation of energy efficient buildings, energy systems and city quarters at the interface between building envelope, building systems, distribution network, automation and control.

BIM-related processes may comprise the coordination of different models of the architecture, engineering and construction (AEC) domains, for example involving advanced rule-based model checker software. On the other hand, BIM may be applied for domain-specific planning tasks within the building services and HVAC domains. Thereby, a CAD model can serve as basis for layout and dimensioning, for engineering and code compliance testing, clash detection, or static and dynamic heat and cooling load calculations, for example.

Today, powerful CAD tools exist for the AEC domain which can be used for design and construction of HVAC systems. Some of these tools provide built-in and proprietary solutions for static or dynamic calculations building on their internal core and data model.

However,

- the lack of open-source solutions to support a tool-chain for BPS model transformation from BIM using open data formats such as the *Industry Foundation Classes (IFC)* makes it difficult to make BIM models available for BPS.

- Other BPS-related data formats such as gbXML are mainly restricted to geometrical issues and disregard parameter which are relevant for describing properties of HVAC components or control sequences.
- Current BIM formats lack the objects and semantics needed to express control logic, e.g., the algorithms that turn measured signals and set-points into actuator signals.
- Defining and generating an integrated building performance simulation model representing the building geometry and topology as well as its energy systems can be a cumbersome and error-prone procedure *[BMOD+11]*.
- Furthermore, a CAD or BIM model cannot be readily transformed into an object-oriented simulation model, as the structure of both prevailing modeling worlds differ significantly *[vTR06]*. Models may be hampered by diverse inconsistencies due to modeling failures or inconsistencies or simply due to conceptual differences between the AEC domains and their modeling hierarchy, especially from a geometrical and topological point of view concerning the issue of space boundaries *[BK07]*.
- The representation of CAD objects and its parameters in the HVAC domain itself differs from the representation which is needed in an object-oriented BPS model such as Modelica. In BIM, objects may not be properly linked with each other, or the way, how these objects are mutually connected may not be eligible for a model transformation into Modelica code which assumes objects are connected as in the real world through fluid ports.

These constraints often make it necessary to manually re-generate a BPS model from scratch instead of converting an existing CAD model to a BPS-like representation.

4.2.5 Overview of the Following Chapters

The next chapters describe the activities conducted in Annex 60.

Activities 1.1 to 1.4 were focused on the development of technologies for modeling, co-simulation, BIM to Modelica translations and workflow automation developed in Subtask 1. Activity 1.1, described in Section 5, gives an overview about Modelica and the Modelica Annex 60 library developed in this project.

Activity 1.2, described in Section 6, introduces co-simulation using the FMI standard and presents FMI compliant tools and FMI capabilities of building energy simulators. Activity 1.3, described in Section 7, introduces BIM and presents an open framework for Modelica code generation from BIM. Lastly, Activity 1.4, described in Section 8, presents tools and examples for workflow automation in building and district energy simulation.

Activities 2.1 to 2.3 were focused on the validation and demonstration of the technologies developed in Subtask 2. Activity 2.1, described in Section 9, presents case studies that involve Modelica-based simulation and optimization at the building scale. Activity 2.2, described in Section 10, provides an overview about district energy systems. It then introduces first efforts to develop a validation test procedure for district energy system simulations called DESTEST, and it closes with examples of district energy simulation using mono-simulation and co-simulation. Activity 2.3, described in Section 11, describes use of Modelica and FMI for Fault Detection and Diagnostics, for Model Predictive Control, and for Hardware-in-the-Loop experimentation.

Concluding remarks can be found in Section 13, and a glossary for technical terms can be found in Section 14.

Chapter 5

Activity 1.1: Modelica

This chapter provides an overview of Modelica, gives an introduction with pointers to further literature about its capabilities, and presents the Modelica `Annex60` library that has been developed within the IEA EBC Annex 60. The library is available from http://www.iea-annex60.org/releases/modelica/1.0.0/Annex60-v1.0.0.zip. The models are documented at http://www.iea-annex60.org/releases/modelica/1.0.0/help/Annex60.html. They will be further developed within the IBPSA Project 1 at https://github.com/ibpsa/modelica-ibpsa.

5.1 Introduction

Modelica is an object-oriented equation-based modeling language. The language has been developed to support design and operation of complex engineered systems that are governed by differential equations, algebraic equations, time- and state events. It is now used in various industrial sectors such as automotive, aerospace, electrical engineering, power plants, robotics, buildings and district energy systems.

At the time of this writing, more than 70 free open-source and commercial Modelica libraries were available, see for example https://modelica.org/libraries and http://impact.github.io/#/all. The Modelica Standard Library version 3.2.1, which is the official library of the Modelica Association, contains about 1500 models, blocks and functions, covering most engineering domains.

The goal of the Annex 60 library development is to provide well documented, vetted and validated open-source Modelica models that will serve as the core of future building and district energy simulation programs.

5.2 Literature Review

Modelica origins in the PhD thesis of Hilding Elmqvist in 1978 at the Department of Control, Lund University, under the supervision of Karl Johan Åström *[Elm78]*. In *[Elm14]*, Elmqvist gives a perspective on the Modelica evolution which is summarized as follows: In his PhD thesis, Elmqvist designed the DYnamic MOdeling LAnguage Dymola, which contained a class object for object-oriented modeling and a structured feature to describe interaction between submodels. In Spring 1996, discussions started to unify the modeling languages Dymola, Omola, NMF and Allan. In Fall 1996, the first design meeting was held as Elmqvist realized that it was time for a global unified language design initiative. The primary reason was that modeling requires reuse of stored knowledge and hence a standard language, rather than the various tool vendors inventing their own language and a new language being created for every PhD thesis on modeling. In 1997, after one year of design, Modelica 1.0 was released. The first version of Dymola that supported Modelica was released in 1999 and three years later the first open-source implementation OpenModelica followed *[FAB+02b]*.

In the buildings community, the use of equation-based languages has its origin in the energy simulation program ENET *[LS82][STLL84]*, which provided the foundation of the SPANK or SPARK program *[SBEW86][SBN89][BEWS93]*. In 1989, Sahlin and Sowell *[SS89]* introduced an equation-based language called Neutral Model Format (NMF) which is used in the commercial software IDA/ICE *[BBE+99]*. In 1993, Klein introduced the equation-based Engineering Equation Solver EES *[Kle93]*.

Felgner et al. *[FAB+02a]* reported in 2002 the earliest applications of Modelica for building energy simulation *[Mer02]*, which led to the ATPlus library *[FAB+02a]* which no longer seems to be developed. In 2005, Nytsch-Geusen et al. presented a hygrothermal building model [NGNHH05]. In 2004, development of a proprietary Modelica library for building and HVAC system simulation started at the United Technology Research Center [Wet06b][Wet06a]. In 2007, Wetter started at LBNL the development of the open-source Modelica Buildings library [WHMS08][Wet09][WZNP14]. Also in 2007, Müller started to develop the model library AixLib for building and urban energy systems at RWTH Aachen University [BM10][FCL+15]. This work was based on his first activities at TU Berlin to model active building components in Modelica [HHTM05]. In 2010, Baetens et al. started at KU Leuven the development of IDEAS, a library for Integrated District Energy Assessment Simulations [BDCVR+12][BDCJ+15]. In 2012, Nytsch et al. presented the Modelica BuildingSystems library developed at UdK Berlin [NGHLR12].

Between 2012 to 2014, the AixLib, BuildingSystems and IDEAS libraries were made open-source to facilitate collaboration within the IEA EBC Annex 60. With the AixLib, Buildings, BuildingSystems and IDEAS libraries, four open-source libraries for building and district energy systems became available; however, due to different design choices and modeling principles, combining models from these libraries was cumbersome if not impossible for most users. Furthermore, because the libraries used different conventions and implementations, sharing development effort was not possible prior to the Annex 60, thereby wasting development resources. Therefore, within Annex 60, the design choices of these libraries were harmonized, models were made compatible, and a common core of a Modelica library, the *Annex60* library, has been developed [WFG+15]. All four of these libraries are now based on the collaboratively developed *Annex60* library whose goal is to provide a well-documented, vetted and validated open source Modelica library that serves as the core of future building simulation programs.

In addition to these open-source libraries, various commercial and proprietary libraries have been developed. In 2005, Modelon presented the *AirConditioning* library [TEP05] for detailed steady-state and dynamic modeling of vapor compression cycles, and TLK Thermo presented a similar library as part of the TIL suite [GKRT10]. Since 2009, XRG develops and distributes the *HVAC* library and *HumanComfort* libraries [ME09]. In 2012, ITI, EA Systems, Honda and TU Dresden introduced the *GreenBuilding* library [USM+12]. In 2014, Électricité de France presented the *BuildSysPro* library [PKL14].

Various literature compares equation-based and procedural modeling languages. Sahlin [Sah96] reports that developing a simulation model in the equation-based modeling language NMF [SS89] was about three times faster than it was in the BRIS simulation program [Bro90], but the computation time in BRIS was three times faster. Sahlin et al. [SEG+04] compared the computation time of IDA ICE with EnergyPlus. In their numerical experiments, IDA ICE required approximately half the computation time that was required by EnergyPlus for a three zone building with natural ventilation and twice the computation time in an initial experiment for a 53 zone building without natural ventilation. In the latter experiments, the COMIS airflow program was not included in EnergyPlus. Sowell and Haves [SH01] compared the computation time of SPARK and HVACSIM+ for a variable air volume flow system that serves six thermal zones. They report that SPARK computes about 15 to 20 times faster than HVACSIM+ and attribute the faster computation to SPARK's graph decomposition and cut set reduction. Wetter et al. [WHMS08] compared the SPARK solver with Dymola for a variable air volume flow system with five zones and concluded that they have comparable computing time. Wetter and Haugstetter [WH06] compared computing time in TRNSYS and the Modelica environment Dymola. TRNSYS simulated about three to four times faster, which most likely was due to the time series representation used in TRNSYS for the conduction heat transfer calculation. Jorissen et al. [JWH15] show how model and solver knowledge can lead to simulation times that are two orders of magnitude lower for large Modelica models when using explicit solvers instead of implicit solvers.

Also, decreasing the computing time for large Modelica models through changes in the Modelica translators and numerical methods is an active area of development, see for example [MBKC13][FBC+14][Cas15][BBCK15][BRC16][CR16][OE17][BCB17][JHB17].

To compare the labor time for model development, Wetter [Wet11b] compared the development time and, as development time often scales with code size, he also compared for cross comparison the code size for models in Fortran, C/C++ and Modelica. Their development time comparison indicates a five to ten times faster development through use of Modelica rather than C++ for the C++ multizone building model of BuildOpt [Wet05]. This translated to about one year of labor savings. The Modelica implementation required 6,000 lines of code while the C++ implementation required 24,000 lines. A similar reduction in labor time was observed for a small heat transfer problem that was independently implemented in Fortran and in Modelica by three engineers who were familiar with both languages. See Fig. 5.1.

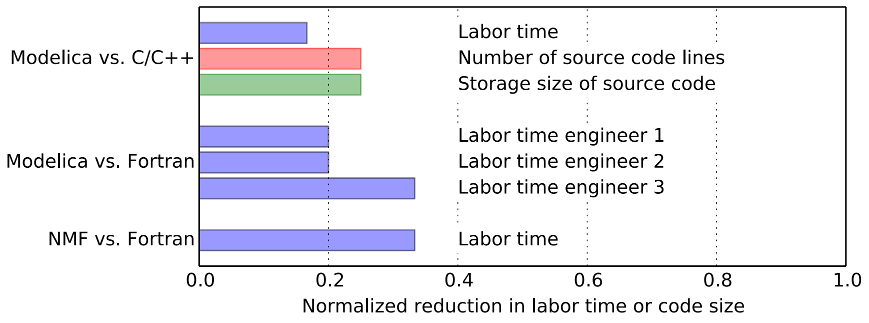


Fig. 5.1: Comparison of relative reduction in labor time and code size for equation-based vs. procedural programming languages.

5.3 What is Modelica

This section gives an overview about the main capabilities of Modelica. The intent is to familiarize the reader with the ideas of the Modelica language and explain how Modelica models are converted to simulation code. While most users will use existing Modelica component models from a library, this section also gives some background information that helps understanding the principles of the language.

Modelica is an equation-based, acausal, object-oriented modeling language that is designed for component-oriented modeling of dynamic systems. Models are described by differential equations, algebraic equations and discrete equations. Using standardized interfaces, the mathematical relations of a problem between its interface variables are encapsulated in a model, which can be represented graphically by an icon. The interface variables of multiple models can be graphically connected with each other in a graphical model editor, without requiring any notion of what is input and what is output of a model.

This encapsulation together with the standardized acausal model interface facilitates model reuse and model exchange. Since Modelica is a standardized language, models can be shared and exchanged by a large user community.

To realize such a flexible modeling environment, the Modelica language embodies the object-oriented modeling paradigm, a term that was coined by Elmqvist *[Elm78]* and summarized by Cellier et al. *[CEO95]* as follows:

Encapsulation of knowledge The modeler must be able to encode all knowledge related to a particular object in a compact fashion in one place with well-defined interface points to the outside.

Topological interconnection capability The modeler should be able to interconnect objects in a topological fashion, plugging together component models in the same way as an experimenter would plug together real equipment in a laboratory. This requirement entails that the equations describing the models must be declarative in nature, i.e., they must be acausal.

Hierarchical modeling The modeler should be able to declare interconnected models as new objects, making them indistinguishable from the outside from the basic equation models. Models can then be built up in a hierarchical fashion.

Object instantiation The modeler should have the possibility to describe generic object classes, and instantiate actual objects from these class definitions by a mechanism of model invocation.

Class inheritance A useful feature is class inheritance, since it allows encapsulation of knowledge even below the level of a physical object. The so encapsulated knowledge can then be distributed through the model by an inheritance mechanism, which ensures that the same knowledge will not have to be encoded several times in different places of the model separately.

Generalized Networking Capability A useful feature of a modeling environment is the capability to interconnect models through nodes. Nodes are different from regular models (objects) in that they offer a variable number of connections to them. This feature mandates the availability of across and through variables, so that power continuity across the nodes can be guaranteed.

The following subsections describe the approaches that are used in Modelica to realize this object-oriented modeling paradigm.

5.3.1 Acausal Modeling

5.3.1.1 Physical Connectors and Balanced Models

A tenet of Modelica is that each component should represent a physical device with physical interface ports called connectors. To accomplish this, Modelica requires models to be balanced. Casually expressed, a model is balanced when the number of its equations equals the number of its variables. A more rigorous definition can be found in *[OOME08]*. Models interact with other models through connectors. The connectors of a model contain all variables required to uniquely define the boundary conditions of the model.

The Modelica Standard Library implements connectors for various domains, such as heat transfer, fluid flow, electrical and translational systems. These connectors declare the interfaces of the models. Table 5.1 shows different connectors and their variables, which are either potential, flow or stream variables. Potential variables are the driving force for a flow, and hence examples are voltage V, position x, temperature T, and pressure p. The associated flow variables are current I, force F, heat flow rate $\dot{Q}$ and mass flow rate $\dot{m}$. Thermofluid flow systems are a special case because the mass flow rate carries properties such as specific enthalpy h, mass fractions X and trace substance concentrations C. These properties therefore change depending on the flow direction. Hence, these are *stream* variables, and are, for numerical performance, treated in Modelica in a unique way *[FCO+09a]*.

When connectors are connected with each other, Modelica automatically generates equations that equate the potential variables. For example, connecting n connectors of heat flow components c_i, for $i \in \{1, \dots, n\}$, results in all c_i having the same temperatures at the connection points, e.g., $c_1\, T = c_i\, T$, for all $i \in \{1, \dots, n\}$. Furthermore, for the electrical, translational and heat flow do-

Table 5.1: Physical connectors from the Modelica Standard Library.

Domain	Potential	Flow	Stream
Electrical	V	I	
Translational	x	F	
Heat flow	T	$\dot{Q}$	
Thermofluid flow	p	$\dot{m}$	h, X, C

main, the sum of the flow variables is set to zero. For example, for the above heat flow components, Modelica will generate the equations $0 = \sum_{i=1}^{n} c_i \, \dot{Q}$. Thermofluid flow components are different because the mass flow rate $\dot{m}$ carries the fluid properties (h, C, X). To allow connecting multiple fluid connectors, Modelica uses so called stream-connectors [FCO+09a]. This allows Modelica to generate for connected thermofluid flow components the equations for conservation of mass $0 = \sum_{i=1}^{n} c_i \, \dot{m}$. For the properties that are carried with the flow, Modelica generates the equations

$$\chi_{mix} = \frac{\sum_i \dot{m}_i^+ \, \chi_i}{\sum_i \dot{m}_i^+}, \tag{5.1}$$

where χ is any of the conserved quantities (h, C, X) and $\dot{m}_i^+ = \max(0, \dot{m})$.

These rules allow to connect one or multiple components to any physical port. For example, if a heat flow port is unconnected, then no equation for its temperature is introduced, and the heat flow rate across this port is set to zero, thereby rendering the port as an adiabatic boundary condition. Whether a port variable is an input or an output to the model is determined when the model is translated. Hence, models impose no causality on the connector variable. This property allows to reuse, for example, a model for one-dimensional steady-state heat conduction for situations where both port temperatures are known and the heat flow rate at the ports need to be computed, or where one port temperature and heat flow rate are known and the other port temperature and heat flow rate needs to be computed.

This encapsulation of balanced models with acausal physical connectors enables a graphical, input-output free model construction, as we will see later in this section. In a schematic model diagram of a physical system, icons correspond to actual components or subsystems and encapsulate the equations that define the physics of the subsystem. Lines between the icons impose the port equations to conserve flow and to equate state variables, or they may propagate signals in a control system. Because these connectors are standardized, models from different libraries can be connected with each other, thereby allowing users to use models from various domains and libraries within one system model.

5.3.1.2 Illustration of a Simple Model

We will now illustrate how a simple heat flow component is built using the heat flow connector of the Modelica Standard Library. This connector is implemented by the following lines of Modelica code:

```
1 partial connector HeatPort
2     "Thermal port for 1-D heat transfer";
3   SI.Temperature T "Port temperature";
4   flow SI.HeatFlowRate Q_flow "Heat flow rate";
5 end HeatPort;
```

On line 4, the type prefix `flow` declares that `Q_flow` is a flow variable and hence need to be summed to zero as explained above. This connector can then be instantiated to define the interface with the outside of the model in a one-dimensional heat transfer element with no energy storage. In Modelica's thermal library, such a heat transfer element is implemented as follows:

```
partial model Element1D
  "Partial heat transfer element with two HeatPorts"
  SI.HeatFlowRate Q_flow
    "Heat flow rate from port_a->port_b";
  SI.TemperatureDifference dT "port_a.T-port_b.T";
 public
  HeatPort port_a "Heat port a";
  HeatPort port_b "Heat port b";
 equation
  dT = port_a.T - port_b.T;
  port_a.Q_flow = Q_flow;
  port_b.Q_flow = -Q_flow;
end Element1D;
```

Lines 3 to 5 contain the declaration of the variables $\dot{Q}$ and ΔT that are typically computed in a one dimensional heat transfer element. Lines 7 and 8 instantiate the `HeatPort` connector to expose to the outside of this model the port temperatures `port_a.T` and `port_b.T`, as well as the port heat flow rates `port_a.Q_flow` and `port_b.Q_flow`. The equations on line 10 to 12 define the relationships among the variables of the two `HeatPort` connectors and the variables of the partial model `Element1D`. Note that `Element1D` does not declare a relation between the heat flow rate and the temperatures, as this relation is different for conduction, convection or radiation. Because this relation is not specified, the model is declared `partial` to indicate that it can be extended by other models to refine its implementation, but that it cannot be instantiated directly as it does not define how temperature and heat flow rate are related. To implement a thermal conductor, the above partial model can be extended as follows:

```
1 model ThermalConductor
2   "Lumped thermal element
3    transporting heat without storing it"
4   extends Interfaces.Element1D;
5   parameter SI.ThermalConductance G
6     "Constant thermal conductance";
7 equation
8   Q_flow = G*dT;
9 end ThermalConductor;
```

This thermal conductor model can then be encapsulated in a graphical icon using drawing elements that are part of the Modelica language standard, and hence, can be interpreted by different Modelica modeling environments. By using a different parameter declaration on line 5 and a different equation on line 8, the semantics can be changed to represent other one-dimensional heat transfer elements such as a model for long-wave radiation between two surfaces. Note that the above code is not pseudo-code, but rather a complete implementation of a heat conductor in the Modelica language, except for the optional graphical annotations that will be explained in the next section. Note that in the implementation above, the model developer only declared the variables and the constraints between heat flow rate and temperatures. Whether the model will solve for the heat flow rate or for a port temperature will be determined by a code generator that analyzes the overall simulation model in order to determine a numerically efficient sequence of computations, as we will show in Section 5.3.4.

5.3.1.3 Graphical Encapsulation

Creating, changing and understanding reasonably large system models without a graphical model editor would be very cumbersome if not impossible. To be able to use models in different graphical model editors, Modelica also standardizes graphical elements such as lines, circles and colors that can be used to graphically render an icon which encapsulates the model. In graphical editors, these icons can be opened to see the actual model implementation, and the icons can be used to graphically assemble components to build system models. Fig. 5.2 shows how the above heat conductor can be used to compute the heat flow rate for a time-varying and a constant temperature boundary condition on the left and right, respectively.

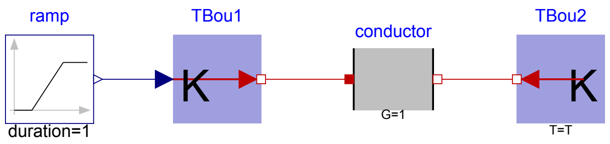


Fig. 5.2: Thermal conductor with varying temperature boundary conditions.

5.3.1.4 Object-Oriented Modeling

Object-oriented modeling allows reusing common functionalities across different models. Duplication of code which would make it difficult to correct coding errors throughout a large library of models when code is copied to different models. In the *Annex60* library, object-orientation is extensively used to implement heat and mass balances of thermofluid flow components. It is also used, for example, to implement two-way control valves, which we will use here to explain the principle of object-oriented modeling.

Two-way valves can be characterized by an opening function

$$\varphi(y) = \frac{k(y)}{K_v}, \tag{5.2}$$

where $y \in [0, 1]$ is the control input, $k : [0, 1] \rightarrow \mathbb{R}$ is the flow rate divided by the square root of the pressure drop and $K_v = k(1)$ characterizes that flow rate for a fully open valve. For a valve with linear opening characteristic, $\varphi(y) = l + y\,(1 - l)$, where l is the leakage of the closed valve, whereas for an equal percentage valve, the function $\varphi(\cdot)$ is more complicated. To share code that is common among these valves, we can implement a base class such as

```
partial model PartialTwoWayValveKv
  "Partial model for a two way valve using Kv"
  extends BaseClasses.PartialTwoWayValve;

equation
 k = phi*Kv_SI;
 m_flow=BaseClasses.FlowModels.basicFlowFunction_dp(
   dp=dp, k=k, m_flow_turbulent=m_flow_turbulent);
 // ... (other code omitted for brevity)
```

Now, we can implement the valve with linear opening characteristics by extending this base class and assigning the function $\varphi(\cdot)$ as follows:

```
model TwoWayLinear
  "Two way valve with linear flow characteristics"
  extends BaseClasses.PartialTwoWayValveKv(
    phi=l + y * (1 - l));
 // ... (other code omitted for brevity)
end TwoWayLinear;
```

Similarly, the valve with equal percentage opening characteristics can be implemented as

```
model TwoWayEqualPercentage
  "Equal-percentage two-way valve"
  extends BaseClasses.PartialTwoWayValveKv(
    phi=BaseClasses.equalPercentage(y, R, l, delta0));
  parameter Real R=50
    "Rangeability, R=50...100 typically";
  // ... (other code omitted for brevity)
end TwoWayEqualPercentage;
```

where the function `BaseClasses.equalPercentage` implements the opening characteristics.

This object-inheritance has various benefits. For users, if they understand what physics is implemented in the base class, for example that the valve linearizes the pressure drop calculations at very small flow rates, then they know that this applies to both valves. For a developer, if a new capability need to be implemented, such as a heat loss of the valve, then this can be done in the base class and will be automatically propagated to both valves models. This is also useful for users that build system models from existing component models. For example, in a variable air volume (VAV) flow system, each VAV damper likely has the same controller. Hence, the user can implement the controller in one object and instantiate it for each VAV damper. Should the control law require changes, then this can be done at a central place and the change be propagated to all instances of this controller.

5.3.1.5 Hierarchical Modeling

Hierarchical modeling supports keeping a well-defined and visually accessible model structure and helps manage complexity. This is aligned with the fact that physical problems are often analyzed by disaggregating them into several parts. Using hierarchical modeling allows having the same view within simulation models.

Consider for example modeling the thermal mass of a building. Standard workflow encompasses disaggregating the building into several thermal zones. In our fictive case, we consider two thermal zones (Fig. 5.3). Each of these thermal zones encompasses a number of wall elements representing e.g. exterior walls, interior walls and floor plates. These wall elements interact with each other by radiative heat exchange and via convective heat exchange through the indoor air volume. Each wall element in turn encompasses the description of heat transfer and heat storage effects, in our case described by thermal resistances and capacities.

Using hierarchical modeling approaches in Modelica, each level of thermal mass modeling can be created by connecting sub-models that represent lower levels. In this way, each level takes care of modeling the interactions between the sub-models.

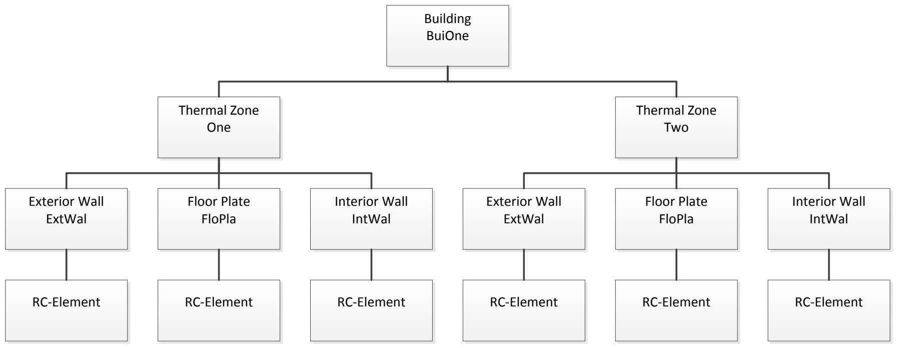


Fig. 5.3: Hierarchical structure of a thermal building model with levels for building, thermal zones, wall elements and RC-elements.

Refering to the system given in Fig. 5.3, the uppermost level is the building itself. At this level, we instantiate two thermal zones, using reduced order models for thermal zones with three wall elements:

```
model Building
  "Illustrates the use of hierarchical modeling"

  RC.ThreeElements thermalZoneOne(...)
  "Thermal zone one";
  RC.ThreeElements thermalZoneTwo(...)
  "Thermal zone two";
```

The visualization in the schematic diagram editor shows at the building level both thermal zones as simple icons without revealing any insights such as their sub-models. These insights become only visible on the thermal zone level as shown in Fig. 5.4. This level implements the interactions of the walls, e.g., the radiative heat exchange. It also instantiates wall elements for exterior walls, interior walls and a floor plate.

Looking into `BaseClasses.InteriorWall`, as shown in Fig. 5.5, reveals the actual implementation of resistances and capacities. These are connected in series using vectors of models to create a finite difference heat transfer model.

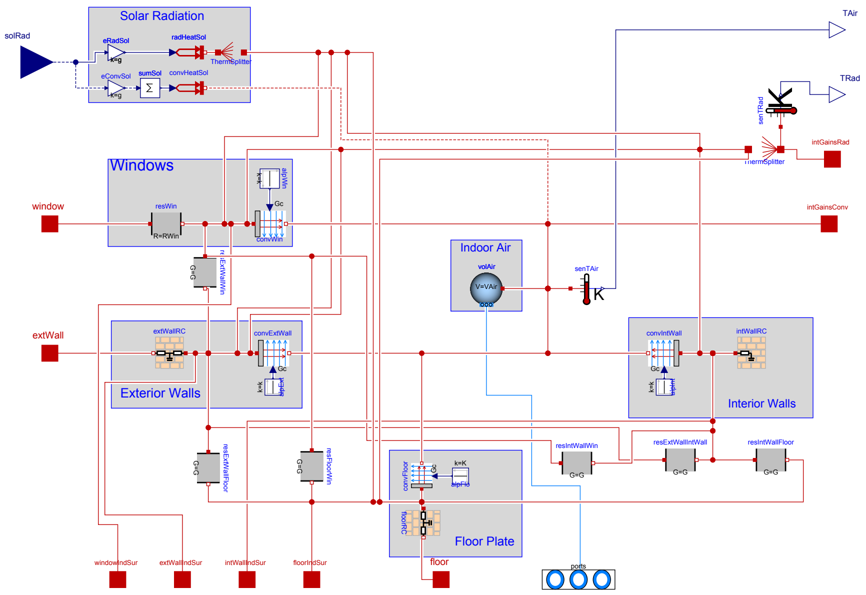


Fig. 5.4: *Diagram view of a thermal zone model with three wall elements (exterior walls, interior walls and floor plate), windows and indoor air volume.*

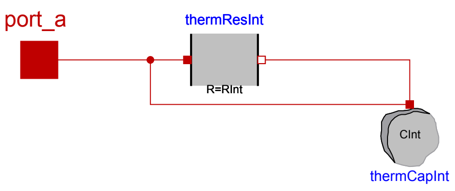


Fig. 5.5: *Diagram view of a thermal network for interior walls, consisting of an array of thermal resistances and capacities.*

5.3.1.6 Architecture-Driven Modeling

When evaluating different building systems, it is convenient to implement a system model of a base line, and then only declare how design variants differ from the base line. Modelica supports such modeling through architecture-driven modeling, which enables declaration of which models can be replaced with a different implementation. For illustration, consider a simple example in which we are interested in evaluating two different valves for a heat exchanger. One valve has linear opening characteristics, whereas the other valve has exponential opening characteristics. This can be accomplished by implementing a system model of the base case, which contains the valve with the linear opening characteristics. We can declare this, using the `TwoWayLinear` valve described above with the following code:

```
replaceable TwoWayLinear valve
  constrainedby BaseClasses.PartialTwoWayValveKv(
  redeclare package Medium = Medium,
  CvData=Buildings.Fluid.Types.CvTypes.Kv,
  Kv=0.65,
  m_flow_nominal=0.04)
  "Replaceable valve model”;
```

Here, we declared that the valve is of type `TwoWayLinear` (a two-way valve with linear opening characteristics) and it can be replaced by any other valve that uses (extends) the base class `PartialTwoWayValveKv`. The valve uses K_v data for the flow coefficient, with $K_v = 0.65$ and a design mass flow rate of $\dot{m} = 0.04\,\mathrm{kg/s}$. These values are also applied to any valve that replaces this `TwoWayLinear` valve. Typically, such a model is part of a larger system model as shown in Fig. 5.6 in which the replaceable valve is graphically rendered by a gray box.

With the few lines of code below, which can be either entered in a textual editor or created from a graphical model editor, we can create a new model that is identical to the previous model except for the valve. The corresponding declaration is:

```
model EqualPercentageValve
  "Model with equal percentage valve"
```

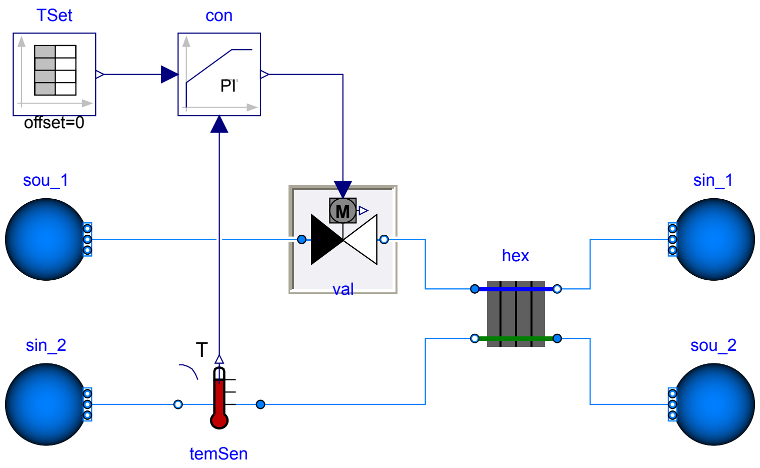


Fig. 5.6: *Model with heat exchanger, feedback control and replaceable valve. The gray background rendered around the valve indicates visually that this model can be replaced by another implementation.*

```
  extends DryCoilCounterFlowPControl(
  redeclare Actuators.Valves.TwoWayEqualPercentage
    valve);
end EqualPercentageValve;
```

This allowed us to change the architecture of the system by replacing a model, while reusing everything else in the base case. In large systems that may have hundreds of models, such modeling is very convenient as it shows only what has changed from one version to another. Note that the replaced model can be another hierarchical system model, for example one can replace in a whole building simulation a heating plant with a gas furnace by a plant that uses a heat pump with a borehole heat exchanger.

5.3.2 Separation of Concerns

When implementing a model, a model developer typically writes computer code that implements functions, variable assignments, computing procedures, numerical methods and routines to obtain input. With Modelica, this is no longer needed, as it suffices to simply state the physical laws and the control algorithm that a model should obey. Therefore, Modelica requires only declaration of the physics and dynamics of a model. How to solve the equations, which variables to assign first, when to read input and update outputs etc. need not be specified. Thus, it is a *modeling* language, as opposed to an imperative programming language such as C/C++. The underlying design philosophy is to separate the concerns between *modeling*, *simulation* and other applications such as *optimization* or *real-time simulation*. In Modelica it suffices to implement a mathematical model of the system without having to specify how to solve the equations. The simulation program is automatically generated from the mathematical model using a Modelica translator that employs computer algebra as explained in Section 5.3.4. This allows for various target applications: For example, from a Modelica model, code can be generated for the following:

A conventional time-domain simulator that uses a classical ordinary differential equation solver, and optionally generate code for sequential or parallel computing *[EMO14]*.

A discrete-event simulator that can handle order of magnitudes bigger models than classical ordinary differential equation solver *[MBKC13]*.

A building automation system that imports control code in a standardized format *[NW14]*.

An embedded real-time controller that may not have any hard disk for file input and output.

A web application in which the model is translated to JavaScript so it can run native in a web-browser *[Fra14]*.

An optimization program that symbolically converts the model to a form that allows computing optimal control functions significantly faster than conventional optimization methods *[WBN16]*.

Such a separation between mathematical model and executable application code is a critical underlying principle for a future modeling and simulation environment for building systems. By following this principle, it is possible to structure the problem more naturally in the way a human thinks, and not how one computes a solution.

5.3.3 Example Model

Fig. 5.7 demonstrates how a simple model can be constructed using components from the Annex 60 library. The model is available in the `Annex60` library as `Annex60.Fluid.Examples.SimpleHouse`. It consists of a simple building, a heating system and a ventilation system, with both systems including a controller. The building model consists of a wall, an air volume and a window. The wall is represented using a simple model consisting of a single heat capacitor and a heat conductor. The zone air is modeled using a control volume with mixed air, which is connected to the wall using a thermal resistor which represent the convective heat transfer. The window is modeled using a component that injects heat onto the wall. Boundary conditions are the outside temperature of the wall and the solar irradiation on the window.

The heating system consists of a radiator, a pump and a heater as illustrated in the bottom of the figure. The heater thermal power is controlled using an on/off controller with hysteresis that tracks a set point for the zone air temperature.

To ventilate or cool the room, a ventilation system is modeled that consists of a heat recovery unit, a fan, a cooling device and a damper. The fan applies a constant pressure difference over the damper. The damper position is modulated by a proportional controller to provide more air if there is a need for cooling.

To build the model, each individual component model was dragged from the component library, placed on the schematic editor window, and connected to other components much like physical components. Model parameters like the pump mass flow rate were set by double-clicking on the respective component, which opened a parameter window. More detailed building models may be implemented using one of the libraries that extend the `Annex60` library and are listed at Section 5.4.2.

5.3.4 Model Translation

The translation of a Modelica model into a simulation program is a fully automated process. When translating a model, symbolic processing is important to reduce computing time since many building system simulation problems lead

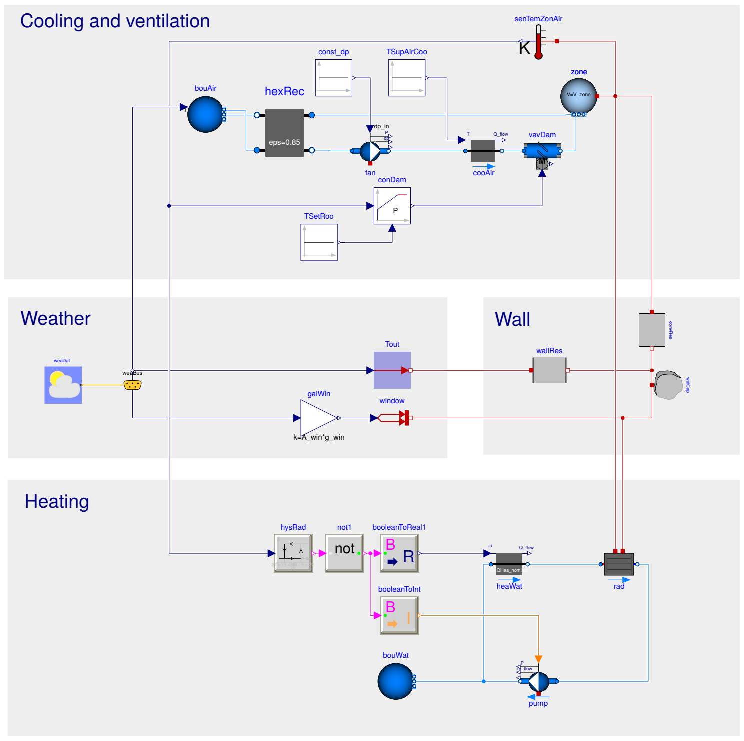


Fig. 5.7: Modelica system model `SimpleHouse` *as rendered by a Modelica graphical model editor.*

Table 5.2: Incidence matrix prior to block lower triangularization.

Equation	$T_1(t)$	$T_2(t)$	$\dot{Q}(t)$
1	x	x	x
2	x		
3	x	x	

to large, sparse differential algebraic equation systems (DAE systems). Symbolic processing is typically used to reduce the index of the DAE system and to exploit sparsity. The following methods are typically used in Modelica simulation environments. To solve a DAE system with ordinary differential equation solvers, the index of the DAE is reduced using the algorithm of Pantelides *[Pan88]*. Next, to exploit the sparsity of the model, Modelica translators generally convert the coupled system of equations to block lower triangular form, by changing the equation order. This process is called block lower triangularization and is discussed in detail by Duff *[DER89]*. A simple example is as follows: Consider the system of equations

$$\begin{aligned} \dot{Q}(t) &= T_1(t)^4 - T_2(t)^4, \\ f_1(t) &= T_1(t)^4, \\ f_2(t) &= T_1(t)^4 + T_2(t)^4, \end{aligned} \tag{5.3}$$

where $f_1, f_2 : \mathbb{R} \to \mathbb{R}$ are known functions of time and $T_1, T_2, \dot{Q}$ are unknowns. A naive implementation would solve directly a three-dimensional non-linear system of equations using a Newton solver. However, we can consider the incidence matrix whose columns are the independent variables and rows are the equations, as shown on the left of Table 5.2. In the incidence matrix, cell (i,j) contains an x if the independent variable i depends on the equation j.

For this system, we can permute the rows so that all cells with an x are below the diagonal, as shown in Table 5.3.

This shows that by rearranging the equations, we can solve sequentially

$$\begin{aligned} f_1(t) &= T_1(t)^4, \\ f_2(t) &= T_1(t)^4 + T_2(t)^4, \\ \dot{Q}(t) &= T_1(t)^4 - T_2(t)^4. \end{aligned} \tag{5.4}$$

Table 5.3: *Incidence matrix after block lower triangularization.*

Equation	$T_1(t)$	$T_2(t)$	$\dot{Q}(t)$
2	x		
3	x	x	
1	x	x	x

Finally, using computer algebra, these equations can be converted into the explicit assignment

$$\begin{aligned} T_1(t) &= (f_1(t))^{1/4}, \\ T_2(t) &= (f_2(t) - T_1(t)^4)^{1/4} \\ \dot{Q}(t) &= T_1(t)^4 - T_2(t)^4, \end{aligned} \tag{5.5}$$

and hence no iterative solution is required.

After block lower triangularization, there may still be sets of equations that require a coupled iterative solution. In this situation, tearing can be used to break the dependency graph of equations and variables to further reduce the dimensionality of the coupled system of equations. The resulting coupled systems of equations are typically small but dense. Tearing can be done automatically or it can be guided by using physical insight with language constructs that can be embedded in a model library [EO94]. A simple example of tearing is as follows: Suppose we have an equation of the form

$$0 = f(x), \tag{5.6}$$

for some $x \in \mathbb{R}^n$, with $n > 1$, that can be written in the form

$$\begin{aligned} L\,x_1 &= \hat{f}_1(x_2), \\ 0 &= \hat{f}_2(x_1, x_2), \end{aligned} \tag{5.7}$$

where L is a lower triangular matrix with constant non-zero diagonals. If this is the case, we can pick a guess value for x_2, solve the first equation for x_1, compute a new value for x_2 from the second equation and iterate until x_2 converges. As $\hat{f}_1(\cdot)$ has fewer equations than $f(\cdot)$, this often results in faster computation as the computing time of many numerical algorithms grows proportional to n^3.

Fig. 5.8 from the Dymola 2016 user manual shows for a mechanical model with kinematic loops that consist of $1,200$ equations how symbolic processing significantly reduces the dimension of the coupled systems of equations.

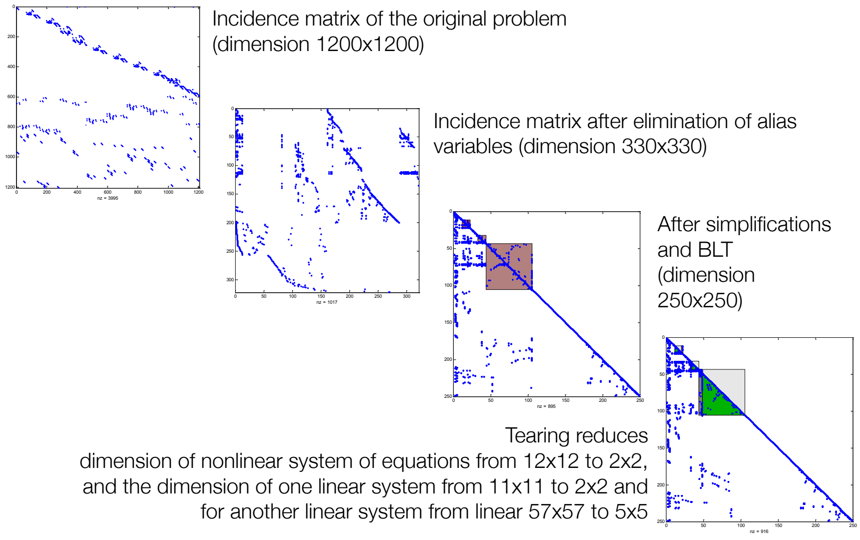


Fig. 5.8: Reduction of the dimension of the system of equations for the case of a mechanical model during the symbolic processing.

Finally, a further reduction in computation time can be obtained by symbolically inserting the discretization formula that represents the numerical integration algorithm into the differential-algebraic equation model, a process called inline integration that was introduced by Elmqvist et al. *[EOC95]*.

5.3.5 Use of External Code

While the above model translation is powerful, there are situations where a human programmer can generate more efficient code. In the buildings domain, a typical example is the solution for partial differential equations for computational fluid dynamics. Partial differential equations often lead to repeating structures, such as a band-diagonal matrix, for which efficient solution algorithms or parallelization methods can be devised or that can simulate faster when parts of the computation is done on a graphical processing unit, see for example *[ZC10]*. For such situations, the Modelica language allows for the calling of C or FORTRAN 77 code, as well as compiled code. Using C code is for example used in the Buildings library to connect Modelica with Python, and compiled code is used to link code for computational fluid flow dynamics for indoor air simulation to the building model in *[ZWT+16]*.

5.3.6 Debugging

When talking about debugging, users typically mean instruction-by-instruction execution of a program code. This makes sense for imperative programming languages such as C/C++, or for signal flow diagrams such as in Simulink, or in actor-based modeling such as in Ptolemy II. However, Modelica, except for the body of Modelica `functions`, has no notion for line-by-line execution. The equations are declarative. A tool can rearrange equations, invert them by changing variables on the left- and right-hand side or by doing other computer algebra before it generates code. Hence, the process of "simulating" a Modelica model is fundamentally different from simulating a model that is expressed in code such as C/C++, Java or Python. Consequently, debugging needs to be approached differently.

To explain how to debug declarative Modelica models, let us first stipulate that the code is translated correctly from Modelica to an executable language such as C/C++, but that the trajectories of the simulation are unexpected. Therefore, one debugs to verify what equations, not what variable assignment, may be wrong. This is often easiest by breaking up large models into small component models and test them individually. Through this process, wrong equations are often easy to detect by looking at the results of a simulation and comparing them to an analytical solution. This is also the reason why good model development should always include the implementation of small unit tests, preferably for different input signals for which the correct solution is known a-priori. Then, assembling models to form larger systems seldom introduces an error due to Modelica's implementation of physical connectors and its requirements of models to be balanced, as described in the Section 5.3.1.1.

Debugging can also consist of understanding why a system model has large linear or nonlinear algebraic loops. In this situation, Modelica translators typically allow for the inspection of intermediate formats that are human-readable model representations after the symbolic manipulation but before generating C code. For example, an intermediate format can show which equations relate the inputs to the outputs and which equations require use of a linear solver or use of a nonlinear iterative solver.

Another purpose of debugging may be to reduce computing time. For this purpose, some simulation environments allow for running diagnostics on the model to see in what equations most of the computing time is spent, what variables dominate the error and the time step size control, and what logical tests produce events that can increase the computing time.

Furthermore, tools typically have intermediate formats and/or debuggers that show what Modelica variable appears in what part of the generated code.

5.4 Annex 60 Library

This section describes the open-source, free Modelica `Annex60` library that has been developed since 2012 and is now used as the core of the Modelica libraries of RWTH Aachen, from LBNL Berkeley, CA, UdK Berlin and KU Leuven.

The goal of the development of the `Annex60` library is to create an open-source, freely available, well documented, validated and verified Modelica library that serves as the core of other Modelica libraries for building and district energy systems, and for whole building simulation programs. Hence, the library should set a foundation for the development of models and whole model libraries that serves the building simulation community over the next decades.

This effort is also expected to support realizing various propositions of Joe Clarke's position paper that he wrote on behalf of the IBPSA Board *[Cla15]*. That position paper calls for a consolidation of models for HVAC and controls that can be used for testing, as a review framework and as a library (Propositions 1, 3, 4, 5, 6, 7, 9, 11 and 12). The stated opportunity is

1. to standardize the approach for how such component and system models are represented, both as data-model and as mathematical models

that formalize the physics, dynamics and control algorithms,
2. to agree upon the physics that should be included in such components for specific use cases, and
3. to share resources for development, validation and distribution of such component and system models

Specifically, proposition 6 states:

> IBPSA will encourage manufacturers to provide more fundamental descriptions of components and make these available within a standard library.

We believe that the `Annex60` Modelica library could serve as the basis of such a standard library.

5.4.1 Approach

The approach of the library distribution is modeled after Linux, which has a kernel that is used by different distributions (e.g., Ubuntu, RedHat etc.), which then provide installation packages, user support and detailed documentation. In a similar fashion, the `Annex60` library provides reliable base classes for building and HVAC component models. Developers of the different model libraries then integrate this library into their Modelica library, add additional models, provide documentation and user support. Through this process, the different model libraries of participating institutions can be further developed, each with their specific focus, while compatibility among the libraries is ensured by the use of the common base classes from the `Annex60` library. In addition to the advantages of having a reliable and well-tested common foundation for model development, further benefits include increased compatibility, exchange and collaboration as opposed to the previously fragmented development of mutually incompatible libraries.

5.4.2 Libraries That Use the Annex 60 Library as Their Core

At the start of Annex 60, four institutes developed their own library in a manner that made models among these libraries incompatible. This led to duplicative effort in model development and validation, and limited the scope of each libray. Within Annex 60, the following institutes worked together, which resulted in their libraries to be all based on the `Annex60` library, thereby allowing users to combine models from these libraries, and reducing development effort:

- RWTH Aachen, Germany, which develops `AixLib`, available from https://github.com/RWTH-EBC/AixLib
- UdK Berlin, Germany„ which develops `BuildingsSystems`, available from http://modelica-buildingsystems.de
- Lawrence Berkeley National Laboratory, Berkeley, CA, USA, which develops `Buildings`, available from http://simulationresearch.lbl.gov/modelica
- KU Leuven, Belgium, which develops `IDEAS`, available from https://github.com/open-ideas/IDEAS

5.4.3 Functional Requirements

The `Annex60` library needs to fulfill several functional requirements that originate from typical use cases in building performance simulation. The use cases include design and analysis of warm water heating and distribution systems, cooling systems, ventilation systems, heat demand calculations of multiple buildings, controls design, co-simulation and export of models for use in other simulators or in building automation systems. The use cases aim at optimal design and operation of thermal energy systems as well as district energy systems and model use during operation. Thus, the `Annex60` library focuses on annual whole building simulation, on single as well as on multiple buildings.

Regarding physical resolution, various effects are modeled to ensure suitability for typical use cases. For example, all mass flow rates are pressure driven to allow simulations of duct and piping networks. Optionally, all equations for pressure drop calculations can be removed from the models through a parameter assignment. Transport delays can be added to fluid flow networks. The media models are compatible with models from `Modelica.Media` to ensure compatibility with other libraries. For air, water vapor and trace substances such as CO_2 or VOC concentrations are tracked as to account for moisture transfer and for simulation of indoor air quality. For cooling devices, no detailed modeling of the state and distribution of refrigerants is conducted as this would lead to computing time that is too large for annual building simulation.[1]

Regarding dynamic system behavior, models with different idealizations are implemented. Where applicable, it is possible to approximate the dynamic response of models using a first or higher order response, and optionally disable the model dynamics to conduct a quasi steady-state simulation. For example, the dynamics of sensors can be approximated by a first order response. This can be disabled to obtain a steady-state sensor, or to add a more detailed sensor response to the output signal of the sensor.

It is also possible to export as a Functional Mockup Unit (FMU) thermofluid components, as described in *[WFN15]*. All models use SI units.

5.4.4 Mathematical Requirements

Component models from this library are typically assembled to form system models that lead to systems of ordinary differential equations, which are coupled to algebraic systems of linear, nonlinear and discrete equations. To ensure that these systems of equations can be solved efficiently, they need to satisfy certain mathematical properties. In this section, we describe the main requirements that have been followed during the development and are satisfied by the `Annex60` library.

For equations that describe the physics, the following mathematical properties need to be satisfied:

1. Equations must be differentiable and have a continuous first derivative that is bounded on compact sets, i.e., they need to be once continuously differentiable on compact sets. This is not only needed for numerical efficiency, but also to establish existence of a unique solution to the system of differential equations *[CL55]*, and it provides a key requirement for the solution of optimal control problems in which the cost function is defined on numerical approximations to the solutions of the differential equations *[Pol97][PW06]*. For example, instead of using $y(x) = sign(x)\sqrt{|x|}$, this equation needs to be approximated by a differentiable function that has a finite derivative near zero because

[1] For such applications, the AC library from Modelon or the TIL library from TLK-Thermo GmbH may be used.

$\lim_{x \to 0} y'(x) = \lim_{x \to 0} 1/(2\sqrt{|x|}) = \infty$. Otherwise, Newton-Raphson solvers may fail if x is close to zero, as the Newton step length is proportional to the inverse of the derivative of the residual function.

2. Equations where the first derivative with respect to another variable is zero must be avoided. For example, let $x, y \in \mathbb{R}$ and $x = f(y)$. An equation such as $y = 0$ for $x < 0$ and $y = x^2$ is not allowed. The reason is that if a simulator tries to solve $0 = f(x)$, then any value of $x \leq 0$ is a solution, which can cause instability in the solver. An example of such a situation is a valve. If it were to have no leakage flow, then any value of the pressure drop would cause zero mass flow rate, which may lead to ill-conditioned equations for some flow networks. More formally, the conditions for the Implicit Function Theorem [Pol97] need to be satisfied as this guarantees existence and differentiability of an inverse of the function.
3. Equations that cause divisions by zero should be avoided.

Clearly, models for controls require discrete variables that can only take on certain values, such as for switching equipment on or off. This certainly is allowed, but must be implemented using a hysteresis to avoid chattering. For example, an equation such as $y = 0$ if $T > 20°\text{C}$ and $y = 1$ otherwise is not allowed as this can lead to chattering in continuous time solvers.

5.4.5 Process for Quality Control

The `Annex60` library is used as a foundation on which different libraries for end users are developed. It therefore needs to provide reliable models and results. Thus, a strict process for quality control was implemented from the start of the library development. This process includes open source code development in a version control system on GitHub, automated regression testing of the entire library, and a review process before authorizing any code changes. This process is supervised by a core development team. However, contributions from the community are encouraged, given that they meet the quality standards and requirements of the library.

Using git for version control of the source code and documentation of the library allows for keeping a record of all development stages and changes of the library. Stable models are kept in the so-called master branch. Additions and code changes are restricted to dedicated branches for feature development that are documented in a corresponding issue tracker. In order to prevent any unintended effects of code changes to existing models, the translation statistics and reference results for each model are automatically created, stored, and managed by using the Python package BuildingsPy (https://github.com/lbl-srg/BuildingsPy). This package includes functions for unit testing. The implemented process for quality control requires contributors to provide a scripted test for every model, so that these tests can be run automatically for the whole library. When introducing a model and its test, the simulation results are saved by the unit testing routine. The unit testing is run before accepting any changes to the master branch. Differences in the translation statistics and simulation results between reference results and the tested model are automatically plotted and need to be accepted or rejected manually, thus guarding against introducing unwanted effects on any of the existing models.

Once code changes have been documented and evaluated by unit testing, they need to be reviewed by a second developer. Thus, the author of the changes issues a pull request to suggest moving the new code into the master branch. Only after all comments from the reviewer have been addressed by the original author of the changes is the new code merged to the master branch. At the time of writing, this process has been applied to over 400 issues and 1000 changes to the code base. Finally, the underlying version control system can serve as a safety net if changes need to be reverted despite the described quality control process. As each step of code change as well as the responsible author is documented at all times, such corrections require little effort.

5.4.6 Requirements for Adding Classes

The `Annex60` library aims at following a coherent set of conventions and requirements to ease maintenance and further development. The library follows and extends the conventions of the Modelica Standard library. In addition, the names of *models*, *blocks* and *packages* must start with an upper-case and be a combination of adjectives and nouns using camel-case to combine multiple words, whereas instances of these models, blocks and packages start with a lower case letter. The names of each Modelica `function` must start with a lower-case character. Instance names are usually a character, such as `T` for temperature and `p` for pressure, or a combination of the first three characters of a word, such as `TSet` for temperature setpoint or `higPreSetPoi` for high pressure set point.

New components of fluid flow systems must extend the partial classes defined in the package `Annex60.Fluid.Interfaces`. If the new class is partial or if it is not intended for the end-user, it must be added to a `BaseClasses` package. Each new model must be used in at least one model contained in the `Examples` or `Validation` package, which illustrates or validates its behavior and is included in the unit tests.

Each class needs to be documented. Every variable and parameter must have a documentation string. The `documentation` section of the model provides a more comprehensive documentation describing the main equations, model assumptions and limitations, typical use, options, validation, implementation and references. Some of these sections are optional as they are not applicable to all models. Each class also contains a list of revisions made by the developers to keep track of the changes and their rationale.

The robustness of the models is also a key requirement, and the guidance explained in Section 5.4.4 needs to be followed. Fluid flow models must be stable near zero mass flow rate even under the assumption that flow rates or heat input are approximate solutions obtained using an iterative solver. Fluid flow models must be well-behaved if the mass flow reverses direction.[2] Furthermore, fluid flow models use the `Annex60.Media` models, which have physical constraints such as the valid temperature range or relative humidity bounds outside which they are not valid. These bounds need to be taken into account by the models in order to avoid non-physical situations or convergence problems. Parameters and variables should whenever possible use units from `Modelica.SIunits` and they should declare bounds for minimal and maximal values. Default values for parameters, which can be adapted by the end-user, should be declared using the start attribute so that the user gets a warning if no other value has been provided.

[2] By well-behaved, we do not mean, for example, that a performance-curve based model of a direct expansion cooling coil computes the right physical results if there is a slight backward flow, but rather that the model is robust. For example, it suffices to add no energy to the air if there is slight backward flow.

Each model must be validated using either measurement data, cross validation with other simulators or with analytical solutions. Validation models need to be added to the `Validation` package and be included in the unit tests.

Finally, new models should be in line with the scope of the library as described in the functional requirements.

5.4.7 Design Decisions

This section describes the main design decisions for the `Annex60` library.

5.4.7.1 Media

In Modelica, component models typically call functions to obtain thermodynamic properties such as the specific enthalpy.

We experimented with two implementations. One approach called `MediaFunctions` was using functions for the thermodynamic properties of a medium, with an enumeration as a function argument that declares the medium type such as air or water. The other approach called `MediaPackages` was using a separate package for each medium type, as is done in `Modelica.Media`.

The main differences between these two implementations are as follows:

1. For `MediaFunctions`, models of HVAC equipment require parameters for the medium type and for the default values of pressure, temperature, mass concentration and trace substances. The medium type needs to be propagated to the functions that compute the thermodynamic properties.
2. For `MediaPackages`, there is one package for each media type, as in `Modelica.Media`. Models of HVAC equipment contain a replaceable parameter for the medium package that needs to be set to medium type. Prior to the model translation, the medium type such as water, air or glycol must be declared, but its concentration can be changed after translation.

Based on these implementations, we decided to organize media in packages as is done in `Modelica.Media`. The benefits are:

1. Full compatibility with `Modelica.Media`.
2. Default values for the medium, such as the default pressure and mass concentration, can be propagated through its declaration because packages can contain constants.
3. Modelica translators can verify that connected fluid ports contain the same medium, and refuse to translate the model if this is not satisfied.

The advantage of using `MediaFunctions` would be that only two sets of FMUs need to be generated, one for the single species medium water, and one for the dual-species media such as air and glycol. However, the cost of separately compiling FMUs for air and for glycol is small as this can be fully automated and be done as part of a simulation engine development.

While the implementation of `Annex60.Media` is compatible with `Modelica.Media`, we generally use simpler implementations. For example, in `Annex60.Media.Air`, water is only present in vapor form even if the water vapor pressure is above the saturation pressure. This allows for the computation of temperature from specific enthalpy and mass concentration without requiring iterations, thereby leading to more efficient models.

5.4.7.2 Fluid Connectors

We decided to use the same fluid connectors as defined in `Modelica.Fluid`. These connectors declare the medium package (used to assert that only models with the same medium are connected), the mass flow rate as the conserved quantity, the absolute pressure as the potential variable, and the following variables that are carried by the mass flow: the specific enthalpy, optionally the mass fraction such as to declare the mass concentration for water vapor in air, and optionally trace substances. Trace substances are concentrations that can be neglected for the thermodynamic calculations such as CO_2 or VOC concentrations.

We also explored using temperature instead of specific enthalpy in the connector. Modelica's concept of `flow` and `stream` variables allows connecting multiple fluid ports. It then automatically generates the conservation equation

$$\chi_{mix} = \frac{\sum_i \dot{m}_i^+ \chi_i}{\sum_i \dot{m}_i^+}, \tag{5.8}$$

where χ is the conserved quantity, $\dot{m}_i^+ = \max(0, \dot{m})$ is the mass flow rate if it is positive and the subscripts mix stands for the mixture and i for the connected ports. If $\chi = h$ is the specific enthalpy, then h_{mix} is always correct. If $\chi = T$ were temperature, then T_{mix} is wrong if the specific heat capacity c_p depends on temperature. We therefore selected the specific enthalpy to be in the fluid connector, as is also used in `Modelica.Fluid`.

5.4.7.3 Package Structure

In Modelica, classes can be collected and organized hierarchically into packages. This section describes how the Annex 60 library organizes classes into the following main packages:

`BoundaryConditions` contains models for weather data reader, solar radiation and sky temperature.

`Controls` contains models for continuous time and discrete time controllers and for set point scheduling.

`Fluid` is the main package that contains thermofluid flow components such as heat exchangers, pumps, valves, air dampers and boilers. We considered introducing subpackages `Fluid.{Air,Water,Glycol}` but have not done so yet as this would lead to duplication of code and documentation. Consequently, users always need to assign the media.

`Utilities` contains the major packages `Psychrometrics`, which implement blocks and functions for psychrometric properties, and `Math`, which provides blocks and functions that are once continuously differentiable approximations to functions such as $\min : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $\text{abs} : \mathbb{R} \to \mathbb{R}$, the Heaviside step function or cubic spline interpolation. The functions in the `Math` package are used to satisfy the earlier discussed differentiability requirements.

Major packages contain user guides, and all packages contain an `Examples` or a `Validation` package that demonstrates how to use the models, and that are used for validation and regression testing.

5.4.8 Main Classes of the Library

The `Annex60` library provides models for simulating HVAC systems in buildings, containing both hydronic and air flow models. These systems combine different types of components such as energy and mass transfer models, media models for air and water, control models and supporting utilities. These types are grouped in separate packages. This section gives the highlights of these packages.

5.4.8.1 Fluid Component Models

A typical HVAC system contains components such as valves, pumps, dampers and heat exchangers. Essentially these are all components that transfer mass and/or energy. These models are therefore grouped in the `Fluid` package. The most important parts of this package are now discussed.

Conservation of Energy and Mass Since all of these models need to conserve mass and energy, the `Fluid` package heavily relies on the model `ConservationEquation` that implements these conservation laws. The model is implemented in a generic way such that it can be used for different types of media: compressible and incompressible and media with or without moisture or trace substances. The energy and mass conservation laws can be configured to be a steady state or dynamic balance. The `ConservationEquation` model is instantiated by the `MixingVolume` model, which is used by most equipment models.

Flow Networks Since the physics of fans and pumps is similar, they are implemented using the same models in the `Movers` subpackage. This package contains four mover models, each using a different control signal:

`FlowControlled_m_flow` This model directly sets the mass flow rate, independent of the head pressure that results from the duct or pipe network simulation.

`FlowControlled_dp` This model directly sets a pressure head, independent of the mass flow rate that results from the duct or pipe network simulation.

`SpeedControlled_Nrpm` This model sets the speed. Mass flow rate and pressures are calculated from similarity laws, from a user-provided pump curve and the duct or pipe network simulation.

`SpeedControlled_y` This model is identical to `SpeedControlled_Nrpm`, except that the input signal is normalized by the nominal speed.

Various types of flow resistance are implemented. For example, `FixedResistanceDpM` implements a pressure curve according to $K_v = \dot{m}_{nom}/\sqrt{\Delta p_{nom}} = \dot{m}/\sqrt{\Delta p}$, where $\dot{m}_{nom}$ is the nominal mass flow rate, Δp_{nom} is the nominal pressure drop, and $\dot{m}$ and Δp are the actual mass flow rate and pressure drop. In a neighborhood around zero, this function is regularized. In valves and dampers, the K_v value is a function of the actuator input y. The library implements models in the `Actuators` subpackage where the relation between K_v and y is expressed using a linear, quick opening or an equal percentage characteristic. Custom opening characteristics using a table can also be used, as well as a pressure independent characteristic. Various dampers, two-way and three-way valves of these types are implemented.

Energy Transfer The library provides the model `HeaterCooler_T`, which is a heater or cooler that maintains an outlet temperature, subject to optional capacity limits. The outlet temperature is obtained from an input signal. The model `HeaterCooler_u` is a variant of this model that takes as an input a control signal that is proportional to the heat to be added to or removed from the fluid. A heat exchanger with constant effectiveness can be used to transfer energy between different fluid streams. A similar model exists for a mass exchanger that transfers moisture. The model `RadiatorEN442_2` implements a radiator model based on the EN 442-2 norm.

Other Models Various models are available for integrating fluid components in larger models. Models from the `Sensors` package can be used for integrating control components and for performing analysis. The `Sources` subpackage implements components for enforcing boundary conditions on fluid ports. The user can configure the model to obtain the mass flow rate or pressure from a parameter or from an input signal.

5.4.8.2 Media

The library contains media implementations for water and air. `Water` is considered to be incompressible with constant thermal properties. `Air` contains moisture, has a pressure-dependent density and constant thermal properties. More detailed implementations are available in the subpackages `Specialized.{Air,Water}`.

5.4.8.3 Control Models

The `Controls` package contains basic control components such as PID controllers and blocks for set point resets. Our intention is to expand this package to provide control blocks and template control sequences that are commonly used in building control systems.

5.4.8.4 Utilities

The `Utilities` package contains models that simplify the consistent implementation of other components. The `Psychrometrics` subpackage, for example, contains functions and models for relating the vapor fraction and partial pressure to the humidity and wet bulb temperature. The `Math` subpackage contains commonly needed once continuously differentiable approximations to mathematical functions.

5.4.9 Merging with other Libraries

To automatically merge the `Annex60` library with libraries that are based on it, such as the libraries `AixLib`, `Buildings`, `BuildingSystems` and `IDEAS`, developers of these libraries use the Python package `BuildingsPy` and execute the following commands:

```python
import buildingspy.development.merger as m
src="/home/joe/modelica-annex60/Annex60"
des="/home/joe/modelica-buildings/Buildings"
mer=m.Annex60(src, des)
mer.merge()
```

These commands copy all `Annex60` library models from the directory `/home/joe/modelica-annex60/Annex60` into the library `Buildings` in the directory `/home/joe/modelica-buildings/Buildings`. The `merge()` command updates all hyperlinks, references to package names and file names that contain the `Annex60` string. Therefore, users will only see the respective library and do not have to combine models from different libraries.

5.5 Getting Started

To get started with Modelica, we suggest the following literature for those who want to use Modelica or want to develop Modelica models.

5.5.1 Literature for Users

The following books are useful for new users to get started:

- The online book with interactive examples of Michael Tiller at http://book.xogeny.com/.
- The books by Michael Tiller *[Til01]* and Peter Fritzson *[Fri11][Fri15]*.
- The tutorials that are listed at https://www.modelica.org/publications.

Although the Modelica Language Tutorial at https://www.modelica.org/documents/ModelicaTutorial14.pdf is for an older version (Modelica 1.4), it is still instructive and relevant to understand the concepts of the language.

The `Annex60` library as well as the libraries that use it as their core contains hundreds of examples of varying complexity that can be simulated and modified to get accustomed to using Modelica.

5.5.2 Literature for Developers

For users who develop new thermofluid models, it is essential to understand the concept of stream connectors. Stream connectors are explained in the Modelica language definition, available at https://www.modelica.org/documents, and in *[FCO+09a]*. Furthermore, developers should have a proper understanding of the implications of flow reversal and know when what types of algebraic loops are generated. This is discussed in *[JWH15]*.

The `Annex60` library uses similar modeling principles, as the `Modelica.Fluid` library. Hence, we also recommend reading the paper about the standardization of thermofluid models in `Modelica.Fluid` as described in *[FCO+09b]*.

Xogeny's Modelica Web Reference at http://modref.xogeny.com/ gives a concise overview, explanation and further links about the Modelica language.

5.6 Conclusions

Using Modelica allowed using and contributing to an open standard for model representation, which is supported by a large ecosystem of tools, has an active development and user community, and enjoys significant investment from various industrial sectors. Using a multi-disciplinary modeling language allowed for interdisciplinary collaboration among communities in engineering, applied mathematics and computing. Collaboration between these research communities that have deeper knowledge of their respective fields has shown to be critical in addressing increasingly challenging demands on computational tools that support the design and operation of high performance buildings and communities.

At the first expert meeting of the planning phase of Annex 60, in March 2013 at RWTH Aachen, Germany, participants were hesitant to open up their proprietary development, open-source their code and embark on a joint development of an open-source library that should become the core of their respective libraries. However, as the collaborations slowly took shape, participants saw the value in avoiding duplicative work, in conducting collaborative research and in jointly developing a core library. Hence, participants ended up investing a considerable amount of work in scrutinizing different implementations, refactoring their respective libraries and sharing previously proprietary code. As a result, the four major Modelica libraries for building and district energy systems now all share the same set of core models, they became more robust, better validated and compatible with each other. With this shared development, Annex 60 created a robust, open-source basis for a model library for the buildings performance simulation community. As of this writing, the further development of this code is being transferred from the umbrella of the IEA EBC Programme to the International Building Performance Simulation Association IBPSA.

Chapter 6

Activity 1.2: Co-simulation and Model Exchange Using the FMI Standard

6.1 Introduction

6.1.1 Motivation for Coupled Simulation

Currently, there is a significant number of sophisticated simulation models and tools available for use in building energy simulation. Models and parametrization are provided by different specialists, with some models being the outcome of many years of development. Hence, different programming languages and modeling approaches are used.

Often, a tool misses functionality provided by others. For many applications it is desirable to use several simulation tools concurrently and make use of results generated by the different models. Consequently, tools running transient simulations often need results generated by other models at runtime. For example, a building energy simulation model computing room air temperatures may require heating loads from an HVAC supply system, with the latter coming from a simulation model external to the building simulation tool.

Pure integration of several existing models into a single simulation tool is technically very difficult, in particular since many tools include their own numerical solver engines and calculation algorithms. Moreover, different simulation tools manage their internal data differently, which further complicates their integration. Also, given different authors, domain experts, copyright limitations, etc. an integration into a single tool may not be meaningful or possible at all. Lastly, future development and maintenance of such an integral model may be difficult to organize. It is also difficult to ensure lasting financial support for development and maintenance of a complex integral simulation tool authored by many individual developers.

Translating existing models into modular equation-based languages such as Modelica could be considered as an alternative, and could benefit from modern Modelica features. Modular, hierarchical design could be used and Modelica tools would be able to optimize the system of resulting equations for performance. However, this approach may not be economically feasible regarding the expected overhead of translating a large number of existing models, and their documentation. Also, copyright and intellectual property protection may not permit integrating their existing model code.

As an alternative, new and existing code of complex systems can be decomposed into sub-systems, the subsystems simulated with the simulator which suits best the specific domain, and interface variables exchanged as the simulation progresses.

Co-simulation is a technique in which simulators are executed simultaneously while exchanging data during run-time. In this context we refer to *simulators*, also called slaves, as individual component models described by differential algebraic or discrete equations.

The simulators typically embed systems with differential-algebraic equations. Each simulator may implement different numerical techniques tailored to a specific physical or mathematical problem. This includes the possibility that through usage of special programming language features, memory layout, parallelization or other specific optimization, the specialized implementation for model equations may perform better than an equivalent model in a generic simulation environment.

6.1.2 Need for a Tool Coupling Standard

When considering coupled simulation of different tools, several options are available:

1. When access to source code is possible, or tool developers can work actively together, tools may *interface directly*. Simulation tool developers may even incorporate calculation capability from another tool into their own code. Such a procedure depends on many prerequisites, such as compatible programming languages, proper license agreements, data and calculation structure compatibility and sufficient developer resources. In many cases the resulting code can be considered as a new integrated tool and not really a coupled tool.
2. In the *one-to-one approach*, two tools implement a specific protocol that regulates exchange of data during runtime. This approach is the most flexible and can support any kind of numerical solution method and also take into account peculiarities of the tools involved. However, the implementation may need to be continuously updated when one of the tools evolves. Also, if tools are developed independently, testing and checking of correct functionality may become time-consuming. Furthermore, interfacing with a third tool typically requires reformulation or adaptation of the protocol and effectively implementing another coupling mechanism tailored to the new tool. Lastly, the development of a clear application programming interface (API) specification and the semantics of the tool coupling is difficult, and subsequent revisions to this API may incur large development costs.
3. To overcome the need for, potentially many, one-to-one exchange protocol implementations a *middleware* could be used. This acts as a hub for several tools that interface with it and relays communicated data to other connected tools. This approach is less flexible than the one-to-one approach, since a common ground on interface capability needs to be agreed upon. When interfacing domain-specific tools, the middleware can be fitted to the specifics of the problem types and applied numerical methods. Tools need to implement only a single interface and communication protocol and only need to revise code when the middleware is upgraded. However, the same problem regarding the API specification as for the *one-to-one approach* remains.

In the past, many tool coupling technologies have been developed and applied, with different mathematical concepts and programming interfaces. Many have been tool-specific proprietary interfaces that may change frequently as the original tool developers see fit. Further, the details of interfaces are often restricted to certain applications and lack functionality needed in other physical domains. This is particularly true for domain-specific middleware approaches. In addition, closed-source middleware implementation details may change from version to version, thus potentially render previously connected tools incompatible.

To overcome these limitations it was necessary to develop an open-source and community driven standard for coupled simulation. In 2008 the Modelisar project started to develop such a standard, named the Functional Mockup Interface (FMI, https://www.fmi-standard.org/). Finally, with the initial 1.0 release, a standard for interfacing simulation tools has become available. See *[BOA+11]* for an introductory paper and https://www.fmi-standard.org/literature for various literature about FMI.

6.1.3 Overview of the FMI Standard

The FMI standard describes the mathematical concept for coupled simulation of several simulators, called *Functional Mockup Units* (FMU). The FMI standard defines the following:

1. A set of C-functions (FMI functions) that allow exchange of data between FMUs. The standard defines the signature of these function, their required behavior, e.g., the semantics, and it also prescribes during what state of the simulation these functions can or must be called.
2. An XML scheme that is used by each FMU to publish meta-data of the model and the simulator. These data are stored in a file called `modelDescription.xml`. It contains meta-data such as the number and type of variables to be exchanged and integrated, the capabilities of the simulator, and the mathematical structure the FMU.
3. A zip file with extension `fmu` that is used to package the XML file, the compiled binaries with the implemented FMI functions, optionally the original source code, as well as resources (databases, project parameters, etc.) required to execute the model.

The FMI standard allows two modes of encapsulating a model: model-exchange and co-simulation. The next sections describe these modes.

6.1.4 Mathematical Aspects of Model Exchange and Co-Simulation

With respect to solving the coupled system of equations arising from all FMUs within one *simulation system*, two different approaches are standardized. The first, called *model exchange*, requires a central time integration algorithm that solves collectively the system of differential-algebraic equations defined within the FMUs.

The second approach, called *co-simulation*, defines a methodology for inter-connecting different encapsulated models and numerical solution methods. The individual FMUs incorporate their own integration method that may be tailored towards the physical problem to solve. Also, existing simulator engines can be re-used by implementing a co-simulation interface. The *co-simulation* master algorithm needs to ensure agreement of all exchanged variables, i.e. that variables computed by one FMU and used by another are correctly synchronized among the tools.

The next sections describe FMI for model exchange and co-simulation, then describe how to synchronize FMUs with a master algorithm, and finally how to generate or export FMUs.

6.1.5 Functional Mockup Interface for Model Exchange

The mathematical concept behind FMI for Model Exchange is based on the idea that a model can be formulated as a function whose output depends on parameters, input variables, state variables and time. Parameters are values that do not change with time. They are assigned at the start of the simulation. Inputs are values that can change with time.

Consider the example illustrated in Fig. 6.1.

The corresponding Modelica code illustrates the functionality of the model:

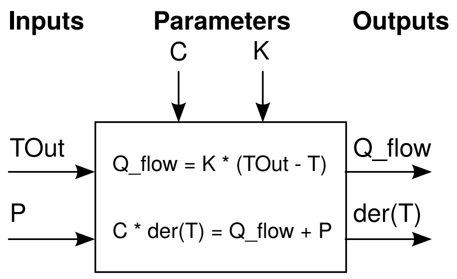


Fig. 6.1: *Simple room model with air change model and externally defined heat source.*

```
model RoomModel
  parameter Real C "Room heat capacity in J/K";
  parameter Real K "Heat loss coefficient in W/K";
  input Real TOut "Outdoor air temperature in K";
  input Real P "Heat added to the room in W";
  output Real Q_flow "Heat exchange with the ambient in W";
  Real T(start=293.15) "Room air temperature in K";
equation
  Q_flow = K * ( TOut - T);
  C * der(T) = Q_flow + P;
end RoomModel;
```

Suppose now a user wants to incorporate this model into a Model Exchange FMU. The inputs to the model are $u = (T_{out}, P)$, the state is T, the state derivative is dT/dt, and the requested output is the heat exchange with the ambient $\dot{Q} = K\,(T_{out} - T)$. The actual calculation code looks like:

```
// We have cached the state variable T and
// inputs TOut and P.

// Compute the output.
Q_flow = K * ( TOut - T);
// Compute the time derivative.
dT_dt = (Q_flow + P) / C;
```

The implementation of the physics of such a model is fairly simple. The model obtains input, caches the state variable and computes the output and time derivative. The time integration will be managed by the simulation master and is not a task of the model. Therefore, no solver to integrate the ordinary differential equation through time needs to be implemented. However, model developers must bear in mind that there is no notion of time step size. Also, the model implementation must not make the assumption that time is always increasing, since the simulation master may request the model to be evaluated several times at the same time, or even requested to go back to a past time instant. See Section 6.1.7 for more detail.

The FMI standard defines a state-machine approach for evaluation of the functions that set inputs, update states, compute outputs, etc. This means, that the model is not evaluated through a single function call passing the complete set of input variables and expecting all outputs and states at once. Instead, the model is treated as an *object with a state*.

The state of an FMU is only modified when input variables or the time are changed by invoking the FMI functions. While the FMI standard prescribes when the state of the object and outputs are updated, the model can make use of efficient implementation. For example, in our implementation above, if the input `P` is updated but `TOut` and `T` remains unchanged, then `Q_flow = K * ( TOut - T)` need not be recomputed, saving unnecessary model computation time.

In the model implemented above, all outputs can readily be evaluated without any iteration. However, some FMUs may involve equations that form an algebraic loop. For example, an FMU may implement a hydraulic network which requires an iterative solution to compute pressures and mass flow rate. In this situation, the numerical algorithm to solve such an algebraic loop, typically a Newton-Raphson method, needs to be part of the FMU. Alternatively, the model can be split into two FMUs, and the task of resolving algebraic loops may be delegated to the master algorithm (see Section 6.1.7).

6.1.6 Functional Mockup Interface for Co-Simulation

Co-Simulation describes a calculation method that requires each *simulator* to integrate its own model equations over a time interval requested by the master algorithm, and exchange outputs only at certain time instants called the *communication points*. At these communication points, the master algorithm retrieves output variables, and distributes them to other FMUs that use these outputs as inputs. Once the inputs have been updated, the master requests each FMU to integrate to the next communication point. If a simulator cannot advance time to the next communication point, for example because its error would be too large, or because an event happened inside the simulator, it can report this to the master algorithm, which then may request other FMUs to redo their time integration over a shorter time interval.

Co-simulation slaves are, similar to Model Exchange simulators, state objects. The key difference is that for model-exchange, FMUs output the time derivatives of their continuous-time state variables, whereas for co-simulation, FMUs need to integrated this states themselves and output the new values of the state variables.

If the room model example from the previous section is implemented as an FMU for co-simulation, the time integration needs to be performed manually. In the example code below, a simple explicit Euler Integration is implemented:

```
// We are requested to integrate over the communication
// interval with length h beginning at current time t.

// The FMU state T is stored at time t.
// We have cached the inputs TOut and P at time t
// which are treated constant over the interval t...t+h.

steps = h/dt; // number of integration steps

for (unsigned int i=0; i < steps; ++i) {
  // compute time derivative
  Q_flow = K * ( TOut - T);
  // Update the state
  T = T + dt * ((Q_flow + P) / C);
}
```

Note that the explicit Euler integration algorithm with fixed time step size may be inefficient. During the time integration from one communication time step to the next, the FMU assumes its inputs to be constant, while in actuality, they may vary more frequently through time. Hence, large communication time steps can incur large errors of the co-simulation.

For example, consider again the thermal room model example above. Suppose the heat added to the room `P` is computed by a heating system in another FMU. The heater may be controlled by the room temperature and reduce heating power when the set point temperature is approached. The room model provides the room temperature as output. At the beginning of the communication interval, this temperature is passed as input to the heating model, and being below the set point temperature, a heating power is computed. This heating power is then passed to the room model. During the time integration of the room model, this heating power remains constant, and may eventually lead to a room temperature above the set point temperature, as illustrated in Fig. 6.2[1].

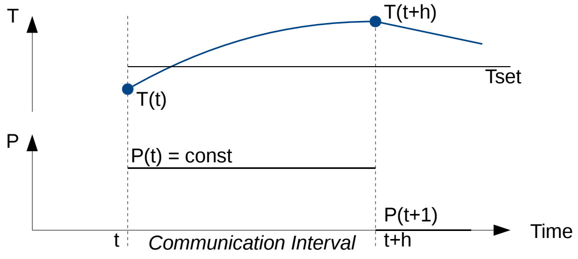


Fig. 6.2: Illustration of room air temperature exceeding the set point temperature due to delayed variable exchange.

While the co-simulation method is a flexible tool coupling approach, it is important to control communication interval sizes and implement suitable algorithms that handle *events* and control the numerical error of the coupled equations.

[1] To allow larger communication time steps, the FMI standard also allows for providing time derivatives of the continuous real inputs.

6.1.7 Master Algorithms

We will now describe simple master algorithms for model exchange and for co-simulation. Master algorithms synchronize the different FMUs, assign inputs to FMUs, and in the case of model exchange, also provide solvers for differential equations and algebraic equations that are formed by connecting FMUs.

Model exchange master algorithms need to solve systems of differential algebraic equations (DAE). Typically, master algorithms provide different choices for such solvers. When solving the DAEs, the FMU is treated as a mathematical function that computes outputs and time derivatives for given time, inputs and states.

We will now use the FMU example model from Section 6.1.5 to illustrate a simple model exchange master algorithm. An instantiated FMU, named simulation *slave*, is evaluated in a time integration loop that is the core of the Model Exchange master algorithm. In the following example, an Explicit Euler integration method is used with a predefined number of integration steps:

```
// ... the FMU slave is instantiated and initialized

// We have stored the initial state of the FMU
// in the variable x.
// The instance m is a pointer to the model.

// Set constant time step sizes based on selected
// steps in the master.
double dt = (t_end - t_start)/steps;

// integrate
for (unsigned int i = 0; i < steps; ++i) {
   // Set new time.
   M_fmi2SetTime(m, t_start + dt * i);
   // Set inputs.
   M_fmi2SetReal(m, ..., &TOut);
   M_fmi2SetReal(m, ..., &P);
   // Set state variables
   M_fmi2SetContinuousStates(m, &T, 1);
   // Retrieve the computed time derivative.
   M_fmi2GetDerivatives(&dT_dt);
```

```
    // Integrate in time.
    T = T + dt * dT_dt;
}
```

In general, a more sophisticated integration algorithm is needed to ensure that the integration error remains below a user-prescribed tolerance, to handle *time events* and *state events*, and to integrate the equations more efficiently. Typically, an adaptive time step method is used that adapts the integration time step based on an estimate of the integration error and events. Furthermore, if *algebraic loops* are present, the master algorithm needs to provide a linear or nonlinear solver. Such algebraic loops are formed if the outputs and inputs of an FMU have *direct dependency*, and the inputs and outputs with direct dependency are connected with each other.

Implicit time integration methods and nonlinear equation solvers typically use a Newton-Raphson algorithm. Since the Newton-Raphson algorithm builds a Jacobian matrix that has the same dimension as the number of variables, this operation can be computationally expensive. The use of sparse Jacobian matrices is helpful, but requires information on dependencies of individual variables on others. The ability for an FMU to publish such dependency information has been added to the FMI standard in version 2.0. The information about dependency also allows usage of efficient asynchronous numerical time integration algorithms such as Quantized State Systems (QSS) methods *[ZL98][Kof03][FK14]*.

Co-simulation master algorithms assign inputs to outputs of other FMUs, assign the requested time step to FMUs, and advance time of the co-simulation. Hence, each FMU is responsible for integrating in time its continuous-time state variables from the current time to the next communication time point.

Next, we illustrate a basic co-simulation master algorithm with a simple example. Suppose, the room model example above was complemented by a controlled heating model which takes the room air temperature as input and computes the required power of the heating unit based on a temperature setpoint. The heating model and the room model both hold a state at a certain time point. Advancing from this time point to the next communication point can be managed by the co-simulation master algorithm as follows:

```
// The FMU slaves are instantiated and initialized.
// s1 is the room model, and s2 is the heater model

// Both FMU slaves are at time t and the master algorithm
// has cached the output variables of both slaves.

t = startTime;
h = communicationTimeStep;
// Loop over all slaves.
while(tc < stopTime &&
      status1 == fmi2OK &&
      status2 == fmi2OK)
  // Retrieve outputs.
  s1_fmi2GetReal(s1, ..., 1, &T);
  s2_fmi2GetReal(s2, ..., 1, &P);
  // Set inputs.
  s1_fmi2SetReal(s1, ..., 1, &P);
  s2_fmi2SetReal(s2, ..., 1, &T);
  // Advance time.
  status1 = s1_fmi2DoStep(s1, t, h, fmi2True);
  status2 = s2_fmi2DoStep(s2, t, h, fmi2True);
  // Increment master time.
  t += h;
}
```

In this example, the master algorithm retrieves outputs, assigns them as inputs to the other FMU, and then requests the FMUs to advance time. Not shown in this example is error handling, as well as handling the situation in which an FMU may reject to advance time to `t+h`.

The choice of the co-simulation master algorithm can have a great impact on overall performance, as well as on the correctness of the result, in particular in the presence of an event. To ensure that the results of co-simulations of hybrid systems are deterministic and, hence, equal regardless of the tool that implements the master algorithm, *[BGL+15]* provides requirements and a test suite with expected results that can be used to verify correctness of master algorithms.

6.1.8 Exporting and Importing of Functional Mockup Units

Simulation programs may generate and export FMUs. During FMU export, the simulation model is exposed in such a way that it can be used with the FMI functions for model exchange or co-simulation. The result of the FMU export is a zip file, called the FMU file, as described in Section 6.1.3.

A simulation master or a simulation environment needs to import FMUs in order to use them. Importing FMUs consist of extracting the FMU file, parsing the `modelDescription.xml` file, and connecting the internal data structure with the FMI functions provided by the FMU. Through these FMI functions, the master or the simulation environment will interact with the model that was provided inside the FMU.

6.2 Use Cases and Applications

Having explained the FMI standard and its philosophy of use, we will now present typical use cases and applications in the design and operation of building and district energy systems in which utilizing FMI offers interesting and innovative perspectives. We will present typical and generic use cases, as well as specific applications that were developed by participants of this activity.

6.2.1 Simulation-Based Building Operation

This use case describes how FMI can be applied to monitor the actual performance of a building relative to the design intent during operation.

During the design, an HVAC designer creates a simulation model of a building, including its HVAC system and controls. The designer then exports the model as an FMU for co-simulation and imports it into a building management system (BMS) such as NiagaraAX. In the BMS, the designer links the model input to measured data. The design model can then be used to compute expected room air temperature or energy consumption, which in turn can be used to compare measured with expected performance. Details of these steps are as follows:

- As a first step, the HVAC engineer develops a model of the building and its HVAC and control system in a modeling language such as Modelica.
- As a second step, the HVAC engineer simulates the building model along with its HVAC and control system to ensure correctness of the control sequence.
  Optionally, to verify the control sequence, the HVAC response and the building response, the model could be decomposed into these three subsystems: Measured data could be fed into the control model to verify actual versus expected control actions; the control signals of the actual system, together with measured outdoor and room conditions could be sent to the HVAC model to verify actual versus expected HVAC efficiency; and the measured HVAC supply conditions and outdoor conditions could be sent to the building model to verify actual versus expected energy provided to the rooms.
- As a third step, the model of the building with its HVAC and control sequence is exported as an FMU for co-simulation and imported into the building management system. The exported system model can then run in real-time to compute the expected performance of the building.

For more information, see [NW14].

6.2.2 Design and Operational Analysis of Energy, Control and Communication Grids

Buildings as components of an energy system consume, store and produce energy. In order to provide more global energy efficient solutions and to facilitate the integration of intermittent energy resources, one should consider the interaction of clusters of buildings with the electrical, gas and, if district heating or cooling is present, thermal networks. Such energy grids may have overlying centralized or distributed control, coupled through communication networks.

Such networks are complex systems composed of multi-disciplinary, multi-physics components that can be difficult to handle by any one functional domain: energy assets, information and communication technology and business processes. Energy assets (electrical components, buildings and HVAC systems, industrial processes, thermodynamics) are conveniently described by differential algebraic equations. In other domains, typically for information

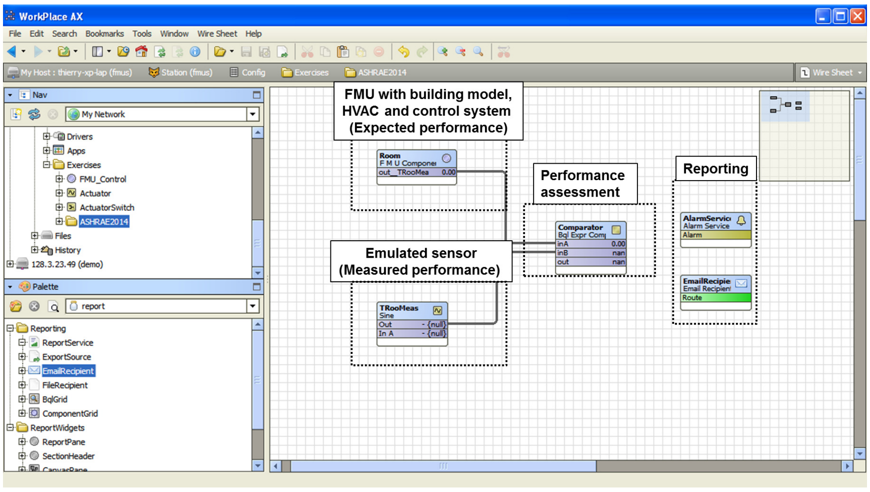


Fig. 6.3: Simulation-based performance monitoring in which an FMU is imported in NiagaraAX.

and communication technology, components are more conveniently expressed by event-based modeling (e.g. representing a market that negotiates energy prices and energy allotments, messaging among distributed equipment, and switching devices).

For such a wide functional scope, a monolithic approach that uses one model with one global solver may require simplified or reduced order models to be computationally tractable. In contrast, distributed simulation, possibly with different numerical solvers for the different domains, is a promising alternative to simulate large systems in parallel. The FMI standard provides a programming interface for models that allows this distributed simulation. Moreover, by encapsulating the domains in FMUs, domain-specific tools may be reused, such as for the simulation of large electrical networks that connect the buildings.

To setup such a distributed simulation, the following four steps are typically done:

1. Generate the models representing the buildings, grid components, control and communication.
2. Export the models as FMUs.

3. Connected the FMUs in a master algorithm, possibly using distributed computing.
4. Control the simulation with a master algorithm.

See also *[CBM+16][RGCJ+16]* for applications.

6.2.3 Co-Simulation via Domus with Modelica, EnergyPlus and ANSYS CFX

During the IEA Annex 60 project, the research version of Domus, Domus-r, described in Section 6.5.7, has been improved for co-simulation. For EnergyPlus and Modelica, an FMI for co-simulation import interface has been added, while for ANSYS-CFX a script-based interface has been added. In a tool-coupling test case with EnergyPlus, a pure conductive heat transfer envelope model was encapsulated as an FMU for co-simulation and imported into Domus, with the objective to validate the Domus FMI interface.

The Domus *pixel counting technique* (Fig. 6.4 a) has been used for accurate and fast evaluation of complex shading at internal and external surfaces. The intention of this use case was to simulate the effect that complex shading has on building energy use, with the latter being computed in EnergyPlus.

As a first step of a Domus co-simulation with CFD, Domus has been used to co-simulate with ANSYS-CFX a hollowed block (Fig. 6.4 b) wall. The results are promising and demonstrate a functional integration between Domus and CFX that can expand the current limits of both simulation tools and allow 3-D simulation of building elements.The introduction of the co-simulation in Domus enables detailed, combined consideration of the three heat transfer modes calculated by CFX and integration of this with different tools such as EnergyPlus and Modelica. This would allow for assessment of indoor air quality and building energy efficiency under asymmetric indoor and outdoor boundary conditions. Nevertheless, the high computer run time for CFD is still a considerable constraint so that further research is needed to make it viable.

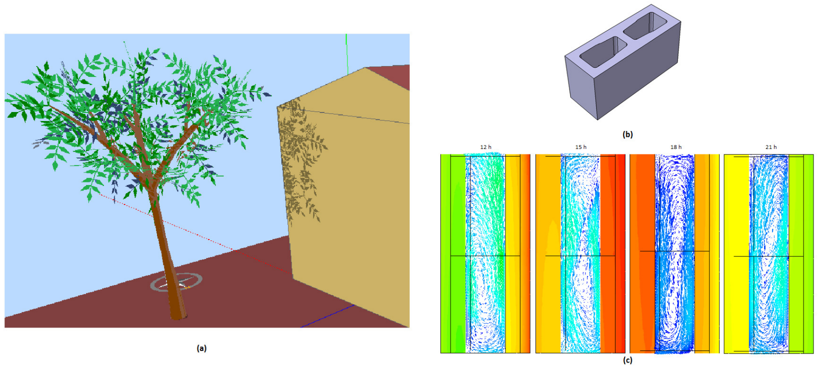


Fig. 6.4: *The use of pixel counting technique for accurate and fast evaluation of complex shading (a) at an external surface due to a non-planar tree. (b) The hollowed block used in the wall ANSYS CFX co-simulation and (c) the preliminary results.*

6.3 FMI-Compliant Tools for Buildings and District Simulation

This section provides an overview of FMI-compliant simulation tools that are either specifically developed for, or useful for, building simulation. While the majority of these tools have been developed outside of Annex 60, they are listed here to guide new users. A more comprehensive list of those that support the FMI standard can be found on the FMI-Standard website (https://www.fmi-standard.org/tools).

Dymola (http://www.3ds.com/products-services/catia/products/dymola/) is an environment that supports modeling and simulation of multi-physics systems implemented in the Modelica language. Dymola can be used to import and export models as FMUs for both co-simulation and model exchange.

JModelica (http://www.jmodelica.org/) is a Modelica-based open source platform for optimization, simulation and analysis of complex dynamic systems. JModelica has been used by the building simulation community for the simulation and optimization of building energy systems. It supports the export and import for FMUs for model exchange and for co-simulation using the PyFMI Python package.

OpenModelica (https://openmodelica.org/) is an open-source Modelica-based modeling and simulation environment. It has been used in the building community for the modeling and simulation of building energy systems. It supports the export and import for FMUs for both model exchange and co-simulation.

SimulationX (https://www.simulationx.com/) is a modeling and simulation tool which can be used for energy efficiency and HVAC systems modeling and simulation. It supports the export and import of FMUs for both co-simulation and model exchange.

MATLAB/Simulink (http://mathworks.com/products/simulink/) also support FMI import and export using the Modelon FMI toolbox (http://www.modelon.com/products/fmi-toolbox-for-matlab/). A building application case and some implementation comparisons can be found in *[WJEP15]*.

Ptolemy II is an open-source software framework *[Pto14]* supporting experimentation with actor-oriented design. Actors are software components that execute concurrently and communicate through messages sent via interconnected ports. Ptolemy II has been used to create the Building Controls Virtual Test Bed (BCVTB) *[Wet11a]*. Ptolemy II supports the import of FMUs which implement FMI 1.0 and 2.0 for co-simulation and model exchange. A large district energy system co-simulation has been implemented in *[ZJB+16]*.

PyFMI (https://pypi.python.org/pypi/PyFMI) is a python package for loading and interfacing with FMUs for co-simulation and model exchange. PyFMI is available as a stand-alone package or as part of the JModelica.org distribution. PyFMI has been used by the building simulation community for applications such as fault detection and diagnostics, and model predictive controls. A building co-simulation example using Dymola models can be found in *[RRDW15]*.

MasterSim (http://mastersim.sourceforge.net) is a co-simulation master algorithm implementation and graphical user interface written in C/C++ (Windows/-Mac/Unix) for coupled simulation of FMUs using FMI 2.0. It supports error controlled co-simulation and iterative master algorithms that set and get the state variables.

EnergyPlus (https://energyplus.net/) is a whole building energy simulation program that engineers, architects, and researchers use to model energy consumption for heating, cooling, ventilation, lighting and plug and process loads, as well as water use in buildings. Import and export of model FMUs are available for co-simulation *[NWZ14]*. Numerous applications can be found in the literature, see for example *[DEP16][ZJB+16]*.

WUFI+ (https://wufi.de/en/software/wufi-plus/) is a hygrothermal whole building simulation program which supports the import of FMUs for co-simulation *[PBR+12]*. The FMU import functionality was added to integrate Modelica HVAC system models to WUFI.

TRNSYS (http://www.trnsys.com/) is a graphically-based software environment used to simulate the behavior of transient systems. Even though TRNSYS itself does not officially provide FMI support, there is a dedicated FMI-based tool coupling solution openly available at http://trnsys-fmu.sourceforge.net. Furthermore, the feasibility of importing FMUs in TRNSYS through a TRNSYS Type has been demonstrated in *[EWP+13][EAWP13]*.

DIgSILENT PowerFactory (http://www.digsilent.com/) is a state-of-the-art industrial-grade modeling and simulation tool for electrical networks. Even though PowerFactory itself does not officially provide FMI support, it provides several APIs for co-simulation and there is a dedicated FMI-based tool-coupling solution openly available (see http://powerfactory-fmu.sourceforge.net) that utilizes these APIs in an FMI-compliant way.

6.4 FMU Export Facilities

This section describes software packages that can be used to export FMUs and were developed by participants of the Annex 60.

6.4.1 THERAKLES (Room Model)

THERAKLES (http://www.bauklimatik-dresden.de/therakles) is a single-zone hygrothermal model with detailed construction models developed at the Institute for Building Climate Control of the TU Dresden. It includes basic equipment component models. It provides modeling of the thermal storage of the wall by use of spatial discretization techniques. Focusing on deployment in engineering practice, it also provides a user-friendly graphical user interface.

For simulation with more advanced equipment and control logic, the THERAKLES model can be exported as an FMU for model exchange and co-simulation according to the FMI 2.0 standard. A graphical dialog is provided to lead the user through the export process. The generated FMUs provide reset functionality for state variables. Thus, all exported FMUs support error-controlled co-simulation.

THERAKLES has its own climate data base and exports calculated climatic data during simulation. The climate model is encapsulated in the CCM-library which supports an internal binary data format but also reads and converts data from `*.epw` file format.

6.4.2 NANDRAD (Building Energy Simulation/Multizone)

NANDRAD (http://www.bauklimatik-dresden.de/nandrad) is a building energy simulation framework developed at the Institute for Building Climate Control at TU Dresden. The NANDRAD tool is a command line application designed for the efficient simulation of complex buildings, including detailed wall models. The resulting large differential equation system is solved by state-of-the-art numerical methods, such as Newton-Krylov-iteration methods and sparse matrix algorithms.

NANDRAD supports FMU export for FMI 2.0 co-simulation and model exchange by a script-based automated procedure. The provided input and output interfaces allow the coupling between the NANDRAD building simulation and an external HVAC model, for example a heating or a hydraulic cycle Modelica model. The import of the FMU into a Modelica environment is directly supported by an automatically generated wrapper model. This wrapper encapsulates the NANDRAD FMU, connects all inputs and outputs with vector valued ports and generates a graphical representation. Additionally, NANDRAD precalculates heating and cooling design data and publishes it in a report file.

Similar to THERAKLES, the NANDRAD FMUs support getting and setting of state variables and climate data export.

6.4.3 EnergyplusToFMU

EnergyPlusToFMU (https://github.com/lbl-srg/EnergyplusToFMU) is a software package written in Python that allows users to export the building simulation program EnergyPlus version 8.0 or higher as an FMU for co-simulation using the FMI standard version 1.0. This FMU can then be imported into a variety of simulation programs that can import a FMU for co-simulation. This capability allows, for instance, for modeling the envelope of a building in EnergyPlus, exporting the model as an FMU, and then importing and linking the model with an HVAC system model developed in a system simulation tool such as the Modelica environment Dymola. EnergyPlusToFMU is released under an open-source BSD-license.

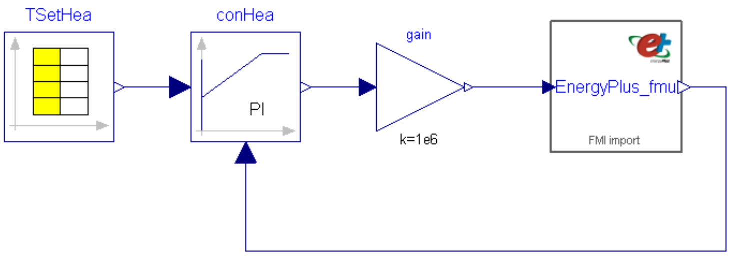


Fig. 6.5: An EnergyPlus room model has been exported as an FMU. This FMU is coupled to a controller implemented in a Modelica simulation environment.

6.4.4 FMI++ TRNSYS FMU Export Utility

Even though TRNSYS itself does not officially provide FMI support, there is a dedicated FMI-based tool coupling solution openly available, called the FMI++ TRNSYS FMU Export Utility. It has been developed using the FMI++ library (see http://fmipp.sourceforge.net/) and enables the export of a TRNSYS model with a Python script as an FMU for co-simulation, using FMI 1.0.

The export tool provides to modelers a special TRNSYS block called Type6139. This type provides external data sent to TRNSYS as inputs to the model and also allows for sending data as external outputs. Apart from the additional input and output block of this type, TRNSYS models are constructed in the usual way as shown in Fig. 6.6.

The FMI++ TRNSYS FMU Export Utility is released under an open-source BSD-license and is available at http://trnsys-fmu.sourceforge.net

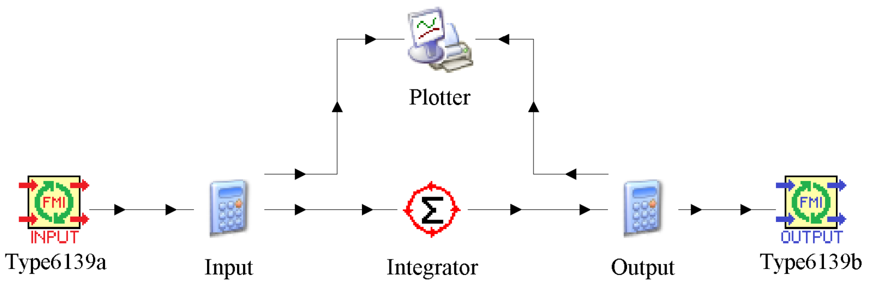


Fig. 6.6: *Example of a TRNSYS model containing blocks of Type6139 for FMU export.*

6.4.5 FMI++ PowerFactory FMU Export Utility

The FMI++ PowerFactory FMU Export Utility is a stand-alone tool for exporting FMUs for Co-Simulation (FMI Version 1.0) from DIgSILENT PowerFactory models, based on code from the FMI++ library (see http://fmipp.sourceforge.net/). It provides a Python script that creates FMUs from certain PowerFactory models, including the XML model description and shared libraries. Additional files (e.g., time series files) and start values for exported variables can be specified. Currently only steady-state simulations are supported, where the system evolution with respect to time comprises a series of load flow snapshots.

The FMI++ PowerFactory FMU Export Utility is released under an open-source BSD-license, available at http://powerfactory-fmu.sourceforge.net

6.5 Simulation Environments and Master Algorithms for FMI

This section describes simulation environments with FMU import facilities and master algorithms that have been developed by Annex 60 participants. It also describes numerical methods that have been implemented to integrate in time FMUs for model exchange.

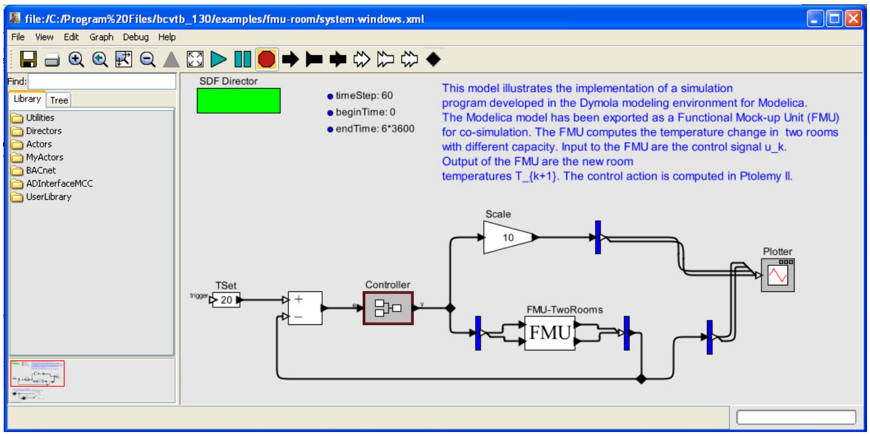
6.5.1 Building Controls Virtual Test Bed

The Building Controls Virtual Test Bed (BCVTB) is a free, open-source middleware based on Ptolemy II (http://ptolemy.eecs.berkeley.edu/). It allows users to couple different simulation tools for data exchange during run-time and for real-time simulation. Programs that are linked to the BCVTB are

1. EnergyPlus, the whole building energy simulation program,
2. Dymola, the Modelica modeling and simulation environment,
3. Functional Mock-up Units (FMU) for co-simulation and model-exchange for the Functional Mock-up Interface (FMI) 1.0 and 2.0,
4. MATLAB and Simulink tools for scientific computing,
5. Radiance, the ray-tracing software for lighting analysis,
6. ESP-r, the integrated building energy modeling program,
7. TRNSYS, the system simulation program,
8. BACnet stack, which allows exchanging data with BACnet compliant Building Automation System (BAS),
9. USB-1208LS, the analog/digital interface from Measurement Computing Corporation that can be connected to a USB port.

Fig. 6.7 shows its graphical user interface with a co-simulation model.

The BCVTB has been used in several applications such as agent-based simulation, real-time simulation, control of networked sensors and actuators, and performance prediction of HVAC systems. The BCVTB is released under an open-source BSD license and available at http://simulationresearch.lbl.gov/bcvtb. For more information, see *[Wet11a]*.



timeStep: 60
beginTime: 0
endTime: 6*3600

This model illustrates the implementation of a simulation program developed in the Dymola modeling environment for Modelica. The Modelica model has been exported as a Functional Mock-up Unit (FMU) for co-simulation. The FMU computes the temperature change in two rooms with different capacity. Input to the FMU are the control signal u_k. Output of the FMU are the new room temperatures T_{k+1}. The control action is computed in Ptolemy II.

Fig. 6.7: The BCVTB is used to couple a controller implemented in Ptolemy II with a room model implemented in Dymola and exported as an FMU.

6.5.2 EnergyPlus

EnergyPlus (https://energyplus.net/) is a whole building energy simulation program that engineers, architects, and researchers use to model both energy consumption and water use in buildings. To extend the capability of EnergyPlus for building simulation, EnergyPlus has been linked through co-simulation to various programs using direct coupling of tools through custom interfaces and through master algorithms in which EnergyPlus acts as a client.

To address the limitation of these custom coupling mechanisms, an FMI for co-simulation 1.0 import interface has been added to EnergyPlus to support the import of FMUs. Also, a facility has been developed to export EnergyPlus as an FMU for co-simulation 1.0. This interface allows users to link the building envelope of EnergyPlus with an HVAC system modeled in Modelica, or in another simulation tool and language. EnergyPlus is released under an open-source BSD license, and available at https://github.com/NREL/EnergyPlus. For more information, see http://simulationresearch.lbl.gov/fmu/EnergyPlus/export and *[NWZ14]*.

6.5.3 DACCOSIM

DACCOSIM lets you design and execute a simulation composed of multiple FMUs on multi-core computation nodes or distributed on clusters.

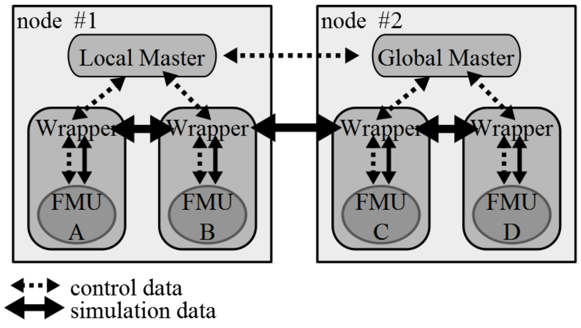


Fig. 6.8: Co-simulation architecture components in DACCOSIM.

A framework is provided to graphically define how the FMUs are connected to each others and on which computation nodes they should be distributed. DACCOSIM automatically generates the associated code and the DACCOSIM library is then used to execute the multi-simulation on the distributed computation nodes. It could be either Java or C++ code.

The main functionalities provided by DACCOSIM are

1. co-initialization with the Newton-Raphson method,
2. inputs extrapolation,
3. support for both constant and variable step size with *rollback*,
4. various error estimation methods that automatically adjust the step size (Euler, Richardson and Adams-Bashforth),

It runs on Windows and Linux, 32-bits and 64-bits and uses FMI-CS. It is developed by the Supelec IDMaD research team and the EDF R&D MIRE department in the RISEGrid Institute (http://www.supelec.fr/342_p_38091/risegrid-en.html).

DACCOSIM is released under the AGPL open-source license and available at http://daccosim.foundry.supelec.fr/.

6.5.4 Ptolemy II and Quantized State Systems (QSS)

Ptolemy II provides a master algorithm for FMUs for model-exchange and co-simulation, which implements the FMI 2.0 specification.

The master algorithm is implemented in the discrete event domain of Ptolemy II. Additionally, a family of Quantized State System (QSS) integrators is provided to integrate state variables of FMUs for model exchange 2.0. Details about the master algorithm and the QSS integrator can be found in *[WNL+15]*. The algorithm is release as part of Ptolemy II, and also in the cyber-physical system simulator CyPhySim *[LNNW15][BLL+15]*, which is a subset of Ptolemy II. Both are available under a BSD-license from http://ptolemy.eecs.berkeley.edu/ptolemyII/ and https://chess.eecs.berkeley.edu/cyphysim/.

The CyPhySim simulator supports classical (Runge-Kutta) and quantized-state simulation of ordinary differential equations, modal models (hybrid systems), discrete-event models, the Functional Mockup Interface (FMI) for model-exchange and co-simulation, discrete-time systems, and algebraic loop solvers. CyPhySim provides a graphical editor, an XML file syntax for models, and an open API for programmatic construction of models. It includes an innovation called *smooth tokens*, which allow for a blend of numerical and symbolic computation, and for certain kinds of system models, dramatically reducing the computation required for simulation. For example, Fig. 6.9 shows a model of a bouncing ball implemented in the CyPhySim simulator. For this problem, a Runge-Kutta 2-3 solver required 14,072 time steps and 3.3 seconds, while QSS required only 46 points in time and completed in 0.085 seconds, or 38 times faster. Furthermore, the QSS solvers have the interesting property that if certain assumptions about the integrator inputs are satisfied, then rollback is never required. If these assumptions are valid, then there is no error due to numerical approximation of the integration, and events such as zero crossings are predictable in advance. For certain cyber-physical systems, such assumptions are indeed valid, and hence computationally exact simulation is possible, where the only source of errors is numerical round-off errors. There is no error due to numerical integration. See *[LNNW15]* for more details.

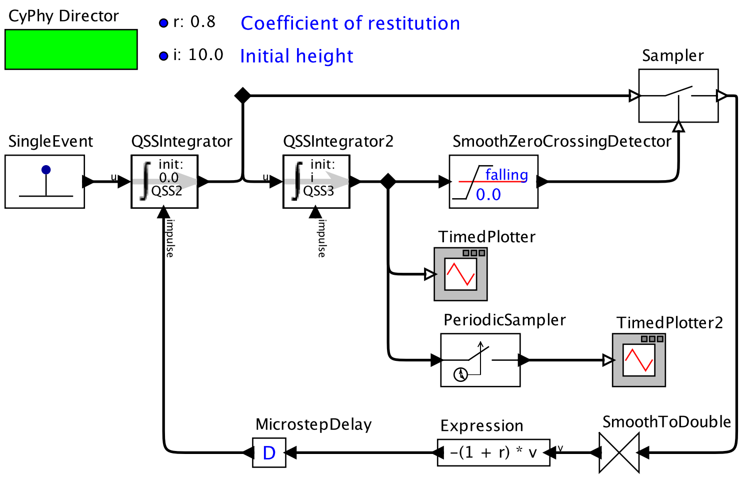


Fig. 6.9: CyPhySim model of a bouncing ball with QSS integration.

6.5.5 FUMOLA - The Functional Mock-Up Laboratory

FUMOLA is a co-simulation framework specifically designed to support the features offered by the FMI specification. FUMOLA is developed on top of the Ptolemy II framework (http://ptolemy.eecs.berkeley.edu/) and the FMI++ library (http://fmipp.sourceforge.net/), mapping Ptolemy II's functions to the functionality provided by the FMI++ library. Therefore, it provides a flexible platform to investigate the full range of possibilities offered by the FMI specification based on advanced simulation concepts. This allows for implementation of various simulation approaches, including discrete event-based simulations *[MW15]*, simulations of coupled physical systems *[WMB+15]* and simulations of closed-loop control systems models *[WJEP15]*.

FUMOLA is available at http://fumola.sourceforge.net.

6.5.6 WRM (Waveform Relaxation Method) as a Proposition of a Simple Master Algorithm

The Waveform Relaxation Method *[Lel82][LRSV82]* has existed since 1982 and was developed and used to solve large systems of ordinary nonlinear differential equations *[IC90]*, in particular for integrated circuit simulations. The method was adapted to be able to be used as a master algorithm for coupling different black box software components such as FMUs *[RRDW15]*.

The idea is to combine several heterogeneous systems that may have different dynamics, and couple them using an iterative method on the waveforms (Fig. 6.10). Each system is solved in time throughout the considered time domain, and its solution, e.g., the entire waveform, is used as an input of the other systems. The procedure can be done for the whole time domain, or for a sequence of sub-domains. This approach is like a strong coupling, and instead of exchanging simple values at a given time, it exchanges the waveform.

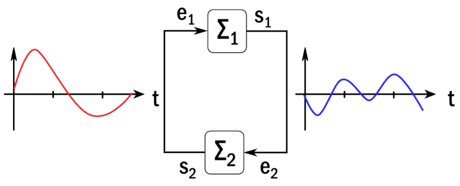


Fig. 6.10: Waveform relaxation algorithm for co-simulation.

WRM has many advantages. The master algorithm is simple and easy to implement. It can be used to couple different components without having to know the internal dynamics and simulation step sizes. In situations where time exchange data between components is important, such as for a web service, the WRM reduces the computation time by reducing the number of calls of the different sub-models. As the simulation time gets longer, the efficiency of the waveform relaxation method increases *[RRDW15]*.

Limitations and disadvantages include less efficient simulations for models that have events. See the application in *[RRDW15]*.

6.5.7 Domus

Domus is a building simulation software written in C++. It has an OpenGL-based graphical interface that has been improved to allow importing models from EnergyPlus 8.0 or higher as a FMU for co-simulation 1.0. Domus has also been improved for

1. shading and sunlit area calculation using a pixel counting technique,
2. reading and writing of EnergyPlus IDF files, and
3. co-simulation with ANSYS-CFD.

This last capability can allow for instance the co-simulation for performance assessment of whole-buildings with 3-D ventilated walls combined with indoor and outdoor air flow under heterogeneous surface boundary conditions. Domus is available from http://www.domus.pucpr.br/. See *[MOG03][MBZF08]* for more information.

6.5.8 MasterSim

MasterSim is an open-source co-simulation master implementation that supports FMI 1.0 and 2.0.

The simulation master was developed specifically for application in building energy simulation, i.e, interfacing of building simulation models with HVAC system models and control systems, or specialized sub-system models. The master employs several algorithms for obtaining stable, efficient and error controlled solutions. It contains

- different master algorithms/iteration methods, such as non-iterative Gauss-Jacobi, and iterative Gauss-Seidel and Newton methods,
- variable communication step sizes with local error control,
- serialization/deserialization for stop-and-restart of the master.

MasterSim is developed in C++ and runs on Windows, Mac OS X and Linux. Also, the master supports a feature that disables automatic unzipping of FMU archives, which allows for using persistent DLL/shared library files, which is important for FMU developers.

MasterSim is available from http://mastersim.sourceforge.net under a GPL v3.0 license.

6.6 Desired FMI Features and Proposal of Evolutions of the Standard

While the FMI standard is well-suited for our classes of problems, as with any other standard that covers such a complex use case, there are certain limitations encountered and improvements that the Annex 60 participants would like to see in future versions of FMI. As a result of Annex 60, some changes have been proposed to the standards committee. For this cases, we list below the ticket number of the FMI development site. This section provides an overview of the key challenges encountered while using FMI version 1.0 and 2.0, with actions undertaken to address some of the limitations.

Change of array size during initialization: The size of arrays used for inputs, outputs and state variables is fixed once the FMU is generated. It would be desirable to allow defining array variables as quantities where the array size is being controlled by a parameter given during the instantiation of an FMU. This would allow pre-compiling FMUs in which array sizes vary for different applications, such as for CFD, ray-tracing, or heat conduction through solids and through windows.

For this item, the FMI committee has a working group.

FMU time unit: The FMI standard does not specify the unit of the time passed to the FMUs. This can result in synchronization problems if slaves assume a different unit for time. Thus it would be desirable to extend the standard to specify the unit of the time passed to the FMU slaves, or conversely, state in the specification that time is in seconds, which seems to be the implicit assumption and corresponds to what most, if not all, tools implement. Currently, FMU exporters have to agree on a time unit and specify this in the interface description.

A ticket has been filed on the FMI issue tracker (ticket #307).

Handling of FMU-specific output and log files: While FMI 2.0 provides a path to a read-only resource location, such as for data files to be read by the model, it does not provide a path to a writable location where output and log files shall be written. Knowledge of such a writable location is important, for example, when the same FMU is instantiated several times within a co-simulation scenario. Then, each FMU instance needs to use a different root directory for its written files in order to avoid overwriting each other's files. The same applies when multiple co-simulation runs are executed in parallel using the same FMU files. Hard-coding writable paths inside the FMU does not work in this situation. Use of the current working directory as basis does not work on the Windows platform, since several Windows API function related to file and directory handling change the working directory. The outcome depends on the order in which the FMUs are instantiated and initialized, and on their internal operation. Therefore, it would be meaningful to pass such a directory, if available on the given platform, during instantiation of the FMU. Without a formal method, FMUs must agree on a specific name of the output directory, and the master algorithm must set this before entering the initialization mode of an FMU.

A ticket has been filed on the FMI issue tracker (ticket #299).

Support for structured variables: FMI 2.0 only defines scalar variables in the `ModelVariables` subsection of `fmiModelDescription` at the root-level of the schema file *modelDescription.xml*. Various domains (e.g. fluid, heat transfer, electrical, magnetic, mechanics) would benefit from supporting structured variables of fixed size, especially one-dimensional arrays, instead of only scalar variables. This would also require new `fmi2GetXXX` and `fmi2SetXXX` functions.

For this item, the FMI committee has a working group.

Event handling in co-simulation: In FMI for co-simulation 2.0, an FMU cannot announce to the master algorithm an upper bound for the step it can simulate. Rather, an FMU will have to reject a step size requested by the master if it is too long. *[BGL+15]* suggests a new function `fmiGetMaxStepSize` that would allow an FMU to announce how long a step it can take. A similar suggestion is also made by *[TCT+16]*, who propose to add the next event time to a new function `fmi21DoStep`. Announcing the maximum step size would be useful, for example, if a model knows its next time event and hence can ask the master to step no further than that time point. With the current implementation, a master algorithm may step over this time instant, then get notified by the FMU that it went too far, and then ask the FMUs to rollback and do a smaller step.

This item is in discussion within the FMI committee.

Efficient mechanism for output writing in FMI for model exchange: Currently it is unclear how to handle FMU-specific outputs efficiently in FMI 2.0 for model exchange. For example, in a building simulation FMU, at certain time points, usually hourly or every 15 minutes, a lot of output data needs to be stored in output files. Passing that amount of output data through the FMI interface seems not practical.

The integrator within the master algorithm may use a higher-order integration method with variable time steps, such as CVODE. Therefore, integration steps and output steps in the FMU generally do not match.

Currently the only means for the ModelExchange FMU to identify a suitable time point for writing past outputs is the `fmi2CompletedIntegratorStep` function. The FMU may than write outputs within the last step at the scheduled time point. However, the FMU does not have information about the higher-order integration method of the integrator. Therefore, backward interpolation will likely give different results compared to those computed by the integrator.

A ticket has been filed on the FMI issue tracker (ticket #326).

Master shall be required to set fixed parameters even if unchanged Currently, when specifying fixed parameters in the `modelDescription.xml` file, the standard does not require that the master calls `fmi2SetXXX` functions for such parameters. Most master implementations currently set the parameters, even if the values have not been changed by the user.

In order to ensure consistency between the default parameter values set during the export of the FMU and the values in the `modelDescription.xml` file, it would be useful to require the master to always set the values during the initialization of the FMU.

A ticket has been filed on the FMI issue tracker (ticket #366).

6.7 Best-Practice Recommendations

This section summarizes best practices proposed by Annex 60 participants for building or district-level simulations using the FMI standard.

6.7.1 Climate Data Export

While simulating coupled FMUs, redundant calculations of the same physical part should be avoided because of the risk of inconsistency. Typically, this problem occurs when climate data calculation are performed by different FMUs simultaneously. An example is a system with two FMUs, one for a building model, and one for an HVAC system, which both require outdoor conditions.

As climate data only depend on time, using a common FMU for climate data will not create cyclic dependencies among FMUs.

Since there are different possibilities for defining climate parameters (units, naming conventions, sun coordinate system, etc.) standardizing an interface for climate data would be beneficial.

6.7.2 FMI Interfaces for Room and HVAC Systems

Different variables could be exposed at the interface of rooms and HVAC systems. In order to make different FMUs compatible, an FMI package has been added to the `Annex60` library which allows for exporting HVAC models, HVAC systems, and thermal zones as FMUs. This package contains connectors that can be exposed to retrieve and to output HVAC system variables. The package also contains FMI adaptors, which have been added to facilitate the export of HVAC systems. See http://www.iea-annex60.org/releases/modelica/1.0.0/help/Annex60_Fluid_FMI.html for documentation. The rationale for the selection of variables that are used as inputs and outputs of the HVAC FMU are discussed in *[WFN15]*. Fig. 6.11 shows a model of a heater with the FMI interfaces. The instances `bou*` convert between causal input and output signals and the acausal fluid ports.

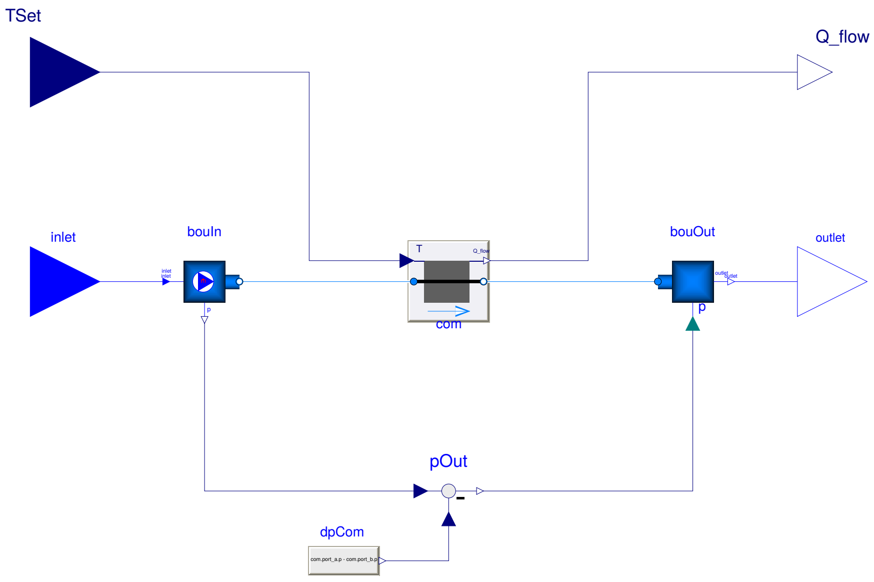


Fig. 6.11: *Modelica block that can be used to export a heater that has acausal fluid ports as an FMU.*

6.8 Conclusions

This chapter demonstrates that having a standard for exchanging models between different simulators and for linking simulators during run-time is an important topic for building and district energy system simulation. As building and district energy simulation typically involve large models and long simulation periods, having efficient master algorithms is important. The participants of Activity 1.2 demonstrated that FMI is useful for various applications. They also demonstrated the capability of tools to export, import, and co-simulate models encapsulated as FMUs. They showed that the FMI approach is suited for solving the multi-physics, multi-component systems encountered in the simulation of buildings, and they contributed to proposals for further improving the FMI standard. Simulation coupling with open-source standards is potentially a key for tackling the scalability issues that will be encountered for district-level simulations. This is one of the big challenges that building and district energy simulations will encounter in the years to come.

Chapter 7

Activity 1.3: BPS Code Generation from Building Information Models

7.1 Introduction

Activity 1.3 of the Annex 60 is dedicated to the complex issue of transforming a digital model of a building and its energy systems into Modelica code that can be readily used for advanced building performance simulation (BPS). It is the conceptual idea and vision of this activity to thoroughly address the prevailing tedious, cumbersome and error-prone process of manual data conversion and model generation and to provide a methodology for automatically, or at least semi-automatically, transforming a digital model into an object-oriented acausal model.

The transformation is hampered by several constrains such as

- models contain inconsistencies and modeling errors and are typically not built by a person skilled in energy performance simulation,
- models originate in the design process from other domains such as the architecture or structural domain,

- objects and parameters in a CAD model significantly differ from the representation needed in Modelica,
- input models are lacking information relevant for BPS,
- a conversion process needs to support multiple Modelica libraries with different model topologies and varying syntax at the same time,
- the object-oriented data schema of the digital building model is continuously updated and subject to changes which need to be dynamically reflected by the tool.

Consequently, several complex requirements form the specification of a software framework that is capable to deal with these constraints. Furthermore, a flexible methodology is required to take expert knowledge into account when it comes to object and parameter mapping. The software framework shall provide interfaces to Open BIM data formats such as the Industry Foundation Classes (IFC). And, as well, the framework must be modularly structured in order to allow for further and distributed developments on an open source basis by an international community. Its structure shall allow to maintain the framework or even most parts of it in due time.

In order to carefully consider these settings, the activity followed the following approach:

- As BIM data format, the Annex supports the Open BIM format Industry Foundation Classes (IFC). Before processing BIM models, these models are checked for integrity concerning two aspects. First, the geometric consistency is accounted for by an advanced model checking process which includes the definition of space boundaries. Secondly, the HVAC model is checked by another model checking toolbox developed in this project.
- Models are then transformed into an intermediate data format called SimXML. Therefore, a flexible module for schema parsing was developed. Relevant data which are missing in the BIM can be added to the SimXML data model.
- In order to manage these SimXML data, a dynamic schema parser was developed in C++. This offers the flexibility to dynamically account for changes and updates of the SimXML schema (and, thus, for changes of the IFC data model as well). An application programming interface (API) between C++ and Python was implemented to efficiently interact

with the data model.

- An object and parameter mapping mechanism as well as respective mapping rules were developed in order to formulate engineering knowledge in a rule-based methodology. These rules are processed by the framework.
- An IFC model view definition (MVD) was developed in order to specify the subset of IFC data relevant to BPS.
- In order to dynamically support multiple Modelica libraries at the same time, a template-based approach was selected as solution and implemented in Python.
- All transformation tools and methods are provided as open source framework for further developments in the community. Researchers are invited to collaborate and to further extend the framework.

The following sections briefly introduce what BIM is, explain the BIM data import process and detail the developed open Annex 60 framework for Modelica code generation from BIM. At the end of the chapter, the use cases applied for testing and development are introduced as well. It shall be noted at this point, that the activity followed a bottom-up approach which was based on these use cases. Therefore, the framework is currently limited to support these cases. As the activity focused on a subset of IFC data, on the other hand, with the resources on hand it was possible to finalize the overall framework as such in the Annex and to provide a modular framework for further development and dissemination.

7.2 Building Information Modeling (BIM)

This subchapter starts by introducing the term BIM and by explaining the changes required when BIM is used in a project. For the evaluation of the developed model transformation and BPS code generation concept and toolchain the use cases and case study are described next. The authors detail current state-of-the-art toolboxes and tools that are relevant for our prototype development. Finally, in this subchapter we summary relevant standards and agreements that are needed for development of BIM based data exchange.

7.2.1 What is BIM?

The acronym *BIM* stands for Building Information Model or Modeling. It involves *buildings* and building designs, *information* about given buildings and/or designs, and to how that information is *modeled*. Searching the term BIM on the web one will find many industry definitions such as: BIM and Sustainability, Green BIM, 4D BIM, BIM and the Bottom Line, BIM for Risk Management, BIM BOP, BIM CON!FAB, BIM Symposia, BIM Consulting, BIM Surveys, and many more. It seems that anybody who is asked to define BIM has their own definition which reflects their own needs, expectations and uses of BIM.

According to Bazjanac *[Baz04]*, the BIM acronym can have two meanings: noun or verb. The science definition of BIM-the-noun is straight forward:

> A *Building Information Model (BIM)* is an *instance* of a *populated data model of buildings* that contains multi-disciplinary *data specific to a particular building* which they describe *unambiguously*.

The science definition of BIM-the-verb is even simpler:

> *Building Information Modeling (BIM)* is the *act* or *process* of *creating* a Building Information Model (BIM-the-noun).

Charles Eastman and colleagues offer a more elaborate and practical working definition of BIM for the buildings industry *[ETSL11]*:

> BIM is a modeling technology and associated set of processes to produce, communicate, and analyze *building models* characterized by

- Building components that are represented with intelligent digital representations and can be associated with computable attributes and parametric rules.
- Components that include data that describe how they behave.
- Consistent and non-redundant data.
- Coordinated data such that all views of the model are represented in a coordinated way.

Similarly, definitions of BIM by buildingSMART International *[bSI15a]*, the U.S. National Institute for Building Sciences *[NIB16]* and the U.S. National BIM Standard Project *[NIB13]* are in full concordance with this definition.

Eastman and colleagues also define what causes a data agglomeration not to be a BIM, disqualifying building "models" which some industry members call BIM:

- Building models that contain 3D data only and no object attributes.
- Building models with no support of behavior (i.e. models that do not utilize parametric intelligence).
- Building models that are composed of multiple 2D CAD reference files that must be combined to define the building.
- Building models that allow changes in one view that are not automatically reflected in other views.

Accordingly, a Building Information *Collection* (BIC) is also not a BIM. A BIC is characterized by an ad-hoc database to which information is supplied as a participant recognizes the need for it, with no systematically rules for database creation. As a result, the database is used indiscriminately from one building to another, and when viewing a datum in the database one cannot be confident what the datum actually represents without examining why and how the datum has been entered in the database.

In order to create a proper BIM, three conditions must be satisfied: (1) valid building data must be available and verifiable; (2) a specific data model of buildings must be used in the instantiation of data; and (3) software capable of properly populating the data model must be available and used in all instances of entering data in the data model. BIM data used in this research project are valid and verifiable for the purposes of the project. The most common data model format used with BIM is Industry Foundation Classes (IFC). The latest IFC version 4 Addendum 2 (IFC4 Add 2) *[bSI15b]* release is the data model used as basis for the BIM applications to share information across multiples software. All software used to populate the BIM is directly or indirectly IFC compatible.

IFC is fully object-oriented and is fully extensible. It is absolutely "neutral" (i.e. does not favor any software environment, platform, market, endeavor or suite of tools). IFC is also the only life cycle model of buildings that is an "open" International Standard Organization (ISO) Standard (no. 16739), and can be used free of charge.

The main purpose of the IFC data model is the enabling and support of software interoperability in the buildings industry. Software interoperability – essential to collaborative design – is the common name of technologies that allow software to communicate to one another and seamlessly share and exchange data. Some advantages of interoperability are detailed below:

- Allows automatic and flawless exchange of relevant common project data between one software application and another.
- Supports automatic data conversion between software that use partially overlapping data.
- Offers consistency management of the various ways that are used to model a building.
- Provides the tracking and management of information consistent with best practices required for security, provenance, etc. in support of contractual and legal frameworks.

Interoperable applications enable the work of multiple disciplines with varied expertise, and automatically translate data to support the different data structuring and viewing needs for different decision-making industry domains.

Any complete BIM of a sizeable commercial building will contain much more data than any single software application can read and manipulate. The main reason for that is not necessarily the size of the building – it is the multi-disciplinary nature of the data in a complete BIM. In most cases only discipline software can populate the BIM with all relevant discipline data, and only discipline software can read such data. To avoid having to deal with data in the BIM that are irrelevant to the discipline, relevant data are grouped and agglomerated in a "view" of the particular BIM *[Hie06]* and such views are defined as Model View Definitions (MVD).

Data included in Model View Definitions (MVD) are subsets of the IFC data model. The subsets specify all discipline (or project type) definitions in the IFC data model that need to be shared or exchanged among multiple software applications in support of work of a given discipline, project type or any other given purpose. bSI certifies software applications' compliance with a given MVD if the application can demonstrate that all data exchange requirements of the given MVD (object/attribute/relationship sets, as well as their form and format) have been implemented by the application.

The BIM Collaboration Format (BCF) is an open XML file format that supports workflow communication and messaging among different software applications engaged in BIM processes [bSI14]. It boosts collaboration in BIM workflows by exchanging only the lean machine-readable BCF-topics with attached BIM-snippets (and not the entire BIM) among participating applications. The RESTful API enables seamless automated exchanges of BCF-topics among software that has implemented this Application Protocol Interface [bSI15c].

7.2.2 Changes Within the Planning/Design Process

The adoption of BIM (as a verb) constitutes a paradigm shift in the Architectural, Engineering, Construction and Facilities Management (AEC/FM) industry. BIM based collaboration is a different way of working as opposed to complementing the traditional 3D CAD approach. The key difference is the opportunity for collaboration over the entire building life-cycle: from conception to design, construction and operation and up to the final demolition. The deployment of BIM in construction can make the industry more efficient, effective, flexible and innovative [THN13]. However, the use and adoption of a new technology begins with a series of decisions that ends in appropriate and effective use of such technologies.

In some countries such as USA, UK, Norway, Finland, Denmark, Singapore and South Korea the use of BIM in projects is obligatory. Despite the industry's awareness of the potential of BIM, construction organizations are yet to use it effectively. Fig. 7.1 shows that BIM is leading a change in resource allocation and associated costs from the later to the earlier stages of the project. By comparing the traditional 3D CAD approach with BIM, BIM is considered more efficient and less-time consuming, therefore less costly.

The use of BIM concentrates stakeholder collaboration throughout the entire building life-cycle. In doing so it provides a central repository of information that can be accessed by any stakeholder, when needed, in order to make the best and most effective use of available information. The major benefits from this approach are design consistency and visualization, cost estimates, clash detection and implementation of lean construction [VSS14]. This digital transformation cannot happen unless there are appropriately skilled personnel to support BIM implementation.

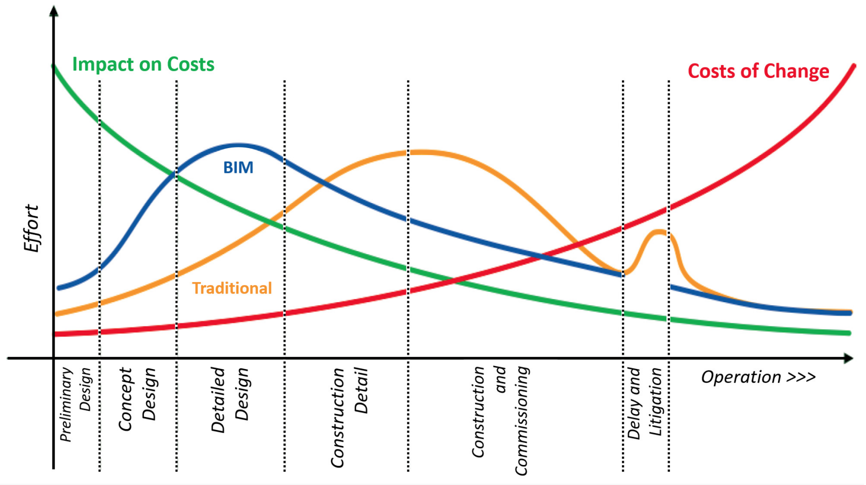


Fig. 7.1: Comparison between BIM-based workflow and traditional workflow (Adapted from [EHLP13]*).*

7.2.2.1 Roles and Responsibilities in the BIM-Based Process

It is expected that the demand for BIM will required new organizational structure and changes of responsibilities within defined roles. *[GT14]* on their study reviewed over 300 jobs description to determine responsibilities and skills required in BIM related roles. They identified three core roles: BIM Manager, BIM Coordinator and BIM Modeler. Others researches also mentioned different terminologies for these roles but with the same responsibilities of management, coordination and modeling *[BS10][PH13][SML13]*.

A BIM Manager is a person familiar with the building and design process from start to finish. This role does not require a domain identifier as this is an overarching role within the organization covering all aspects such as collaborative information management, standards management and process planning. This person can work directly with BIM Coordinators and BIM Modelers and with management designing processes. The BIM Coordinator is the person responsible for facilitating the transition into BIM based work practices and subsequently enhance the performance of BIM based activities. This person's key role is to help with the model and other BIM related issues. Lastly the BIM

Modeler requires a domain identifier and skillset in their respective. This person focuses on a specific performance aspect of the project. Therefore, the success of a project depends on meaningful integration of the roles involved, through an efficient and sustainable use of BIM.

7.2.2.2 Agreements in the BIM Process

Different national and international BIM guidelines and standards are available today to help organizations to implement BIM in their processes *[EHLP13][AECUKC15][BIMGW12]*. The benefits of using BIM depends on the manner in which information will be created, maintained and shared within project stakeholders. People involved in the BIM project are expected to have the ability to produce, manage and share any information required in a standard format. According to Egger *[EHLP13]*, the success of a project in the building industry depends on four factors: people, processes, guidelines and technology (Fig. 7.2). With the exception of technologies that are well established in the construction industry, the remaining three factors present problems that need to be overcome, especially the lack of professional training and guidelines.

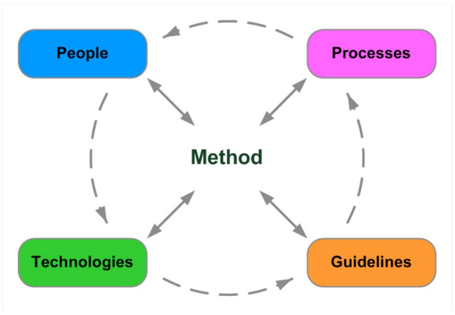


Fig. 7.2: Main factors of influence on the use of BIM method [EHLP13].

Within BIM projects all information is coordinated in a common data environment and it is very important that all objectives and deliverables are defined within this environment at the beginning of the project. In this manner, BIM servers provide a multi-disciplinary collaboration platform that maintains a repository of the building data, and allows applications to import and export files from the database for viewing, checking, updating and modifying the data. The information contained in the common environment will form the basis of the BIM Execution Plan. The execution plan is a crucial document that reflects the employer information requirements to ensure that design is developed in accordance with their needs. As a consequence of this new process, a higher quality and consistency is provided and administered in BIM projects.

7.2.3 Relevant BIM Standards and Implementation Agreements

In context of building energy performance of buildings two types of standards exist: data exchange standards and energy code standards. The data exchange formats most relevant are gbXML, IFC and SimModel, which we describe in the next subsection below. Energy code standards have a local character and can generally be further divided into standards that are based on static calculations and standards that are more detailed and based on simulation. For example, the German energy code (EnEV) is a static calculation whereas the US energy code (ASHRAE 90.1) is using simulation models. For further details on those energy standards we refer to van Treeck et al (*[vTWM15]*). Since this project is focusing on Modelica simulation, which is a more detailed approach, energy standards could become a byproduct in the future with additional development.

7.2.3.1 Data Exchange Formats

In the traditional approach, the majority of information is exchanged via 2D drawings that consist of lines, but with the introduction of BIM-based exchange there is a possibility to link or embed data into BIM objects. Several data exchange formats are available today for BIM applications to share information among each other. The most common are:

- Green Building XML schema (gbXML) facilitates the exchange of data among CAD tools and energy analysis software;
- IFC is the most used data format in the AEC/FM industry, mainly because of its ability to represent elements of a building as objects with

properties and references to others objects [ARHZ15]. The latest release IFC4 contains many improvements in the HVAC domain and some enhancements for space boundaries.

- SimModel is a simulation domain specific data model. It combines content from both IFC and gbXML as well as other relevant data models. As a simulation domain specific data model it also contains data that are relevant to simulation only.

With IFC4, energy-related data of a building and its systems can be exchanged. This includes space boundary information for representing thermal zones. Compared to previous versions it also contains new and more detailed HVAC components. A data format for the exchange of vendor-specific product data is the ISO Standard 16757. The standard provides a data format for the description of manufacturer product data and can be seen as a supplement to the IFC data model.

All three data models can be also represented in XML format (IFC is mostly defined in the STEP format, whereas SimModel has also a binary representation) and are object oriented data models. Thus, they include classes, attributes and references. The IFC data model contains many relational objects that link two different objects together. For example, the IfcRelAggregates object can link the IfcSite and IfcBuilding object. Since some of these relational objects are very generic, there exists a large number of possibilities to link objects. SimModel simplifies this by combining these relational objects into the main objects. For the example above, the SimSite object contains a property that directly links to the SimBuilding object. While SimModel is still very close in its structure and hierarchy to the IFC model, this adds another level of detail for the definition of objects. In IFC classes can be further specified by using an ObjectType, SimModel uses the additional ObjectSubType to further specify objects. gbXML does not provide the necessary level of detail of HVAC components and was thus not further considered in this project.

While IFC is seen as the right data format to retrieve building data from the architect and other domain experts it does not cover special requirements for simulation purposes. For that reason, the simulation domain specific data model SimModel (see Section 7.3.6) is used as an intermediate step towards the Modelica simulation. Due to the mentioned flexibility to define and link objects in an IFC model as well as it large scope, the mechanism to limit scope is needed and discussed in the next subsection.

7.2.3.2 Implementation Agreements

BIM standards like IFC need further implementation agreements to limit the theoretically possible representation options within the data model to a practical subset. BuildingSMART has introduced the concept of Model View Definitions (MVD) that identifies a subset of IFC that is needed to support a selected set (partial model) of use cases. The definition of an MVD is described in Section 7.3.5. Other domains, such as the structural domain are not relevant in the simulation context and can be excluded from the scope via an MVD. Geometric representations with corresponding material properties, internal loads, thermal zones, HVAC systems and HVAC components are topics that would be entailed in a simulation MVD. Besides the data itself, the MVD also defines so-called concepts that show how objects should be connected to form an interconnected data model. For example, a concept to define topology connections or a concept to place HVAC components in a system. As shown in Section 7.3 MVDs can be formalized using the mvdXML format so that automatic checking for required objects and properties is possible. This checking will be introduced with the software certification for IFC4 to enable more and better quality checks of IFC implementations. If a software passes all test cases defined for an MVD, then it will be certified by buildingSMART to meet defined quality criteria. The MVD concept and corresponding tools have been further developed for this latest IFC4 version and now cover a wider range of features. In previous versions, so-called implementer agreements were essential to define concepts in a more loose manner, but to properly document them for software implementers. For example, detailed space boundary definitions were specified in special add-on of implementers agreements in IFC2x3. Since the IFC4 version is quite new as well as the improved MVD, time will tell if implementers agreements will still be an important aspect in addition to MVDs or if they will be mostly replaced by MVDs.

7.2.4 Review of Existing BIM Tools for Visualization and Model Checking

For the development of software prototypes based on the IFC data model, it was important to reuse existing tools, resources and technologies to optimize the project output. Due to the large number of existing IFC tools a proper review was needed and conducted within this project. We illustrate this review in context of the development of the model checking and conversion components. Here, two major feature areas were of particular interest:

- Free and open source IFC-based viewing components
- BIM Model Checking Tools including their analysis features

The following tools have been selected and analyzed:

Open source IFC-based viewing components

- xBIM Explorer (sample application from xBIM Toolkit), V 2.4.1.28
- IfcPlusPlus Viewer (sample application from IFC++ Toolkit)

Free IFC-based viewing components

- Constructivity Viewer, V 0.9.7.0
- DDS-CAD-OpenBIM-Viewer, V 8.0.2012.101
- FZKViewer 4.1, V 4.1
- IFC JAVA VIEWER (Part of IFC-Tools Project) V 2.0 ,
- IFC Engine DLL: Sample Viewer, V 1.0.0.1
- Solibri Model Viewer, V 8.1.0.80
- Tekla BIMsight, V 1.8.5002.18178 ,

Free BIM-Model checking tools

- DDS-CAD-OpenBIM-Viewer, V 8.0.2012.101
- FZKViewer 4.1, V 4.1
- Tekla BIMsight, V 1.8.5002.18178

We evaluated the available tools and components in terms of their capabilities to represent, model and appropriately describe specific aspects as described below with respect to:

- support of IFC2x3
- support of IFC4

- support of the STEP IFC base syntax
- support of the XML IFC base syntax
- reliability and performance of IFC import
- platform independence
- capability to validate IFC files
- visualization features
- selection of attributes
- clash detection

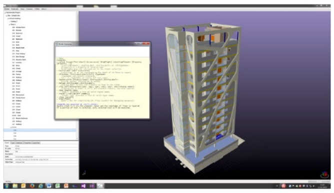
Since the ifcXML syntax was essential to a subset of our tools, it was an important criterion for our review. In addition, the reliability and performance of the IFC import is obviously an important criterion. But the evaluation of the IFC 4 import was difficult due to the early stage of the related tool development efforts. Thus our review of this functionality is limited. Since a basic validation of IFC files is essential before any conversion process is meaningful, we also investigated if tools can validate the IFC file against the schema. If it supports rule-based validation where the rules can be predefined within the tool, potential users can apply those to their IFC files. In addition, some tools support the recalculation of surface areas and volumes to verify those properties with the IFC file content. The tools were also assessed on their visualization features. We investigated view and render options as well as support for transparency. Object selection and filtering based on attributes is also a common feature among those tools and was added as criterion.

The free model checking tools such as FZKViewer and Solibri Viewer do support basic checking features such as clash detection or quantity takeoff. However, we could not identify a free model checking tool for the required HVAC consistency check of the IFC model (Section 7.4.3.3). This is why we reused our previously developed components for this model checking implementation. In context of the ifcXML support only the RDF Viewer based on the IFCEngine.DLL and the FZKViewer do currently support import and export of the ifcXML file. Only the former is also available as open source.

Based on these criteria, two toolkits can be identified as the most promising ones to serve basic functionalities: the open source XBIM tool kit (Fig. 7.3) and the IFC Engine DLL toolkit (Fig. 7.4).

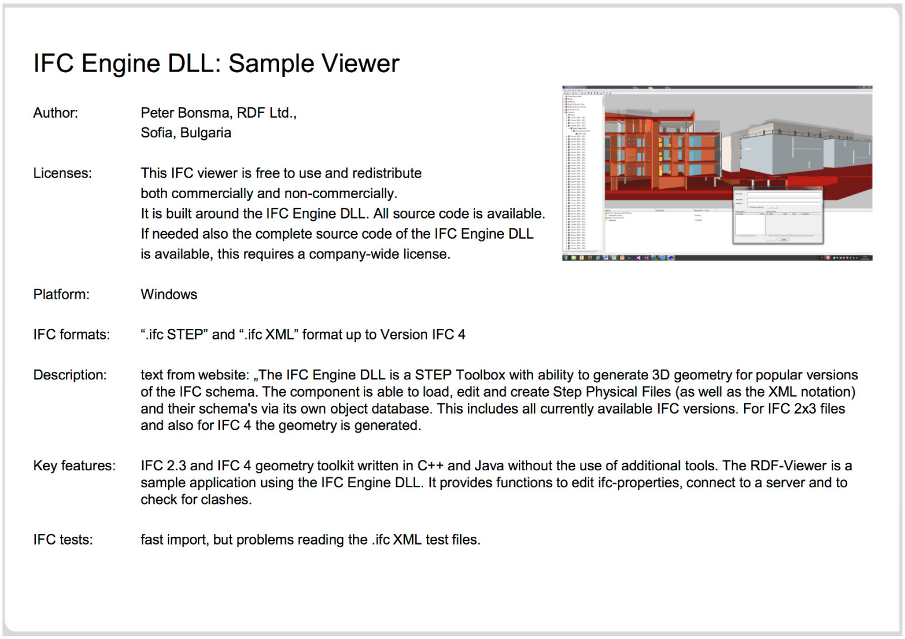
For the development of the Annex 60 model checking tool (ref Section 7.4.3) the second tool was selected due to its simpler enhancement possibilities.

xBIM Explorer (sample application from *xBIM Toolkit*)



Author:	Prof. Steve Lockley Northumbria University, United Kingdom
Licenses:	open source: CDDL and OC: LGPL-like with certain differences GPPG: New BSD
Platform:	Windows
IFC formats:	".ifc STEP" and ".ifc XML" format up to Version IFC 2.3
Description:	text from website: "The eXtensible Building Information Modelling xBIM Tookit (eXtensible Building Information Modelling) is an open-source, software development BIM tool that supports the BuildingSmart Data Model (aka the Industry Foundation Classes IFC). xBIM allows developers to read, create and view Building Information (BIM) Models in the IFC format. There is full support for geometric, topological operations and visualization. In other words, xBIM can be used to create bespoke BIM middleware for IFC-based applications."
Key features:	extensible and open source IFC 2.3 visualization toolkit written in c# and c++ with Visual Studio 2010. Based on GPPG-Tools, Open Cascade and the Helix 3D Toolkit. A sample viewing application and a web based JSWebViewer is available. Reads and saves own xBim file format. Sample viewing app provides a 'Model Query' functionality. Nice translucent visualization option.
IFC tests:	Problems reading ".ifc XML" format, relative slow import.

Fig. 7.3: Brief technical description of the XBIM tool



IFC Engine DLL: Sample Viewer

Author:	Peter Bonsma, RDF Ltd., Sofia, Bulgaria
Licenses:	This IFC viewer is free to use and redistribute both commercially and non-commercially. It is built around the IFC Engine DLL. All source code is available. If needed also the complete source code of the IFC Engine DLL is available, this requires a company-wide license.
Platform:	Windows
IFC formats:	".ifc STEP" and ".ifc XML" format up to Version IFC 4
Description:	text from website: „The IFC Engine DLL is a STEP Toolbox with ability to generate 3D geometry for popular versions of the IFC schema. The component is able to load, edit and create Step Physical Files (as well as the XML notation) and their schema's via its own object database. This includes all currently available IFC versions. For IFC 2x3 files and also for IFC 4 the geometry is generated.
Key features:	IFC 2.3 and IFC 4 geometry toolkit written in C++ and Java without the use of additional tools. The RDF-Viewer is a sample application using the IFC Engine DLL. It provides functions to edit ifc-properties, connect to a server and to check for clashes.
IFC tests:	fast import, but problems reading the .ifc XML test files.

Fig. 7.4: Brief technical description of the IFC Engine DLL Toolkit

7.3 Importing BIM Data to BPS

This section describes the transformation from Building Information Modeling to Building Performance Simulation in Modelica. Starting with a first and initial draft of the data transformation process adopted in this Annex, requirements for BPS in different stages of the building design are described. We detail the possibilities and requirements to generate Building Information Models with a focus on BPS in this section. The section further illustrates how to extract related information from a BIM and transforming it to be used as basis for BPS, respectively.

The use of Building Information Modeling implies interoperability between software applications and the collaboration of different disciplines. Fig. 7.5 illustrates the process how to derive Modelica models for different Modelica libraries from BIM. Each step requires different actors and software. The architect and HVAC engineer use domain specific BIM-based CAD software and may export their models to an Industrial Foundation Classes (IFC) file (left side). This file is converted into SimXML which is the file format for SimModel, a data model for the simulation domain. SimModel extends information provided by IFC with simulation specific parameters. The SimXML file is then converted to a valid Modelica model with the software framework developed in this Annex. This framework is setup to support multiple Modelica libraries as well.

This section puts energy modeling into the context of Building Information Modeling, including IFC, SimModel and tools to check the BIM for integrity.

7.3.1 Overall Transformation Process

For the task of converting BIM into Modelica models (as outlined in Section 7.1), we used a case study approach for the development of the tool chain (Section 7.4).

Fig. 7.6 illustrates the data transformation process within the developed methodology, in which the IFC data model is instantiated in the BIM Platform and exported as an IFC4 file. In order to read and operate with these data, the

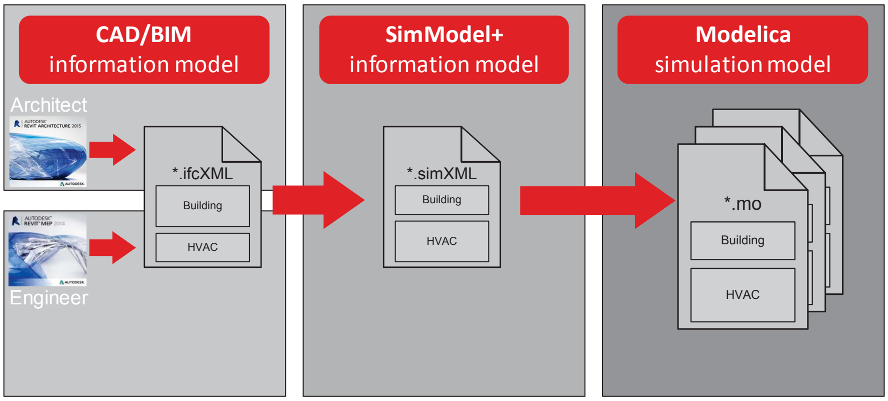


Fig. 7.5: *Overview of the transformation process from IFC to Modelica models using different Modelica libraries.*

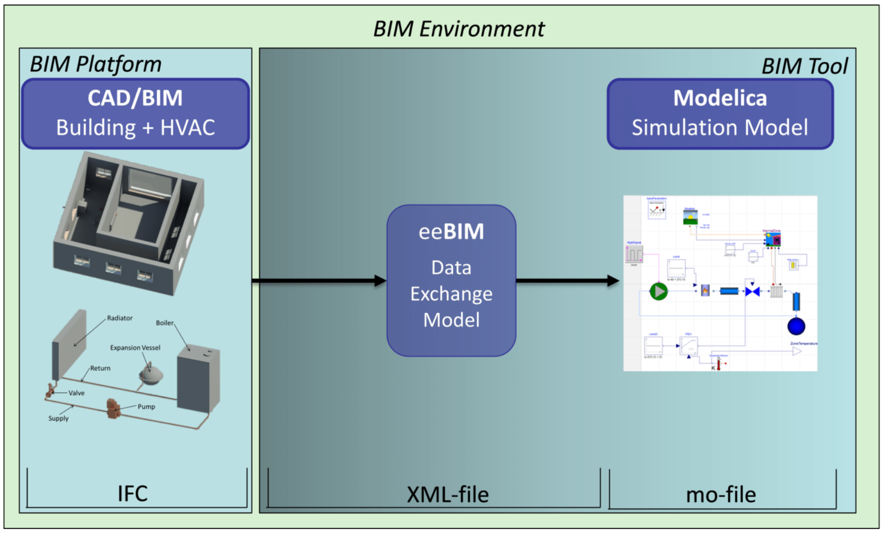


Fig. 7.6: *Data transformation from BIM to Modelica*

model is converted into a specific exchange model for BPS. SimModel is such a data exchange model specifically developed for BPS and thus serves as the intermediate format to support the exchange of data from BIM to Modelica in this Annex. SimModel offers a flexible data model allowing enhancements and additions as described in Section 7.3.6. The final process in the tool chain generates a Modelica file.

The conceptual and software development of this methodology was driven by a set of simple use cases (described in Section 7.5).

7.3.2 Requirements for BPS Models Across Different Stages of the Building Design Process Based on Uncertainty

BPS models can be used at different stages of the design process, for example to investigate the thermal comfort and energy consumption of a building or to support the planning of the HVAC configuration and controls. Depending on the respective application of a BPS model and the corresponding design phase, different requirements must be satisfied in terms of the nature of the model and the associated input data.

It is important to know the level of detail which is required at each design stage. This allows for the BPS engineer to generate reliable results. Here, it has to be taken into account that simulation results always comprise some extent of uncertainty. This uncertainty arises due to unknowns within the inputs of the simulation model. Certain input uncertainties decrease in the course of the design process, particularly as detailed planning proceeds and available information is increasing. Other input uncertainties are more or less constant over the entire planning process, for example: climate conditions or user behavior. The challenge is to identify which level of detail, represented by subsets of data contained in the BIM, is necessary for a BPS model at a specific stage of the design, which is a crucial consideration given the corresponding uncertainties at that stage.

To provide answers to this challenge, methods for uncertainty and sensitivity analysis can be used. In *[BJH10]* a methodology is described which allows for consideration and evaluation of uncertainties in BPS models using Model-

Table 7.1: Overview of the uncertain input parameters and their respective distributions.

Parameter	Unit	Minimum	Maximum	Distribution
zoneParam.RRest	K/W	0.03	0.05	uniform
zoneParam.R1o	K/W	0.0003	0.005	uniform
zoneParam.C1o	J/K	1.0e06	2.0e06	uniform
zoneParam.Aw	m^2	5	9	uniform
zoneParam.g		0.5	0.7	uniform
pipe.dp	Pa	5000	20000	uniform
valve.dp	Pa	5000	20000	uniform
flow.temperature	℃	70	90	uniform

ica. Additionally, the methodology can be used to determine the impact of the uncertain inputs on the output uncertainty (sensitivity analysis) and, thus provides a means to rank model inputs according to their relative importance in terms of the simulation result. The most sensitive inputs are highly relevant for providing valuable simulation results while less sensitive inputs (lowest ranking) have little influence on the simulation results.

Accordingly, one can argue that a simulation model should account for the most sensitive inputs (even if the respective inputs are not exactly known) with respect to the output of interest. Less sensitive inputs can be initialized with default values or ignored in the model.

In the following, the procedure for uncertainty and sensitivity analysis is briefly explained and illustrated using the example of use case 1.1 (Section 7.5.1) (standard office building modeled as a simple zone with radiators and a boiler). From the results of this type of analysis one can estimate the output uncertainties and deduce BPS model requirements for different design stages.

The first step is to identify the uncertain model input parameters and estimate their respective distributions. Here, we assume that building parameters are not fully known, which leads to uncertainties in the heat demand, and the exact HVAC layout has not been specified yet. This situation can, for example, appear in the outline planning when first simulations are run, in order to support code compliance or later certifications, while significant uncertainties have to be considered.

The assumed input parameter distributions are shown in Table 7.1. Here, the first five lines correspond to building (zone) parameters, while the last three lines correspond to HVAC parameters. RRest, R1o and C1o describe thermal resistances and a thermal capacity in the reduced order building model according to VD16020. These parameters cannot be directly related to physical quantities in the building, but represent aggregates. Aw and g denote the area and the solar transmittance of the windows, respectively. The parameters valve.dp and pipe.dp are associated with the pressure losses of valves and pipes.

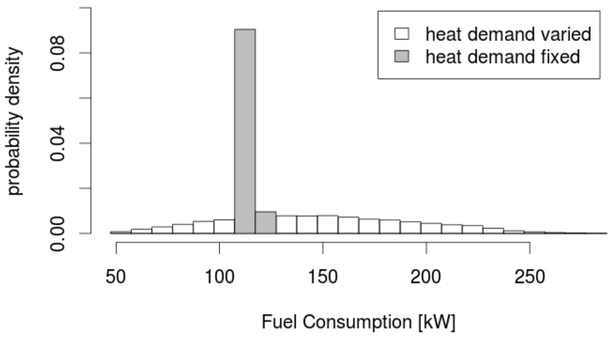


Fig. 7.7: *Distribution of the fuel consumption over all simulation runs. White bars correspond to a varied heat demand according to the building parameter distributions given in Table 7.1. Gray bars correspond to a fixed heat demand and only HVAC parameters are varied according to Table 7.1.*

Then, Monte Carlo Simulations are performed, where for each uncertain input parameter 4096 random samples from the given range are chosen. The simulation period is six weeks, starting from 1st of January. The considered output is the total energy consumption of the HVAC system which consists of the fuel consumption of the boiler and the electricity consumption of the pump, summed over the simulation period.

Fig. 7.7 and Fig. 7.8 show the resulting distributions of the outputs, fuel consumption and pump power, when the input parameters are varied according to Table 7.1 (white bars). For comparison, the corresponding distributions are shown when the heat demand is assumed to be known (associated with fixed building parameters) and only the HVAC parameters are varied (gray bars). It can be seen that the uncertainty in heat demand has major influence on the total energy consumption: the estimate of the total energy consumption is much more precise when the heat demand is known. The remaining, relatively small, uncertainty if the heat demand is fixed results from the uncertainties in the HVAC layout. In comparison to the building parameters, the HVAC layout plays a negligible role for the overall output uncertainty.

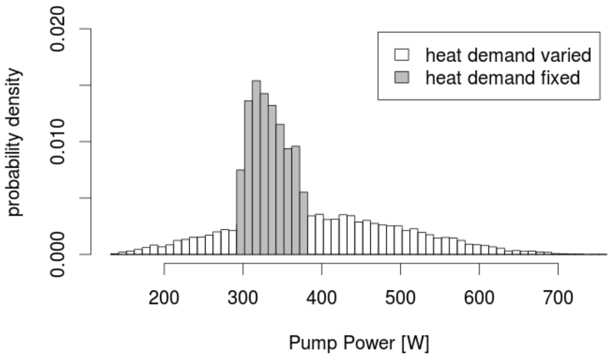


Fig. 7.8: *Distribution of the pump power over all simulation runs. White bars correspond to a varied heat demand according to the building parameter distributions given in Table 7.1. Gray bars correspond to a fixed heat demand and only HVAC parameters are varied according to Table 7.1.*

This effect can be quantified via the calculation of sensitivity indices. The variance-based first order sensitivity indices for all varied input parameters with respect to fuel consumption and pump power are shown in Fig. 7.9. A higher index indicates a greater influence on the observed output uncertainty. Also from this figure, it becomes clear that, in this setting, the building parameters have a greater impact on the energy consumption than the HVAC layout. In particular, the thermal resistance parameter of the simplified zone model and the window area turn out to be most influential.

So, one can deduce from these results that for a reliable estimate of the energy consumption the heat demand should be known relatively well, while a rough picture of the HVAC system configuration is sufficient at this stage. When it comes to optimizing the operation of the HVAC system, having eliminated other uncertainties to large extent, the exact HVAC layout becomes important.

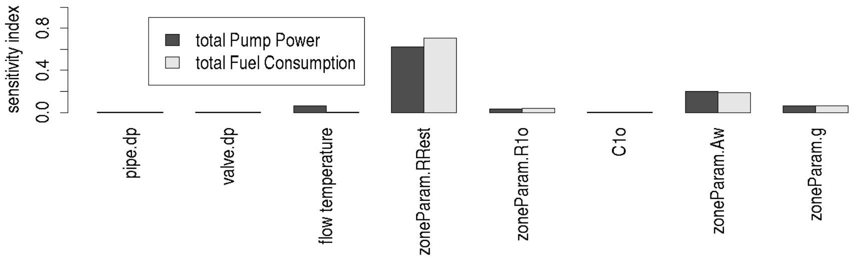


Fig. 7.9: *Sensitivity indices for the outputs fuel consumption and pump power with respect to the varied input parameters. The variation of the building parameters has much more effect on the outputs than the variation of the parameters concerning the HVAC layout.*

In a similar way, one can identify the following model requirements, based on the simple use case 1.1 (Section 7.5.1), for four different design stages:

7.3.2.1 Preliminary Planning

A simple building simulation model is used to estimate the energy demand of the designed building. Uncertainties in weather conditions, internal loads and building parameters (e.g. materials, areas) have to be taken into account. The resulting energy demand yields a probability distribution.

The resulting energy demand distribution from the building simulation is used to perform an HVAC simulation and to support the HVAC planning. Therefore, a simple HVAC model is used which only comprises the boiler and radiator, and yet uses the uncertain energy demand as input. If applicable, several different HVAC concepts with alternative heat sources or heat exchangers are compared. A first estimate of the energy consumption in terms of a probability distribution is provided.

7.3.2.2 Outline Planning

As more details concerning the building geometry and the materials are known, a corresponding building simulation yields a more precise heat demand, which, however, still comprises some uncertainty due to unknown internal loads and weather conditions.

An HVAC simulation is used to determine the optimal dimensions for the boiler and radiator. Uncertain heat demand, unknown set values and controls have to be taken into account. A model-based optimization of set values (e.g. flow temperatures) and controls (e.g. night set-back) could be pursued. As a result of the HVAC simulation, the estimated energy consumption is obtained and serves as a basis for code compliance and certification.

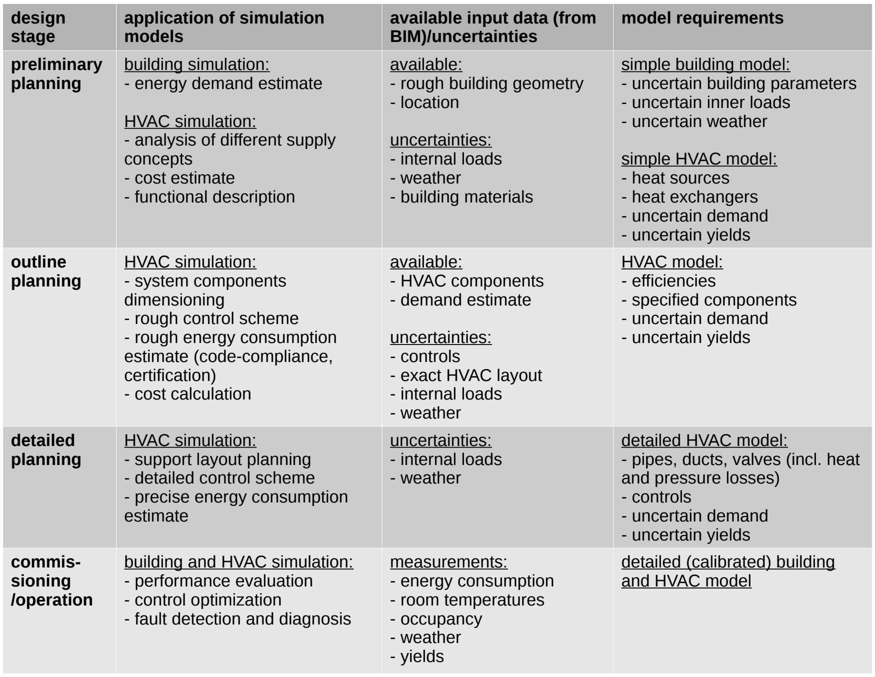
7.3.2.3 Detailed Planning

A detailed HVAC simulation is used to optimize the HVAC configuration and controls. Pipes and valves are included in the model. Uncertain heat demand, unknown pressure and heat losses are taken into account. As a result of the HVAC simulation a more precise distribution of energy consumption is obtained and serves as a baseline for building operation.

7.3.2.4 Operation Phase

A detailed as-built building and HVAC simulation model is calibrated with measurement data. The calibrated model is used for model-based on-line fault detection and optimization or model-based control.

The BPS model requirements for this simple example arise from a sensitivity analysis. For more elaborated systems with storage and renewable energy usage the picture could change slightly. In particular, controls would have a more important role to play during the early design as they tend to have a large impact on the energy consumption of innovative or low energy systems.



design stage	**application of simulation models**	**available input data (from BIM)/uncertainties**	**model requirements**
preliminary planning	building simulation: - energy demand estimate HVAC simulation: - analysis of different supply concepts - cost estimate - functional description	available: - rough building geometry - location uncertainties: - internal loads - weather - building materials	simple building model: - uncertain building parameters - uncertain inner loads - uncertain weather simple HVAC model: - heat sources - heat exchangers - uncertain demand - uncertain yields
outline planning	HVAC simulation: - system components dimensioning - rough control scheme - rough energy consumption estimate (code-compliance, certification) - cost calculation	available: - HVAC components - demand estimate uncertainties: - controls - exact HVAC layout - internal loads - weather	HVAC model: - efficiencies - specified components - uncertain demand - uncertain yields
detailed planning	HVAC simulation: - support layout planning - detailed control scheme - precise energy consumption estimate	uncertainties: - internal loads - weather	detailed HVAC model: - pipes, ducts, valves (incl. heat and pressure losses) - controls - uncertain demand - uncertain yields
commis-sioning /operation	building and HVAC simulation: - performance evaluation - control optimization - fault detection and diagnosis	measurements: - energy consumption - room temperatures - occupancy - weather - yields	detailed (calibrated) building and HVAC model

Fig. 7.10: Overview of the usage and requirements of simulation models in different design stages.

An overview of the use of simulation models in different design stages is given in Fig. 7.10, as is the availability of relevant input data for each respective phase and the data requirements for the BPS model. Although this representation is not exhaustive and can certainly not be applied to all building processes, the image indicates typical dependencies between available information and targeted BPS results. As a result, this information can serve as a basis for *Information Delivery Manuals (IDM)* as described in sub:numref:*sec_IDM*.

7.3.3 Building Geometry Generation and Processing

Building geometry definitions for use in Building Energy Performance (BEP) simulation modeling usually originate in CAD software. They are defined by CAD in 2D, 2.5D or 3D, and are delivered in electronic form or on paper.

Geometry information on paper (drawing prints) generated by 2D CAD tools, such as AutoCAD *[Aut16]*, must be manually transcribed into electronic geometry definitions that can be read by BEP simulation programs and engines. This is a very tedious, time-consuming, frustrating and extremely error-prone process that can, depending on the size and complexity of the modeled building, consume more than a half of the total simulation project budget.

If the 2D building geometry is available in electronic format (DWG or DXF) which contains scalable dimension information, the transcription into geometry definitions readable by BEP simulation programs and engines can be a bit faster (using copy-and-past) and somewhat more efficient. Still, the transcription requires heavy human modeler involvement.

State-of-the-art 2D CAD software, such as AutoCAD, can provide the third dimension by extruding the information defined in 2D plans vertically. Such geometry is called 2.5D, as it is not true 3D geometry. While 2.5D geometry can further expedite the geometry transcription for use in BEP simulation, it can have the opposite effect when different adjacent buildings have different extrusion length.

“Model-based” architectural CAD tools, such as Revit *[Aut15]*, ArchiCAD *[Gra16]*, Allplan *[Nem16]* and MicroStation *[Ben15]*, offer definition of building geometry in true 3D formats – geometry definitions that are object-oriented. Each of these CAD tools exports geometry in its own proprietary file format, but also includes utilities for export and import of geometry in IFC format. Third party software, such as the Space Boundary Tool *[RB13]* and Simergy *[Alc16]*, can automatically or semi- automatically transcribe geometry defined in IFC format into input for EnergyPlus. Bentley’s AECOsim Building Designer *[Ben15]*, transcribes building geometry originally created by MicroStation for seamless use in EnergyPlus. The OpenStudio platform can import building geometry defined by SketchUp (not a “model-based” CAD tool) or other CAD tools that export geometry in gbXML format *[GbX16]* and use it directly in EnergyPlus without any additional transcription.

Most CAD tools allow the modeler to define "center-line" building geometry models in which walls, floor slabs, ceiling slabs, roof slabs, windows and doors are represented by flat surfaces parallel to the sides and placed in the center of each particular object; "center-line" representations have no thickness. While "center-line" modeling simplifies the modeling process and may save time, "center-line" geometry definitions result in BEP simulation results that significantly overestimate building energy consumption *[BMNG16]*.

Building geometry definitions as required for use in detailed Modelica geometry algorithms should be as precise as possible. Any current high-end "model-based" CAD tool can facilitate the necessary modeling precision; it is *up to the modelers* to define building geometry *precisely* *[BMNG16]*. Thus it is paramount to always verify the correctness and precision of any generated building geometry.

7.3.4 Energy Modeling Using IFC

IFC based BIM has the potential to provide significant other inputs for BPS modeling, thus reducing the time, effort and expense associated with model creation *[RB13][vTR06]*. To date the majority of work in the area of BIM to BPS has focused on geometry transformations *[Sim15][GLG+15][Gra11][JKC+16][JS16][OMR+13]*. Commercial BPS tools such as RIUSKA *[Gra11][LOK10]*, Simergy *[SHS+11]*, IDA-ICE *[Sim15]* and TRNSYS *[CRR+11]* focus mainly on the import of geometrical information. Section 7.3.3 describes geometry transformations in detail but such transformations require high quality IFC models which are typically not delivered in practice. Previous efforts such as the Design BIM 2010 (*[BDA12]*) MVD development did go beyond just geometry data, in adding internal loads data as well as high level data on space requirements and HVAC systems (CDB2010 is supported by Simergy).

In addition to geometric information, HVAC, controls, operating schedules and simulation parameters should be contained within a BIM. However, when mapping HVAC systems to BPS tools, a number of complex and interrelated issues arise. Primarily, the broad variation in representation of HVAC systems within software tools results in bespoke mapping solutions from BIM to each target BPS engine. As an example, HVAC duct designers typically use a supply and return convention for loop structure while EnergyPlus uses a supply and demand loop structure. Modelica on the other hand, only requires system topology information and a different parameter set compared to parameters typically entered by HVAC system designers.

At the data level, many of the objects required for BPS are not contained within IFC based BIM or are not inserted into IFC by appropriate BIM based CAD tools. This shortcoming is particularly noticeable in the HVAC controls domain where the object and property definitions required by simulation are not yet rigorously defined and have not been in focus of data exchange developments. As a result a formal definition of the data requirements for BPS would significantly assist data transfer to the BPS domain and MVD is an ideal platform to service this need.

7.3.5 Model View Definition (MVD) for Energy Related BIM Support

In order to facilitate data exchange between applications using the IFC schema, an *MVD* was developed to meet the requirements needed by BPS tools. The MVD focuses on the definition of IFC entities, attributes, relations and properties that satisfy this specific exchange scenario. This defines a subset of the building product model schema that provides a complete representation of the information concepts needed for a particular use case in an AEC/FM workflow *[PWM+16]*.

The MVD also defines the business rules and agreements necessary to assist the implementation of import and export functions by BIM applications. Thus, a first quality assurance can be done by checking for mandatory entities and property sets. In cases where information is not currently represented in the IFC specification, properties can be defined to extend the IFC schema.

7.3.5.1 Available MVDs and Current Status

In 2016, buildingSMART published the IFC4 Addendum 2 (IFC4 Add 2), together with two new MVDs to support IFC4 implementation: Reference View (RV) and Design Transfer View (DTV). The main purpose of the first one is to define a standardized subset of the schema that supports exchange in one direction only. The RV is characterized by the ability to provide workflow for the widest array of software applications. The second MVD proposed is an extension of the Reference View. Contrary to the RV, the DTV enables model editing by design software platforms. DTV is considered the successor of the former IFC2x3 Coordination View and is compatible with IFC2x3 import *[Bui16]*.

Since the introduction of the MVD concept, organizations started developing their own MVDs to support internal processes based on the IFC2x3 schema. There are some related to energy performance such as: Holistic Energy Efficiency Simulation and Management of Public Use Facilities HESMOS *[LSW+13]* and Concept Design BIM 2010 (CDB) *[BDA12]*. Fig. 7.11 shows a short comparison of the energy related MVDs. The HESMOS project did actually not define an MVD, the project focused on the definition of exchange requirements (ER) only. HESMOS and CDB focused on the building envelope, but added some additional HVAC related data as a first step towards a simulation MVD. CDB MVD requires 2nd level space boundaries, whereas HESMOS only demands first level space boundaries. The HESMOS project developed a conversion tool that transforms from 1st into 2nd level space boundaries. Both projects attached the internal loads to a space. HESMOS covers most of the relevant data for a static simulation, whereas CDB's focus lays on a more general definition of the energy analysis zone. HVAC systems are only described as existing flags or with basic parameters. HVAC components itself are completely missing. In addition to that, renewables like photovoltaic components are part of the simplified definitions.

7.3.5.2 Identification of Data Requirements and Mapping to IFC

As shown in Section 7.3.2, the exchange requirements for energy simulation can be different at different stages of the building life cycle. Within this project, several use cases were defined using different levels of complexity from a single room to a large commercial building. The main purpose of creating these use cases is to analyze several HVAC systems that are used in the building sector. Based on these use cases, it is possible to identify the scope of information needed by BPS tools and subsequently map it to the IFC schema. This process involves the IDM methodology (Section 7.4.1), which is a standardized methodology to create information exchange requirements *[WK10]*.

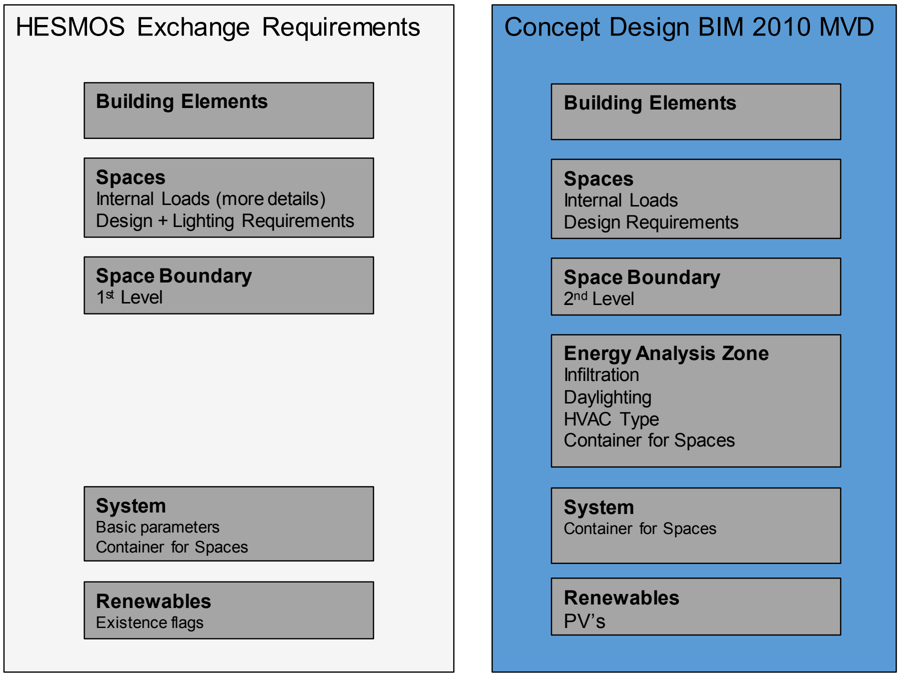


Fig. 7.11: Contrasting juxtaposition of existing energy related ER/ MVDs.

Working with proprietary software to generate IFC files we were bound to the content it offers. In this project, we used mainly the IFC4 version due to its added content in the HVAC component area. During the runtime of the Annex 60 project, only limited IFC functionality existed since official buildingSMART certification of IFC4 was not yet available. Through our cooperation with software vendors during the project significant effort is put into the support of IFC4 export. Over time better support for IFC4 will become available. Besides these described difficulties originating in the early stage of the implementation of the IFC4 data model, this also provides an opportunity for feedback and possible influence on further development.

In general, new versions of the IFC schema provide new entities to cover the wider scope and support additional areas of the AEC industry. Along with new entities, existing entities can change and in special cases even be deprecated. These changes have to be incorporated into the import and export functionalities of software which is then submitted for certification. Additionally, IFC supports property sets for the export of product characteristics. Some tools support these property sets with corresponding library objects, while other tools may only provide the properties through manual input. Once certified, a detailed certification report can be found on the buildingSMART website listing the level of support for individual entities.

7.3.5.3 Specification of the MVD

An MVD is composed of the following elements: ModelView, ConceptRoots, ExchangeRequirements, ConceptTemplates and Concepts. The ModelView is the description of an MVD and is specific to an IFC schema release. It groups zero-to-many ExchangeRequirements and ConceptRoots thereby defining the scope of the MVD. The ExchangeRequirements define the information necessary for a particular exchange scenario and may add additional constraints to the use of Concepts. The ConceptRoot is represented by a collection of available Concepts, each of which references a specific IFC entity, e.g. IfcSpace. Each Concept describes rules for common subsets of information (e.g. space attributes) within the context of the particular root. The Concept is backed by a ConceptTemplate describing a graph of object instances, relationships and constraints. The information contained inside of the ConceptTemplate enables the generation of instance diagrams of the MVD *[CLW13]*. The MVD is defined by using the open source tool ‘IFC Documentation Generator’, provided by buildingSMART ([bib]). The information defined in the existing energy related MVDs are integrated in the creating of the new MVD.

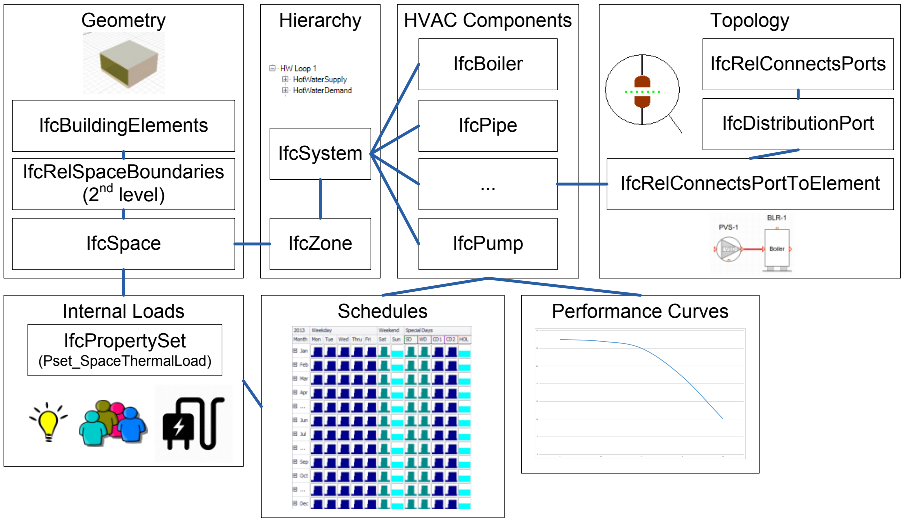


Fig. 7.12: Overview of the main topics to define the energy related MVD.

The fundamental elements for an energy related MVD are shown in Fig. 7.12.

Geometry: The Geometry defines the basis of the model. Second level space boundaries link spaces to building elements and are used as basic surfaces for the heat transfer between spaces internally and between spaces and the outdoor. These should be well defined and correct (see also Section 7.4.3.1).

Internal loads: Space contains the definition of the internal loads as property sets. IFC4 already has designated properties for internal loads as follows:

- Maximum/Minimum temperature of the space
- Maximal number of people
- The total sensible heat or energy gained
- Lighting loads
- Percent of sensible load to radiant heat

These simplified properties originate from earlier MVDs. Most importantly the dynamic nature of schedules is hereby missing which is crucial for BPS. Schedules in general are discussed next.

Schedules: The ability to define complex schedules in combination with the already defined properties allows for example the definition of detailed internal loads. BPS defines this schedule requirement that has not been introduced by other disciplines. Currently in IFC, schedules can be defined in three different ways:

- IfcWorkCalendar: On/off status through the year with predefined patterns
- IfcRegularTimeSeries: A regular time series that defines values based on a regular time interval (e.g., hourly).
- IfcIrregularTimeSeries: A time series that defines values on a irregular time series.

Typically, schedules for BPS follow a regular pattern, mostly daily, weekly and seasonal patterns are used. The ability to define patterns would simplify and reduce the data representation of schedule significantly. The IfcWorkCalender uses the IfcRecurrancePattern that is able to define such patterns, but can only contain an on/off signal and no values. For example, the presence of a certain number of people in the thermal zone on a weekday, which changes over the course of the day, or the opening range of a valve on a national holiday cannot be described with a Boolean flag. This is a significant limitation of the IFC model and creates the need for a workaround solution.

In order to enable the definition of these schedules, it is necessary to define a new concept. The IFC Documentation Generator (IfcDoc tool) is used to define the new concepts, based on the newest IFC4 data model. The top level of the concepts hierarchy is IfcObject. IfcObject is best suited for applying schedules to each possible spatial and non spatial element, such as a zone, space or HVAC component. The object is assigned via a relating control to the IfcPerformanceHistory. IfcPerformanceHistory offers the recurrence patterns, using the entity IfcWorkCalender. IfcPerformanceHistory is also defined by properties, which in turn allow the definition of complex properties and ir/regular time series. If IFC 4 allows the definition of recurrence patterns for time series, the application of detailed schedules on an hour basis over the course of a day is possible. To improve this limitation, we suggested changes for the coming version of the IFC data model. The application of this concept allows for dynamic simulation using detailed schedules of the typically complex properties for a thermal zone.

Hierarchy: The hierarchy for the thermal zones is defined using the composition concept, provided by buildingSMART. IfcSpace is the lowest level, composed to IfcBuildingStorey, which is in turn composed of IfcBuilding. IfcZone is used as a grouping mechanism, which allows for a definition of the thermal zones in combination with the space boundaries. The HVAC system is defined using IfcSystem as a top hierarchy entity. The system can be separated into subsystems. For a single loop heating system, the distribution system would be defined as supply and return pipe systems in which the radiators represent a thermal sink and the boiler a thermal source, which are also part of the subsystems. No previous agreements exist on the hierarchy of systems that is why we are proposing a simple hierarchy with one parent system for each loop that contains a supply and demand subsystem. For simulation purposes this differentiation can be useful to know which components are supplying a loop and which are demanding cooling or heating needs.

HVAC components: In the latest IFC4 version, several enhancements to the HVAC component were made. In particular, components definitions became more detailed. E.g., an IfcBoiler used to be a generic IfcEnergyConversionDevice. This great enhancement makes the representation of HVAC component more explicit. However, there are a number of HVAC component that did not get a more detailed class, such as heat pumps and combined heat and power units. Those need to be represented with a more generic class IfcUnitaryEquipment.

Topology: Since most HVAC systems are based on some kind of fluid flow, topology connections are one of the key concepts to consider. In IFC this is done by defining an IfcDistributionPort, which is connected to a IfcDistributionElement. The IfcRelConnectsPorts relationship connects two ports and forms the basis for the topology.

Performance curves: Another aspect of HVAC components are performance curves. Typically, performance curves are used to describe specific behavior of a component at varying conditions. E.g., a boiler efficiency curve that depends on the water return temperature. In IFC4 this concept does not exist. We did realize the definition of performance curves with the use of IfcComplexProperties which in turn is a list of IfcProperties. This list of properties then defines the coefficients of the curve, minimum and maximum values as well as other relevant properties.

	Object Type Definitions	Project Units	Project Representation Context 2D	Project Global Positioning	Project Document Information	Object Typing	Property Sets for Objects	Property Sets for Types	Quantity Sets	Software Identity	Revision Control	User Identity	Site Attributes	Building Attributes	Storey Attributes	Space Attributes	Door Attributes	Windows Attributes	Element Type Predefined Type	Material Single	Material Layer Set	Material Layer Set Usage	Material Profile Set	Material Constituents	Spatial Composition	Spatial Decomposition	Port Nesting	Type Port Nesting	Element Voiding	Distribution System Assignment	Zone Assihnment	Spatial Container	Spatial Containment	Space Boundaries 2nd Level	Path Connectivity	Port Connectivity	Product Shape	Product Local Placement	Product Type Geometric Representation	Internal Loads-Schedules
IfcBoiler	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	□	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcBuilding	■	■	■	■	■	□	□	■	□	□	□	□	■	□	■	■	■	■	■	□	□	□	□	□	□	□	■	■	■	■	■	□	■	■	■	■	□	□	■	□
IfcBuildingStorey	■	■	■	■	■	□	□	■	□	□	□	□	■	■	□	■	■	■	■	□	□	□	□	□	□	□	■	■	■	■	■	□	■	■	■	■	□	□	■	□
IfcDistributionElement	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	□	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcDistributionElementType	■	■	■	■	■	■	■	□	■	□	□	□	■	■	■	■	■	■	□	□	□	■	□	□	■	□	■	□	■	■	■	■	■	■	■	■	■	■	□	■
IfcDistributionPort	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	■	■	■	■	■	■	■	■	■	□	□	□	■	■
IfcDistributionSystem	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	■	□	□	■	□	■	■	■	□	■	■	■	■	■	■	■	■	■	■
IfcDoor	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	□	■	■	□	□	□	□	□	■	□	■	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcElement	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	■	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcElementType	■	■	■	■	■	■	■	□	■	□	□	□	■	■	■	■	■	■	□	□	□	■	□	□	■	□	■	■	■	■	■	■	■	■	■	■	■	■	□	■
IfcMaterial	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
IfcOpeningElement	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	■	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcPipeFitting	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	□	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcPipeSegment	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	□	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcProduct	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	■	■	■	■	■	■	■	■	■	■	□	□	■	■
IfcProject	□	□	□	□	□	■	■	■	■	□	□	□	■	■	■	■	■	■	■	□	□	■	□	□	■	□	■	■	■	■	■	■	■	■	■	■	■	■	■	■
IfcPump	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	□	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcRoot	■	■	■	■	■	■	■	■	■	□	□	□	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
IfcSite	■	■	■	■	■	□	□	■	□	□	□	□	□	■	■	■	■	■	■	□	□	□	□	□	□	□	■	■	■	■	■	□	■	■	■	■	□	□	■	□
IfcSlab	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	■	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcSpace	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	□	■	■	■	□	□	□	□	□	□	□	■	■	■	■	■	□	■	□	■	■	□	□	■	□
IfcSpaceHeater	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	□	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcSpatialElement	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	□	□	■	■	■	■	■	□	■	■	■	■	□	□	■	□
IfcSpatialZone	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	□	□	■	■	■	■	■	□	■	■	■	■	□	□	■	□
IfcSystem	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	■	□	□	■	□	■	■	■	■	■	■	■	■	■	■	■	■	■	■
IfcTank	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	□	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcValve	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	□	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcWall	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	■	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcWallStandardCase	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	□	□	□	■	□	■	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcWindow	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	□	■	□	□	□	□	□	■	□	■	■	□	■	■	■	□	■	□	■	□	□	■	■
IfcZone	■	■	■	■	■	□	□	■	□	□	□	□	■	■	■	■	■	■	■	□	□	■	□	□	■	□	■	■	■	■	□	■	■	■	■	■	■	■	■	■

Fig. 7.13: Overview of the concepts used to define the MVD for use case 1.1.

The developed MVD is based on well-defined ConceptTemplates provided by buildingSMART [bia]. For instance, the definition of properties and connections of HVAC components were defined using ConceptTemplates such as "Port Nesting" and "Property Sets for Objects". Fig. 7.13 shows the concepts applied to a subset of the MVD. On the left hand side are the objects listed that are used for the use case 1.1. The top shows a list of the used concepts for the MVD. The white boxes show that the concepts can be applied to the object and the grey boxes show that the concepts are not applicable.

As already mentioned the 2nd level space boundaries are mandatory for the MVD and this concept is assigned to IfcSpace. The concepts “Port Connectivity” and “Port Nesting” are necessary to connect the Ports on distribution elements and to define placement, indicating the position and outward orientation. The different materials of the numerous entities are also defined using the given concepts, such as “Material Layer Set” or “Material Constituent”. If something is undefinable using this method, an Implementer Agreement can be used (see Section 7.2.3).

It is important to notice that the process of certification is done on the basis of MVDs and not the general IFC schema. This process requires time and depends on buildingSMART as well as software developers. However, there is already some work being done to provide IFC4 support within 2016.

7.3.5.4 Outlook

The MVD developed in the Annex 60 project applies the methodology from buildingSMART, and uses the recommended ifcDoc tool for its formal and descriptive definition. IfcDoc enables the export of HTML documentation in the mvdXML format (official BSI format) for software certification and the generation of a subset schema containing all relevant entities and properties of our MVD.

Available ConceptTemplates are re-used as far as possible. While it covers all requirements identified by the various use cases, the MVD has been designed to be a general purpose view for thermal simulations, i.e. it is not restricted to the Modelica simulation package. Since the MVD is developed based on our set of use cases it has a limited scope in terms of HVAC objects, but it contains all key concepts related to BPS. Feedback on what is missing in the IFC4 data model from a BPS perspective as well as a description and discussion of key concepts and issues form important steps towards a first BPS MVD.

Besides the MVD itself, we also discovered a number of inconsistencies or missing concepts in the IFC data model.

- Missing pattern definition for schedules
- Missing explicit classes for heat pumps and CHPs
- Missing performance curve concept

- Missing agreement on system hierarchy

The next step will be to submit the MVD proposal to buildingSMART for further review by the community as well as to extend its scope to cover all relevant HVAC objects. As the current proposal already includes definitions from previous work, in particular the space boundary add-on view to fill the current generic simulation MVD gap. In parallel to the review period the software certification process can be prepared, which means to specify test cases and expected quality criteria. For this, the developed use cases and prepared example data can be used as a starting point. They already cover the main aspects of our use cases, but may need to be extended to check further aspects of the MVD.

7.3.6 SimModel: A Data Model for BPS

SimModel is primarily used as an internal data model by the Simergy software developed at LBNL and continued by Digital Alchemy *[LBN13][SHS+11]*. This tool was first conceived as a platform that facilitates data flow to and from BPS simulation tools to and from potentially any building modeling tool *[BMR+11]*. Bi-directional data flow is possible to and from IFC BIM, DOE-2 software or tools that use the DOE-2 engine, EnergyPlus, and tools with gbXML export (Fig. 7.14). These tools are typically used for BPS simulations. Data from any of these environments can be mapped to and from the SimModel data model using the Simergy software *[LBN13]*.

SimModel is an object-oriented data model which defines all object /attribute/relationship sets used for BPS. The primary objective of SimModel is to accommodate the existing input data requirement of EnergyPlus, while allowing mapping from/to other domain data models and easy incorporation of new definitions *[BMOD+11]*. At its core, SimModel is represented using the XML markup language *[OSM+11]*. This representation is closely aligned to the IFC data model, in order to link to incoming or outgoing IFC information (Fig. 7.14), a motivating factor for using SimModel in this project. In this context, SimModel also reduces some of the complexity of the IFC data model by simplifying relationship objects though direct object references.

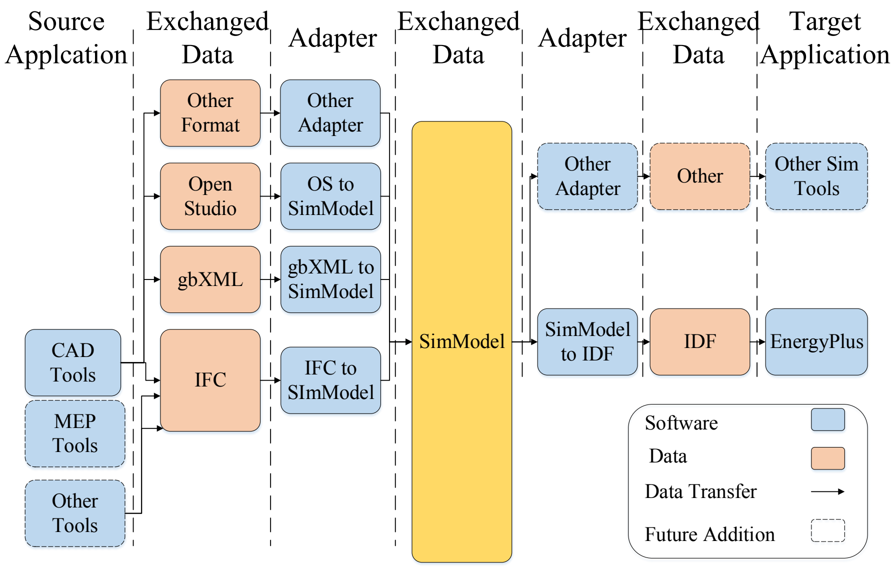


Fig. 7.14: Interoperable data exchange enabled by SimModel. This solution enables re-use of original project data as contained in IFC based BIM (bi-directional mapping) and data from other sources (import only)

7.3.6.1 SimModel Design

SimModel incorporates a number of features that address current domain weaknesses (as detailed in [OSM+11]), a set of requirements for a shared simulations model and is easily extensible to account for future domain advances. This data model ensures interoperable exchange of simulation data within the simulation domain and most importantly across an entire building project. The unique design enables interoperable data exchange and uses a number of features to do so, these include: mappings to/from existing domain models; structured yet flexible class definitions; property set definitions; object type definitions; model ontology; templates and resources. The following subsections detail only the key features.

7.3.6.2 Data Model Mappings

The data model should facilitate seamless data exchange for the extended building-simulation domain and even for the entire AECOO industry [NRE10]. Version 1.0 supports BIM concepts from IFC, gbXML, IDF, and OpenStudio [NRE10]. Version 2.0 adds support for SDD (Standards Data Dictionary) to enable the foundation for code compliance analysis (here California Title 24) ([MHS15]). The quality of data varies with each file format so customized adapters enable single or bi-directional mappings on a case-by-case basis. SimModel can also accommodate bi-directional mapping to other data models as the need arises, e.g. IDA-ICE or IES-VE.

7.3.6.3 Class Definitions

Data element/entity ontologies vary greatly between SimModel, the EnergyPlus schema called Input Data Dictionary (IDD) and gbXML. The EnergyPlus IDD contains ~650 element types and other relevant data model schemas contain several hundred classes. SimModel uses approximately 120 data model classes to represent the merger of the EnergyPlus IDD and other model schemas. This approach results in fewer software classes, less code to maintain and simplified model evolution.

The streamlined approach uses a type/subtype hierarchy for each data model class. An example best illustrates this concept. SimMaterial is a data model class that represents material types. However opaque and transparent are types within that class, where each type then contains the relevant subtypes, e.g. class = SimMaterial, type = OpaqueMaterial, subtype = NoMass.

The type/subtype approach also acts as a filter for data on an object instance to ensure that only properties relevant to the subtype are used. This approach enables schema evolution and application specific schema variants. This property filtering is a key feature that is not supported by most other data models.

7.3.6.4 Model Ontology

The SimModel ontology introduces two concepts that were previously undefined in simulation data models: 1) projects and 2) design alternatives. These new concepts enable an efficient re-use of existing data and minimize the overhead associated with tracking changes between design alternatives. Other features include geometric entities, HVAC systems, HVAC components, groups, controls, simulation parameters, and outputs. The concept of modeling systems and zones as groups is unique when compared with other data models in the simulation domain. This feature aligns with the definition of Thermal Blocks as contained in COMNET (a set of rules and procedures for energy modeling) *[RES10]*. Explicitly defined properties enable collections of group members. SimModel also surpasses IFC with respect to loosely defined building element assemblies by formalizing definitions for curtain walls, ramps, roofs, stairs, transportation systems, site assemblies, day-lighting assemblies and ventilation assemblies.

7.3.6.5 Resources

SimModel takes advantage of a number of resource objects that are absent from other simulation domain data models. These include actors in a project, which can include people, organizations, or people in organizations (as in IFC). Examples that have come into SimModel from CDB-2010 (*[BDA12]*) include the building owner, the architect, and building occupants. Actors are also used to support the fact that simulation tools require not only heat generated by occupants but also their behavior and presence. For the purposes of collaboration, applications may also associate an actor with the ownership of each individual object instance (called the OwnerHistory as in IFC). Other new resources include templates and library object entries that support the reuse of library and template content. With this model in place we now describe a transformation process from SimModel to Modelica.

7.3.6.6 Extensions

Due to the nature of SimModel, the majority of concepts, objects and parameters did already exist and could be used in this project. The project initiated the addition of a small number of parameters as well as a couple of new object subtypes. In particular, controller representations are simple in SimModel (originating in the simple representation in EnergyPlus) and needed some of these extensions. In addition, we added a couple of parameters such as area and normal direction to the space boundary object. This addition eliminated the need for 3D geometric processing for our conversion from SimModel to Modelica.

7.3.7 Formal Transformation Process

This subchapter formally describes the data transformation process mathematically.

7.3.7.1 Overall Definitions

In order to map SimModel objects and properties to relevant Modelica objects and properties, in the Annex 60 project a set of generic mapping rules was developed. This mapping of objects and parameters consists of four mapping rules using a formal mathematical description (Fig. 7.15). To illustrate the mapping appropriately, the rules are defined by using set theory *[PD00]*:

Let U be a universal set. U contains the input data for SimModel (represented by the subset S) and Modelica (represented by the subset M), where S is a

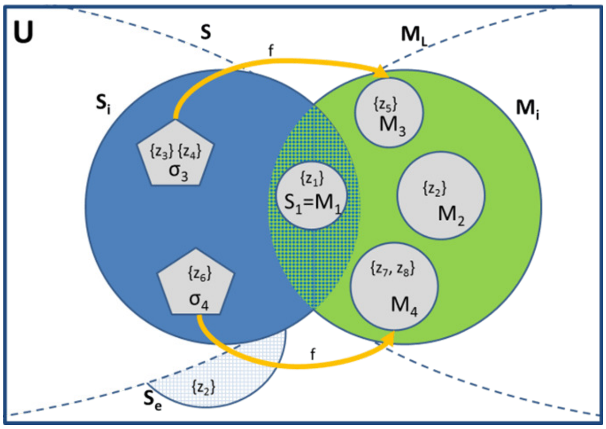


Fig. 7.15: *Venn diagram for the mapping details between SimModel and Modelica*

subset of U. It represents the SimModel objects with corresponding parameters (7.1).

$$S = S_1, S_2, \dots, S_n \tag{7.1}$$

S_e represents the extension of parameters and objects beyond the data set of SimModel (S). This is necessary for representing the required additional data within the targeted Modelica libraries (7.2),

$$S_e = S_{n+1}, \dots \tag{7.2}$$

$S_i(i = 1 \dots n)$ is a subset of elements of S, where S_i contains objects and parameters as relevant input data for the targeted M_i (7.3) + (7.4),

$$S_i = z_1, z_2, \dots, z_m \tag{7.3}$$

z is an element of the set U and represents a single parameter.

$$S_i \subset S \tag{7.4}$$

$M_L(L = \mathit{AixLib}, \mathit{Buildings}, \mathit{BuildingSystem}, \mathit{IDEAS})$ is a subset of U. M_L represents the objects and parameters necessary for a specific library in Modelica (7.5),

$$M_L = M_1, M_2, \dots, M_n \tag{7.5}$$

$$S \cup M_L \subset S_e = U \tag{7.6}$$

$M_i(i = 1 \ldots n)$ is a subset of M_L. It represents a set of objects and parameters (7.7) + (7.8),

$$M_i = z_1, z_2, \ldots, z_m \tag{7.7}$$

$$M_i \subset M_L \tag{7.8}$$

For an optimal communication between the two data models, the interface represents a minimum data flow. To meet this requirement, the following boundary condition needs to be fulfilled (7.9).

$$|S_i| \rightarrow min \tag{7.9}$$

The following two sections define the formal transformation rules. Thereby, two different kinds of mappings are distinguished. The first section considers the so-called object mappings while the second section focuses on parameter mappings.

7.3.7.2 Object Mapping

The object mapping defines how a mapping between objects of the two domains essentially takes place. The type of mapping thereby depends on the issue if objects can be mapped directly (i.e., one to one), if a single object needs to be mapped to more than a single object or vice versa (referred to as many to one or one to many mapping), or if such a mapping is not possible at all (gap) which requires to extend the model, respectively.

One to One Mapping The One to One mapping formally represents an intersection of M_L and S (7.10) and identical subsets (7.11):

$$M_L \cap S \tag{7.10}$$

$$M_i = S_i \tag{7.11}$$

Example: Identical fans in SimModel and Modelica.

Many to One (7.12) / **One to Many Mapping** (7.13) Several different subsets S_i represent the necessary elements for a single M_i or vice versa

$$S \supseteq (S_1 \cup \cdots \cup Sn) = M_i \subseteq M_L \tag{7.12}$$

$$S \supseteq S_i = (M_1 \cup \cdots \cup M_n) \subseteq M_L \tag{7.13}$$

Example: A valve is part of the radiator object in SimModel whereas in the used Modelica library models the valve and radiator are separate objects.

Gap The SimModel project instance does not contain the targeted object (7.14) and needs an extension by S_e (7.15) or this missing object needs to be added to the corresponding instance of the data model.

$$S \cap M_i = \varnothing \tag{7.14}$$

$$|M_i \cap (S \cup S_e)| > 0 \tag{7.15}$$

Example: A SimModel project instance does not contain an expansion vessel for hot water systems.

Combination Possible combination of the above rules.

7.3.7.3 Parameter Mapping

Similar to the object mapping, model parameter need to be mapped between the two models. We formally distinguish between the following situations.

One to One mapping Intersection of a subset S_i and M_i (7.16) with identical parameters (7.17)

$$M_i \cap S_i \tag{7.16}$$

$$M_i = z_1, \dots, z_m = S_i \tag{7.17}$$

Example: Power of a radiator (W).

Gap The specific data model defined in SimModel does not contain the parameter (7.14), so that it needs to be extended by S_e (7.15) or added to this specific data model instance.

Example: A SimModel project instance does not contain the maximum pressure for a tempering valve.

Transformation Rule The transformation rule represents a special case. Technically, it describes a gap, as there is no corresponding parameter in SimModel (7.14). However, with a transformation of a subset it is possible to create the required data. With this rule, a new subset σ is accordingly defined.

σ represents a set of parameters which are similar to the definition of parameters in M_i (7.18)

$$\sigma \sim M_i \tag{7.18}$$

To accomplish the mapping with a union of M_i with σ, it is necessary to transform the elements in σ via an algorithm (7.19)

$$f : \sigma \rightarrow M_i \tag{7.19}$$

To combine several parameters, it is necessary to define transformation algorithms (7.20) + (7.21)

$$f(\sigma_i = z_1, \dots, z_m) = M_i = z_1 \tag{7.20}$$

$$f(\sigma_i = z_1) = M_i = z_1, \dots, z_m \tag{7.21}$$

The transformation rule covers a conversion of parameters as well.

For example, a simple unit conversion or a more complex conversion of one function to another function needs to be handled by a certain algorithm.

Combination Possible combination of the rules above.

Fig. 7.15 exemplarly conceptualizes the mapping rules for the object mapping with the universal set U, the corresponding subsets S, M_L and S_e, and the sub-subsets. The figure specifically illustrates the parameter mapping. S_1 and M_1 contain identical parameters and represent the first rule. M_2 represents the gap, which is closed by embedding the missing information in the SimModel data model via S_e. σ_3 and M_3 represent a transformation from many parameters in the SimModel set to a single parameter on the Modelica side. At this point, it is necessary to use the transformation rule, as the user needs to implement algorithms (illustrated by the function f) to combine the parameters. The outcome of this algorithm is a single parameter on the Modelica side. The link between σ_4 and M_4 demonstrates the use of a single parameter at the SimModel side to define multiple parameters on the Modelica side.

7.4 Open Framework for Modelica Code Generation from BIM

This subchapter presents and describes tools developed in this Annex 60 project required for the conversion from Building Information Models to Building Performance Simulation using different Modelica libraries. The tools result in a tool-chain which is highly modular and supports the reuse of different parts in follow on projects. We identify three necessary steps:

- Data model generation
- Information selection, enrichment and verification
- Simulation model generation

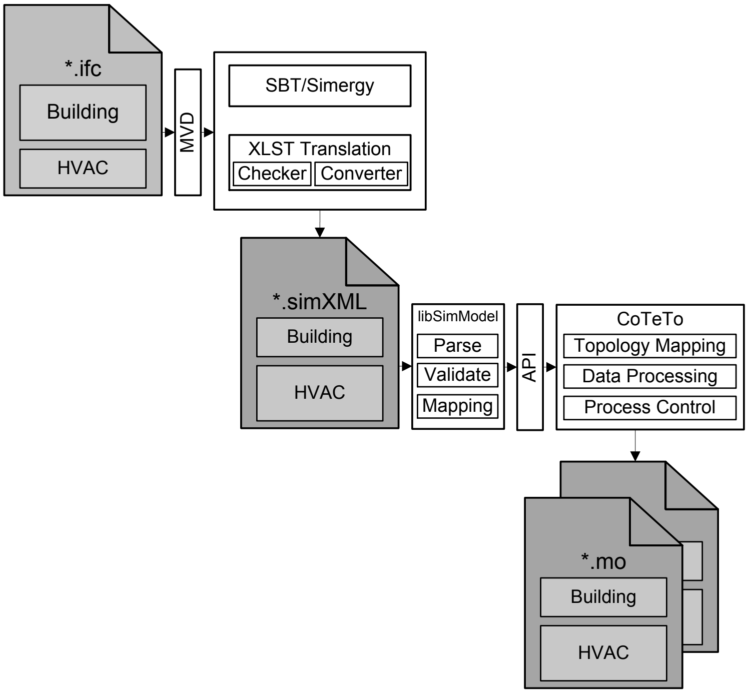


Fig. 7.16: Overview of the transformation tool-chain from IFC to SimXML and Modelica model

7.4.1 Conversion from BIM to BPS

This section details the process from BIM to Modelica models. Fig. 7.17 shows this overall process. There are three actors involved. The architect generates a geometry model, verifies its correctness and generates 2nd level space boundaries. The HVAC engineer uses this geometry model as a starting point and adds the HVAC model of the building. He also performs a validation check. The simulation team then gathers necessary data to enrich the simulation model, runs the simulation and analyzes the results. This process view illustrates the IDM (Information Delivery Manual) that goes along with the MVD defined by this project.

7.4.1.1 Data Model Generation

An obligatory requirement for the framework is a valid and well-formed BIM model. A well formed BIM in the context of BPS means the definition of geometry (e.g. different constructions and corresponding space boundaries) as well as the HVAC system. We distinguish between geometry, building physics and HVAC components as semantic model parts within a BIM. The building geometry is created with BIM-based CAD software (Fig. 7.16). Within this software physical and semantic properties of the building objects are defined as well. Based on the geometry of the building, HVAC engineers further add information about the energy system to the BIM. Using a single file format for data exchange between different actors creates added value for actors and users. However, there is need for an actor who coordinates data exchange and data migration to the model. This BIM manager requires knowledge in different domains (e.g. geometry, building physics and HVAC components) as well as in the field of data exchange. As some planning is done in parallel, the BIM manager is as well responsible for merging partial models. This is important for the overall consistency of the BIM within these collaborative activities. Model quality is highly important for the use of the presented tool-chain. We assume to start with a well formed and valid IFC file. Starting from this IFC file, the transformation process is initiated using the tools from the tool chain as shown in Fig. 7.16.

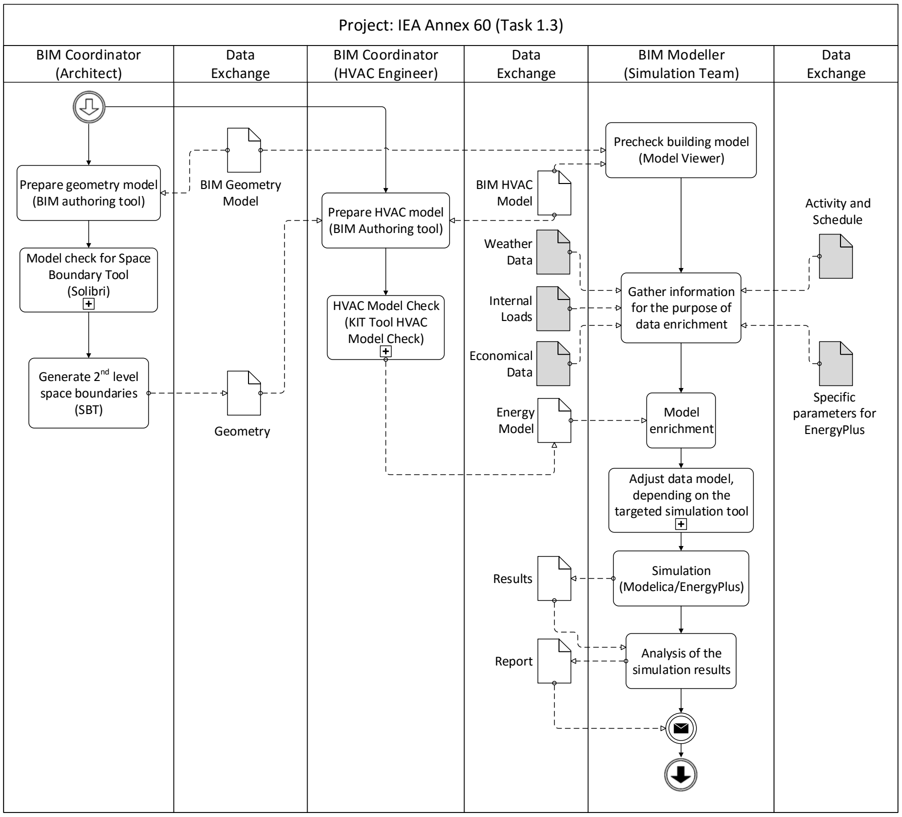

Project: IEA Annex 60 (Task 1.3)
BIM Coordinator (Architect)
Data Exchange
BIM Coordinator (HVAC Engineer)
Data Exchange
BIM Modeller (Simulation Team)
Data Exchange
Prepare geometry model (BIM authoring tool)
Model check for Space Boundary Tool (Solibri)
Generate 2nd level space boundaries (SBT)
BIM Geometry Model
Geometry
Prepare HVAC model (BIM Authoring tool)
HVAC Model Check (KIT Tool HVAC Model Check)
BIM HVAC Model
Weather Data
Internal Loads
Economical Data
Energy Model
Results
Report
Precheck building model (Model Viewer)
Gather information for the purpose of data enrichment
Model enrichment
Adjust data model, depending on the targeted simulation tool
Simulation (Modelica/EnergyPlus)
Analysis of the simulation results
Activity and Schedule
Specific parameters for EnergyPlus

Fig. 7.17: *Overall process diagram (IDM)*

7.4.1.2 Information Selection, Enrichment and Verification

The IFC format is able to store detailed information about various disciplines, for example, fire protection. Not all of the exchanged data is needed for BPS, therefore the energy consultant uses a specific Model View Definition, which is described in Section 7.3.5, to read the relevant data for BPS in Modelica (middle part in Fig. 7.16). An MVD is a subset of the IFC-schema that defines discipline-specific exchange requirements. As an example the energy consultant needs spatial surface geometry defined in the MVD as a basis for thermal zones. On the other hand, the simulation of HVAC equipment might need further information that is not part of the information model of IFC. To store and exchange this additional information, the energy consultant migrates the collected data from the MVD to the intermediate file format SimXML. The transformation is done by using Space Boundary Tool and Simergy for the geometry part of the IFC mode, see Section 7.4.3.1 and Section 7.4.3.2. The HVAC related objects are transformed with a tool developed as part of this project. It uses the XSLT language to translate the model into SimXML (Section 7.4.3.4). The information model behind SimXML is called SimModel (see Section 7.3.6) which is a simulation domain specific data model. Both tools add further, missing information to the SimXML file. This particularly refers to space boundary and semantic information, needed for dynamic building simulation. The resulting file is automatically verified against the XML schema definition. Section 7.4.3 describes technical details of the transformation from IFC to SimModel.

7.4.1.3 Simulation Model Generation

The last step is the generation of valid Modelica code. This task divides into the parsing and validating of the SimXML file, mapping from SimModel to a specific Modelica library (e.g. BuildingSystems or AixLib) and textual output of the code itself. For parsing and validating the SimXML file we developed a C++ framework, called libSimModel (Section 7.4.4). In addition the code maps objects and parameters defined by mapping rules from SimModel to Modelica library. Furthermore, individual processing and the mapping of object topologies (as these differ in different libraries) need to be considered. Through an API we expose relevant data available in libSimModel to the Python programming language and perform these individual steps in a tool called Code Templating Tool (CoTeTo). Both object and parameter as well as the topology mapping needs to be defined by a simulation expert, who has fundamental knowledge of the considered library and SimModel. CoTeTo collects now all information and prints out Modelica code with regard to a correct Modelica syntax. The process is described in detail in Section 7.4.6.3.

7.4.2 Data Transformation

The primary role of a Building Information Model (BIM) is to serve as a comprehensive repository of data that are retrievable by multiple software applications which participate in the same AECO industry project. Data placed in a BIM by one software application are retrieved and used by other applications. Retrieved data are at times not useable by the recipient application in exactly the same form and/or format as received; in such cases the received data are manipulated and/or transformed before they can be used *[BK07]*.

A software application is entering new data in a BIM, it is *authoring* those data. Such data constitute the original BIM data that can then be used by other, usually downstream, software applications. Downstream applications may, and often do, support different disciplines and have implemented different model views than the authoring application.

CAD applications generate building geometry and, in the process of documenting it, are usually the first to create original data. Additional data are subsequently authored by downstream applications. Because the need to exchange data among CAD applications is relatively infrequent, the real payoff from software interoperability is seamless data exchange with and among downstream applications. But that data exchange is not always automatic or straight forward.

A software application imports or generates itself all data it manipulates. To obtain valid results, data imported by an application must not only be in a form and format that is readable by the application, but also must represent values within the range the application expects to import. For example, if a downstream application expects to import a value for floor-to-floor space height, the imported value must represent floor-to-floor and *not* floor-to-ceiling space height. Because of the diversity of applications (and their internal data structures) that may participate in a given data exchange, "original" data must often be *transformed* before they can be used by a downstream application – data sets must be reduced and/or simplified, or data must be translated and/or interpreted.

In general, data transformation can be classified in four types:

1. *Data set simplification.* An original data set can be too "rich" to import by a downstream application. For example, the original definition of a floor structure may include the precise geometry of all material layers in the floor, while a downstream application can only import a generalized "sandwich" definition of the floor. The redefinition of the floor "sandwich" that loses some geometry detail but still accounts for all material layers amounts to a simplification of the original geometry. Geometry approximation usually falls in this category.
2. *Data set reduction.* If the "rich" original geometry data set contains definitions that cannot be used by the downstream application, these are omitted from the exchange set. This is the case when, for example, a column in its original is partially embedded in a wall, but the downstream application cannot import column definitions; the column definition is excluded from the exchange set and the wall of the original definition is extended over the space previously occupied by the column.
3. *Data set translation.* Most of this type of data transformation involves units of measurement, such as conversion of metric to imperial units. Another type of translation is the textual definition of surface orientation which in the original definition is expressed by the "normal" vector, and the downstream application requires textual identification of the orientation (e.g. "exterior").
4. *Data set interpretation.* When data required by downstream applications have not been *explicitly* defined before, it may be possible to derive them from original data *which themselves may not be needed by these applications*. The interpretation of original information that yields derived data then becomes a process that can involve recognition, extraction, sorting, and calculation. This process can be relatively simple, as in deriving the floor-to-wall area ratio not included in the original BIM data, or fairly complicated, as in deriving the proper "net" building area as specified by local code.

Legitimate data transformation is performed following rules which are agreed upon and standardized. These rules cannot be arbitrary and should be embedded in data transformation software to be executed faithfully. Manual data transformation is acceptable only if it follows applicable rules.

7.4.3 IFC to SimModel

The process of converting the IFC data model into SimModel consists of various tools that convert both the geometry as well as the HVAC data. For the geometry, the Space Boundary Tool generates 2nd level space boundaries. Simergy will transform the geometry data from IFC to SimXML. For the HVAC data, we first describe the checking tool and then the converter.

7.4.3.1 Space Boundary Tool (SBT)

SBT is a tool that calculates all levels of space boundaries that define surface geometry which constitutes the building geometry model in BEP simulation using EnergyPlus and other simulation engines with similar internal geometry models *[Baz10]*. The main SBT algorithm is based on graph theory to convert a three-dimensional architectural building model without defined thermal space boundaries into geometry suitable for import into a whole-building energy performance simulation engine such as EnergyPlus *[RB13]*. The algorithm expects input specified as an *Industry Foundation Classes (IFC)* model, accepts a wide variety of input geometry, and is capable of accounting for a building's construction material configuration as well as its geometry. The described approach is limited to solid-to-solid heat exchange; solid-to-fluid and fluid-to-fluid heat exchange is not considered.

In addition, SBT performs two other functions: It (a) automatically performs data transformation that is necessary so that EnergyPlus- like simulation engines can read and use information contained in SBT export files without further data transformation, and (b) it automatically corrects modeling errors that can be automatically corrected. All data transformation is performed according to rules embedded in SBT. The following is the list of 16 geometry data transformation rules embedded in SBT:

- Skipping of internal wall objects when walls are entirely contained within the same thermal zone;
- Reversal of the order of construction material layers for ”other side” second-level space boundaries for walls and slabs which have asymmetric construction;
- Redefinition of embedded columns as separate wall objects;
- Definition of the remaining wall construction parts when columns are only partially embedded in walls;
- Recognition of exterior building shade types;
- Positioning of exterior building shades right outside the exterior space boundaries of exterior walls;
- Detection and redefinition of virtual walls and slabs;
- Assignment of “virtual constructions” to virtual walls and slabs;
- Identification of floor and ceiling surfaces of a slab;
- Subdivision of slabs with voids into “void-free” segments;
- Redefinition of “exterior ceilings” as roofs;
- Connection of slab-on-grade objects to the ground object;
- Creation of the parent wall’s space boundaries for windows (if missing);
- Adjustment of window area to effective glass area;
- Linking of glazing definitions to Window 6.2 tool;
- Linking of material and construction objects to the EnergyPlus library of materials’ thermal properties.

SBT’s automatic model correction is limited to surface connecting errors typically prevalent in BIM geometry models generated by industry modelers: surfaces not connecting when connection is intended and/or necessary, and surfaces penetrating each other when such penetration is not intended. In modeling cases where not connecting surfaces is intentional (but SBT detects a gap between surfaces as defined in the IFC input file), SBT allows its user to set tolerance that identifies the intended gap and does not treat such instances as errors.

In addition to flat planar geometry, SBT can generate geometry models of curved surfaces and spaces. It calculates the geometry of single flat segment connections between end points of curved surfaces, but overrides surface area and space volume data associated with flat surfaces with actual values of curved surfaces and spaces. Since IFC definitions of curved surfaces is limited to 2D curves, SBT can itself only accept 2D curves. The tool can also interactively map thermal properties of construction materials from ASHRAE specifications to construction materials identified in the imported IFC geometry.

The tool generates two export files: an updated IFC file (with added space boundary definitions) and an IDF file that delineates the building's geometry for direct input and execution in EnergyPlus. The latter exports a 3D wire-frame view of the building geometry imported into EnergyPlus, which facilitates visual inspection of correctness of the imported geometry. Section 7.5.9 shows an example of this transformation.

7.4.3.2 Simergy

Simergy is a simulation front end that currently supports EnergyPlus simulations and analysis for the California Energy Code (Title24). With its data model SimModel (Section 7.3.6) aligns closely to the IFC data model and this supports import and export of IFC files as well as other data formats (e.g., gbXML or EnergyPlus input data format (IDF)). The promise of Simergy is to allow easy reuse of data in form of library object entries as well as data and system templates. This enables quick setup of detailed simulation models that can easily be adjusted and optimized.

In context of this project and its tool chain, Simergy provides mainly two functions. The first one is the import of IFC data files and the export of SimXML files containing geometry data. For this function it can either use space boundaries generated by SBT or generate them internally. The second feature is the addition of internal loads data. While previous efforts (such as Concept Design BIM 2010, *[BDA12]*) tried to enable the definition of internal loads in CAD applications, currently support is only partial. Especially scheduling data is not supported by CAD applications. Since schedules are a major part of internal loads definition, we add them for this project within Simergy. Besides the geometry and internal load data, HVAC data is an important part of the simulation model and the relevant data transformation and checking is illustrated in the next subsections.

7.4.3.3 IFC Model Checking

Quality assurance of the initial BIM model has been identified as an important precondition for a qualified simulation process. Thus, the objective was to provide a coherency check of the IFC data to ensure that data entering the overall tool-chain passes quality requirements. The conception of the tool builds on existing generic software components and packages that were used to implement the tool.

Requirements Section 7.3.5 highlights general requirements for quality assurance within the BIM-based processes. In the context of the BIM to Modelica conversion process the required data quality was examined in a scenario-based requirements analysis. Based on these results specific rules were derived which specify model quality and IFC contents. For example, all pipe parts of a pipe circuit have to be correctly connected (referenced). The rules aim to detect inconsistencies within the designated source data of the simulation, before they are transformed to the intermediate format SimModel.

Concept The tools core architecture is based on concepts and generic components of the BLM collaboration toolKIT. There are two main underlying concepts, such as the workflow based conception of the user interface, and the concept of independant rules written in XML files to define the domain-specific model checking aspects. In order to support non-experts the UI was designed as a four-step procedure to check a model file against given rules. These rules are formalized in *[EvB13]* and are stored in different XML files. The rule-based method builds a base to depict the simulation-related data quality demands and enables to check whether a designated model complies with them. Primarily the defined rules in natural language have to be formalized in order to build the logical (domain-specific) base for the machine-interpretable business logic (rule code). Methodically the first step of formalizing the rules takes necessary elements of the specified IFC4 syntax on a binary level into account (e.g. paths to respective object classes). These first rule parts or elements build the base for deriving collections, resolving relations etc. as needed to formalize the overall natural language content (rule logic) into rule code. The other developed components which adopt the reused concepts, together with the new developed viewing component are finally integrated into a standalone tool. This application together with the separated rule base is available at https://download.building-lifecycle-management.de for free. The application is used to support the designated users of the projects tool-chain with respective quality measures to examine their potential IFC-based input data.

Implementation The model checking tool is released using the component-based-development approach based on the .NET framework. The developed rules are either required or optional and following rules were developed for the context of the project.

Obligatory rules:

- building context information, e.g. proper hierarchy, location information
- building elements have material (layers) connected
- space entities have space boundaries connected
- all (HVAC) IFC components have to be connected to a IfcSystem (or derivate)
- air systems are connected to a thermal zone
- mechanical air systems contain at least one fan component

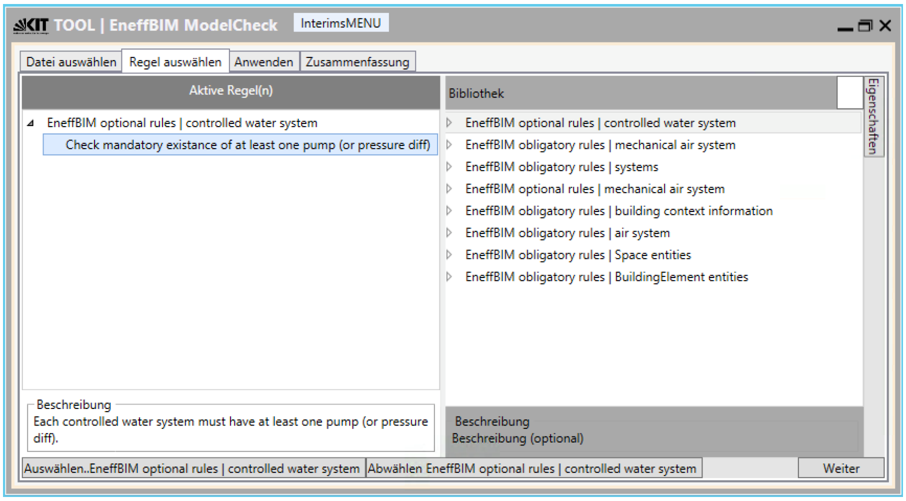
Optional rules:

- mechanical air systems are either fresh air or are closed loop systems
- controlled water systems contain at least one pump component

Fig. 7.18 shows the current development stage of the user interface of the standalone software tool "KIT EnEff-BIM Model Check". The workflow of model checking is presented in four steps:

1. load model file
2. select rules
3. execute model check
4. summarize results

The workflow is displayed in a corresponding four tab view to the user. Besides the possibility to export a respective (error-) report to a spreadsheet file in tab 4, as well a call to the conversion tool has been implemented. In case of a fault free model check the already loaded and checked ifcXML model file is passed as an argument to the standalone console application Section 7.4.3.4. This converter is described in the next subsection.


KIT TOOL | EneffBIM ModelCheck InterimsMENU

Datei auswählen | Regel auswählen | Anwenden | Zusammenfassung

Aktive Regel(n)

EneffBIM optional rules | controlled water system
Check mandatory existance of at least one pump (or pressure diff)

Bibliothek

EneffBIM optional rules | controlled water system
EneffBIM obligatory rules | mechanical air system
EneffBIM obligatory rules | systems
EneffBIM optional rules | mechanical air system
EneffBIM obligatory rules | building context information
EneffBIM obligatory rules | air system
EneffBIM obligatory rules | Space entities
EneffBIM obligatory rules | BuildingElement entities

Eigenschaften

Beschreibung
Each controlled water system must have at least one pump (or pressure diff).

Beschreibung
Beschreibung (optional)

Auswählen..EneffBIM optional rules | controlled water system | Abwählen EneffBIM optional rules | controlled water system | Weiter

Fig. 7.18: Screenshot “KIT EnEff-BIM Model Check” Version 0.9 rule selection tab.

7.4.3.4 XSLT-Based IFC to SimXML Converter for HVAC Elements

The main objective of this tool is to provide a mechanism that converts the HVAC partial model in IFC4 to the SimModel format.

Requirements For the development of the converter tool, the Annex 60 use cases define the requirements and build accordingly the logical base for the transformation of the IFC4 HVAC partial model to SimModel. The web-based BIM*Q platform (*[316]*) was used to collect and specify the requirements needed for the translation. The IFC entities related to the use cases and their parameters form the mapping information of the respective SimModel objects.

Concept The concept of the conversion tool is based on the existing (technical) World Wide Web Consortium (W3C) approach that specifies the eXtensible Stylesheet Language (XSL) Transformations - XSLT - as a transformation technology for XML syntax. The static translation of XML-based IFC data into elements of the XML-based target format SimModel is addressed by an established, scalable and extensible solution. The core principle of the XSLT 1.0 technology can be described in four steps (cf. Fig. 7.19):

- load the source model as stream object,
- load the XSLT files (transformation templates),
- apply the templates to the source data stream and populate the emerging Result Structure Tree (RST), and
- write the RST into a designated target file.

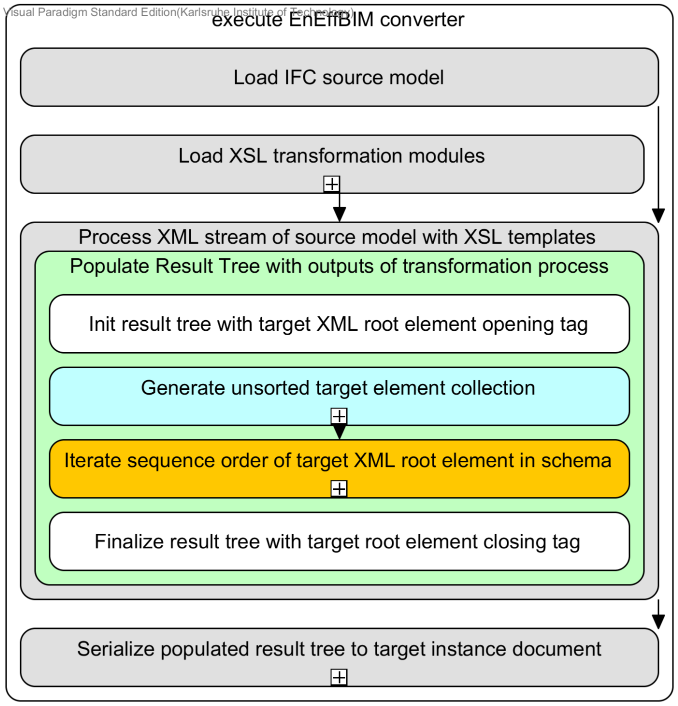

Visual Paradigm Standard Edition(Karlsruhe Institute of Technology)
execute EnEffBIM converter
Load IFC source model
Load XSL transformation modules
Process XML stream of source model with XSL templates
Populate Result Tree with outputs of transformation process
Init result tree with target XML root element opening tag
Generate unsorted target element collection
Iterate sequence order of target XML root element in schema
Finalize result tree with target root element closing tag
Serialize populated result tree to target instance document

Fig. 7.19: Overview of the concept building blocks.

The static mechanism of XSLT requires an a priori definition of the exact translation instructions, and therefore persists of static templates within the framework of the XSLT application. Due to the joint development of the mapping requirements, the static approach of XSLT was supplemented with generic solutions. These are specified in the following four concept blocks that define the core architecture of the tool:

1. Prefilter the relevant IFC model entities from the source model data stream. Several “Traversing Templates” (for processing the IFC topology) are used on side of the technical development to optimize the overall conversion procedure (in the subsection “Implementation” we describe how this concept block is realised and how to bypass it)
2. Dynamic injection of the externally defined translation instructions between source and target model (the used concept of a simple syntax for the mapping logic is introduced in the following sub-section)
3. Trigger the (element-wise) generation of the target model contents by starting the conversion procedure through applying exactly one framing (main) XSLT template and loosely coupled “Production Templates”. This XSLT core principle (cf. Fig. 7.19) builds the base for further thematical extensions of the conversion tool.
4. Final assembly of the output (SimModel file) supported by the target model schema.

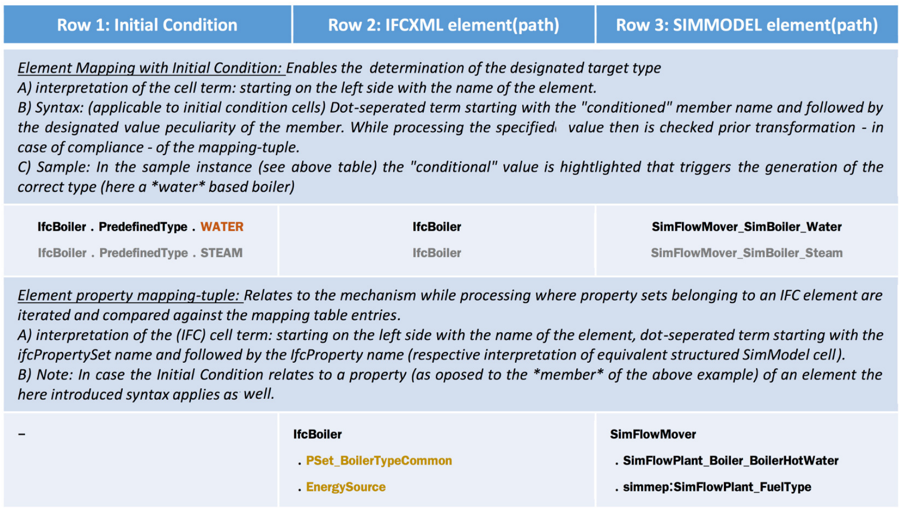
Referring to the “Divide and Conquer” principle these four concept building blocks were consolidated into a comprehensive transformation process (cf. concept building block 3; see Fig. 7.21).

Concept of Machine-Readable Mapping Table Entries In order to transfer the mapping logic into the tool a syntactic machine-readable structure was defined that refers to the contents of the table-based BIM*Q platform. This structured information is then handed off to the conversion mechanism by exporting it to the Comma-Seperated Values (CSV) format. Conceptually the two columns of the mapping-tuple - IFC column and SimModel column - were extended by a third column that defines an initial condition for conversion of the respective tuple. Following mapping case example (IfcBoiler to SimFlowPlant_Boiler_BoilerHotWater) exemplarily depict the simplified model instances:

```
<ifcXML>
  (...)
  <IfcBoiler PredefinedType=WATER>
    <IsNestedBy>
      (...)
      <IfcPropertySet Name=PSet_BoilerTypeCommon>
        <ifc:IfcPropertySingleValue Name="EnergySource">
        (...)
```

```
<SimModel>
  (...)
  <SimFlowPlant_Boiler_BoilerHotWater>
    <simmep:SimFlowPlant_FuelType>NaturalGas
    </simmep:SimFlowPlant_FuelType>
    (...)
```

And Fig. 7.20 illustrates the corresponding (commented) excerpt of the CSV table for the above mapping case.



Row 1: Initial Condition	Row 2: IFCXML element(path)	Row 3: SIMMODEL element(path)
*Element Mapping with Initial Condition: Enables the determination of the designated target type A) interpretation of the cell term: starting on the left side with the name of the element. B) Syntax: (applicable to initial condition cells) Dot-seperated term starting with the "conditioned" member name and followed by the designated value peculiarity of the member. While processing the specified value then is checked prior transformation - in case of compliance - of the mapping-tuple. C) Sample: In the sample instance (see above table) the "conditional" value is hightlighted that triggers the generation of the correct type (here a *water* based boiler)*		
IfcBoiler . PredefinedType . WATER IfcBoiler . PredefinedType . STEAM	**IfcBoiler** IfcBoiler	**SimFlowMover_SimBoiler_Water** SimFlowMover_SimBoiler_Steam
*Element property mapping-tuple: Relates to the mechanism while processing where property sets belonging to an IFC element are iterated and compared against the mapping table entries. A) interpretation of the (IFC) cell term: starting on the left side with the name of the element, dot-seperated term starting with the ifcPropertySet name and followed by the IfcProperty name (respective interpretation of equivalent structured SimModel cell). B) Note: In case the Initial Condition relates to a property (as oposed to the *member* of the above example) of an element the here introduced syntax applies as well.*		
–	**IfcBoiler . PSet_BoilerTypeCommon . EnergySource**	**SimFlowMover . SimFlowPlant_Boiler_BoilerHotWater . simmep:SimFlowPlant_FuelType**

Fig. 7.20: *Commented syntax example of the CSV mapping table*

The simple "point-syntax" of the term further enables e.g. the specification of mappings between nested elements, and completes the machine readable mapping concept. Based on this mapping syntax definition and without having the mapping table fully formulated in detail, a (dynamic) requirement basis for the implementation of the designated conversion tool is given.

Implementation For realizing the converter with XSLT 1.0, the above introduced concept blocks one, two and four were integrated into an XSLT transformation procedure which is described in the third concept block. Fig. 7.21 shows the overall procedure where the green templates - ifcXML root element and Production Templates for respective IFC elements - realize the overarching main procedure.

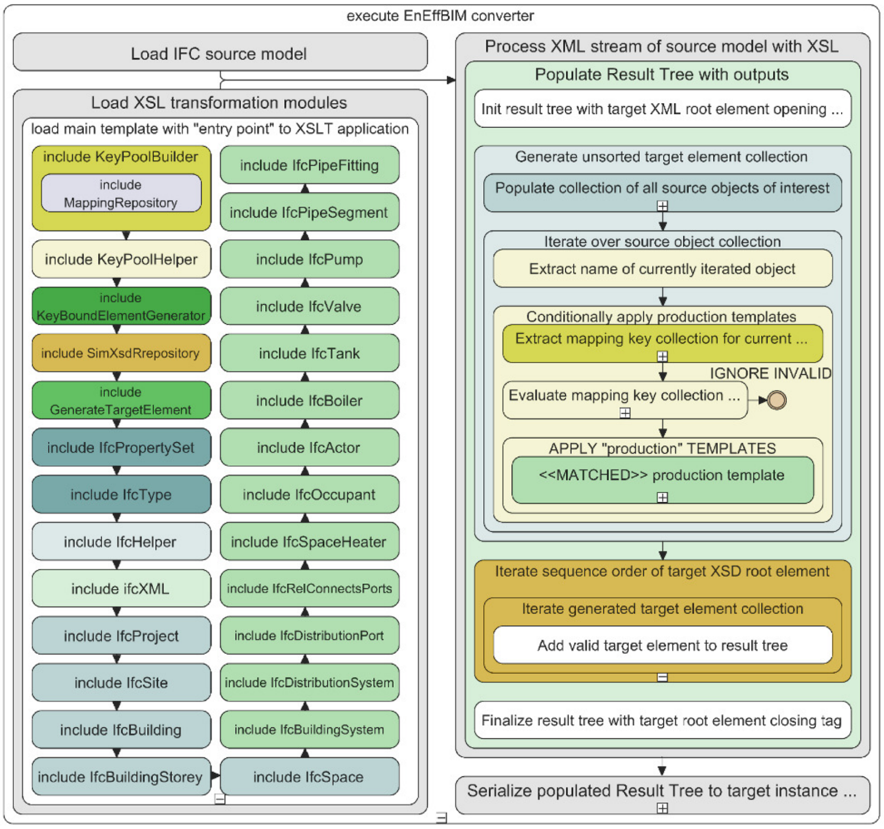
Thereby recursively nested templates of the first (blue) and second (yellow) concept block extend the static procedure with mechanisms to

1. a priori collect all relevant IFC entities from the source model stream and
2. inject the conversion instructions from a CSV table that is delivered by the BIM*Q platform.

Fig. 7.21 depicts the generation of the target root element "SimModel". Within the green box "MATCHED production templates" a recursive sub-procedure handles the generation of the target elements and their content. For further reading we refer to *[EvB16]*. In summary, a demand-led, dynamic and scalable solution was realized. Future developments (or other configurations) are supported by providing the full implementation of the tool as open source (access via https://download.building-lifecycle-management.de).

7.4.4 libSimModel C++ library

This section describes the transformation system developed for linking BIM with different Modelica libraries. In the system, SimModel acts as the BIM container to save BIM data generated from IFC, for example, HVAC, geometry, property data of HVAC systems and equipment, simulation configurations, etc.


execute EnEffBIM converter
Load IFC source model
Load XSL transformation modules
load main template with "entry point" to XSLT application
include KeyPoolBuilder
include MappingRepository
include KeyPoolHelper
include KeyBoundElementGenerator
include SimXsdRrepository
include GenerateTargetElement
include IfcPropertySet
include IfcType
include IfcHelper
include ifcXML
include IfcProject
include IfcSite
include IfcBuilding
include IfcBuildingStorey
include IfcPipeFitting
include IfcPipeSegment
include IfcPump
include IfcValve
include IfcTank
include IfcBoiler
include IfcActor
include IfcOccupant
include IfcSpaceHeater
include IfcRelConnectsPorts
include IfcDistributionPort
include IfcDistributionSystem
include IfcBuildingSystem
include IfcSpace
Process XML stream of source model with XSL
Populate Result Tree with outputs
Init result tree with target XML root element opening ...
Generate unsorted target element collection
Populate collection of all source objects of interest
Iterate over source object collection
Extract name of currently iterated object
Conditionally apply production templates
Extract mapping key collection for current ...
IGNORE INVALID
Evaluate mapping key collection ...
APPLY "production" TEMPLATES
<<MATCHED>> production template
Iterate sequence order of target XSD root element
Iterate generated target element collection
Add valid target element to result tree
Finalize result tree with target root element closing tag
Serialize populated Result Tree to target instance ...

NOTE: The colors relate to the four concept blocks of the xslt 1.0 tool. (1) yellow - read mapping info from CSV (2) blue - populate source object collection (3) green - "plain old XSLT" (4) orange - order output by target XSD

Fig. 7.21: Recursively superimposed concept blocks of final tool specification

7.4.4.1 SimXML Data Binding and Syntax Validation

The data file of SimModel is an XML-based file named SimXML. It saves all the SimModel data as a structured XML document that is in accordance with the syntax defined by the SimModel schema. Thus, this sub-section introduces the XML data binding technique for accessing SimXML and validating its syntax. XML data binding is the process of extracting data from a structure representation of XML documents and presenting it as a hierarchy of objects that correspond to a document vocabulary. This allows us to manipulate SimXML data in a more direct and efficient way. We selected the open source, cross-platform CodeSythesis XSD *[Cod14]* as our system XML binding parser. It is an efficient framework whose parser can be adjusted for custom applications. The automated XML data binding of CodeSythesis XSD will generate a C++ API for accessing the data stored in SimXML after parsing the SimModel schema. For the SimModel schema version 2.2, 2611 C++ classes representing different SimModel objects are generated for the data manipulation in a given SimXML file. The XML syntax validation performs a number of checks on the XML document to prevent the construction of an inconsistent object model, such as an object model with missing required attributes or elements. Our SimXML validation relies on the underlying Xerces-C++ XML parser embed in CodeSythesis XSD. It checks the SimXML data against the given SimModel schema, and outputs the errors found into a log file. The syntax validation is enabled by default and is very useful during the development stage to detect problems with the data model at an early stage. For a user of the framework, this validation does not play a significant role anymore, since the changes to the SimModel model are finalized.

7.4.4.2 SimModel Hierarchy Parsing and Visualization

A model hierarchy is an arrangement of the model elements, e.g., objects, names, values, categories, etc., in which the elements are represented as being “above,” “below,” or “at the same level as” one another. Consequently, the SimModel consists of a hierarchical tree structure saving such relationships between different SimModel data elements (see Fig. 7.22).

As illustrated in Fig. 7.22, a SimProject class object is normally the root node of this hierarchy tree representing a unique simulation project. This root node can

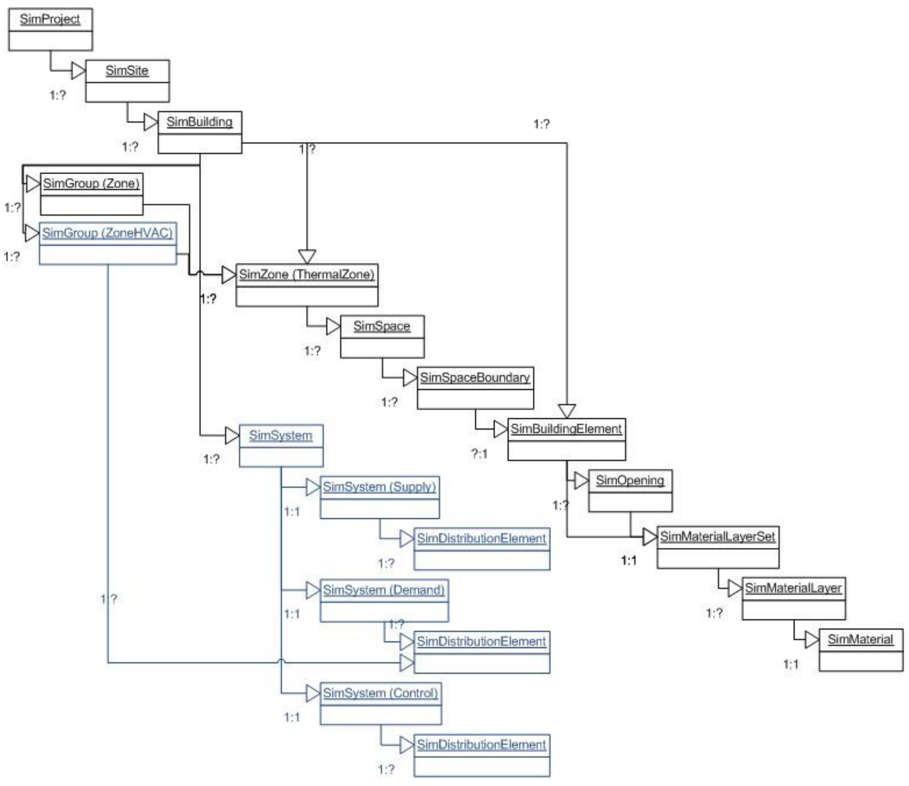


Fig. 7.22: SimModel hierarchy tree structure.

store multiple links that refer to different building design alternatives. Each design alternative also refers to a set of building elements, zones, HVAC systems distributed inside the building, etc. Therefore, the SimModel hierarchy saves a set of different SimModel elements as well as the links between them in a tree-based structure. At the bottom of the SimSystem sub-tree, different types of SimDistributionElement nodes are created to save different HVAC components located at different sides of the SimSystem, e.g., a hot water boiler will be saved as a child SimDistributionElement node of the SimSystem water supply side. At each SimDistributionElement node, a list of child nodes are created for saving different physical connections between these HVAC components, e.g., the pipe connections within a hot water looping system for connecting the boiler and pump. The physical connections saved by these child nodes represent the SimModel topology.

In SimXML, each model element is given a unique long type ID that distinguishes it from all other elements. Each parent element of the SimModel hierarchy links to a child element by saving its ID. As a result, “parse SimModel hierarchy” is a recursive algorithm that detects each SimModel element, locates its position in the hierarchical tree and creates a tree node with a link to its data. After that, we can also recursively iterate each node of this hierarchy, retrieve the link to the SimXML data element and print out the element data for visualizing the created hierarchy.

7.4.4.3 SimModel to Modelica Mapping Rule Schema and Rule Filter

The mapping rule concept was first introduced in mathematics, representing a particular transformation. This transformation describes the conversion of a source model data into a target model data under the constraints specified by a given equation system. As SimModel is significantly different from the data model of a specific Modelica library *[CMO+14]*, e.g., AixLib or BuildingSystems, we also need to define a set of mapping rules that can handle the difference between these two different data models. *[WCR+15][WMO+14]* proposed a set of different mapping rules that can convert the SimModel data into the Modelica model data defined by a specific BPS library AixLib *[EBC14]*. Based on this work, we developed a mapping rule schema in XML Schema Definition (XSD), containing the data model of the mapping rules between SimModel and different Modelica libraries *[CWT+15]*. We can thus efficiently re-use this mapping rule schema, originally developed for our transformation system, to define different sets of mapping rule instances for different target Modelica libraries. Mapping rules are classified according to three different levels in the schema:

- The first level is library mapping, which is designed to link different mapping rule instances for different Modelica libraries.
- The second level is component mapping, which is responsible for mapping SimModel components, e.g., a boiler of the HVAC system, into the corresponding component of the Modelica library specified by the first level mapping.
- The third level deals with the internal properties mapping of the components defined by the upper-level mapping rules.

Czarnecki and Helsen *[CH06][CH03]* discussed fundamental work on data model transformation. In context of their work we can view transforming SimModel data model to Modelica code as a special case of model-to-model transformations. We only need to provide the meta-models for the source model and the target programming language as well as the transformation defined with respect to the meta-models. A transformation engine transfers the source model data into the target programming language model. In our case, the source model is SimModel and the target language is Modelica. The transformation is an instance of the mapping rule schema saved as XML file format and the transformation engine contains a set of filters translating the model data by filtering the mapping rule instance. The source model data is stored in SimXML file and the target programming language model is the Modelica code, based on a specific BPS library.

7.4.4.4 SimModel Python API for Modelica Code Generation

The main parts of the transformation system are implemented in the C++ programming language, in order to satisfy the requirements of model transformation speed and the access to low-level system features, such as virtual memory allocation. Script programming languages, like Python, are heavily used for pre- and post-processing of Modelica. They are more flexible and easier to use for controlling the procedure of Modelica code generation based on techniques like pre-defined code templates and interpretation engines. Therefore, this sub-section introduces a generic API developed for interfacing SimModel C++ library and the script programming language Python, in order to better control the Modelica code generation by separating the translation logic from code generation.

We developed our prototype for this generic API based on a technique named language binding. In computer science, a binding from a programming language to a library is an API providing the glue code to use that library in a particular programming language. In the context of our generic API, bindings are wrapper libraries that bridge the C++ and Python programming languages in order to re-use the SimModel API generated for C++ in Python.

According to the testing on different binding libraries, e.g., ctypes, Cython, Boost.Python, we use SWIG *[SWI15]* as the binding libraries to wrap the SimModel API for data access out of Python. Compared to other binding libraries, SWIG is easier to port the interface of a large C/C++ library to other languages, like Python. It is an open-source and battle-tested binding software used by large companies, e.g., Google. It keeps both the C++ and Python code clean, i.e. not altering Python itself such as Cython is doing. SWIG provides C++ compatible data types, and allows calling functions in the Dynamic Link Library (DLL) or the other types of shared libraries from Python. In the system prototype, we exposed the mapped or translated SimModel components and their internal properties into a Python-based Modelica code generator via the SWIG wrapper. In order to provide Python with full controls on SimModel internal data, we also exposed more data objects from the SimModel hierarchy as well as the topology into Python via the generic API. Fig. 7.22 illustrates SimModel hierarchy exposed for Python.

7.4.5 Mapping Rules

Several steps need to be considered to complete the mapping between the SimModel data model and the Modelica libraries. Fig. 7.23 and Fig. 7.24 shows two *BPMN (Business Process Model and Notation)* process diagrams to illustrate the decisions and steps needed for a successful mapping.

As indicated in Section 7.3.7, in the formal mapping process we distinguish between object and parameter mapping.

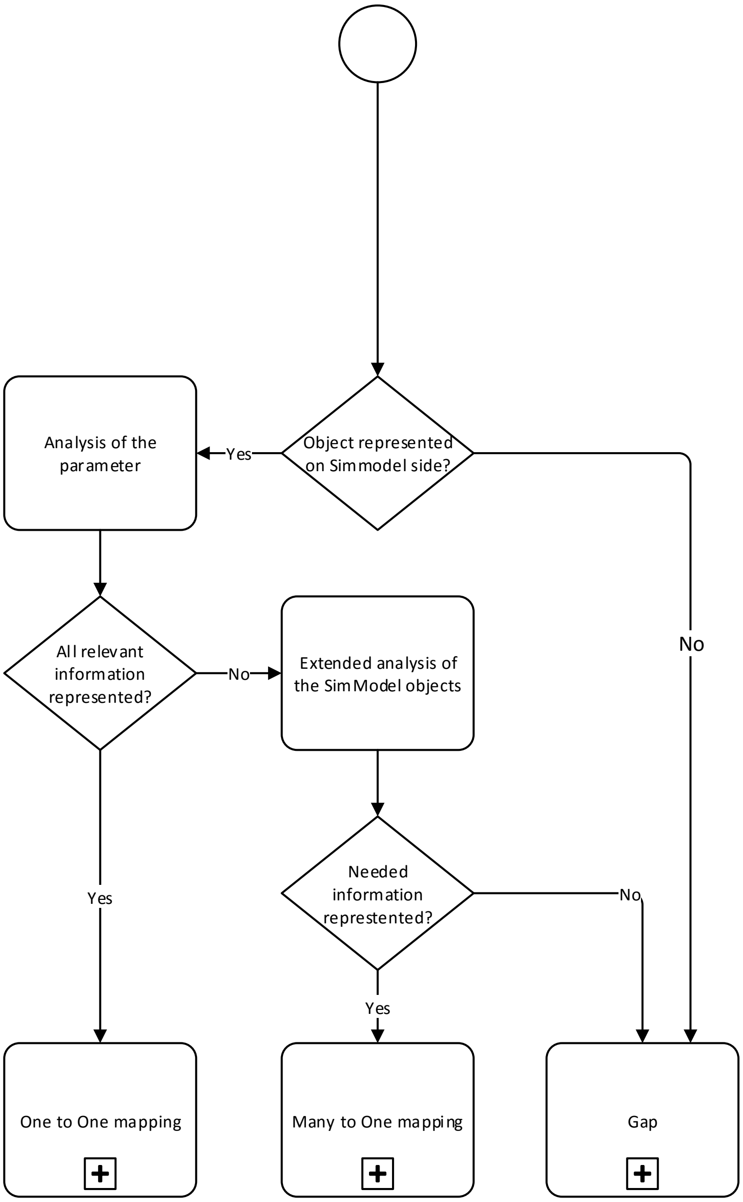

Object represented on Simmodel side?
Yes
No
Analysis of the parameter
All relevant information represented?
No
Extended analysis of the SimModel objects
Needed information represtented?
Yes
No
Yes
One to One mapping
Many to One mapping
Gap

Fig. 7.23: *BPMN process diagram for the object mapping*

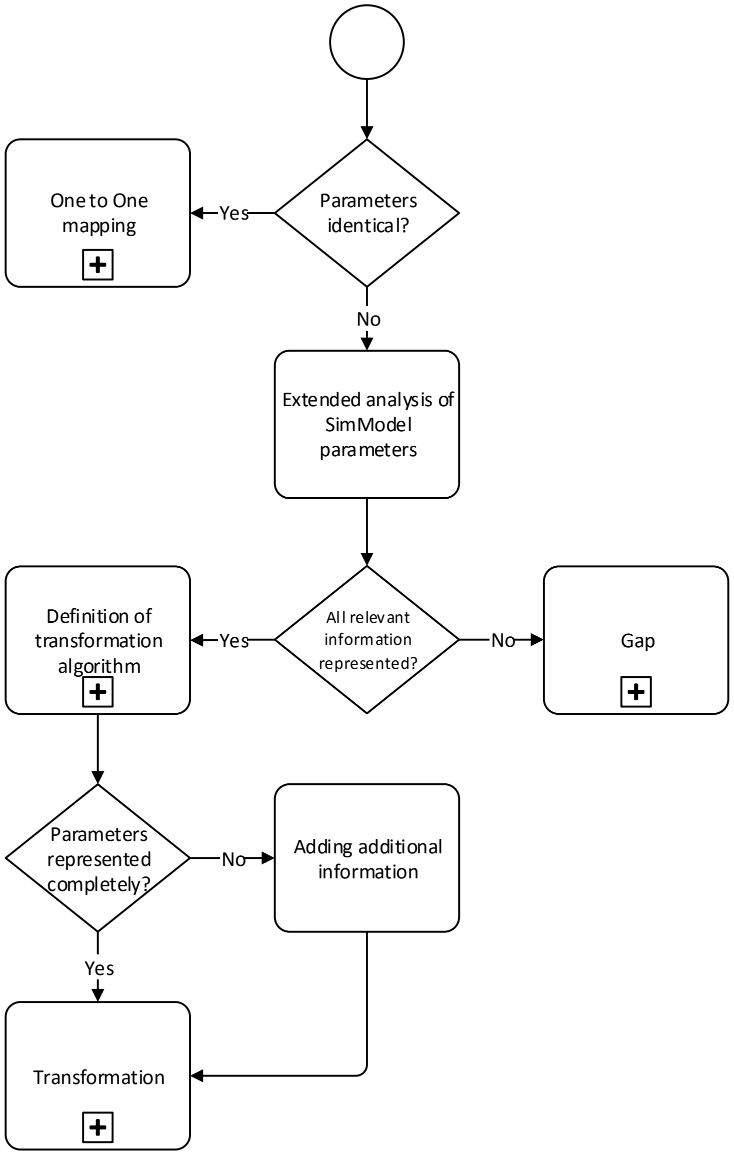

One to One mapping
Yes
Parameters identical?
No
Extended analysis of SimModel parameters
Definition of transformation algorithm
Yes
All relevant information represented?
No
Gap
Parameters represented completely?
No
Adding additional information
Yes
Transformation

Fig. 7.24: *BPMN process diagram for the parameter mapping*

The object mapping is performed as a first step in order to find all the relevant objects to represent the data needed in the specific Modelica library. Fig. 7.23 shows the sequence of the object mapping. Checking if the object needed for Modelica is represented in the SimModel data schema is the first step. To understand the objects of each side, the SimModel and the Modelica library documentations should be considered. If an object is not defined in SimModel, either the missing object needs to be added to the existing data schema or the Gap mapping rule needs to be applied. If it is represented, the next step is to check if all parameters in the SimModel object are sufficient for the targeted Modelica object. One to One mapping can be applied, if all relevant data are available. If further information are needed, other SimModel objects need to be considered in addition. In case several objects are needed to provide the relevant information, the Many to One mapping rule defines those. If still some information is lacking, the data schema of SimModel needs to be extended again by applying the Gap rule.

Fig. 7.24 illustrates the next steps to complete the mapping between the SimModel data schema and the Modelica libraries. If the parameters on both sides are defined identically, a simple One to One mapping rule can be applied. If additional information is needed, further parameters should be considered. Several SimModel parameters can be combined to define a single parameter on the Modelica side. To perform a combination of parameters, the transformation mapping rule needs to be applied. For defining the transformation algorithm, it can be necessary to add external data.

A specific example of the transformation rule is demonstrated below. The value of the set temperature for a gas boiler on the SimModel side is defined using the unit Celsius, whereas the unit for the Modelica side needs to be defined in Kelvin.

The transformation algorithm for this mapping process is

$$f = a + 273.15$$

where a is the temperature setpoint of a gas boiler in the SimModel data model.

As shown, the algorithm transforms the set temperature from Celsius to Kelvin, adding 273.15. Thus, the SimModel value is transformed and describes the relevant data for the Modelica side.

7.4.6 Python Tools for Modelica Code Generation

Python as an interpreted language is used widely in the Modelica community for scripting, workflow automation, pre- and post-processing and as a glue language. Using the wrapper for the libSimModel library generated by SWIG, most functions from the C++ library can be called from Python. The Python tools using the libSimModel API so far are a Qt based viewer and editor for the SimModel object tree, a high-level API that adds further functionality and implements the topology mapping and the code templating tool CoTeTo, that is used for generating Modelica code.

7.4.6.1 SimModelTreeView

Based on the API generated by SWIG, a simple viewer for the SimModel object tree has been implemented as a PyQt widget. It uses QTreeView and reads data from the API on demand when unfolding the tree. A second widget is implemented for viewing and editing the attributes of objects. Both widgets can be called on their own or be embedded in another GUI or be imported as a module.

7.4.6.2 High-Level API

The API as generated by SWIG reflects the functions defined in C++. For user convenience, a set of Python classes that can be loaded as a module has been implemented on top of that API. A UML diagram for these classes is shown in Fig. 7.25.

The classes each have attributes that are accessible using the dot notation. One attribute is, where applicable, an iterable list of child objects. The Python classes as well implement the topology mapping which is described in the following paragraph.

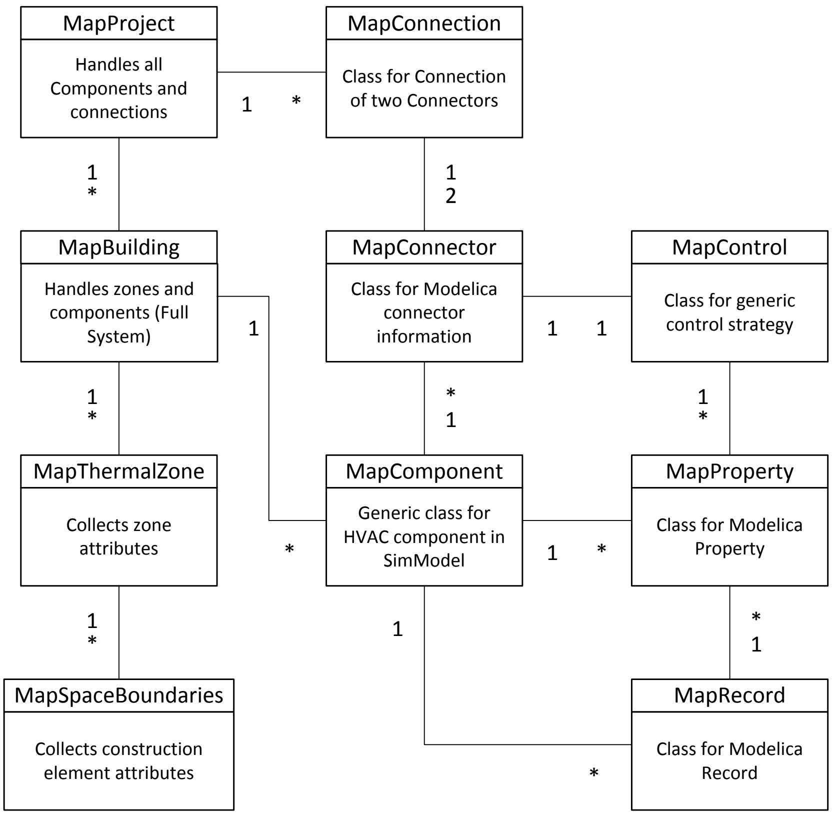

MapProject
Handles all Components and connections
MapConnection
Class for Connection of two Connectors
MapBuilding
Handles zones and components (Full System)
MapConnector
Class for Modelica connector information
MapControl
Class for generic control strategy
MapThermalZone
Collects zone attributes
MapComponent
Generic class for HVAC component in SimModel
MapProperty
Class for Modelica Property
MapSpaceBoundaries
Collects construction element attributes
MapRecord
Class for Modelica Record
1
*
2

Fig. 7.25: *UML diagram for the high-level API.*

7.4.6.3 CoTeTo: Code Templating Tool

The actual Modelica code generation is implemented as a tool named CoTeTo, which stands for Code Templating Tool *[TRR+15]*. Although designed for this project, this tool was implemented in a way that it can be used standalone and in other software environments. CoTeTo is released under the MIT open-source license at https://github.com/UdK-VPT/CoTeTo .

In this project, Modelica models for a set of different model libraries have to be generated using a common data source. Each library needs separate filtering and output of data because of different modeling approaches. These libraries are currently under development and are likely to change in the future as well. This requires a flexible and generic data conversion framework to allow for future changes. Thus, the framework should allow flexible output components for different libraries in multiple versions as well as flexible input components, both should be easy to maintain even for non-programmers.

The workflow of CoTeTo and the coupling to other tools within the toolchain is shown in Fig. 7.16.

We designed CoTeTo to be used by graphical, command line and library level interfaces. The multiple access possibilities open the framework to be used by a larger community.

The fact that Python does not require extensive compilation cycles helps with rapid development. The following section will give an overview of the components and their functionality. We divided CoTeTo into input components (*Data APIs*) and output components (*Generators*). A *Generator* depends on a specific *Data API* (defined by its name and version).

The Template Approach There are two general concepts for the generation of textual output within a computer program. One approach is to embed `print()`-statements for text strings and data in the structure of a program. This is useful for nearly static, well-defined structures of the data set and of the textual output.

The other approach is template-based, where placeholders for the content are embedded in a text file (a template for the output). Besides placeholders templates also offer control structures. Thus, template-based model generation allows complying with fixed Modelica language syntax and adding flexible model content in the same file. One advantage is the flexibility for the end user, who does not necessarily need to dive into the programs' internal structure, but can enrich the template file with placeholders and simple programming constructs, whenever the used Modelica models change. This workflow is much like the form letter function in office software, which fills some variable address fields in a text document from a database.

The template approach fits well into the flexible structure of the CoTeTo framework, as it is independently usable for different information sources. From the list of available template engines Mako *[Bay14]* and Jinja2 *[Ron14]* seem to fit best into CoTeTo. At this point support for both is implemented.

Input - Data APIs A *Data API* is a Python module that defines a prescribed way to fetch data sets from a data source. Although we use the Python language to write the CoTeTo, *Data API* functions can interface with other languages.

Different *Data APIs* and different versions can be used in parallel. Sample modules for reading JSON, XML and CSV files exist in CoTeTo. This allows flexible processes during development and testing. There is no definition for the structure of the returned data items, since different data sources contain different types of data (tree, table, graph, map). It is the job of the corresponding output *Generator* to understand the data delivered by the related *Data API*.

The most important *Data API* in the Annex 60 context is the interface to the C++ library libSimModel, which handles the SimModel parsing and the mapping to Modelica. Seen from the Python framework and from CoTeTo it defines the data source used to fill the placeholders in the output templates.

Output - Generators Once all relevant data have been loaded into CoTeTo, it is passed to the output component, called *Generator*. We designed the *Generator* to contain all items needed to generate the code for a specific Modelica library. This includes

- filter functions,
- the meta model structure,
- text templates,

- additional configuration and information and
- additional files.

The filter functions, meta model structure and text templates are used and applied by CoTeTo. Additional files like the mapping rules XML file can be stored inside the *Generator*.

Some data need manipulation that may not fit well into the mapping rule mechanism. For this purpose, *Generators* can include filter functions (Python code) that we call between the data API and the templates. The filters are custom-built to the used library. In our case, they may include simplification of geometric relationships and calculation of model specific parameters. Another application of filters would be the creation of annotations for the graphical appearance and placement of model components in the Modelica code.

One major challenge in the automated generation of Modelica models is the flexibility of Modelica. Generally said, setting up useful models needs the knowledge of an experienced user. We are following the approach to encapsulate this knowledge in library specific meta-models and templates. One essential task is the appropriate connection of components. The API returns the connection information corresponding to the SimModel ontology, which differs from the one in a Modelica library. The meta model checks if the connection is applicable, if not, it manipulates it.

The text templates are the last step in the process chain. The template engine combines the data structures returned by the *Data API* and possibly manipulated by filters with the text templates to files with valid Modelica code. The templates in a *Generator* can be split into several files to ease maintenance.

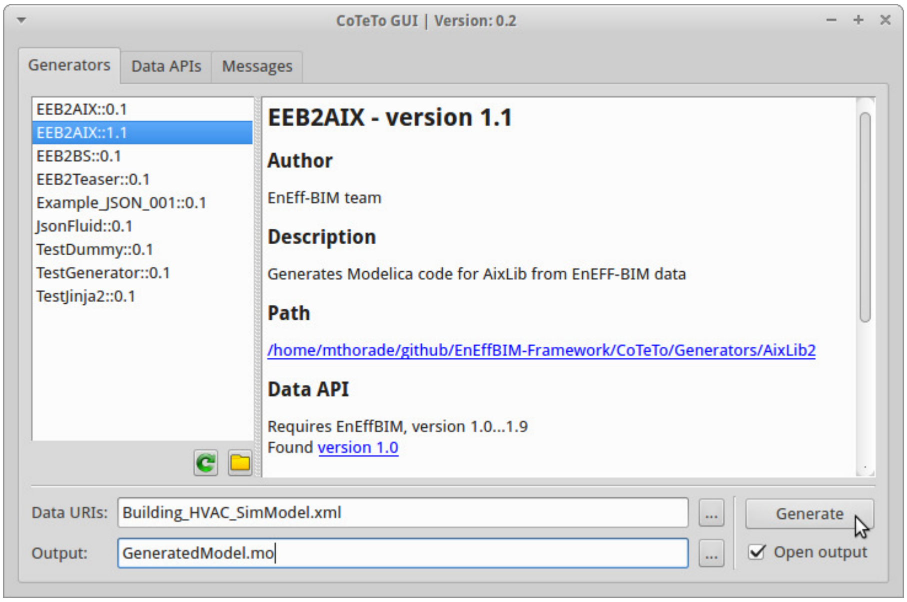
Generators can be easily exchanged between different installations, as they are simple folders or even zip-files with a defined structure. *Generators* can be maintained and edited with standard system tools like a file manager and a text editor. Creating a new *Generator* is as simple as copying a folder with an existing *Generator* and changing the name or version number in a text file.

One example for a *Generator* template using Mako template engine is given in following listing. Static elements can easily be implemented as plain text in the template (e.g. *within* line 1). A Dollar symbol, followed by curly brackets mark the placeholder, e.g. *MapData.used_library* in line 1 is replaced by the Python class attribute *used_library* in the complied template. Control elements, like for loops, are identified by a percentage symbol and follow the Python syntax, with the exception that indentation is not supported and the control element has to be closed (e.g. *%endfor*, line 6). This example template generates a simple model, by printing the library location (*target_location*) and individual name (*target_name*) in the first loop from line 4 to 6 and their corresponding connections in line 9 to 13.

```
1  within ${MapData.used_library};
2  model ${MapData.name}
3
4  %for comp in MapData.hvac_components:
5  ${comp.target_location} ${comp.target_name}();
6  %endfor
7
8  equation
9  %for con in MapData.connections:
10 connect(
11    ${con.con_a.parent.target_name}.${con.con_a.name},
12    ${con.con_a.parent.target_name}.${con.con_a.name});
13 %endfor
14
15 end ${MapData.name};
```

User Interface and Handling There are currently three ways to use CoTeTo:

- CoTeTo can be imported in Python software as a module library. CoTeTo works both with Python 2.7 and 3.3+. All functions are usable via the modules API.
- A command line interface can be used interactively or called from other software. It allows listing the available *Data APIs* and *Generators* and executing a *Generator* with a data source URI to produce the text output.
- The graphical user interface (GUI) is implemented using PyQt4. It allows flexible browsing and editing of all components and included files and the execution of selected *Generators*. The GUI can be used as a standalone tool (see Fig. 7.26) or embedded in PyQt4-based applications as a widget.



EEB2AIX::0.1
EEB2AIX::1.1
EEB2BS::0.1
EEB2Teaser::0.1
Example_JSON_001::0.1
JsonFluid::0.1
TestDummy::0.1
TestGenerator::0.1
TestJinja2::0.1

EEB2AIX - version 1.1

Author

EnEff-BIM team

Description

Generates Modelica code for AixLib from EnEFF-BIM data

Path

/home/mthorade/github/EnEffBIM-Framework/CoTeTo/Generators/AixLib2

Data API

Requires EnEffBIM, version 1.0...1.9
Found version 1.0

Data URIs: Building_HVAC_SimModel.xml

Output: GeneratedModel.mo

Fig. 7.26: *Screenshot of the CoTeTo user interface*

7.4.7 Topology Mapping

As already stated in the introduction of this section, the Mapping Rules (Section 7.4.5) deal with objects and parameters from SimModel to a specific Modelica library. In addition to the Mapping Rules in its current status we need information about model topology (e.g. number, type and name of Modelica connectors attached to the model) to generate valid Modelica models.

We define model topology as the way that the model in a certain object oriented language or information model is described in terms of connection to other models. In addition it offers the possibility to characterize which parts of the model are included in the object and which additional objects need to be connected in order to work appropriate (e.g. different control elements). This topology differs from SimModel to Modelica libraries. One advantage of the presented work is that the transformation process complies and adapts with the BIM and the specific model topology in Modelica libraries. Neither the topology of the BIM model nor the topology of the Modelica library need to be changed.

Furthermore the topology of models in different Modelica libraries may differ, even if the models describe the same behavior/object. To extend the information provided by the BIM with these Modelica and library specific information we developed a set of Python classes. This section describes the different topologies of SimModel and Modelica in more detail, taken the AixLib as an example. Furthermore, the Python classes and the usage for other libraries are demonstrated.

In SimModel as well as in Modelica different objects are connected by *connections* with different *connectors*. However, the type of connections differs for these languages. For the given use cases and with a special focus on BPS we identified four connector types in Modelica to be significantly more important. Table 7.2 lists these Modelica and the corresponding SimModel connector types. In addition, Fig. 7.27 show the different connectors in Modelica, the left column shows the connectors dedicated for design input, the right for design output respectively. However, for Fluid and thermal connectors this is just a guideline as these connectors are acausal. The Fluid connector can be used for various media (e.g. water or air) by changing the *Medium* attribute inside the connector.

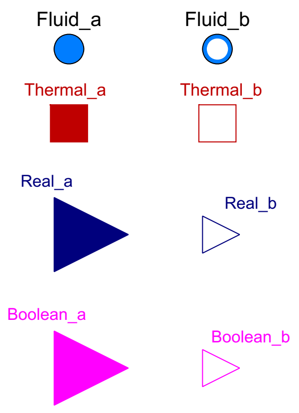


Fig. 7.27: Different connector types in Modelica (graphical representation).

Table 7.2: Comparison of Modelica and SimModel connector types.

MSL	SimModel
Fluid connector (Medium: water)	Water connector (cold, hot)
Fluid connector (Medium: air)	Air connector
Thermal connector	Air connector
Real connector	Control connector
Boolean connector	Control connector

As Table 7.2 shows one connector type in Modelica may correspond to several connector types in SimModel and vice versa. The topology mapping needs to consider the different connectors and in the code generation distinguish between them. This may be done by setting the correct parameters in the model, for example in the case of a water connection in SimModel, the corresponding media of the fluid is also set to water, by an mechanism in the Python topology classes.

Not only the type of connections differs from SimModel to Modelica but also the number of connectors per specific model and in particular the function and name of the connectors can differ from library to library (e.g. in the case of different Pumps). Thus, it has an effect on the topology of the models in this library and the library itself. As the transformation process was designed to be flexible enough, the library developers do not have to change their library, but adapt the transformation process. This is advantageous as the library can be developed for different purposes, without delimiting the development because of BIM required restrictions. Other pre- and post-processing tools of the individual library do not need to be changed.

In the following we present three examples and compare the topology and connections of the Modelica objects from AixLib (the topology may differ in other libraries) and SimModel. The considered objects are:

1. Pump
2. Radiator in a thermal zone
3. Temperature controlled valve

7.4.7.1 Pump

The first example considers a pump. In SimModel the specific object is called *SimFlowMover_Pump_VariableSpeedReturn*. It has three different connector types:

- *SimNode_HotWaterFlowPort_Water_In*
- *SimNode_HotWaterFlowPort_Water_Out*
- *SimNode_DigitalControl_HWLoop_DigitalSignal_In*

Water_In and *Water_Out* illustrate the causal approach of SimModel and are used to connect the pump within a hydraulic loop. The *Digital-*

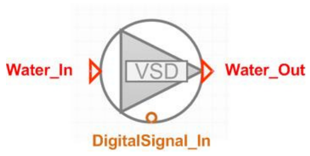


Fig. 7.28: *Connector types in Simergy shown on a pump (graphical representation)*

Control input is used to attach a certain control object to the model. The AixLib holds five different implementations that could be used as a pump. We take *AixLib.Fluid.Movers.Pump*, which is AixLib specific, and *AixLib.Fluid.Movers.FlowControlled_dp*, which is inherited from an Annex60 library example to point out differences (see Fig. 7.29). Both models have two *Fluid* connectors to connect them within a hydraulic loop. This concept aligns with the topology in SimModel. Further, both Modelica models have inputs to control the pump. *AixLib.Fluid.Movers.FlowControlled_dp* uses a real input named *dp_in* to imprint the pressure difference of the pump, which may come from arbitrary control strategies. In contrast to that the developers of *AixLib.Fluid.Movers.Pump* implemented a control strategy in the model itself, to use the model for specific applications. The control strategy uses a boolean signal to switch between two different operational characteristics. This input is called *IsNight*. Both the type and the name of all three connectors need to be added within the topology mapping. In addition, the control strategies of the pumps need additional model objects (e.g. a boolean pulse), the instantiation and correct connection of these additional objects can also be assured in the topology mapping.

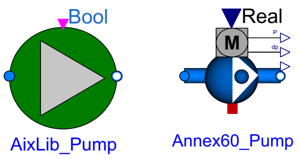


Fig. 7.29: *Comparison between* AixLib.Fluid.Movers.Pump *and* AixLib.Fluid.Movers.FlowControlled_dp

7.4.7.2 Radiator in a Thermal Zone

To distribute thermal energy into a thermal zone, radiators can be used. The integration into a hydraulic loop is very similar to the procedure described in the above example and thus not part of this section. To semantically connect a radiator and the thermal zone, the concepts of SimModel and the approach used in AixLib differ, see Fig. 7.30. In SimModel the radiator is attached (by referencing it with an ID) to a *ZoneHvacGroup*, which is a collection of all HVAC equipment used or aligned to that certain zone. In AixLib the distribution of thermal energy is done by thermal connectors, that calculate the heat flux for convective and radiative heat transfer respectively. Therefore both, the radiator and the thermal zone hold each two thermal connectors. Between these connectors the topology mapping needs to instantiate the correct connections [*connect(Radiator.heatPortCon, ThermalZone.internalGainsConv)* and *connect(Radiator.heatPortRad, ThermalZone.internalGainsRad)*]. It is worthwhile mentioning that the names of the connectors and models are again model and library specific. Further standardization would simplify the conversion process in the future.

7.4.7.3 Temperature Controlled Valve

This example illustrates the different handling of controls in SimModel and AixLib. We consider a PID controlled valve for flow regulation in a hydraulic loop. The measured variable in this case is the air temperature in the thermal zone, set point is a constant value and the manipulated variable is the opening of the valve. SimModel has no such control, the radiator is controlled by an ideal controller that is connected with *SimNode_DigitalControl_HWLoop_DigitalSignal* connectors to the thermal zone and the *ZoneHvacGroup*, respectively. For AixLib we need to instantiate a PID controller, a constant block and a temperature sensor (see Fig. 7.31, all taken from MSL) and connect them in the correct manner with the valve and the thermal zone air node.

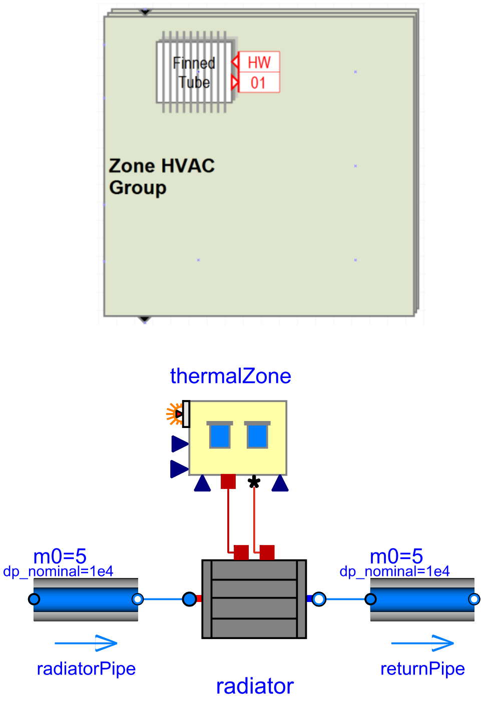


Fig. 7.30: *Connection between a radiator and a thermal zone in SimModel and AixLib*

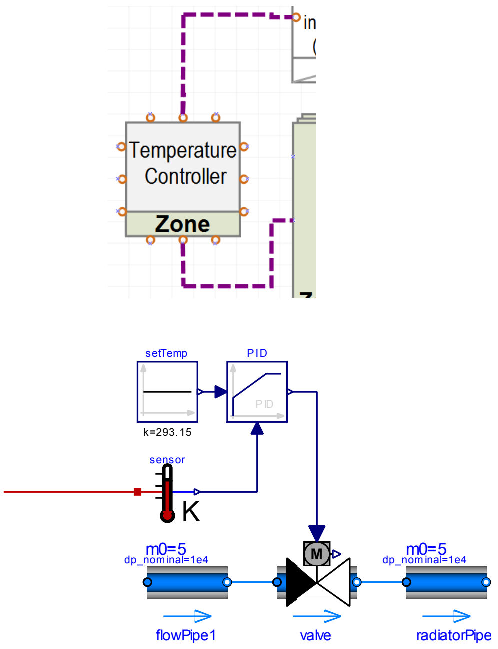


Fig. 7.31: *Temperature controlled valve in SimModel and AixLib.*

7.4.7.4 Python Classes for Topology Mapping

The set of python classes used for topology mapping can be obtained from Fig. 7.32. The idea is to provide highly reusable classes with general functions for the transformation of SimModel and Modelica, including the processing of data from libSimModel API and providing functions for topology mapping. Individual library can directly use the classes (e.g. MapProject or MapBuilding) or inherit them and extend it with library specific information (e.g. MapThermalZone and MapComponent). The Python classes and an example implementation for the AixLib models can be found at https://github.com/EnEff-BIM/EnEffBIM-Framework.

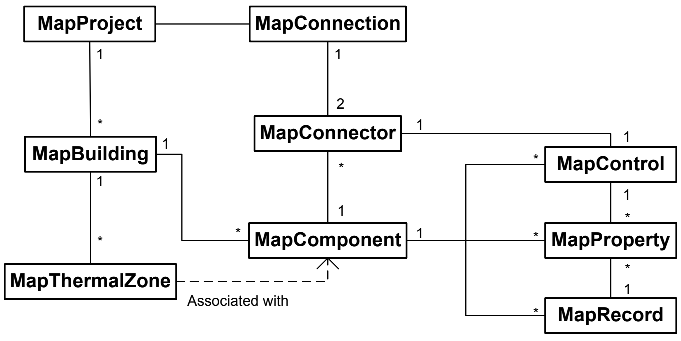


Fig. 7.32: Simplified UML diagram of the Python classes used for applying information from libSimModel API and topology mapping.

7.5 Use Cases and Demonstration

To test the tool chain and to analyze the different data models, in the Annex 60, a set of nine use cases was defined. Each use case is applied in the different steps of the tool chain. By modeling each of these use cases in IFC, SimXML and Modelica each (where possible, see more at the end of this section) it was possible to compare data structures and to determine requirements for

Table 7.3: Overview of the use cases for transformation from BIM to Modelica.

Identifier and name:	HVAC System (generation + distribution)
1.1 Boiler	gas boiler + radiator
1.2 Boiler	gas boiler, buffer storage for domestic hot water + radiator
2.1 Heat Pump	heat pump, buffer storage + radiator
2.2 Heat Pump	heat pump + floor heating
3 CHP	combined heat and power unit + radiator
4.1 AHU	air handling unit heating
4.2 AHU	air handling unit cooling
5 Multi Zone	combination (Boiler 1.1, AHU 4.1, AHU 4.2)
6 Rooftop Building	test case for geometry processing

the transformation from BIM to Modelica. While the information structure in IFC and SimModel will not change, the structure of Modelica libraries differs, as already described. Table 7.3 provides an overview of the use cases. All use cases contain different HVAC-systems, to ensure that different components with multiple model topologies are considered. For the sake of better readability, this list does not include other necessary components like pipes, ducts, pumps or control elements, however, these components are part of the use cases as well.

The transformation of BIM to Modelica in this project focusses on HVAC components. Thus, we use the same thermal zone for the first seven use cases to calculate the thermal losses. This test room is parameterized according to German guideline VDI 6007-1 (Fig. 7.33). Basic characteristics of the room are:

- One room with a window, on the second floor of a three story-building
- Floor area: 17.5 m^2
- Outer wall area: 3.5 m^2
- Window area: 7 m^2
- Adiabatic (no heat transfer through) internal walls and slabs. Heat transfer only through the outer wall and the window
- internal loads: occupancy, plug loads, lighting

- Weather-data: TRY dataset for Germany, zone 5 (Aachen)

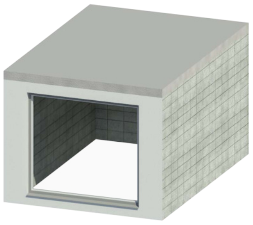


Fig. 7.33: *The single room architecture for the use cases 1.1 - 4.2.*

For weather boundary conditions we are using a Test Reference Year (TRY) from Aachen, Germany. Internal loads for persons, plug loads and lights are taken from DIN 18599 and SIA 2024.

Use case eight was defined as multi-zone test case. Use case nine is dedicated to geometry processing and testing.

7.5.1 Use Case 1.1: Boiler with Radiator

Short identification

HVAC-system of Use Case 1.1 (Fig. 7.34) consists of the following items:

- gas fired boiler
- pump
- expansion vessel
- radiator with P-controlled valve
- pipes to connect components

Gas fired boiler

The gas boiler has a maximum heat output of 1300 W. The flow temperature is set to a constant value of 340 K. The volume of the water inside the heat exchanger is 0.01 m^3. The efficiency is indicated in the form of Table Table 7.4 as function of part load ration.

Pump with night set back

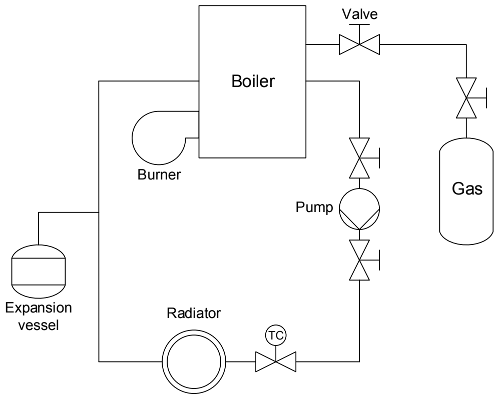


Fig. 7.34: Use case 1.1.

Table 7.4: Efficiency of gas fired boiler as function of part load ratio.

Part load ratio	Efficiency
0.0	0.78
0.2	0.78
0.4	0.82
0.6	0.84
0.8	0.86
1.0	0.88

Table 7.5: Head of pump as function of volume flow.

Flow [m^3/h]	min head [m]	max head [m]
0.0	0.6	5.0
0.5	0.4	4.5
0.75	0.3	3.0
1.3	0.0	2.5
1.5	0.0	1.5
2.5	0.0	1.0
3.0	0.0	0.5
3.5	0.0	0.0

The pump has a maximum head of 5 m and a maximum flow of 3 m^3/h (if the control strategy requires a maximum flow). The flow characteristics of the pump are given in Table 7.5 as dependence of the flow (in m^3/h). During 6 pm and 5 am the night signal is turned on and the minimal characteristics are used.

Radiator

The radiator type is a single-row radiator with one convector and cladding. The characteristics of the radiator can be taken out of Table 7.6. These characteristics follow the description of the European standard ISO EN 442 .

Control valve of radiator

The valve is connected to a P-Controller. The gain is 0.1, lower and upper limits are 0 and 1, respectively. The set temperature for the controller is set to 20 ℃.

Pipes

All pipes used to connect components have the same properties. They are 3 m long, have a diameter of 0.03 m and a roughness of $2 * 10^{-}5m$.

Expansion vessel

The expansion vessel ensures a constant pressure of 1 bar on the suction side of the pump.

Table 7.6: *Radiator characteristics according to ISO EN 442*

Parameter	Value
Nominal power	979 W
Nominal flow Temperature	75 ℃
Nominal return Temperature	65 ℃
Nominal room Temperature	20 ℃
Volume	3.15 l
Mass of steel	19.58 kg
Percent of radiative heat	0.35
length	1 m
height	0.6 m
Exponent for heat transfer	0.35

7.5.2 Use Case 1.2: Boiler with Radiator and Domestic Hot Water

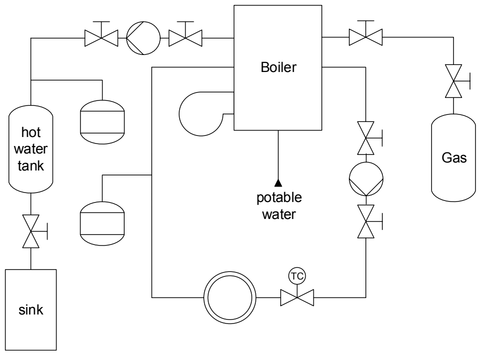


Fig. 7.35: *Use case 1.2.*

Short identification

HVAC-system of use case 1.2 (Fig. 7.35) consists of following items:

- gas fired boiler (see Section 7.5.1)
- pump (see Section 7.5.1)

Table 7.7: Buffer storage characteristics.

Parameter	Value
diameter	0.72 m
storage height	1 m
volume of heat exchanger inside storage	0.05 m^3
heat exchanger heat transfer coefficient	1500 W/(m^2 K)
heat exchanger area	20 m^2
thermal conductivity of insulation	0.04 W/(m K)
thickness of insulation	0.2 m
internal heat transfer coefficient	1500 W/(m^2 K)
external heat transfer coefficient	15 W/(m^2 K)

- expansion vessel (see Section 7.5.1)
- buffer storage for domestic hot water
- three way valve for switching loading/unloading of buffer storage
- radiator with P-controlled valve (see Section 7.5.1)
- pipes to connect components (see Section 7.5.1)

Gas fired boiler

The domestic hot water system and the water for the heating system is seperated. The potable water has a seperate port connected to the public water system.

Hot water tank

The hot water tank is only used for domestic hot water. The usage of domestic hot water is triggered by a simple time table to indicate when hot water is used. The set value of the flow temperature is 70 °C. The buffer storage has following parameters:

Three way valve

The three way valve is used to control loading of the buffer storage, with a simple on/off strategy. The storage is supplied with hot water (valve fully opened) when the room temperature is warm enough and the temperature on the top level of the storage is too cold. In any case the valve opening to the buffer storage is set to zero.

Table 7.8: *Electric power as function of sink and source temperature*

Sink/source temp. [K]	273.15	283.15	288.15
308.15	203 W	212 W	217 W
328.15	295 W	323 W	337 W

7.5.3 Use Case 2.1: Heat Pump with Radiator

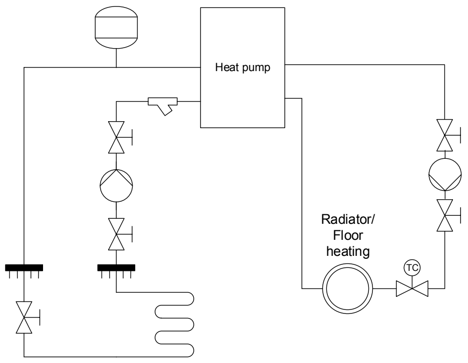


Fig. 7.36: *Use case 2.1 & 2.2.*

Short identification

HVAC-system of use case 2.1 (Fig. 7.36) consists of following items:

- heat pump with fixed source temperature
- pump (see Section 7.5.1)
- expansion vessel (see Section 7.5.1)
- radiator with P-controlled valve (see Section 7.5.1)
- pipes to connect components (see Section 7.5.1)

Heat pump

The heat pump characteristics are given in Table 7.8 and Table 7.9. Based on these tables the electric power and the heat flows of the condenser can be calculated as a function of source and sink temperatures.

Table 7.9: Condenser power as function of sink and source temperature

Sink/source temp. [K]	273.15	283.15	288.15
308.15	885 W	1162 W	1300 W
328.15	811 W	1060 W	1185 W

Table 7.10: Buffer storage characteristics.

Parameter	Value
Pipe spacing	0.2 m
external diameter of pipe	0.02 m
thickness of pipe	0.0025 m
thermal conductivity of pipe	0.35 W/(m K)
thickness of concrete above pipe	0.02 m
thickness of concrete below pipe	0.02 m
thermal conductivity of concrete	0.35 W/(m K)
density of concrete	0.35 kg/m^3

7.5.4 Use Case 2.2: Heat Pump with Underfloor Heating

Short identification

For this use case the thermal zone is slightly changed. Instead of a radiator for heat distribution into the thermal zone an underfloor heating system is used (Fig. 7.36).

HVAC-system of use case 2.2 consists of following items:

- heat pump with fixed source temperature (See Section 7.5.3)
- pump (See Section 7.5.1)
- expansion vessel (See Section 7.5.1)
- underfloor heating
- pipes to connect components (See Section 7.5.1)

Underfloor heating

The underfloor heating is installed for the whole area (17.5 m^2) of the obtained room. Further characteristics are described in Table 7.10.

7.5.5 Use Case 3: Combined Heat and Power (CHP) Unit

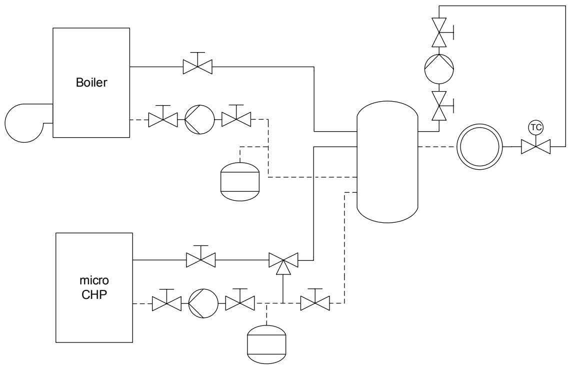


Fig. 7.37: *Use case 3.*

Short identification

HVAC-system of use case 3 (Fig. 7.37) consists of following items:

- CHP unit
- pump (See Section 7.5.1)
- expansion vessel (See Section 7.5.1)
- radiator with P-controlled valve (See Section 7.5.1)
- pipes to connect components (See Section 7.5.1)
- buffer storage (See Section 7.5.2)
- three way valve for loading and unloading

Combined Heat and Power unit

The boiler is replaced by a Combined Heat and Power unit with following characteristics:

Three way valve

The three way valve is used to control unloading and loading of the buffer storage. If the thermal zone requires thermal energy, first the buffer storage is completely unloaded, until it is not able to satisfy the flow temperature. The

Table 7.11: CHP characteristics.

Parameter	Value
Electrical output	1000 W
Thermal output	2500 W
Electrical efficiency	0.27
Thermal efficiency	0.67
Fuel utilization rate	0.92
Modularity	0.5 - 1.0

storage is loaded if the temperature in the thermal zone is above the set temperature and the buffer storage temperature reached the flow temperature plus a additional temperature difference of 5 K.

7.5.6 Use Case 4.1: Air handling Unit for Heating (AHU Heating)

Short identification

HVAC-system of use case 4.1 (Fig. 7.38) consists of following items:

- Air ducts
- supply/ return air damper
- silencer
- pressure controlled fans for supply/ return air (radial/ axial)
- air filter for supply/ return
- electric device for heating

Damper

The dampers for supply and exhaust air are set to a constant opening fraction of 0.7. The face area of the dampers are 0.1 m^2.

Fan

The fan is pressure controlled to supply the zone with heated air. The pressure difference is controlled with a PID-Controller PID (gain = 100, time constant integrator = 20 s, time constant deriative = 4 s). Lower and upper limits for

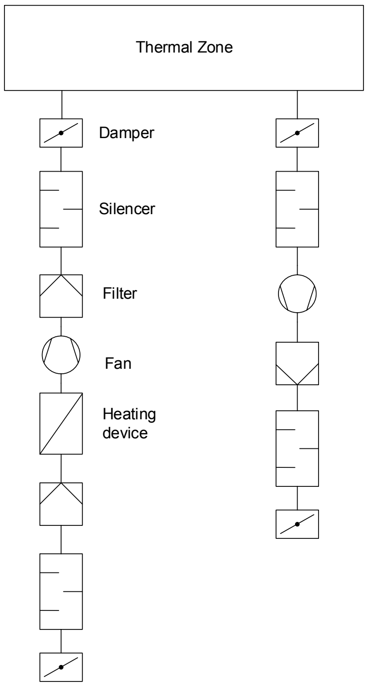


Fig. 7.38: *Use case 4.1.*

pressure difference are set to 18 Pa and 550 Pa, respectively. The heat added by the fan is negligible.

Filter, silencers, ducts

Several components like the filter, the silencer or the ducts introduce pressure losses into the system. These pressure losses are combined and set to 300 Pa.

Heater

The heater increases the supply air temperature to reach the room set temperature (20 ℃). The maximum temperature of the supply air is set to 40 ℃. The power of the heater is set 1300 W and controlled with a P-Controller, the pressure loss is 200 Pa.

7.5.7 Use Case 4.2: Air Handling Unit for Cooling (AHU Cooling)

Short identification

HVAC-system of use case 4.2 (Fig. 7.39) consists of following items:

- Air ducts (see Section 7.5.6)
- supply/ return air damper (see Section 7.5.6)
- silencer (see Section 7.5.6)
- pressure controlled fans for supply/ return air (radial/ axial) (see Section 7.5.6)
- air filter for supply/ return (see Section 7.5.6)
- Evaporative, adiabatic cooling device

Evaporative, adiabatic cooling device

The cooler reduces the supply air temperature to reach the rooms set temperature, by adding moisture the air stream. The temperature of water that is added to the fluid stream is set to 8 ℃. The added water is controlled with a P-controller using the zone set temperature of 20 ℃.

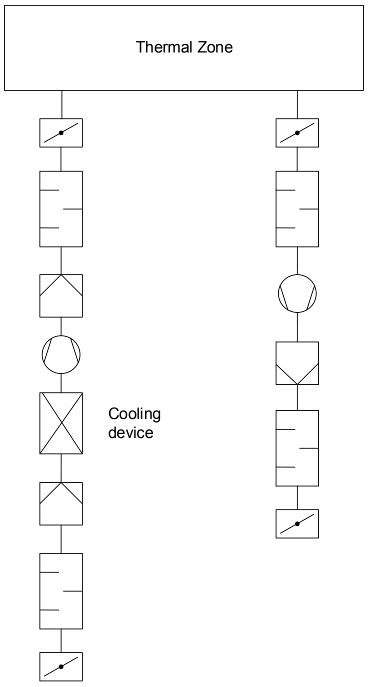


Fig. 7.39: *Use case 4.2.*

7.5.8 Use Case 5: Multi Zone Model

Short identification

This case study building represents a three story office building, located in Aachen.

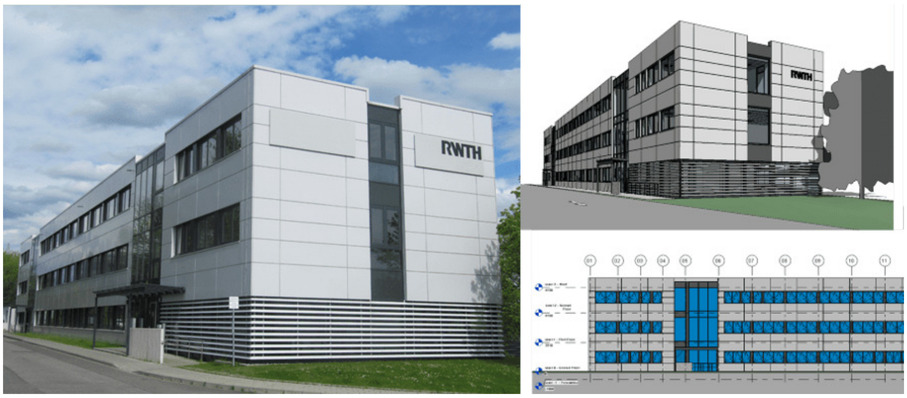


Fig. 7.40: Case study building, address: Mathieustraße 30, 52074 Aachen, Germany.

The building primarily comprises office spaces but also contains kitchens, washrooms, corridors, mechanical rooms and a server room. The heating system consists of a gas boiler and radiators (see Section 7.5.1). The BIM model developed in Revit Architecture is shown on the right and a picture of the building is shown on the left of Fig. 7.40. The heating systems, also modelled with revit is shown in Fig. 7.41

The thermal zones are defined, according to the DIN V 18599. It defines the thermal conditions for the room type. The office type rooms should have an air temperature of 21 ℃. The presence of persons is between 7:00 and 17:30. The metabolic rate is equal to a sitting person, emitting 80 Watt per square meter skin surface. The light switches on at 17:00 and other equipment, like computers and screens are activated the whole time people are present. Because the offices are heading in different directions the operations of the zones to guarantee the thermal comfort is different. One side of the building is heading mostly to to the west and the other side is heading mostly to the east. The

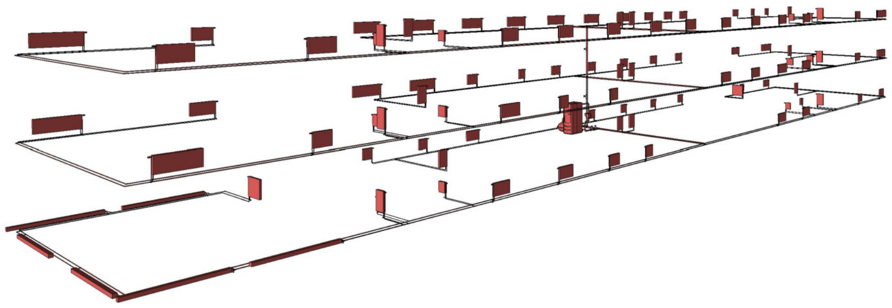


Fig. 7.41: *Whole heating system of use case five, modelled with Revit 2016.*

Table 7.12: *Multi zone use case characteristics.*

Parameter	Value
Thermal zones	2 on each floor
Thermal output	80490 W
Pump performance	9068 Pa
Volume flow	2.22 l/s
Number of radiators	96

differences in the solar radiation requires different operations for the heating systems. The corridor should have an air temperature of at least 15 ℃ and the light is turned on at 16:30. Nearly no solar radiation hits the corridor directly. The different thermal zones are shown in Fig. 7.42, differently coloured.

Use Case 5 modeled in Modelica, using the AixLib library is shown in Fig. 7.40. The six thermal zones are represented on the top. To improve the simulation performance, the thermal zones were simplified in an additional step. The nine thermal zones have been comprised to six thermal zones. The corridor is separated, so that one half of the corridors's thermal zone is part of the eastern and the other part belongs to the western thermal zone. The heating system is modeled below the zones. All 96 radiators are represented, to serve the building.

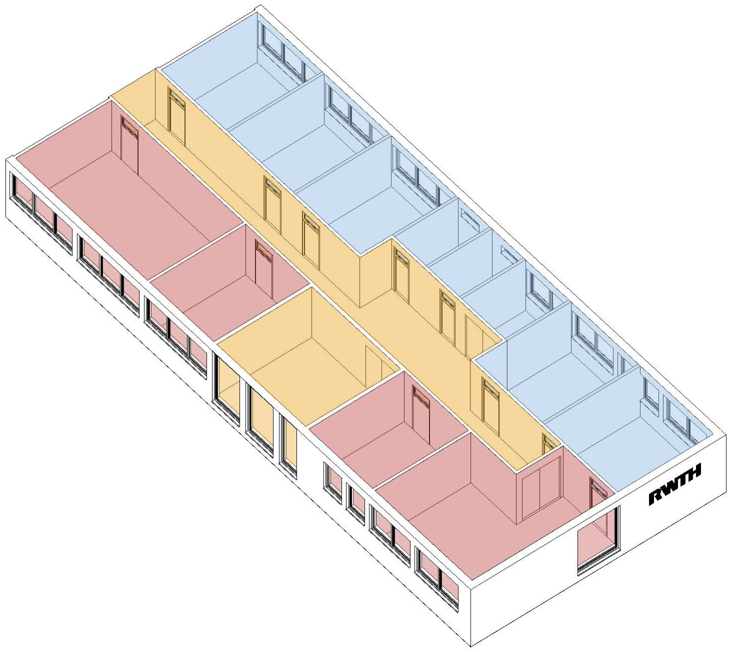


Fig. 7.42: *Thermal zones for the western and eastern offices and for the corridor, defined in Revit 2016.*

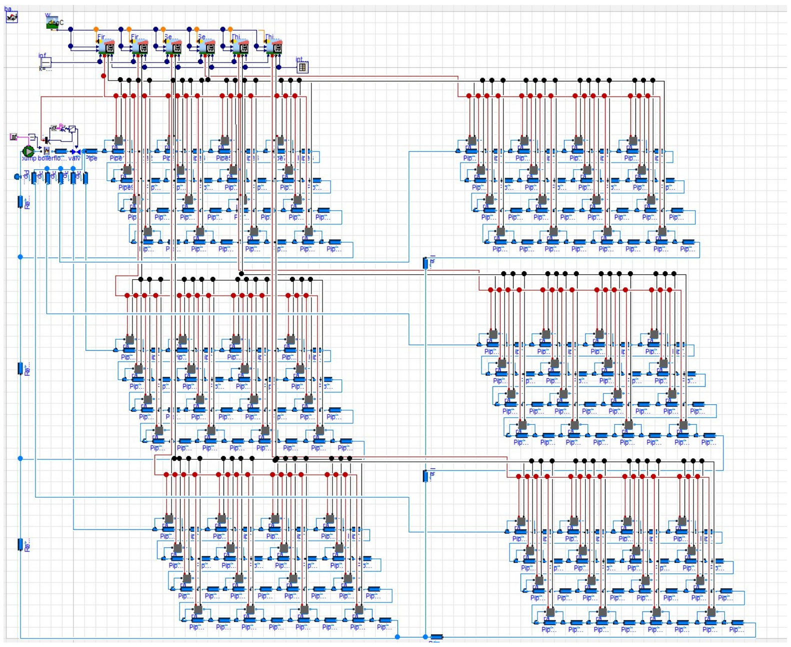
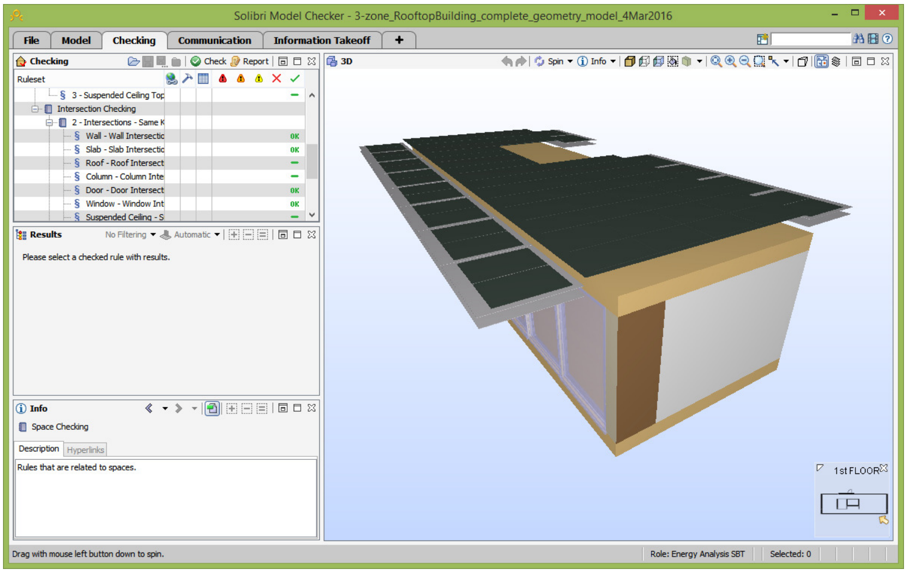


Fig. 7.43: *Use Case 5 modeled with the AixLib Modelica library.*

7.5.9 Use Case 6: Rooftop Building

Building geometry modeling for use with Modelica algorithms must be precise and must include all relevant detail, must be “clean” (i.e. error-free) and must be thoroughly checked and validated. If not, Modelica algorithms can lose their effectiveness. As virtually all major model-based CAD tools have the ability to generate building geometry models that are precise, it is always the modeler’s task to create models precise enough to meet the geometry definition needs of Modelica algorithms, and to assure that these models are technically error-free and verifiable *[BMNG16]*. Technically error-free models do not contain modeling errors and pass all tests with model checking tools. An example of such a model checker 3D view after successful validation is shown in Fig. 7.44. Yet, such models may still not be verifiable because buildings data used in the modeling process may be incorrect. Even the most competent modeler cannot create an error-free and verifiable model without access to trustworthy data about the building.



Ruleset	
§ 3 - Suspended Ceiling Top	–
Intersection Checking	
2 - Intersections - Same K	
§ Wall - Wall Intersectio	OK
§ Slab - Slab Intersectio	OK
§ Roof - Roof Intersect	–
§ Column - Column Inte	–
§ Door - Door Intersect	OK
§ Window - Window Int	OK
§ Suspended Ceiling - S	–

Please select a checked rule with results.

Space Checking

Rules that are related to spaces.

Drag with mouse left button down to spin.

Role: Energy Analysis SBT

Selected: 0

Fig. 7.44: Geometry model of the Rooftop building in Solibri Model Checker.

In the cases of modeling existing buildings, ideally the modeler has access to "as-built" documentation that provides reliable data about building construction. Unfortunately, "as-built" documentation seldom defines what was actually built – it does not document last-moment construction and installation changes. This inevitably results in "as-built" model inaccuracies, which contribute to the gap between simulated and measured building performance.

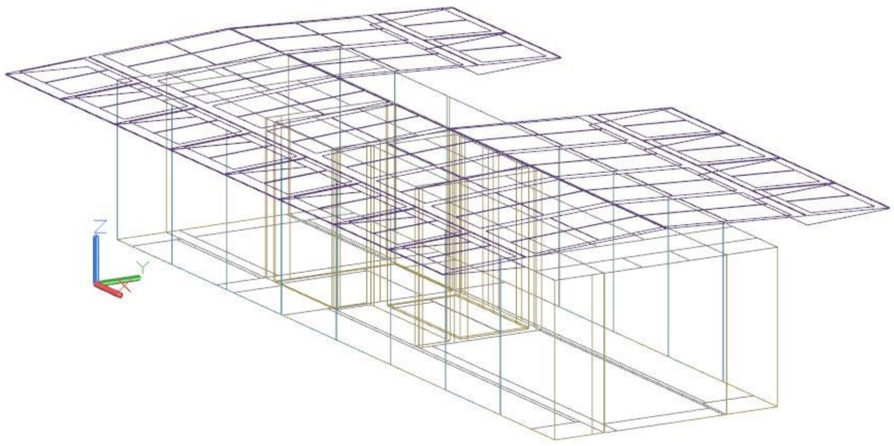


Fig. 7.45: *Geometry model of the Rooftop building in passive state as imported into EnergyPlus, showing space boundaries, PV panels and shutters in their original position.*

The Rooftop Building is a good example of "best-practices" modeling for BEP simulation (Fig. 7.45). Its geometry model was built specifically to provide precise geometry data for use in the execution of Modelica algorithms that model energy systems installed in the building. As shown in Fig. 7.45, the building features an elaborate system of fixed and movable PV panels; panels directly above the building are fixed, while those projecting over the building sides are movable to optimize solar incidence angles during the day, as well as provide needed shading and shuttering. The building was an entry in the Solar Decathlon Europe 2014 *[UBTB14]*. The Decathlon entry was as well documented as one can expect at the end of design. Yet, some of the critical modeling data are missing in the documentation; other critical data contradict each other.

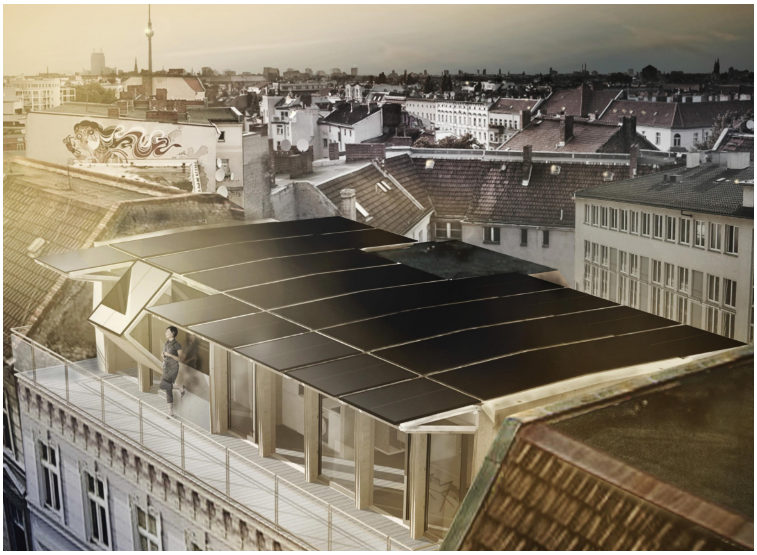


Fig. 7.46: *Computer generated perspective image of the Rooftop building placed in Berlin city-scape (source* [GFMAE14]*).*

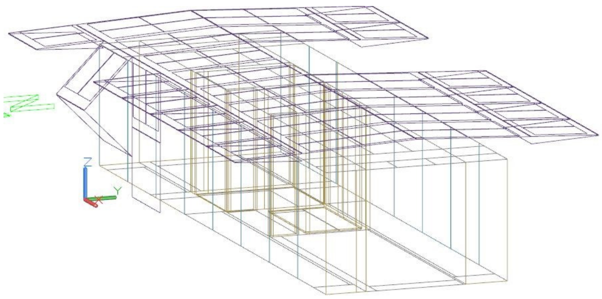


Fig. 7.47: *Orthogonal projection of the Rooftop building's geometry model in active state as imported into EnergyPlus, reflecting PV and shutter positioning shown in* Fig. 7.46.

Fig. 7.46 is a computer generated "artist's view" of the building, copied from the project documentation. Fig. 7.47 shows the building geometry model, generated using best available information from the same project documentation that mimics the positioning of movable PV panels and shutter shown in Fig. 7.46. The modeler had to improvise and guess some of the missing information and, in the case of conflicting data, to subjectively decide which data to use. Visual comparison of Fig. 7.46 and Fig. 7.47 shows what was expected (Fig. 7.46) and want appears to have been built (Fig. 7.47). Finally the resulting Modelica model is shown in Fig. 7.48. Fortunately, the Rooftop Building is being reconstructed and instrumented at the UDK- Berlin campus. This is providing a unique opportunity to record true "as-built" data, re-model the building geometry and see how closely simulation results can match energy performance measurements in the building.

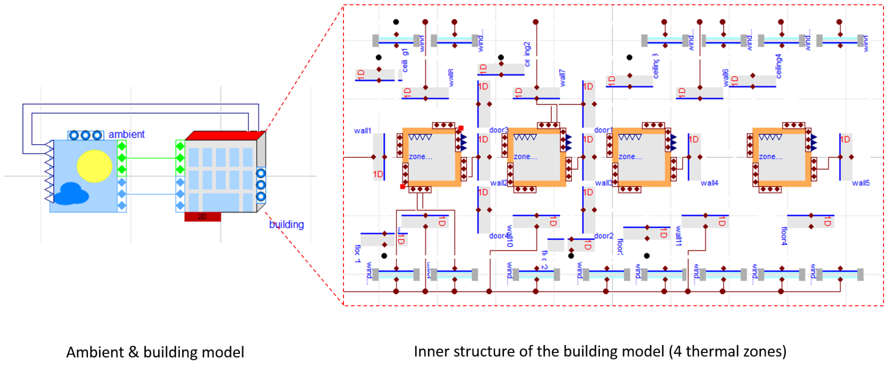


Fig. 7.48: Screenshot of the Modelica model of the roof top building.

7.6 Limitations of the Framework and Future Work

This subchapter describes the limitations of the framework focusing on generic limitations rather then today's technical limitations which are improving over time. Furthermore, we discuss future work that is enabled by this development.

7.6.1 Limitations

The basic limitations of this framework or tool chain lies in the fact that various data models and tools are used. The tool chain is restricted to the functionality of each tool and data model. For example, HVAC components that are not available in Modelica libraries reduce the list of supported components. The same is true if any other tool or data model is not supporting a particular HVAC object. The distributed development in small tools also makes changes that influence multiple tools time consuming and sometimes difficult. E.g., changing a parameter in SimModel does create the need for updated mapping rules from IFC to SimModel as well as from SimModel to Modelica. In that context, the implementation of control objects turned out to be quite cumbersome since no tools exist that define controls and have a relevant IFC export and/or import functionality. In this area we developed a so-called workaround and attach relevant properties to the object that is controlled. This was necessary since controller objects are not available in the MEP CAD applications. One of the biggest challenges in this subtask was the flexible nature of Modelica and its libraries as well as their continuous development. While significant progress towards a standardized Modelica library development has been made through the Annex 60 base library, there still exists potential to standardize more aspects of the Modelica libraries. From a data model and mapping perspective additional standardization and library creator agreements would simplify future mapping and conversion efforts. E.g., currently connector names of components can be any arbitrary string. Yet another limitation is the quality of BIMs. While model checking tools exist and are further developed, the understanding of providing data models of high quality is not present in practice yet. Another aspect is that the generated Modelica models are just a starting point and need further fine tuning at the instance level of the Modelica models.

7.6.2 Future Work

While the developed tool chain is limited in scope and is a first prototype, the following developments could be based on it:

- Further specification of the MVD to include further relevant HVAC components

- Officially certified MVD that is implemented by various software applications.
- Extensions of the IFC data model for missing concepts that we identified within this project (see Section 7.3.5.3)
- Further development of the prototype tools to cover more HVAC components.
- Further development of the Modelica libraries to include missing components.

In order to allow for international collaboration and further developments, all in this project developed code and models related to the transformation process are made available as open source. The Annex 60 framework therefore provides a solid and well-documented base for further developments.

Chapter 8

Activity 1.4: Workflow Automation Tools

8.1 Introduction and Motivation

Today, workflow automation tools play an important role in the scope of building and community energy performance simulation as models, tools and engineering tasks become incleasingly complex. Running a script in order to start tools in batch mode and to preprocess variables is common practize in simulation. This is especially true if, for example, multiple domains are coupled, different scenarios shall be investigated, if input parameters are frequently varied, if multiple times a similar task shall be performed or if a conversion of input parameters or data formats becomes necessary. Another issue is that computational capabilities are increasing steadily following Moore's law. Hence, dealing with simulation models becomes more complex for both, users and developers. Challenges for practitioners in the field of building energy simulations therefore include the handling of bigger amounts of input/output data (many or huge data files) *[SN14]*. Furthermore, a greater number of simulations will need to be run in order to perform sensitivity or uncertainty analysis of single or multiple parameters *[HH11][BJH10]*. Analysis of the resulting data by means of visualization or statistical figures *[RK11]* as well as mathematical operations on the inputs and outputs will increase in importance. The various data formats involved in simulation need to be converted and adjusted in order to allow comparisons and communication between models.

These tasks do not require scientific expertise or any specific engineering skills, rather they are often very time consuming and repeating. If performed multiple times manually, not only valuable time will be wasted, it is also conceived to be error prone and tedious. As a consequence, building simulation researchers should be able to use scripting languages or other automation environments to gain more efficiency while ensuring the quality of their results. According to *[SN14]* the iterative nature of the building simulation workflow can be leveraged by every step in the process which is automated using Python scripting. Fig. 8.1 is exemplifying the whole simulation process in this context starting from Modelica models based on the Annex 60 Library.

The importance of workflow automation is also identfied by *[SN14]* when working with big data. It is recommended to write scripts that automate large batch processes for simulation tasks. Thus, building simulation researchers and practitioners benefit from basic parsing and scripting knowledge.

This chapter provides an overview of associacted Python packages and recommended third party packages. Several Python tools and packages are available for developers and for model users. Those Python tools and packages enhance the workflows for developing and using Modelica and FMI-based models.

They will assist developers in unit tests and checking libraries with conformance to coding guidelines. Furthermore, users will be able to pre-process, run and post-process batches of simulations, such as for parametric studies or for uncertainty propagation, including post-processing of different data formats created by FMUs. It will additionally be possible to integrate Modelica or FMI based models with optimization packages for design optimization and Model Predictive Control.

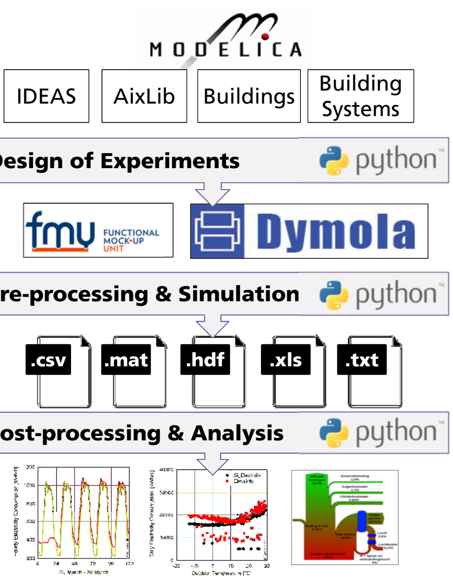


Fig. 8.1: *Schematic of typical building simulation workflow to demonstrate use cases for Python based process automation*

8.2 Tools and Methodology

This section provides an overview of existing tools and Python packages for automation of building simulation workflows. In the Annex, the necessary capabilities of such tools have been identified and compiled into a common working document. After discussions within the expert groups, a set of mature and utile Python packages was selected, under consideration of their capabilities, license type and implementation language. From the analysis of existing software and features for building simulation automation, a structured list of use cases was derived which reflects the high level requirements for workflow automation tools.

8.2.1 Functional Requirements

The main focus of automation tools lies in the possibility to execute simulations from a script. This is a precondition to efficiently apply pre- and postprocessing functions. These functions are then able to generate the desired efficiency gain for repeatedly used operations. The following paragraph concludes the desired functional requirements of automation tools.

8.2.1.1 Running Simulations

In order to provide a broad foundation for working with simulation inputs and results, the tools should help the user to run different simulation engines like Dymola, OpenModelica, JModelica, FMUs, etc. Simulations of similar types should be processed in parallel. Execution of simulations, especially parametric studies should be done in automated batch setups. An efficient allocation of ressources will be a valuable advantage in this regard. Furthermore, also interaction with external programs in order to couple different simulators is desired. This can also be realized by supporting co-simulation interfaces like the FMI standard.

8.2.1.2 Preprocessing Operations

The initial treatment of simulation input data can be relevant when using changing data sources on the same simulation model, investigating parameter variations or preparing external data for the simulation. In order to support these processes, commonly used scripts should support the transformation of input data (e.g. from BIM data). Also, automated preparation of input data from a database (e.g. product catalogues) is to be considered. This can also be realized based on templates for the simulation model. Parameters for the model can therefore be set instantly. Furthermore, simulation settings, like paths for input data or result files as well as specific simulation parameters (start/end time, verbosity, logging) and solver settings (type, tolerance, integration method) should be handled. Besides the instantiation of parameters, functionality for importing dynamic inputs serving as boundary conditions to the simulation, like specific profiles as data from .xls or .csv files needs to be included. Common data-readers in the Annex60 or other Modelica libraries as well as standardized weather data (TRY, TMY) are only a few valuable input sources to be considered.

8.2.1.3 Postprocessing Operations

Increasing amounts of computational resources, modeling capabilities and memory has led to enormous amounts of data that can be generated from simulations. Automated processing of result files has therefore become a time intensive task. In order to reduce this effort, postprocessing of a set of similar result files, e.g. from a parametric study, must be possible. Furthermore, filtering of result data based on different criteria helps to reduce the data capacity that needs to be handled. However, a suitable way for storing data in larger files or database systems is still necessary, especially when different data sources are to be merged. In addition, computation of typical result criteria like thermal comfort or annual energy consumption needs to be supported.

8.2.1.4 Data Analysis

Gathering parameter sets or patterns of simulation data is often only a first postprocessing step. Various analysis methods, e.g. statistical functions like computing the mean, maximum, minimum or standard deviation of a result variable, can be very useful when analyzing the computed values. More advanced statistical methods involve the computation of a moving average, auto-correlation-functions and ARMA-models. Postprocessing scripts should also be able to compare results with a baseline design option and perform regression analysis such as a linear least square method, trend estimation or curve fitting. The computation of the R^2 goodness of fit enables fast comparisons between two data series. The ASHRAE Guideline 14 [ASH02] states further values for result interpretation. Besides statistical analysis, frequently used base functionalities for timeseries, like integration, arithmetic or logical operations, as well as frequency analysis, event counting or discarding irrelevent data from a timeseries, can help a user to interpretate and work with simulated data.

8.2.1.5 Data Visualization

Graphical representation of simulation results is a valuable measure to present, compare and analize simulations. Post-processing scripts should therefore inherit typically used options including

- multiple graph plots,
- plot subsets of simulations for comparison,
- various plotting routines for time series, parameter values, box plots, bar charts, carpet plots, scatter plots, histograms, Sankey diagrams, Bode plots or Nyquist plots.

8.2.1.6 Parametrization

Once provided with the opportunity to automatically instantiate and initialize a simulation, methods to execute studies on a single simulation model are necessary. This includes the automated parametrization of models in order to optimize the design. During such a study, often a vast amount of input data is generated and needs to be organized. Sampling methods (e.g. Latin Hypercube Sampling) help to design the experiment (DoE) in order to decrease the data space. A limited number of input parameters are thereby sampled through a probability density function within the design space. The results allow to perform sensitivity analysis and compute uncertainty propagations of the results.

8.2.1.7 Data Conversion

Multiple simulation environments and various data sources lead to the requirement of converting data between different formats. Useful script tools should therefore provide functionalities to handle the container format HDF5, .mos files, tables from .csv and .xls files, .mat files and figures, diagrams as images in .png or .jpg format.

8.2.1.8 Verification

Simulations often serve as supporting measures to better understand a systems behaviour. However, previous verification of such models is important to prevent errors. Automated model check algorithms are therefore a valuable tool to increase productivity. Their functionality should include the verification of model correctness and completeness as well as compliance and model compatibility checking. Furthermore, unit testing can prevent time intensive and cumbersome search for smaller errors in the model. Erroneous input data can be eliminated with fault detection algorithms based on an isolated view of the data itself or in conjunction with test runs of the model.

8.2.1.9 Optimization

After previous simulation generation and analysis, further algorithms can help to support optimization of a system. These incorporate

- derivative-free optimization that evaluates a model for different parameter values; these can be heuristic such as genetic algorithms or particle swarm optimization, or deterministic such as generalized pattern search methods *[WW04][PW06]*,

- gradient based parameter optimization that either numerically approximate derivatives, or that use computer algebra to obtain analytic expressions for derivatives *[AGT09]*.

8.2.2 Associated Packages

The requirements for Activity 1.4 were mapped to existing tools and packages and the missing functionalities and capabilities needed for building simulation tasks (i.e. validation and demonstration) within the scope of Annex 60 were identified. In order to illustrate the advantages arising from these packages, the following ones are highlighted in the example applications.

- BuildingsPy (http://simulationresearch.lbl.gov/modelica/buildingspy/) from LBNL allows for running Modelica simulations in Dymola and inherits functionality to process result files. It furthermore enables unit tests and the refactoring of Modelia libraries.
- awesim from KU Leuven (https://github.com/saroele/awesim) provides functionality to pre- and postprocess Modelica models. It is possible to compile models, set parameters and solver options. Simulations can be run in parallel. Plots based on the result files can be created and filtering of results based on filenames and parameter sets is possible.
- ModelicaRes (http://kdavies4.github.com/ModelicaRes/) is a package to generate simulation scripts for Dymola. Result data can be loaded, analyzed and plotted. The package interacts with the popular pandas package for Python, allowing for numerous statistical analysis.

8.2.3 Third-Party Packages

Other third-party Python packages exist which provide additional capabilities which are useful for the purposes of building performance simulation such as

- DyMat (http://www.j-raedler.de/projects/dymat/), which is a package for handling Dymola's or OpenModelica's .mat output files. Browsing for variables and exporting their content to various formats is supported.

- PyFMI (https://pypi.python.org/pypi/PyFMI) is a framework to incorporate the Functional Mockup Interface (FMI) in the Python environment. Functional Mockup Units can be loaded, instantiated, simulated and modified or queried using the provided functions by the standard.
- matplotlib (http://matplotlib.org/) serves to plot and visualize data of various kind. Several graph types can be generated and saved in different formats like .png or .jpg.
- scipy (http://www.scipy.org/) is a package for scientific computing (based on numpy). It includes functionalities imitating Matlab-like treatment of data in the Python environment.
- pandas (http://pandas.pydata.org) is a package for data and time series analysis (similar to the statistics tool ‘R’). Huge amounts of data can be efficiently stored, queried and analyzed in data frames.
- StatsModels (http://statsmodels.sourceforge.net/) includes statistical methods (similar to ‘R’) that allow for statistical tests, intensive data exploration and the generation of statistical models in Python.
- pyTables (http://www.pytables.org) serves to manage hierarchical datasets. Huge amounts of data can be handled efficiently and easily. The package is based on the HDF5 library.
- pysimulator (https://github.com/PySimulator (LGPL)) is a simulation and analysis environment for Functional Mockup Units and Modelica models in Python.

8.3 Examples of Application

The following examples cover some of the previously stated requirements through self-explaining Python code. The IPython Notebook was chosen to serve as the implementation platform of these examples. A Notebook is an interactive Python computing environment that enables users to author workflow automation scripts that include executable code, interactive graphs and plots, textual comments, images and governing equations.

While executing a Notebook it generates output from the executed Pyhton kernel that is running in the background. The multi-media output is fully embedded in the notebook giving a complete and comprehensible record of a computation.

The sequential reading, executing and verifying of computation results in a step-by-step walkthrough facilitates productive and re-usable code for either, model developers and users.

These Notebooks can as well be published as interactive web application (see http://jupyter.readthedocs.io). Notebook documents available from a public URL (e.g. on GitHub) can be shared via `nbviewer`. A web service ist loading it from the URL and displays the Notebook as static web page to be shared with others.

The Notebooks created for the Annex 60 project start with a short description of their purpose, the used packages and the used simulation model as far as it is relevant to fully understand the Notebook. The simulation models are identical or slightly adapted examples from the Dymola libraries involved in the project scope. Within the Notebooks, code cells are bundled to execute specific tasks. Short text descriptions before and comments in the code help to get a clear picture of these fragments. Some of the cells produce output like graphs, dataframes or pure text. These are shown directly beneath the code cells. The requirements for the notebooks to work are the following:

- all used Python packages must be installed. These include the following: numpy, os, BuildingsPy, pandas, pyDOE, ModelicaRes, pyFMI, csv and datetime;
- the FMI library must be installed and set as a system environment variable in order for pyFMI to be able to call FMI functions;
- the path to the executable of the used Dymola version needs to be included in the path system environment variables.

8.3.1 Single Simulation of a Building using BuildingsPy

Running simulations from script can proof beneficial when applying identical pre- and postprocessing patterns to a single simulation. Especially the effort for statistical and graphical evaluation of the results can be significantly reduced. In this IPython Notebook, a framework including relevant function calls to adress these issues for a single simulation in Dymola is provided. In particular, the Python library BuildingsPy is used, supplemented with elements from pandas and NumPy. The model serving as an example in this case is a slightly modified version of the Annex60.Fluid.Examples.SimpleHouse. It consists of a simplified building envelope, a ventilation system including heat recovery and a heating loop with a radiator. In order to start the process, some basic information such as model name, its location as well as the folder for the result files need to be defined. In order for Dymola to compute the model autonomously, a path to relevant packages is defined.

```
import os

model = "SimpleHouse"
resultFile = "SimpleHouse"

model_dir = os.getcwd()+"\Resources\Examples\SimpleHouse"
resultFile_dir = os.getcwd()+"\Resources\Results_Nb1"
# set directory to dymola libraries
libs_dir = os.getcwd()+"\Resources"

os.chdir(model_dir)
```

8.3.1.1 Preprocessing

Prior to the simulation run, various settings can be customized. These involve simple parameters like start and stop time of the simulation as well as the desired time step. In addition to that, preprocessing statements and solver settings are given. Finally the simulation is started.

```
from buildingspy.simulate.Simulator import Simulator

t_start = 0
t_end = 86400*1
h_step = 60

s = Simulator(model, "dymola", packagePath=libs_dir)
s.setStartTime(t_start)
s.setStopTime(t_end)
n = (t_end-t_start)/h_step
s.setNumberOfIntervals(n)
# kills the process if it does not finish after 600 seconds
```

```
s.setTimeOut(600)
s.setSolver('dassl')
s.addPreProcessingStatement("Evaluate:=true;")
s.printModelAndTime()
s.setResultFile(resultFile)
s.setOutputDirectory(resultFile_dir)
s.simulate()

Model name       = SimpleHouse
Output directory = .
Time             = Thu Jun 02 12:08:13 2016
```

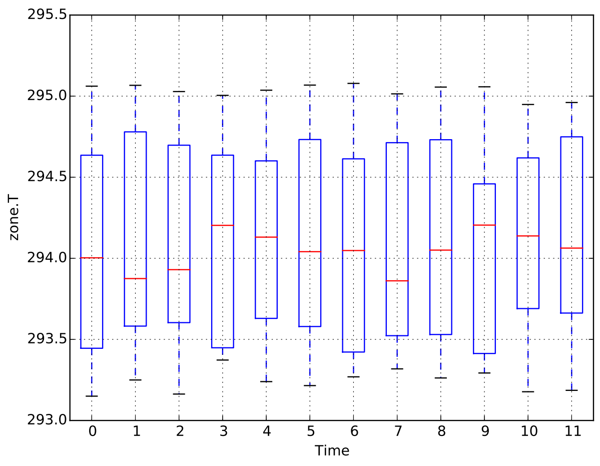
8.3.1.2 Postprocessing

A reader object from BuildingsPy serves to retrieve the values of defined result variables and saves them into the IPython workspace as a pandas dataframe. Since the Dymola simulation yields an irregular time grid, an interpolation is performed to match the result values to the above defined timestep. Finally, the results are exported to a .csv file to ensure the possibility of further, individual analysis.

```
from buildingspy.io.outputfile import Reader
from buildingspy.io.postprocess import Plotter
import pandas as pd
import numpy as np

# variable names to be extracted from the result file are
# determined
variables = ['zone.T','Q_heating']

r = Reader(resultFile_dir+"\"+resultFile, "dymola")
# the original time grid is defined
tSup = np.linspace(t_start, t_end, h_step)
ResultValues = pd.DataFrame(columns=['time']+variables)
for var in variables:
    (time,temp) = r.values(var)
    # results are reduced to the original time grid
    ResultValues[var] = Plotter.interpolate(tSup,
                                            time,
                                            temp)
ResultValues['time'] = tSup
ResultValues.to_csv(resultFile_dir+"\"+resultFile+".csv")
print ResultValues.head()

          time      zone.T    Q_heating
0     0.000000  293.149994     0.000000
1  1464.406780  293.889343  1380.513502
2  2928.813559  294.936695  1840.707414
3  4393.220339  295.025860   702.443914
4  5857.627119  294.288450   241.524324
```

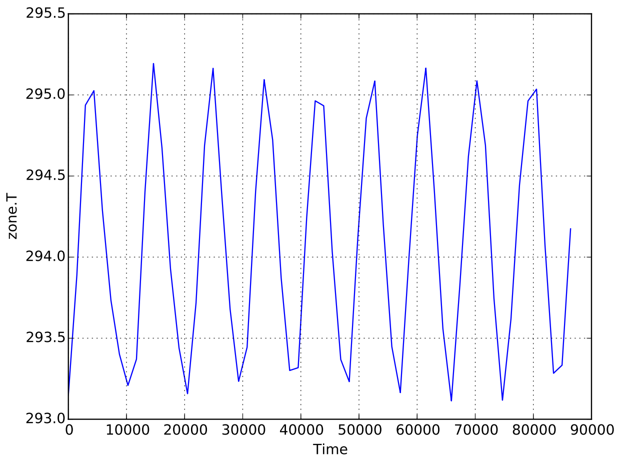
The above defined variable outputs are visualized in two ways using plotting functions from BuildingsPy. The first plot is a boxplot. To execute the function, a time interval needs to be provided over which the boxplot is to be computed. Furthermore, the number of evaluations of the boxplot must be stated. The result is a figure consisting of various boxplots, each representing the given time intervall in sequential order. A second graph produces a simple line plot with the result values represented over the simulation time. Formatting functions provided by the matplotlib can be used within the BuildingsPy package to customize the appearance of the graphs.

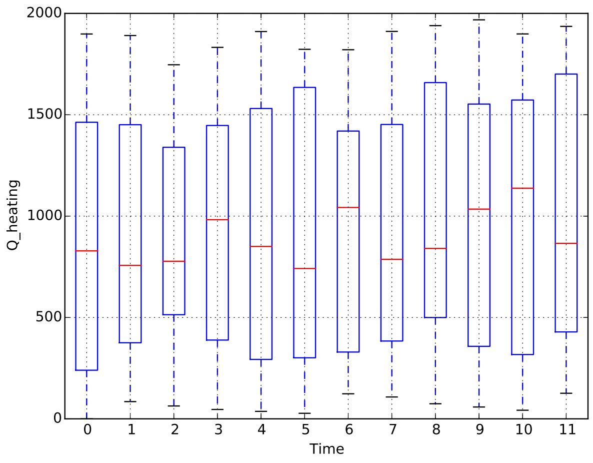
```
for var in variables:

    # 12 statistical evaluations for 600s time intervals.
    # In this case, one hour is divided into 5 minute
    # intervals. For each interval the statistics
    # to create a boxplot are computed.
    plt=Plotter.boxplot(t=list(ResultValues['time']),
                        y=list(ResultValues[var]),
                        increment=600,
                        nIncrement=12)
    plt.xlabel('Time')
    plt.ylabel(var)
    plt.grid()
    plt.savefig(resultFile_dir+"\"+var+"_boxplot.png",
                dpi=300)
    plt.show()

    # a line plot over the simulation time
    plt.plot(list(ResultValues['time']),
             list(ResultValues[var]))
    plt.xlabel('Time')
    plt.ylabel(var)
    plt.grid()
    plt.savefig(resultFile_dir+"\"+var+"_lineplot.png",
                dpi=300)
    plt.show()
```

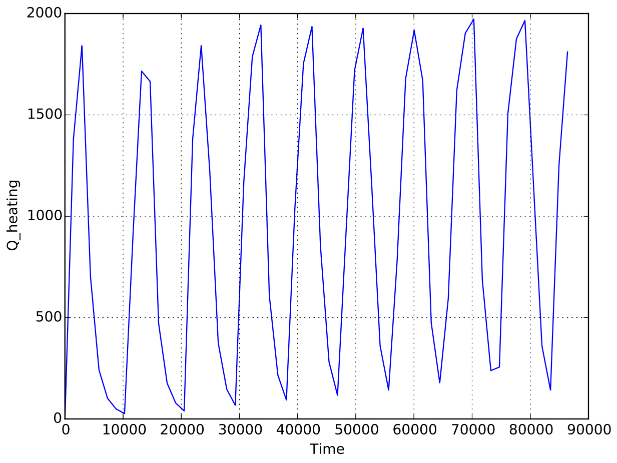


The NumPy package provides several useful functions for simple statistics. The following cell computes various measures of descriptional statistics and saves them to a pandas dataframe. The above applied command can be used again to export the dataframe effortlessly to a .csv file.

```
ResultStatistics=pd.DataFrame(columns=variables)
for var in variables:
    ResultStatistics.loc['Minimum',var] =
        np.min(ResultValues[var])
    ResultStatistics.loc['Maximum',var] =
```


zone.T
Time
295.5
295.0
294.5
294.0
293.5
293.0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000


Q_heating
Time
2000
1500
1000
500
0
0
1
2
3
4
5
6
7
8
9
10
11



```
        np.max(ResultValues[var])
    ResultStatistics.loc['Mean',var] =
        np.mean(ResultValues[var])
    ResultStatistics.loc['Median',var] =
        np.median(ResultValues[var])
    ResultStatistics.loc['25th percentile',var] =
        np.percentile(ResultValues[var], 25)
    ResultStatistics.loc['50th percentile',var] =
        np.percentile(ResultValues[var], 50)
    ResultStatistics.loc['Standard Deviation',var] =
        np.std(ResultValues[var])
    ResultStatistics.loc['Variance',var] =
        np.var(ResultValues[var])

ResultStatistics.to_csv(resultFile_dir+"\"+resultFile+
        "_statistics.csv")
print ResultStatistics

                       zone.T  Q_heating
Minimum              293.1131          0
Maximum              295.1935   1972.613
Mean                 294.0912   941.9056
```

```
Median                 294.052  881.5341
25th percentile       293.4302   232.922
50th percentile        294.052  881.5341
Standard Deviation   0.6809249  708.8307
Variance             0.4636587  502440.9
```

Some useful BuildingsPy functions can also help to quickly assess certain performance indicators. In this case, an integral function is applied to the required heating power. As a result, the cumulative heating energy over the simulated time period can be evaluated.

```
print "Total heating energy sums up to" ,
       r.integral('Q_heating')

Total heating energy sums up to 81547626.5705
```

8.3.2 Parametric Study using BuildingsPy and ModelicaRes

Single simulations can assist practitioners to better understand and assess a system. However, knowing the influence of parameter variation on the system's behaviour is crucial for system optimization and provides valuable additional insights. The following script serves to automatically run such a parameter study in Dymola based on a Latin Hypercube Sampling (LHS) of chosen variables. The package pyDOE is used to generate the sample for the study. The setup and start of the simulation is executed with BuildingsPy. ModelicaRes is used to label and compare the simulation runs graphically. The modified version of the Annex60.Fluid.Examples.SimpleHouse example is again taken to demonstrate the procedure. Parameters under investigation are the outside wall area of the building, its mean U-value as well as the nominal heating power. The study is performed with the purpose to find suitable combinations of the three parameters in oder to ensure comfortable temperatures in the building. In a first step, the usual settings for model name, directories etc. must be defined. Again, the directory containing relevant Modelica libraries for the model needs to be provided.

```
import os

model = "SimpleHouse"
model_dir = os.getcwd()+"\Resources\Examples\SimpleHouse"
result_dir = os.getcwd()+"\Resources\Results_Nb2"
# set directory to dymola libraries
libs_dir = os.getcwd()+"\Resources"
os.chdir(model_dir)
```

8.3.2.1 Parameter Sampling

In order to get a set of simulations, each based on different parameter values, the variables in question need to be determined. A minimum and maximum value for each parameter ensures that the samples are amongst the defined range. These values can represent certain constraints or requirements that might occur during planning. To create the sample, the Latin Hypercube Sampling method, implemented in the pyDOE package, is applied. It creates a multidimensional set of parameter values for the batch simulation. Besides the minimum and maximum values of the variables, the number of total simulation runs must be determined. The output of the following code cell shows the created samples in a pandas dataframe.

```
from pyDOE import *
import numpy as np
import pandas as pd

# Varying parameters in the study
varList = ['A_wall', 'U_wall', 'Q_RadNominal']
MinMax = [(200,300), (0.4,0.8), (1000,3000)]
# Number of simulation runs
n_simulations = 5

# generation of samples with latin hypercube sampling (lhs)
design = lhs(len(varList), samples=n_simulations)
diff = []
mins = []
# the normalized design values are mapped to the parameter
# ranges
for pair in MinMax:
    diff.append(pair[1]-pair[0])
    mins.append(pair[0])

Sample = pd.DataFrame(design*np.array(diff)+np.array(mins),
         columns=varList)
print Sample

       A_wall    U_wall  Q_RadNominal
0  270.708852  0.730839   2151.677135
1  289.948056  0.700511   2313.956082
2  252.777500  0.548079   1369.952720
3  220.802008  0.587404   2900.892276
4  213.898259  0.437846   1661.255155
```

8.3.2.2 Batch Simulation

To run the created parameter sets in batch mode, the BuildingsPy package is used. It creates a folder for every run in the result directory. Naming follows the enumeration corresponding to the parameter set number. After defining start and end time as well as the timestep, simulations with the updated parameter sets are executed in a loop and saved to the corresponding folders.

```
from buildingspy.simulate.Simulator import Simulator

t_start = 0
t_end = 86400*15
h_step = 60

for i in range(0, n_simulations):
    parameters = {}
    for var in varList:
        parameters[var] = Sample[var][i]
    s = Simulator(model, 'dymola', packagePath=libs_dir)
    s.setOutputDirectory(result_dir+'\run'+str(i))
    s.addParameters(parameters)
    s.setSolver('dassl')
    s.addPreProcessingStatement("Evaluate:=true;")
    s.setStartTime(t_start)
    s.setStopTime(t_end)
    n = (t_end-t_start)/h_step
    s.setNumberOfIntervals(n)
    s.simulate()
    del s, parameters
```

8.3.2.3 Batch Postprocessing

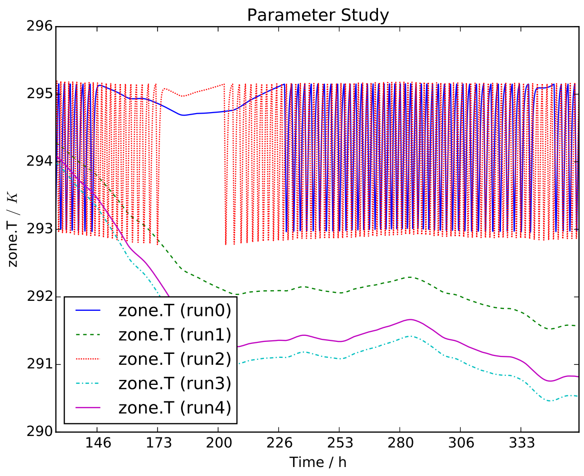
In order to retrieve the results from the individual folders, ModelicaRes is used. Every simulation run receives a label. This label can later on be used to differentiate between the result values e.g. in a graph.

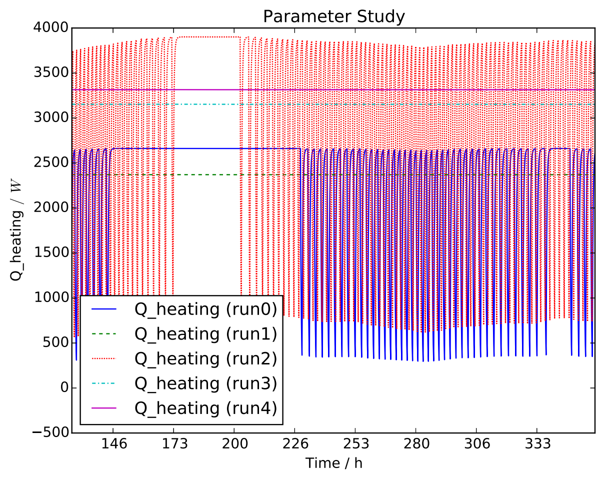
```
from modelicares import SimResList

sims = SimResList(result_dir+"//*//*.mat")

# labelling
i = 0
for sim in sims:
    label = 'run'+str(i)
    i+=1
    sim.label = label
```

In the following cell, defined outputs of the simulation are compared. ModelicaRes therefore collects the values of each result variable from each simulation run and plots them in one graph. Start and end time of the graph can be provided within the simulation period. The graphs are finally saved to the results directory. Possibilities to customize the graph representation can again be found in the matplotlib documentation.

```
# define result variables for comparison.
variables = ['zone.T', 'Q_heating']

t_start_plot = 86400*5
```

```
t_end_plot = 86400*15

for var in variables:
    fig_temp = sims.plot(ynames1=var, ylabel1=var,
                title="Parameter Study")
    fig = fig_temp[0]
    fig.set_xbound(lower=t_start_plot, upper=t_end_plot)
    labels = fig.get_xticklabels()
    labels_h = []
    for i in range(0, len(labels)):
        labels_h.append((t_start_plot+(t_end_plot-
        t_start_plot)/(len(labels)-1)*i)/3600)
    fig.set_xlabel('Time / h')
    fig.set_xticklabels(labels_h)
    fig.figure.savefig(result_dir+"\"+var+".png", dpi=300)
```





8.3.3 Parametric Study of a Boiler FMU from Dymola using PyFMI

In certain cases, planners might interact by exchanging simulation models as FMUs for immediate testing of models from other disciplines on design changes. This tutorial on hand shows how to load and run a parametric study of a single FMU exported from Dymola using PyFMI. In this case, the boiler simulation model from the AixLib library is considered as an example. It is a single thermal zone heated by a radiator. The valve between the boiler and the radiator is controlled by a proportional controller. It is the intention to observe the influence of the temperature set point on the total energy consumption of the boiler over a year. To do so, the model is run a few times with different values for the set point.

In order to extract the FMU, this model has been previously compiled in the Dymola environment in the Windows operating system. It is available as the AixLib_Boiler_DymolaWin.fmu file. The PyFMI python package will be used to call the solver and run the simulations. Since the FMU has been exported with Dymola, a Dymola license is needed to run this as well as the following example. Just like in the previous examples, folders and model name must be defined to start with the tutorial.

8.3.3.1 Loading an FMU

```
from pyfmi import load_fmu
import os

model = 'AixLib_Boiler_DymolaWin.fmu'
model_dir = os.curdir + "//Resources//Examples//Boiler"
result_dir = os.getcwd()+"//Resources//Results_Nb3"
boiler = load_fmu(fmu=model, path=model_dir)
os.chdir(model_dir)
```

8.3.3.2 Execute Parametric Study in Loop

In the following, parameter values to be applied in the study need to be defined. A loop executes the simulation several times providing the current parameter value as an input signal to the FMU. For each simulation run, the cumulative heating energy is computed in an integral function and saved to a growing list.

```
from scipy.integrate import trapz
import numpy as np

t_start = 0
t_end = 3.1536e7

T = [291.15, 292.15, 293.15, 294.15, 295.15, 296.15]
Q = []

for TAirSet in T:

    boiler.reset()

    boiler.set('setTemp.k', TAirSet)

    try:
        res = boiler.simulate(start_time=t_start,
                              final_time=t_end)
    except:
        print "One of the simulation cases has failed."
    # The heat flow from the boiler is recovered from the
    # simulation results
    res_time  = res['time']                          # s
    res_Q = res['boiler.heatDemand.Q_flow_out'] # W

    Q.append(trapz(res_Q, res_time) / 3600 / 1000)
```

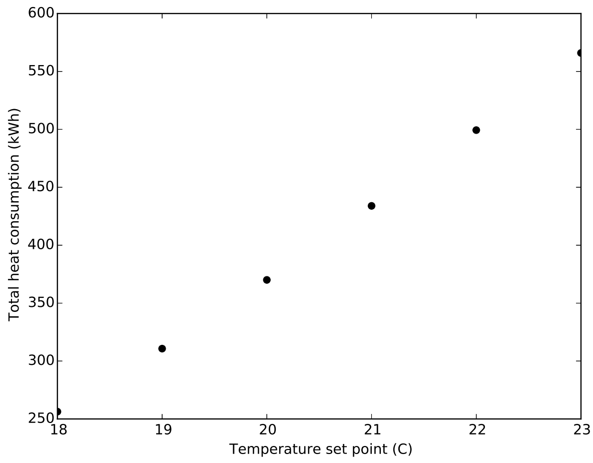
8.3.3.3 Postprocessing

After finishing all simulations, a graphical postprocessing can be started. In this case, a simple point plot is created to demonstrate the dependency of heating energy demand on the temperature setpoint.

```
import matplotlib.pyplot as plt

plt.figure()
plt.plot([x-273.15 for x in T], Q, 'ok')
plt.xlabel('Temperature set point (C)')
plt.ylabel('Total heat consumption (kWh)')
plt.savefig(result_dir+'\Total heat consumption.png',
            dpi=300)
```

8.3.4 Import Data to Co-Simulation FMU using PyFMI and Pandas

Input data for simulations can often come from measurements, other simulation results or any external data sources. In order to consider this data in FMU simulations, co-simulation FMUs can be used. Co-simulations offer a step-wise execution of the simulation. After every time step the simulation can be stopped and input values are updated. The package pyFMI offers the necessary function calls to fulfil this purpose. In the following, the SimpleHouse example from above serves as the FMU. In contrary to the original model, weather data is missing in the FMU. Instead, temperature and solar irradiation



are read from a .csv file. To start with the notebook the FMU name and its directory must be defined.

```python
import os
from pyfmi import load_fmu

model = "SimpleHouse.fmu"
model_dir = os.getcwd()+"\Resources\Examples\SimpleHouse"

FMU = load_fmu(fmu=model, path=model_dir)
```

Input and output variables of the FMU can be determined with the following function calls. Later in this script, the found variable names serve as connector to the exchanged weather data.

```python
input_vars = FMU.get_input_list()
print input_vars.keys()

output_vars = FMU.get_output_list()
print output_vars.keys()

['T_amb', 'Irr_HGloHor']
```

```
['T_air']
```

8.3.4.1 Import Tabular Data

A .csv file is imported to IPython in the following cell. The column called “time [s]” is used as a time index. The start date for the time series is chosen in the variable “date”. These formatting measures are necessary in order to be able to map a possibly different time grid of the input data to the time grid of the simulation.

```
import csv
import numpy as np
import pandas as pd
from datetime import datetime

# Definition of Input .csv file
weatherFile = "Weather.csv"
weatherFile_dir = model_dir

# Data is formated with a time index
date = datetime(2015,1,1)
InputFile = pd.read_csv(weatherFile_dir+'\'+weatherFile)
InputFile.set_index(date+pd.to_timedelta
        (InputFile['time [s]'], unit='s'), inplace=True)

                     time [h]  time [min]  time [s]  T_amb␣
↪[K]  T_amb [C]  \
time [s]
2015-01-01 00:00:00         0           0         0     278.
↪15          5
2015-01-01 01:00:00         1          60      3600     277.
↪15          4
2015-01-01 02:00:00         2         120      7200     276.
↪15          3
2015-01-01 03:00:00         3         180     10800     275.
↪15          2
2015-01-01 04:00:00         4         240     14400     275.
↪15          2
```

```
2015-01-01 05:00:00          5         300      18000      275.
↪15          2
2015-01-01 06:00:00          6         360      21600      274.
↪15          1
2015-01-01 07:00:00          7         420      25200      274.
↪15          1
2015-01-01 08:00:00          8         480      28800      274.
↪15          1
2015-01-01 09:00:00          9         540      32400      274.
↪15          1

                     Irr_HGloHor [W/m2]
time [s]
2015-01-01 00:00:00                   0
2015-01-01 01:00:00                   0
2015-01-01 02:00:00                   0
2015-01-01 03:00:00                   0
2015-01-01 04:00:00                   0
2015-01-01 05:00:00                   0
2015-01-01 06:00:00                 150
2015-01-01 07:00:00                 250
2015-01-01 08:00:00                 300
2015-01-01 09:00:00                 400
```

After definition of the simulation time grid, the input data can be edited to the given time base. Therefore, the relevant columns of the input file are chosen, resampled and interpolated using pandas. The time indexing within this package is especially practical for this purpose. After preparation of the input data, the FMU co-simulation algorithm can be initialized.

```python
import pandas as pd
from pandas.tseries.offsets import Second

SimInputs = ['T_amb [K]','Irr_HGloHor [W/m2]']

# simulation time grid
t_start = 0
t_end = 86400*1
h_step = 60
```

```
t_s = list(np.arange(t_start, t_end+h_step, h_step))

InputVars = pd.DataFrame()
for var in SimInputs:
    InputVars[var] = pd.TimeSeries(data=InputFile[var],
      index=date+pd.to_timedelta(InputFile['time [s]'],
      unit='s')).resample(Second(h_step), how='mean',
      label='right', closed='right')

if InputVars.isnull().values.any():
    InputVars.interpolate(method='time', inplace=True)

print InputVars.head(n=10)

                      T_amb [K]  Irr_HGloHor [W/m2]
time [s]
2015-01-01 00:00:00  278.150000                   0
2015-01-01 00:01:00  278.133333                   0
2015-01-01 00:02:00  278.116667                   0
2015-01-01 00:03:00  278.100000                   0
2015-01-01 00:04:00  278.083333                   0
2015-01-01 00:05:00  278.066667                   0
2015-01-01 00:06:00  278.050000                   0
2015-01-01 00:07:00  278.033333                   0
2015-01-01 00:08:00  278.016667                   0
2015-01-01 00:09:00  278.000000                   0
```

8.3.4.2 FMU Co-Simulation Algorithm

The co-simulation process requires the exchange of variable values after every time step. In this case, the input variables of the FMU are fed by the imported .csv data. The entire procedure starts with the FMU setup and its initialization. In a loop running through the simulation time grid the following steps are performed. First, the input variables of the FMU are provided with their input values at the corresponding time step. Secondly, the simulation of the current time step is performed. The desired variable results at the computed time step can be retrieved to a pandas dataframe where they are available for postprocessing.

```python
# variable names to retrieve result values from FMU after
# each time step
variables = ['zone.T','Q_heating']

FMU.setup_experiment(start_time=t_start, stop_time=t_end)
FMU.initialize()

ResultValues = pd.DataFrame(columns=variables,
                            index=date +
                              pd.to_timedelta(t_s, unit='s'))

# while loop through simulation steps with i as running
# variable
i=0
while i <= len(t_s)-1:

    # current input values are taken from prepared input
    # file
    T_amb = InputVars['T_amb [K]'][date+
            pd.to_timedelta(t_s[i], unit='s')]
    Irr_HGloHor = InputVars['Irr_HGloHor [W/m2]'][date+
            pd.to_timedelta(t_s[i], unit='s')]

    # Input values from data file are written to FMU inputs
    FMU.set('T_amb', T_amb)
    FMU.set('Irr_HGloHor', Irr_HGloHor)

    try:
        res = FMU.do_step(current_t= t_s[i],
              step_size=h_step, new_step=True)
        if res != 0:
            print "Failed to do step", t_s[i]


    except ValueError:
        raw_input("Error...")

    # variable names defined earlier serve for extracting
    # variable values
    for var in variables:
        ResultValues[var][date+pd.to_timedelta(t_s[i],
            unit='s')] = float(FMU.get(var))

    i+=1
```

8.4 Summary

Workflow automation is a key tool to automize pre- and postprocessing processes and simulation control in order to perform perform efficient and reliable building performance simulations. Workflow automation is an enabler for massive parameter studies and a precondition as tool for uncertainty and sensitivity analysis and the like. The complexity of systems and associated computational models imposes additional requirements to building researches and simulation engineers. Handling sophisticated building simulations requires basic skills in programming, data processing and statistical analysis. Beginning with a thorough investigation of already existing resources to start from, Activitiy 1.4 compiled and presented useful methods, tools and packages for the aforementioned tasks.

As Python is already established as quasi-standard in the scientific world, it was chosen to be the core scripting language. IPython Notebooks were used as means to present some of the features of existing Python packages in order to assist building simulation engineers and researchers to achieve a higher level of workflow automation.

Chapter 9

Activity 2.1: Design of Building Systems

This chapter gives an overview about energy-efficient design and optimization of building systems based on the application of Modelica building energy simulation libraries and corresponding optimization methods.

9.1 Introduction

In Activity 2.1, we mean by a building system the combination of the building construction and the corresponding HVAC system. One important advantage of the modeling approach of Modelica is that complex technical systems can be configured in one overall system model. Hence, in the case of the design of building systems, all interdependencies between the thermal building models and the models of the HVAC system can be considered in one coupled numerical problem.

This section is structured as follows:

First, nine case studies are presented in which building energy simulation analysis and building energy designs were performed based on existent Modelica libraries. Second, component models commonly used in these case studies, such as buildings, ducts, pumps and solar collectors, were analyzed to figure out the relevance of the base models of the Annex 60 library. Third, methods for optimal design (OD) and optimal control (OC) for building systems are described and applied in three case studies. Fourth, a Modelica-based building simulation scenario for the exemplary evaluation of OD- and OC-methods is specified. For this purpose, a system model for a solar heating building was developed, which is largely based on the Annex 60 library of Activity 1.1.

9.2 Case Studies

This section describes several case studies, in which Modelica was used to design, analyze and optimize building energy and control systems. Ten different systems of residential and non-residential buildings and their corresponding energy, HVAC and control technologies were modeled by different research institutes.

The description of the different case studies also demonstrates which advantages Modelica offers for building and plant simulation in comparison to other simulation tools and approaches.

The locations of the case studies are distributed over seven countries, namely Germany, Belgium, the Netherlands, Denmark, Iran, Egypt and the USA. Therefore, building energy supply systems both for moderate and for hot climate were evaluated.

The case studies are:

- Development of PV-cooling systems for residential buildings in the MENA region (Section 9.2.1, TU Berlin, UdK Berlin, Germany),
- Control optimization of geothermal heat pump systems combined with thermally activated building systems (Section 9.2.2, Fraunhofer ISE, Germany),
- Investigation of the role of buildings in an European greenhouse gas emission free energy system (Section 9.2.3, KU Leuven, Belgium),
- Implementation of Model Predictive Control for the HVAC system of a Belgian thermally activated office building (Section 9.2.4, KU Leuven, Belgium),

- Modeling for the design of an energy and water efficient hotel (Section 9.2.5, University of Miami, UCI Engineering, USA),
- Design of an innovative two-pipe chilled beam system for both heating and cooling of office buildings (Section 9.2.6, Aalborg University, Denmark and LBNL, USA),
- Integrated optimal design and control of office buildings using renewable energy sources (Section 9.2.7, KU Leuven, Belgium),
- Development of a Virtual Computational Test-bed for Building Integrated Renewable Energy Solutions (Section 9.2.8, TU/e, Netherlands)
- Influence of German energy saving ordinances on heat demand of a residential building (Section 9.2.9, RWTH Aachen, Germany)

9.2.1 Development of PV Cooling Systems for Residential Buildings in the MENA Region

This case study deals with the simulation-based development of PV-cooling systems for residential buildings, located in different countries within the Middle East and North Africa (MENA) region. The hot climate in the MENA region leads to a relevant cooling demand in the summer in residential buildings. The standard used active air-conditioning technologies in the MENA region are water evaporation coolers, often used for dry climate conditions, and also electric-driven small vapor-compression chillers (mostly split units), in particular in regions where natural drinking water resources are scarce (see Fig. 9.1):

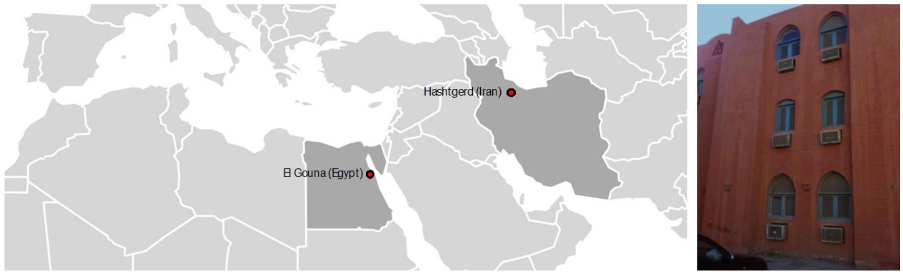


Fig. 9.1: Locations of the case study in Iran and Egypt (left) and typical residential building in El Gouna with room-wise installed split unit devices in the facade (right).

As an alternative, the available high solar irradiation potential in the MENA region, for example 1,900 kWh/(m^{2}a) in Hasthgerd New Town (located in the northern part of Iran, 100 km west of Tehran) or 2,400 kWh/(m^{2}a) in El Gouna (located at the coast of the Red Sea, 500 km south of Cairo) can be used to minimize the CO2-emissions by fossil building air-conditioning (see both locations in Fig. 9.1). Within the Young Cities project, different energy concepts for residential building cooling were developed for a new planned 35 ha district in Hashtgerd *[HNG11]*. One of the proposed technical solutions was a PV-based cooling system, which can store the produced electricity in an electric battery and the cooling energy from the compression chiller in a cold water tank, before it is used for room cooling purposes in a multi-family house with 278 m^{2}net floor area.

Fig. 9.2 illustrates the principle of a PV cooling system for air-conditioning of residential buildings:

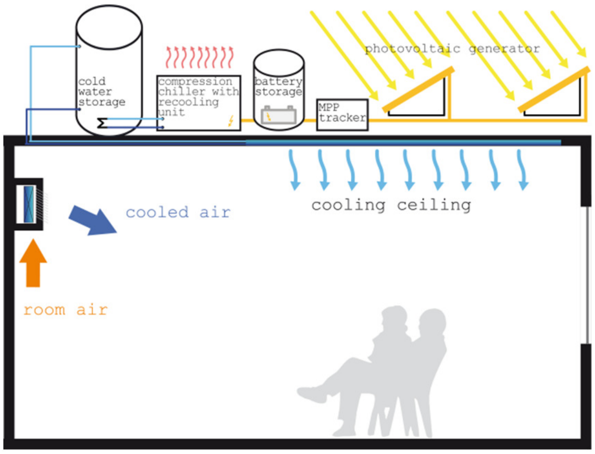


Fig. 9.2: PV cooling system, which supports direct air cooling and indirect surface cooling

The new developed PV cooling system was modeled in Modelica as a base for different simulation analyses for the locations Hashtgerd (North-Iran) and El Gouna (Red Sea, Egypt). The main objectives were the optimization of the energy system design and its most important system parameters, namely the size of the PV-generator, the capacity of the electric battery, the volume of the cold water tank and nominal power of the compression chiller. Furthermore, suited control strategies for the energy management were considered.

A corresponding simulation system model in Modelica is shown in Fig. 9.3:

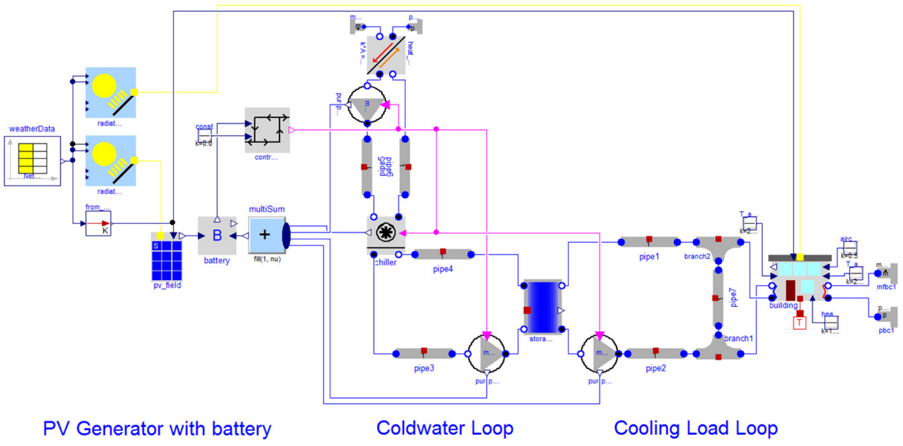


Fig. 9.3: *Plant diagram of the PV cooling system, based on the Modelica BuildingSystems library*

The model diagram illustrates the different energy transformation steps from the solar irradiation to the generated and stored electricity (PV generator with battery), the cold water production with a compression chiller (cold water loop), the storage in a cold water tank and the use of the cooling energy by the building (cooling load loop).

The system models for the simulation analysis were configured with the component models of the `BuildingSystems` library *[NGHLR12]*.

The whole energy system was modeled in Modelica and simulated with Dymola. A simplified thermal building model with one zone was used. In a more detailed analysis, a thermal multi-zone building model, modeled in EnergyPlus, was combined with the energy plant model in Modelica. Both sub-models were integrated in a co-simulation, realized with the BCVTB *[Wet11a]*. For the optimization of the system parameters such as the nominal power of the PV generator or the capacity of the electrical battery, the GenOpt optimization program (http://simulationresearch.lbl.gov/GO/) was used.

9.2.2 Control Optimization of Geothermal Heat Pump Systems Combined with Thermally Activated Building Systems

The simulation analysis focuses on the control optimization of space heating and cooling concepts that utilize environmental energy as heat source and sink. In this study a simulation-based analysis was performed for the inHaus2 located in Duisburg, Germany, as shown in Fig. 9.4. The inHaus2 is a building owned by the Fraunhofer-Gesellschaft for the demonstration and evaluation of building related technologies (http://www.inhaus.fraunhofer.de/en.html). It consists of 3 building segments with a total net useful area of 3,700 m^{2}for laboratory, research, conference and office areas. The heating and cooling energy demand of the building is approximately 60 kWh/(m^{2}a) and 14 kWh/(m^{2}a) respectively.

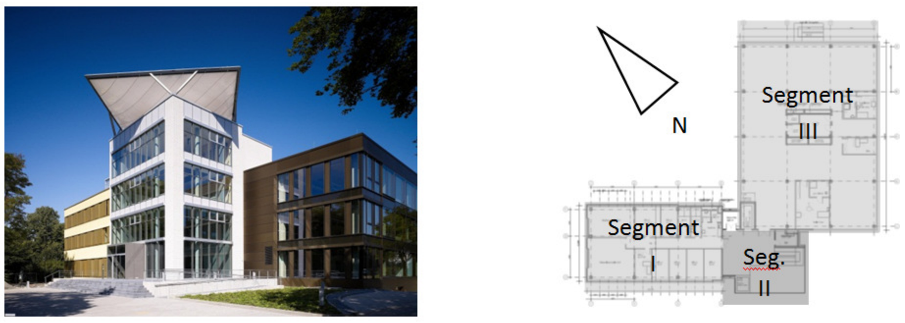


Fig. 9.4: The inHaus2 Building and the orientation of the 3 segments

The space heating and cooling is mainly realized by a heat pump system using near surface geothermal energy with 12 borehole heat exchangers, each with a depth of 120 m. The nominal heating power of the heat pump is 75 kW with a COP of 4.4 at the operating point B0/W35. Mostly thermally activated building systems (TABS) in terms of concrete core activation are used for the heating and cooling of the rooms. Cooling of the building is by means of passive cooling with the borehole heat exchangers and by active cooling using the heat pump as a chiller with a nominal cooling power of 69 kW and an EER of 6.0 at the operating point B35/W18. A monitoring system is installed in the energy supply system for the evaluation of the system performance. The data can also be used for the validation of the simulation models.

Fig. 9.5 shows the HVAC components that are integrated in the energy supply system for heating and cooling of the inHaus2.

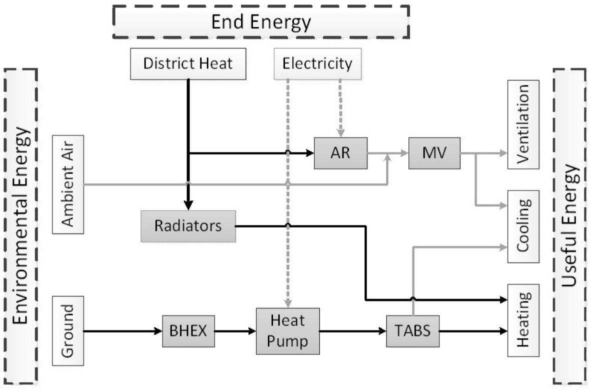


Fig. 9.5: *HVAC Schematic of inHaus2: AR-Adsorption Refrigeration, MV-Mechanical Ventilation, BHEX-Borehole Heat Exchanger*

The objective of the study was to develop optimized control strategies for the thermal as well as the hydraulic system of the geothermal heat pump system. The thermally activated building systems have a high thermal mass and therefore cause controls difficulties for the room temperature which can lead to over- or under-supply of the thermal zones. Furthermore, the hydraulic heat distribution system causes high energy consumption for the circulating pumps due to the long pipe network and high volume flow rates in the hydraulic circuits for the TABS, as well as the borehole heat exchangers.

It has been observed that the auxiliary energy in such systems can amount up to 30 percent of the total end energy consumption of the building *[KH09]*. Therefore, this study aimed at a thermo-hydraulic optimization of the control of the whole system concerning minimized end energy consumption under consideration of the thermal comfort requirements of the building.

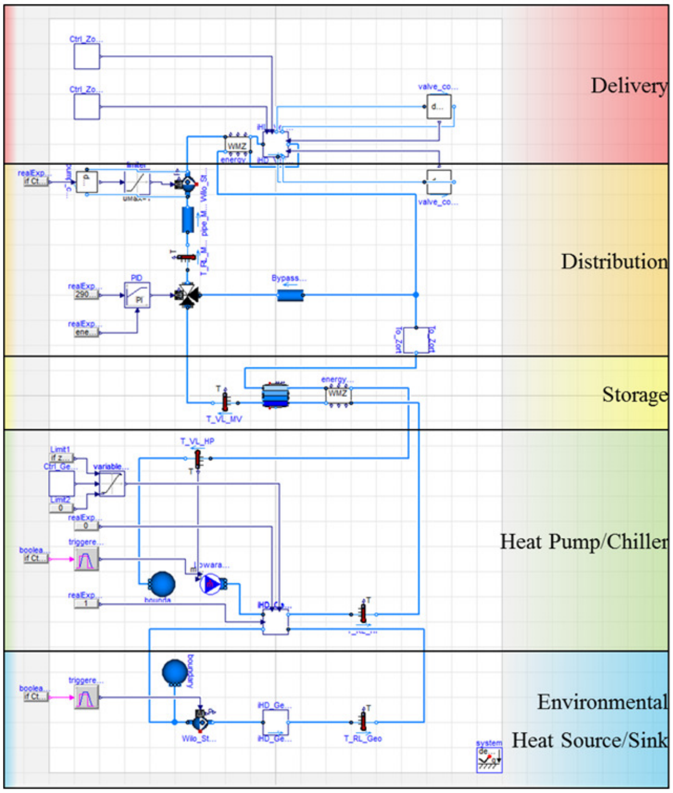


Fig. 9.6: *Representation of the geothermal heat pump system of inHaus2 in Modelica*

Fig. 9.6 shows the representation of the geothermal heat pump system of the inHaus2 HVAC system in Modelica. The system model for the simulation analysis is built using component models of the `Buildings` library *[WZNP14]*, the Modelica Standard Library (MSL) and models that are developed at Fraunhofer ISE.

For the optimization analysis, an executable dymosim file and an input file with the initial states and parameter values were generated from the system model in Dymola. Using these files, the optimization was realized with the optimization tool GenOpt (http://simulationresearch.lbl.gov/GO/). The system control optimization also had to take into account predictions for weather and internal heat gains. Therefore, a Python-framework for the realization of a model-based predictive control (MPC) was built, which coupled the system model and the optimization tool GenOpt.

9.2.3 Investigation of the Role of Buildings in an European Greenhouse Gas Emission Free Energy System

This case study was part of a project called "A fundamental study of a greenhouse gas emission free energy system" which studies the European energy system in 2050 assuming that the energy system would be all-electrical. An important part of this project was on the potential of demand-side response of electric vehicles, appliances and heating and cooling of buildings, as shown in Fig. 9.7.

The study focuses on residential buildings. A large range of technologies was considered, namely these which use local renewable energy sources (such as biomass, solar thermal, PV and geothermal) or electricity (such as heat pumps and electrical resistance heaters), eventually combined with thermal grids.

This study followed a back casting approach. The aim was to determine which combinations of energy technologies in 2050 make it possible to have a greenhouse gas (GHG) emission free energy system. For this purpose, a number of scenarios were simulated. The building stock was represented by a set of simplified models from which the energy demand and the flexibility of this demand were extracted (see Fig. 9.8).

The validity of these simple models was checked against detailed models in Modelica. In a later step, optimizations were performed, determining for each scenario what the optimal combination of HVAC systems is. The study ran from October 2011 to October 2015. Fig. 9.9 shows an exemplary system model for the above described approach.

The IDEAS library *[BDCVR+12]* was used for the verification of the simple models used in the operational model. The operational model was implemented in GAMS, using Matlab for pre- and post-processing. The verification

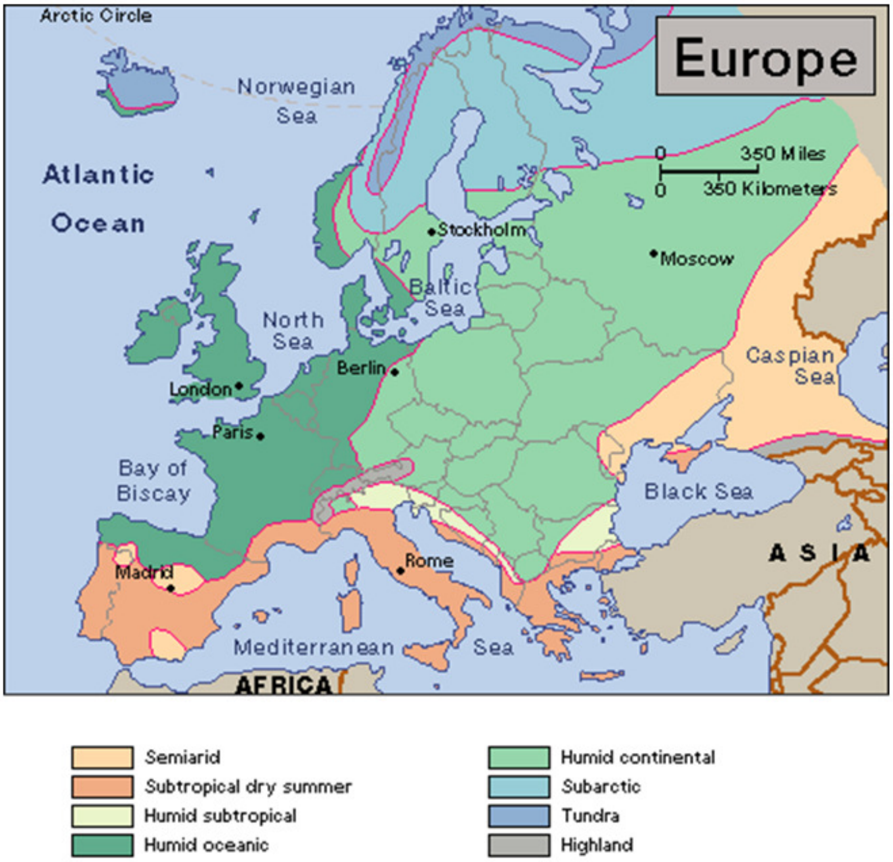


Fig. 9.7: *Europe's building stock will be studied, taking the different climates and current building stock into account. (Source: http://drmrenfrew.files.wordpress.com/2013/08/europe_climate-map-western-europe-atlas.gif)*

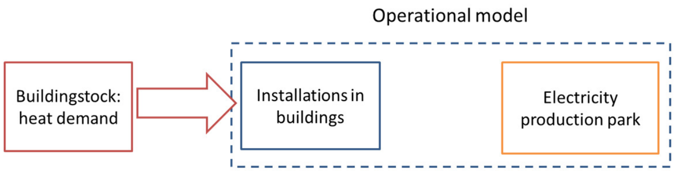

Operational model
Buildingstock: heat demand
Installations in buildings
Electricity production park

Fig. 9.8: *The interaction between the heating and cooling systems in buildings and the electricity production park was studied with an operational model which has information of both systems* [Bal13].

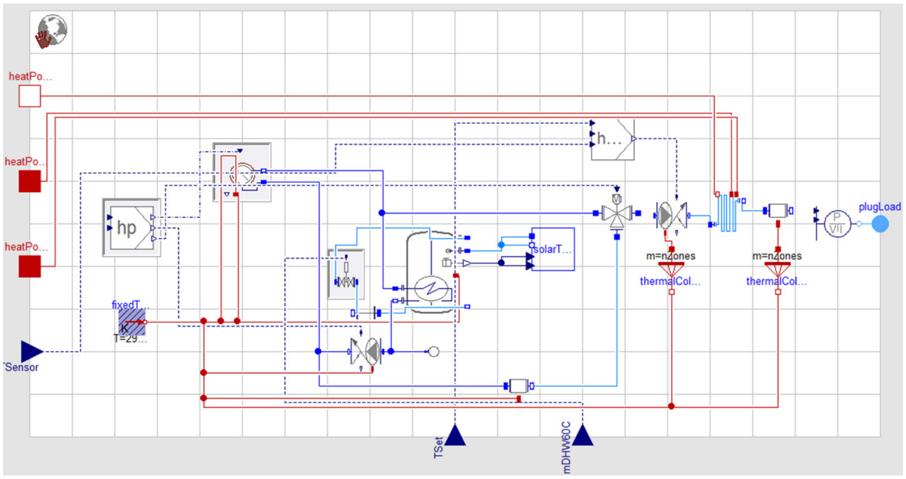


Fig. 9.9: *Results of the operational model will be checked with simulations in Modelica using the IDEAS library* [BDCVR+12]. *This figure is an example taken from the IDEAS library.*

of the results were done by performing more detailed simulations in Modelica using Dymola. The second part of the work, namely the optimization of the heating and cooling systems in different scenarios was performed in Matlab by using the yalmip toolbox *[Lof04]*.

9.2.4 Implementation of Model Predictive Control for the HVAC System of a Belgian Thermally Activated Office Building

The case study building, called “Hollandsch Huys” and shown in Fig. 9.10, is located in Hasselt, Belgium. Its construction was finished in 2007. Designed to be a low-energy, innovative office building, it makes use of thermally activated building systems (TABS), in which the thermal mass of the concrete floors is activated by a water piping circuit for both heating and cooling. The heat and cold are produced by a ground-coupled heat pump (22 single-U-tube bore holes of 75 m) and with a gas-fired boiler of 60 kW as back-up for the air handling unit.

Fig. 9.10: *Hollandsch Huys building (Hasselt, Belgium)*

The building and its systems are monitored by a large set of built-in sensors. Historical data of these measurements since 2007 are stored, allowing a detailed analysis of the technical performance. The main difficulty in interpretation of these data lies in the fact that the building has not been fully rented and occupied by the time this study started. In 2007, about 75 percent of the second floor and the roof apartment was still vacant. From 2010, the second floor was used. The systems commissioning phase was not fully completed yet, resulting in control set points being tuned during the measurement period.

Since January 2013, the building is controlled intermittently during the heating season using model predictive control (MPC) instead of the traditional rule-based control (RBC). The MPC has been developed by the University of West Bohemia, Czech Republic (Jan Siroky) and the Czech Technical University in Prague, Czech Republic *[Cig13]* in collaboration with KU Leuven. The project (SMART GEOTHERM) aimed at analyzing the improvements obtained by the reduced-model-based MPC strategy, generalizing the results for different types of thermally activated building offices using geothermal energy and extracting a new RBC algorithm based on the MPC results. The analysis was done by a combination of simulations and experiments.

Fig. 9.11 shows the components of the thermally activated building.

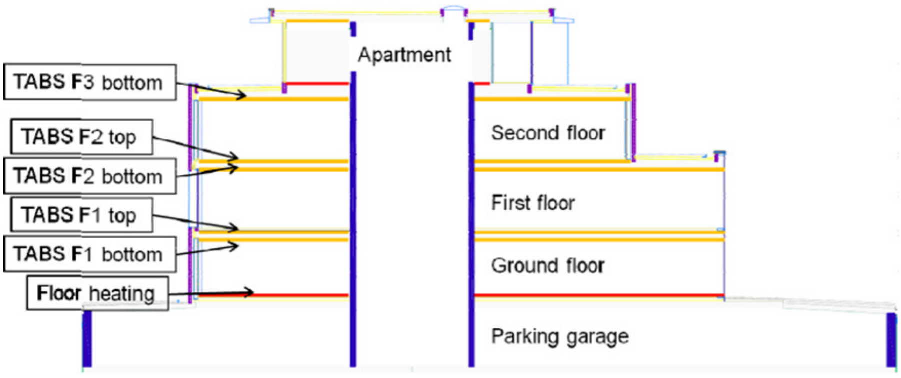


Fig. 9.11: Scheme of the thermally activated building components (floor heating: red, TABS: yellow)

The production unit in heating mode and the general layout of the simulation model are shown in Fig. 9.12 and Fig. 9.13, respectively.

The building was simulated using the IDEAS-library developed by KU Leuven. The whole energy system was modeled in Modelica and simulated with Dymola.

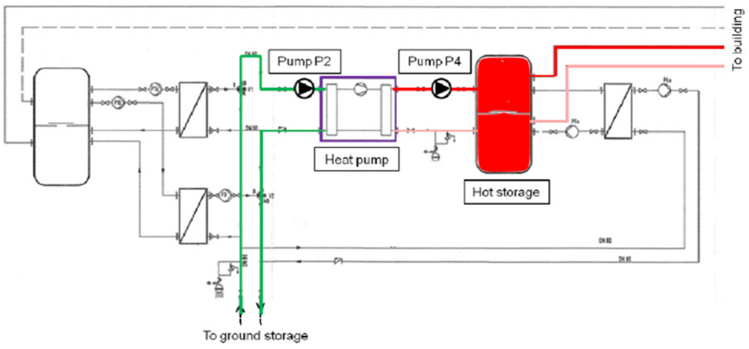


Fig. 9.12: *Production unit in heating mode*

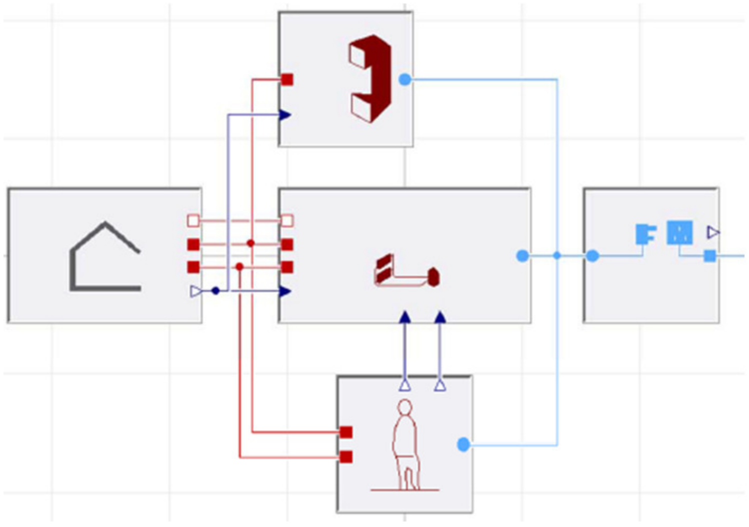


Fig. 9.13: *General layout of the Modelica system simulation model, using the IDEAS-library. Left: building structure. Top: air handling unit. Center: heating/cooling production system. Bottom: internal gains due to occupancy. Right: electrical grid.*

9.2.5 Modeling for the Design of an Energy and Water Efficient Hotel

The case study deals with the HVAC, non-portable water and domestic hot water system of the newly constructed Grand Beach Hotel at Surfside, Florida, USA *[MHB+15]* (see Fig. 9.14).

Fig. 9.14: Grand Beach Hotel at Surfside, Florida

To achieve the efficiency in energy and water, the HVAC, non-portable water and domestic systems are highly coupled. The collected rain water is used for non-portable water usage, such as flushing the toilet, and used as makeup water for the cooling tower. When condition allows, the domestic water will be pre-heated using the waste heat from the heat pump, and it will be used to reduce the cooling energy that is to be provided by the chilled water.

The simulation analysis was used for the system design and optimization. It helped the engineers decide the potential energy saving, balance the water pressure in the pipe system, and evaluate the control sequence.

Fig. 9.15 and Fig. 9.16 show the schematic HVAC and domestic water system and the corresponding Modelica models.

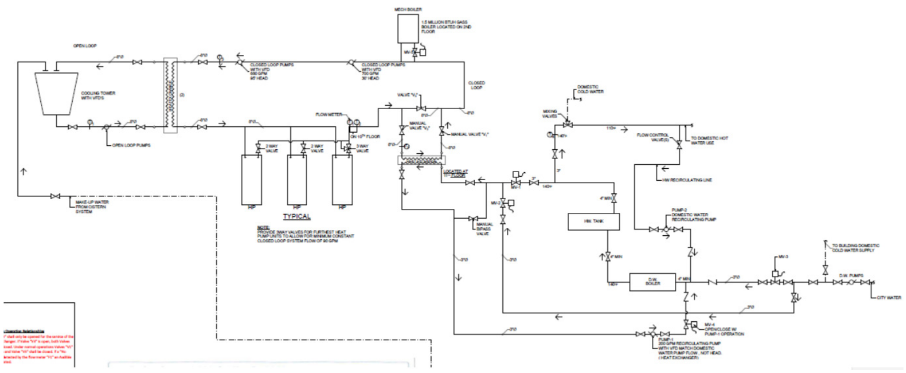


Fig. 9.15: Schematic of the HVAC and domestic water system

The system models for the simulation analysis were configured with the component models of the LBNL's Buildings library *[WZNP14]*. The whole energy and hydraulic system was modeled in Modelica and simulated with Dymola.

9.2.6 Design of an Innovative Two-Pipe Chilled Beam System for Simultaneous Heating and Cooling of Office Buildings

This case study aims to design and evaluate energy performance of an innovative two-pipe system in active chilled beam application. The system enables simultaneous heating and cooling of office buildings by transferring energy between zones, using one hydronic circuit only. The system is designed to have the same inlet water temperature on the entire circuit. This temperature is controlled between 20°C to 23°C, depending on the room air temperatures. Outlet hot and cold water is mixed together and as a result the system only needs to

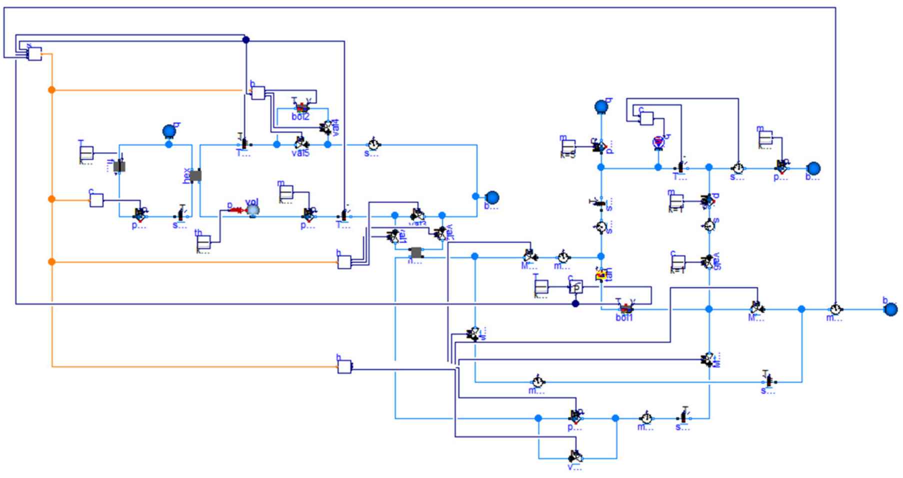


Fig. 9.16: *Modelica system model of the same system*

cool or heat the water to reach the inlet temperature again. By only having to cool or heat the water, no energy is wasted by doing both at the same time.

The possibility to design such a system was explored through a preliminary simulation-based research at the Danish Building Research Institute, in collaboration with Lindab A/S *[ANHB13]*. The system was modeled with BSim, a Danish building energy simulation tool, and integrated into a typical office building model. Standard internal gains for offices were set and thermal properties for the building envelope elements were selected according to Danish regulations. The building is located in Copenhagen (Denmark) and the correspondent weather file was used for the simulation. Results showed that the two-pipe configuration uses 3 percent less energy than a conventional four-pipe system.

However, BSim does not offer all the necessary features for a detailed design of the two-pipe system and some assumptions in the model were made. In order to be able to represent and control each single component of the system Modelica was chosen for more detailed studies.

Fig. 9.17 illustrates the principle of the two-pipe active chilled beam system. In this particular example, the inlet water temperature in the circuit is 22°C.

The room located on the north facade needs heating and the outlet water temperature is 20°C. Conversely, the space located on the south facade needs cooling and the outlet water temperature is 23°C. As a result, the outlet water temperature in the circuit was 21.5°C. The system only needs to provide energy to reach 22°C. Fig. 9.18 shows the system model in Modelica used for the preliminary analysis.

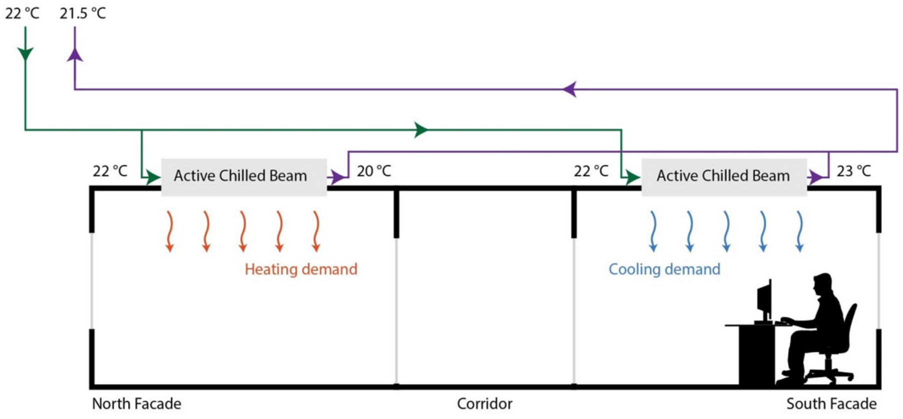


Fig. 9.17: *Two-pipe active chilled beam system for simultaneous heating and cooling*

After the preliminary study, a more detailed study was conducted, using a typical office building model. Simulations were run for two construction sets of the building envelope and two conditions related to inter-zone air flows. To calculate energy savings, a conventional four-pipe system was modelled and used for comparison. The conventional system presented two separated water loops for heating and cooling with supply temperatures of 45°C and 14°C, respectively. Simulation results showed that the two-pipe system was able to use less energy than the four-pipe system thanks to three effects: useful heat transfer from warm to cold zones, higher free cooling potential and higher efficiency of the heat pump. In particular, the two-pipe system used approximately between 12% and 18% less total annual primary energy than the four-pipe system, depending on the simulation case considered *[MWA+17]*.

The Buildings library from LBNL (*[WZNP14]*) was used for the system modeling, the energy analsysis and the controls design.

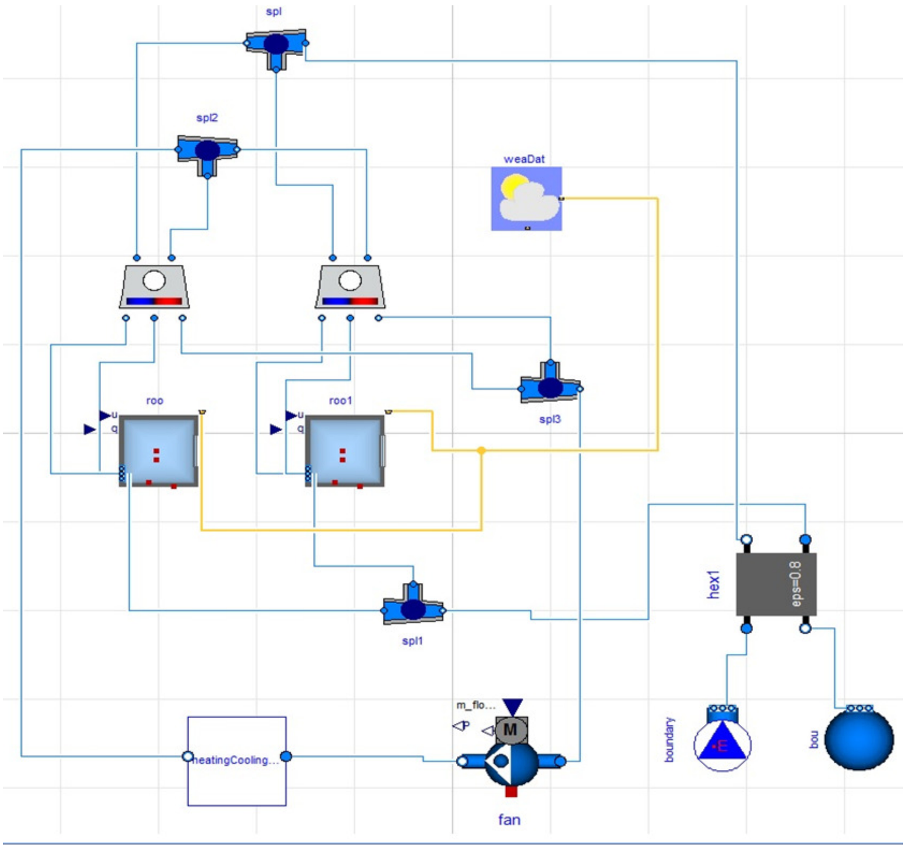


Fig. 9.18: Modelica model of the building system

9.2.7 Integrated Optimal Design and Control of Office Buildings Using Renewable Energy Sources

This case study was performed in the PhD project “Integrated optimal design and control of office buildings using renewable energy sources” at the KU Leuven. The aim of this PhD project was to find a methodology for integrating optimal control and design of office buildings (compare with Fig. 9.19).

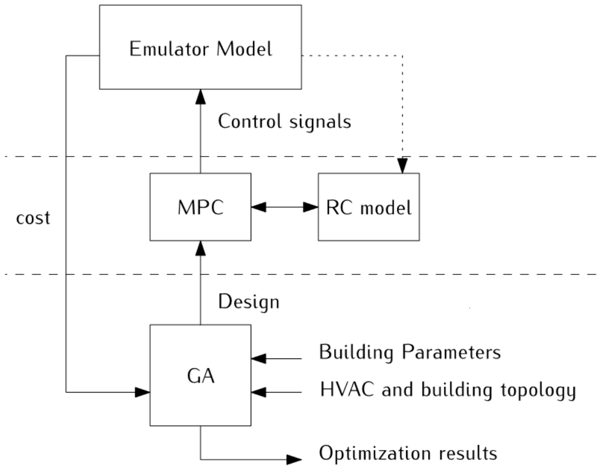


Fig. 9.19: *General outline of the case*

Usually, optimal control and design are solved as two independent problems. Optimal control is performed on a fixed design and optimal design is performed by assuming a simplified control sequence. However, technologies such as Concrete Core Activation (CCA) typically operate in a transient regime due to their large time constants. Furthermore the added value of renewable energy sources is difficult to assess without considering the entire system. This PhD project aimed to assess the potential of optimal control and optimal design using an integrated approach and dynamic simulations. This research proposed a methodology that optimally exploits this potential in practice so that the total costs (investment and running costs) are minimized. Modelica was used to create a parameterized emulator model of the building that needs to be optimized. A Model Predictive Controller (MPC) performed the optimal control of the building's HVAC-system. A controller model (RC model) was used to limit the computational effort. The performance of this controller was evaluated using the Modelica emulator model. This control optimization was nested in a stochastic optimization that optimizes the design of the building with respect to building parameters, different HVAC-components, etc.

The IDEAS library *[BDCVR+12]* and the Buildings library *[WZNP14]* was used for this research in combination with Dymola.

9.2.8 Development of a Virtual Computational Test-Bed for Building Integrated Renewable Energy Solutions

In this case study, a virtual computational test-bed for building integrated renewable energy solutions was developed using Modelica. This test-bed helps assessing the full scale performance potential of existing products based on pilot scale testing results from the Eindhoven University of Technology (TU/e) campus and simulations which are calibrated using the pilot scale testing measurements. The pilot scale testing facility is a cooperation known as SolarBEAT (http://www.seac.cc/projects/solar-beat) which has been established between the TU/e and the Solar Energy Application Centre (SEAC), an independent research organization. This facility is being constructed on the roof of the Vertigo building, home of the Department of the Built Environment at TU/e. It consists of dummy buildings with building-integrated photovoltaic/thermal (BIPVT) solar collectors.

The end product of this work will be a complete building system simulating photovoltaic modules, solar thermal collectors, occupancy, ventilation, infiltration, heating system, domestic hot water demand and heat storage, as shown in Fig. 9.20. The virtual computational test-bed helps assessing the performance of different technologies. Those technologies will consist of BIPVT systems from different companies which are interested in testing their products and assessing their performance.

The building is a single family house for 4 occupants. It is assumed that the house has 10 thermal zones: 3 zones at the ground floor including the kitchen, living room and the entrance, 5 zones in the second floor including 3 bedrooms

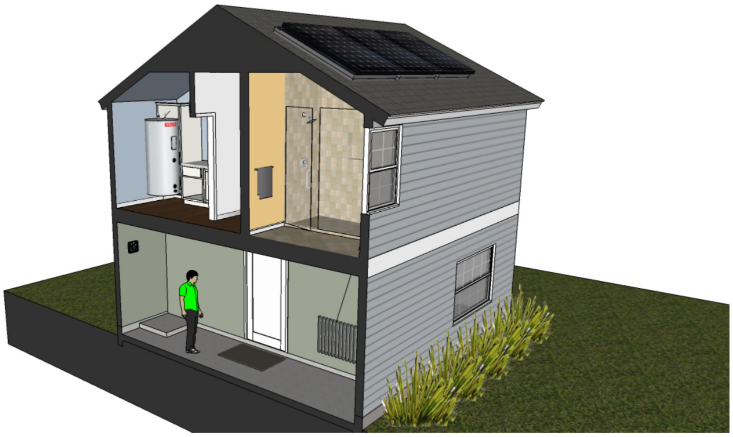


Fig. 9.20: *Full home system schematic*

and the second floor/attic which is divided into 2 rooms. The different zones are connected to each other thermally through the common walls and through the doors which can be controlled to be open or closed. However, the air connection between the different floors was not modeled.

The Modelica model consists of three major parts: the PV model, the solar thermal model and the house model. The house model was used to model the heating demand of the home. The first two models were used to model the generation of renewable energy for the given location of interest. Using the combined simulation, it was possible to analyze the dynamic interaction between the renewable generation systems and the energy demand of the home. The house is heated using typical radiators which are connected to the water heating system.

A complete heating system was modeled in this case study. The schematic in Fig. 9.21 illustrates a more detailed depiction of the solar thermal, space heating and domestic water heating system.

This system contains a heat storage tank connected by a heat exchanger to an array of solar collectors. This tank is also connected to a boiler which heats the water to a specific temperature required by the heating system. The heating is provided by radiators in each of the thermal zones. Each of the

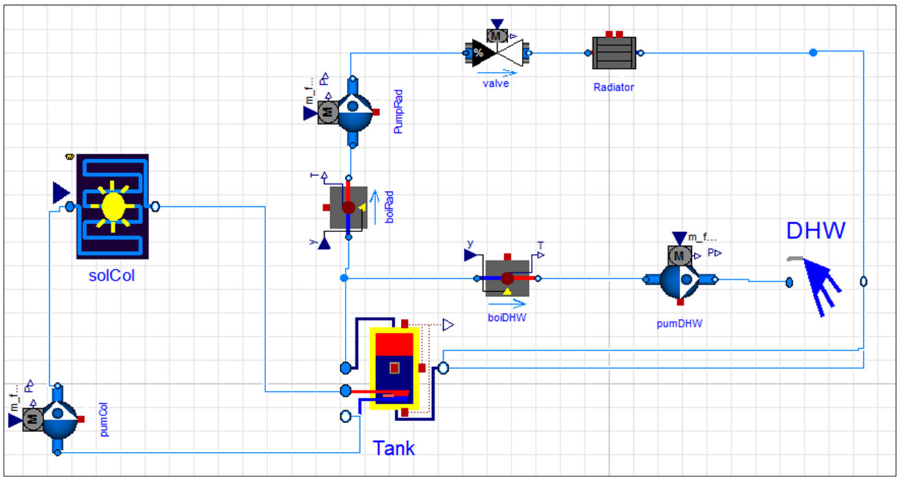


Fig. 9.21: *Heating system schematic*

radiators is controlled by a valve according to the set-point temperature. A pump in the collector circuit and another pump in the heating circuit ensure hot water circulation. Moreover, the domestic hot water circuit contains another boiler to ensure sufficient hot water temperature, and it contains a pump for the water circulation. All the components are controlled depending on the required temperature for the day, night and the outside temperature.

The electrical model should consist of an electrical network, representing the dwellings distribution system to which the PV and all power consuming equipment are connected. However, modeling different components inside the house and their energy consumption is a complicated task which requires the availability a well determined usage and occupancy profile. In the case of this work, the decision was made to represent the electrical demand using average measured data varying throughout the year.

The electrical power demand can vary according to the user behavior, the number of occupants and the resolution of the data available. For instance, a set of electric power demand data with a resolution of 5 minutes is more accurate than one with a resolution of 15 minutes. The 15 minutes data has flatter peaks of power demand due to its temporal averaging. Therefore, the higher the resolution, the more representative is the data *[SHK14]*. That being said, it is difficult to get access to detailed specific data from power companies due to privacy issues. The most complete data available for this application is an average consumption data for one house with a resolution of 15 minutes. The electrical network contains also a resistor representing losses and a voltage source to set the operating voltage as shown in Fig. 9.22.

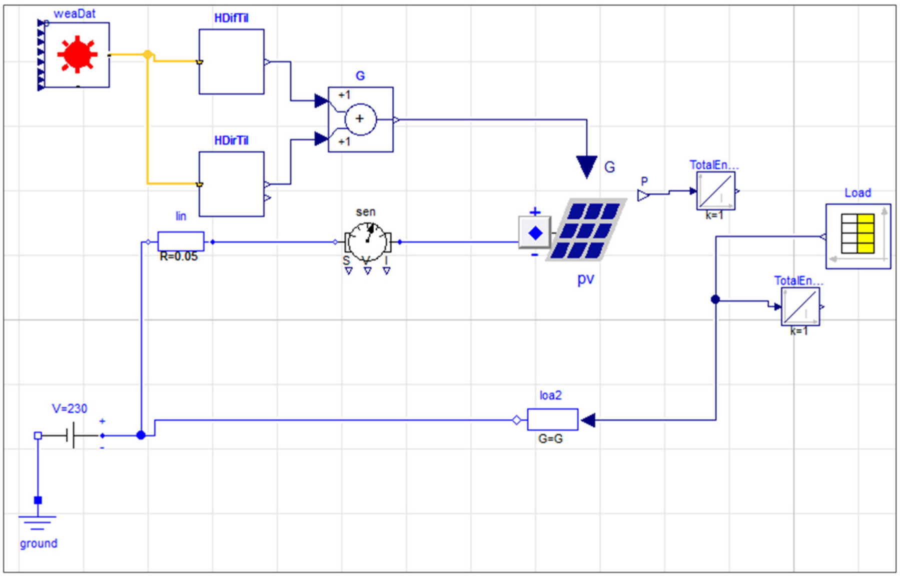


Fig. 9.22: Electric model containing the weather data, PV and the load profile

Fig. 9.23 represents the implementation of the model in Modelica. The block on the left contains the BIPVT and the heating system, while the block on the right represents the single family Dutch house. The top block is the weather data input.

The system models for the simulation analysis were configured with the component models of the Buildings library from LBNL *[WZNP14]*. The whole energy system was modeled in Modelica and simulated with Dymola.

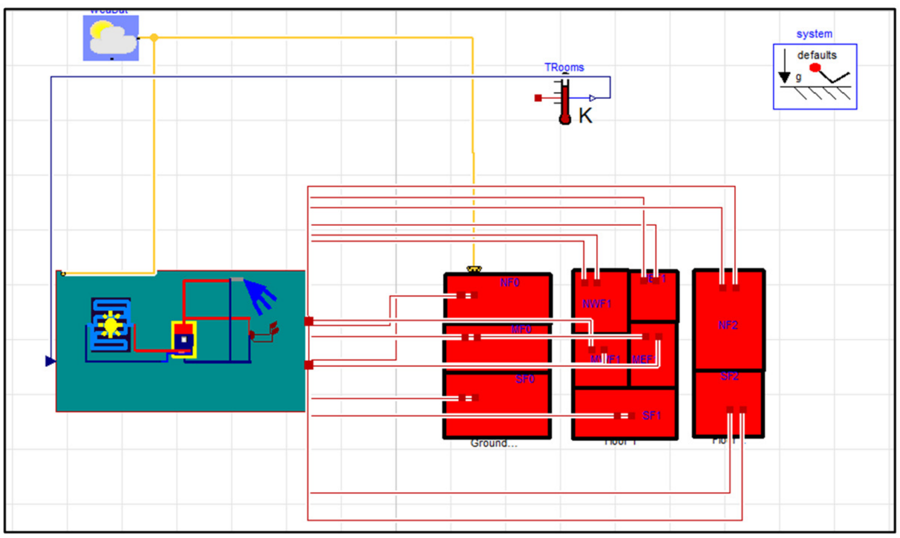


Fig. 9.23: Model implementation in Modelica

9.2.9 Influence of German Energy Saving Ordinances on Heat Demand of a Residential Building

This case study investigated the influence of different German energy saving ordinances on the heat demand of a residential building. In the past, German government had issued several energy saving ordinances, which define limits for the properties of building envelopes in order to reduce heat demands in new building construction. As these properties define the minimum requirement for all buildings built in a certain time period, they can help estimating a building's properties given only its year of construction. Therefore, reference simulations of buildings with these standard properties can serve as benchmarks for a large share of Germany's building stock and as the foundation for further investigations of the system or its parts.

As a reference building for this case study, a one family dwelling with an area of 150 m^{2}over two floors (see Fig. 9.24) was used. In order to only quantify the influence of the building envelope, each room is equipped with an ideal heater. For the location of this reference building, central Germany with

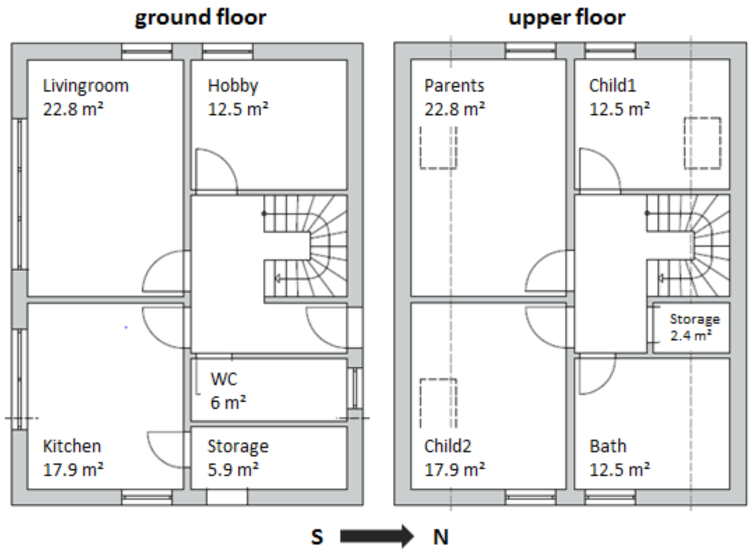


Fig. 9.24: Floor plan for the reference building

weather input from the German test reference year region 12 was selected. The same building setup and weather input was used for building instances with envelope properties of energy saving ordinances from 1984, 1995, 2002, and 2009. Because this case study shall serve as a foundation for further studies, it was included as a demonstration example in the Modelica library AixLib *[BM10][FCL+15]*. Further details can be found in *[CSM14]*.

The Modelica system model was built in a modular way, starting with wall elements including windows and doors. Several wall elements were grouped together to form a room model. The room models were connected to model a single floor. The floors can be connected to the outside environment and to heating system models to form a model of the entire building. In order to test the influence of different building envelope properties, upper-level parameters specify the wall constructions according to a chosen energy saving ordinance. Further parameters can be used to specify a light, medium or heavy building construction and air exchange rates. Thus, there is little manual effort needed to compare different building setups in a certain context. Through the modularity of the modeling approach, the building model could also be connected to a more elaborate heating system model with a prototype controller to test the control performance subject to different building properties. This allows engineers to quickly test their developments with a variety of typical German building setups.

The Modelica model of this building setup is shown in Fig. 9.25. The whole energy system was modeled in Modelica and simulated with Dymola.

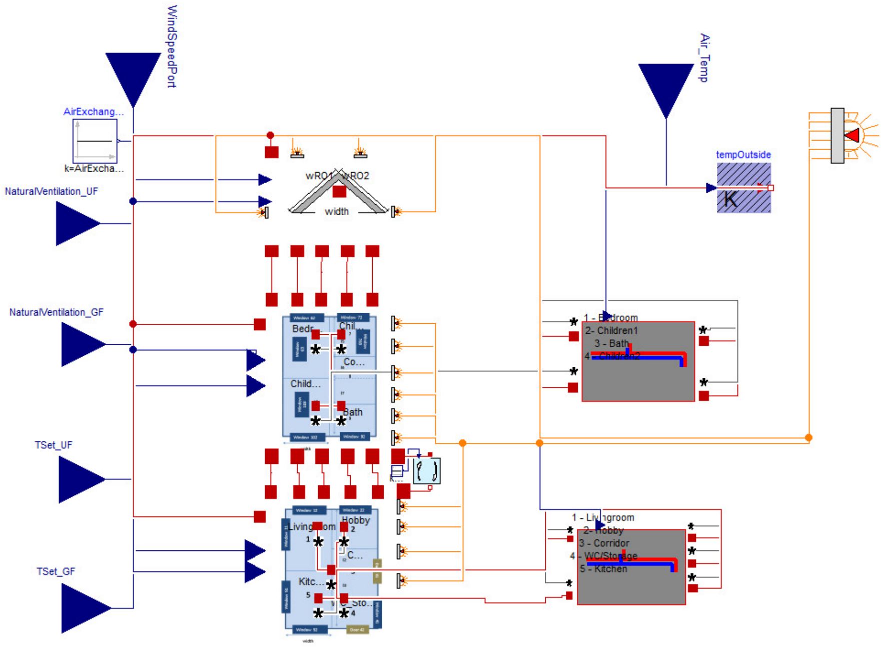


Fig. 9.25: *This case study is an illustrative example of the library AixLib. It uses components from AixLib's Building package as well as basic models from the Modelica Standard Library*

9.2.10 Comparative Analysis of all Case Studies

A comparison of all nine case studies regarding the used Modelica libraries, the type of the component models and the used Modelica tools provides following information:

Used Modelica libraries: Different Modelica libraries were used for the case studies, 3 times the Buildings library from LBNL, 3 times the IDEAS library from KU Leuven, once the BuildingSystems library from UdK Berlin. Further specialized Modelica models were used by the Fraunhofer institute ISE, which are not part of the publicly available Modelica libraries.

Used Modelica component models: A wide range of energy plant models and thermal building models were used in the case studies:

- Auxiliary models:
  - Controllers (Two point-controller, PI controller)
  - Data reader & interpolation (e.g. weather data, user behavior)
  - Radiation calculation on tilted surfaces
- Energy transformation models:
  - PV module and generator with maximum power point tracking
  - Solar thermal collector
  - Electric driven heat pump and chiller
  - Borehole heat exchanger for ground coupled heat pumps
  - Cooling tower
  - Boiler (e.g. condensing gas, wood pellet)
  - HVAC: adiabatic cooling, mechanical cooling
- Energy storages Models:
  - Electric battery
  - Thermal water storage (cold water, hot water)
  - Bore-field
- Energy transport models:
  - Thermo-hydraulic components: pipe, branch, two-way and three way valves, constant and variable speed pump
  - HVAC components: ducts, ventilators, heat recovery components
  - Heat emission systems such as radiators and floor heating
  - Heat exchangers
  - Heat emission components: TABS, radiators, active beams
- Building models:
  - Simplified thermal building model (one zone)
  - Detailed building model (multiple zones)

Used Simulation Tools: In all of the case studies, Dymola was used as the

Modelica simulation tool. (When the case studies were conducted, Dymola was the only tool that could simulate all used models. However, as of Spring 2017, JModelica fully supports the Annex 60 library and the Buildings library.) In two cases, EnergyPlus was used in addition by co-simulation, based on BCVTB (https://simulationresearch.lbl.gov/bcvtb). In one case study GAMS (http://www.gams.com) was used for an operational model.

In two of the case studies, optimization analysis with GenOpt(https://simulationresearch.lbl.gov/GO/) were done. One case study used a Python-framework for the realization of a model-based predictive control (MPC). One case study used the MATLAB toolbox YALMIP (http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Main.HomePage) for optimization.

9.2.11 Reasons for the Use of Modelica

The reasons for the use of Modelica as stated by the case study participants can be clustered into the following groups:

Flexible and extensible object-oriented modeling approach: The object-oriented modeling approach leads to a high working efficiency and supports a transparent and clear understanding of component and system models. Hereby, an easy manipulation, re-use and/or adaption of existing component models is possible and missing component models can be easily implemented. The graphical and hierarchical modeling approach enables a well-structured and clear modeling process and supports system models with different levels of detail in the sub-models (e.g. a detailed plant model in combination with a simplified thermal building model). The combined use of existing large Modelica libraries enables new system models, which are normally not present in existing simulation tools.

Acausal equation-based modeling approach: The equation based approach leads to an easy understanding of the physical modeling of existing models and newly implemented models. It also gives a good physical insight in the problem and the components. The result is a clear understanding of the underlying physical-technical equation system at the component and system level. Especially the non-causal approach of Modelica supports necessary features in energy plant simulation domain such as the bi-directional mass flow within thermo-hydraulic networks.

Multi-domain approach: The multi-physical modeling approach of Modelica including the domain specific common interface definitions supports the integration of domain specific sub-models into a common system model (e.g. building physics, hydraulics and electrical grids).

Numerical reasons: The automatically translation of the textual Modelica model into a running simulation model with time integration allows the modeler a clear separation between the physical-technical model itself and its numerical solution.

External interfaces to Modelica: Modelica-models can be easily coupled with external optimization software (e.g. GenOpt and GAMS). The scripting language Python can control Modelica simulation tools and can serve for pre-and post-processing steps. Co-simulations of Modelica with other simulation tools are possible, e.g. with BCVTB and/or FMI. One typical example is the co-simulation of a HVAC system, modeled in Modelica and a multi-zone building model in EnergyPlus.

Collaborative development: The modular approach of Modelica supports common software development within single teams, and also within a broader distributed model development such as done for the Annex 60 library that was developed in Activity 1.1 and is used in major Modelica libraries for building energy applications.

9.2.12 Drawbacks of Using Modelica

The authors of the case studies have stated the following most important drawbacks by using the Modelica language, Modelica libraries, or Modelica tools in the context of building energy simulation:

Long simulation times for detailed whole building simulation: Whole building simulations are often slower than with dedicated building energy simulation programs, in particular if detailed HVAC models are combined with multi-zone building envelop models. This has several reasons: First, Modelica models often include more detail than for example an EnergyPlus or TRNSYS simulation. Typical examples include modeling of realistic control sequences and pressure-driven mass flow distribution in pipe and duct networks. Second, numerical solvers in the tools use the same time step for all equations. However, for building envelope heat conduction, much larger time steps could be used than what is required for control loops. Here, the use of multi-rate solvers with error control is promising. Third, if an *event* is triggered by a model, then the whole time integration is stopped, and the integration is restarted, typically with a low order integration method. However, many events could be treated in a subsystem, in particular if they affect other parts of the model through integrators, as integrators smoothen discontinuities. Fourth, Modelica translators typically convert the models to a dense system of equations rather than preserving the sparsity, which then could be exploited using sparse solvers. Fifth, ordinary differential equation solvers for stiff equations that scale cubic in the number of state variables are often used. This can lead to large computing time if the building envelope model, with many slow evolving states, and the HVAC model, with few but fast evolving states, are coupled. However, research and development is ongoing on all of these issues, and they are not an inherent problem of the Modelica language which for simulation is converted to efficient C-code, as described in Section 5.3.4. Rather, it is a problem of the solvers that are currently in use and need to be improved as users simulate larger Modelica models than they did in the past. At the time of this writing, various research in such model translations and numerical methods that allow faster simulation is ongoing, see for example *[MBKC13][FBC+14][Cas15][BBCK15][BRC16][CR16][OE17][BCB17][JHB17]*. As Modelica separates declarative model formulations from solution methods, these methods can be made available in simulation environments with little to no changes to the underlying model libraries.

Reliance on one commercial Modelica tool: During the research phase of this project (2013 to 2016), only Dymola supported the entire Modelica language specification. Hence, complex Modelica system model run only partly with other commercial or open Source Modelica tools. Since Spring 2017, the situation is improving as the open source Modelica tool JModelica works with the Annex 60 library and the Buildings libary, and other tool providers stated that they are working on increasing the support of models that are used for building energy simulation.

Connector types were not standardized: At the start of the Annex, some Modelica libraries used connectors and function that compute enthalpy as a function of temperature that were different from the ones standardized in the

Modelica Standard Library. This made combining existing models difficult as adapters had to be implemented. In the meantime, the Annex60, AixLib, BuildingSystems, Buildings and IDEAS libraries all use the same adapters and media functions, making their models compatible.

Fewer validated models than established building simulation programs: Modelica offers fewer models than for example EnergyPlus or TRNSYS, and since the libraries are new compared to these established tools, the range of validated models is smaller. However, Modelica libraries are more transparent in comparison to EnergPlus, in particular regarding how components and systems are controlled. Also increasing the scope of models and their level of validation was a key motivation for the development of the Annex 60 library as a common base for different Modelica libraries. See also Section 5.4.5 for the quality control in the Annex 60 library.

9.3 Model Analysis

Outgoing from the described case studies, an analysis about the used Modelica models of the four libraries for building energy and plant simulation AIXLib, BuildingSystems, Buildings, and IDEAS was performed. Because the quantity of all of the used Modelica component models was large, only models used in one or several of the case studies were considered in the analysis.

In the second step, a comparison between the spectrum of the used models and the present models of the Annex 60 library (version during Fall 2014) were conducted. In this manner, commonly used models were identified. This served as a basis to prioritize which new component models to add to the Annex 60 library. This fits with the idea of the Annex 60 project of providing a core Modelica library, while allowing individual libraries to differ in the specialized models that they provide.

The following ranking was obtained by the statistics of how often a Modelica model was used in all of the case studies and *was not present* during Fall 2014 in the Annex 60 library:

1. Compression chiller/heat pump; Thermal zone models or building models: 6 x

2. Radiation calculation for tilted surfaces: 5 x
3. Thermal water storage (hot water/cold water): 4 x
4. Data reader / interpolator: 3 x
5. Borehole heat exchanger; Cooling ceiling / radiant slab; Photovoltaic model: 2 x
6. Solar thermal collector model; thermally activated building systems; electric battery model; Cooling tower: 1 x

Starting from this case, the following five most used models should be integrated in the future versions of the Annex 60 library:

A simplified thermal building model: In the majority of case studies, the building energy and plant simulations were coupled, with the focus of the analysis being on the plant. The building model delivered the dynamically calculated heating and cooling demand, typically in yearly simulations. Hence, a simplified thermal building model is needed.

A thermal water storage model: Many thermal energy concepts used a thermal storage tank to store hot or cold water. They are integrated in hydraulic concepts in different manners (e.g. internal/external heat exchangers) and also often the thermal stratification is important (e.g. for solar thermal systems). Hence, a thermal horizontal 1D-discritized tank model with flexible inputs and outputs should be part of the Annex 60 library.

A compression chiller/heat pump model: Many energy concepts are based on (reversible) heat pumps and/or compression chillers. Hence, a simplified model should be included in the library, which uses characteristic curves for fast system simulation analysis.

A solar radiation calculation model: One basic feature consists in the transformation of horizontal solar radiation as obtained from the weather data into the incident irradiation on tilted surfaces (e.g. building envelope, solar thermal collectors, photovoltaic modules).

A data reader model: All building and plant system models have to read discrete, typically hourly, weather data, which have to be transformed into continuous boundary conditions for the building and its HVAC system. Hence, a data reader model which supports different types of interpolation should be part of the Annex 60 library.

9.4 OD/OC Method Analysis

Methods for optimal design and optimal control were used in following of the case studies:

- Multi-criteria optimization of PV-cooling systems for residential buildings (Section 9.4.1, TU Berlin, UdK Berlin, Germany), and
- Model predictive control framework (Section 9.4.2, KU Leuven, Belgium).

9.4.1 Multi-Criteria Optimization of PV-Cooling Systems for Residential Buildings

Optimization problem: A residential building in a hot climate region with space for two families (net floor area of 278 m^2) is air-conditioned by a solar PV cooling system. The optimization problem was analyzed for two different climate locations in the MENA region: The city Hashtgerd in Iran and the city El Gouna in Egypt *[NGHN13]*. The maximum air temperatures in Hashtgerd and El Gouna are nearly the same (41 ℃), but the mean temperature during the summer period in El Gouna (30.4 ℃) is much higher than in Hashtgerd (26.0 ℃). The yearly horizontal global solar radiation is about $1,800\,\mathrm{kWh/(m^2\,a)}$ in Hashtgerd and $2,400\,\mathrm{kWh/(m^2\,a)}$ in El Gouna.

The mean U-value of the thermal envelope of the considered building was $0.593\,\mathrm{W/(m^2\,K)}$ for walls, windows and roof, and $0.350\,\mathrm{W/(m^2K)}$ for the floor. The potential surface for PV modules on the roof was about 135 m^2.

The study aims to solve a multi-criteria optimization of the PV cooling system. Furthermore, the PV cooling system has to supply 100 percent of the cooling demand by solar energy without any use of additional electricity from the grid.

For this purpose a cost-function with three different design criteria was defined:

1. Over temperature: The first criteria was to avoid overheating for an optimal thermal comfort. If the indoor air temperature exceeds the set temperature of 26 ℃, then the difference *TAir-TSet* (over temperature) was integrated over time.

2. Investment costs: The second criteria was the minimization of the investment costs. The following assumptions to the specific investment costs were made: (PV generator: 0.7 Euro/W_{peak}; Battery: 600 Euro/kWh; Compression chiller: 0.4Euro/W_{el}; Cold water storage: costs in $Euro = 1649.81\ V^{-0.464}$ where V is the storage volume.
3. Durability of the components: The third criteria was the reduction of switching on/off events of the compression chiller.

These three criteria were weighted with the factors $f_1 = 1$, $f_2 = 1$ and $f_3 = 2$. Electricity costs were not considered, because the PV cooling system works grid-independently.

These chosen optimization parameters and the possible parameter spans for the optimization algorithm were as follows:

1. Number of PV modules: 1 to 83,
2. Capacity of the electric battery: 1 to 80 kWh,
3. Nominal power of the compression chiller: 0.1 to 5.0 kW and
4. Volume of the cold water storage: 0.1 m^3 to 10.0 m^3.

Used Modelica system model for optimization: The PV cooling system was modeled in Modelica by the use of the BuildingSystems library. It consists of the following seven main sub-models which are also shown in Fig. 9.26:

1. A weather data reader and its conversion to different surfaces (PV surfaces and building facades),
2. a PV generator system model,
3. an electrical battery model,
4. a model of a compression chiller with its recooling unit,
5. a cold water storage model,
6. a simplified thermal building model,
7. controller models.

Several controllers were implemented within this system model. The first controller manages the cooling load pump. This pump switches on if the room air temperature reaches the set temperature with an offset of 2 K. The second controller measures the water temperature in the top of the cold water storage. If the temperature rises above 7℃, the compression chiller, the recooling pump and the cold water loop pump turns on, but only if the battery load state is not lower than 20% (discharge protection).

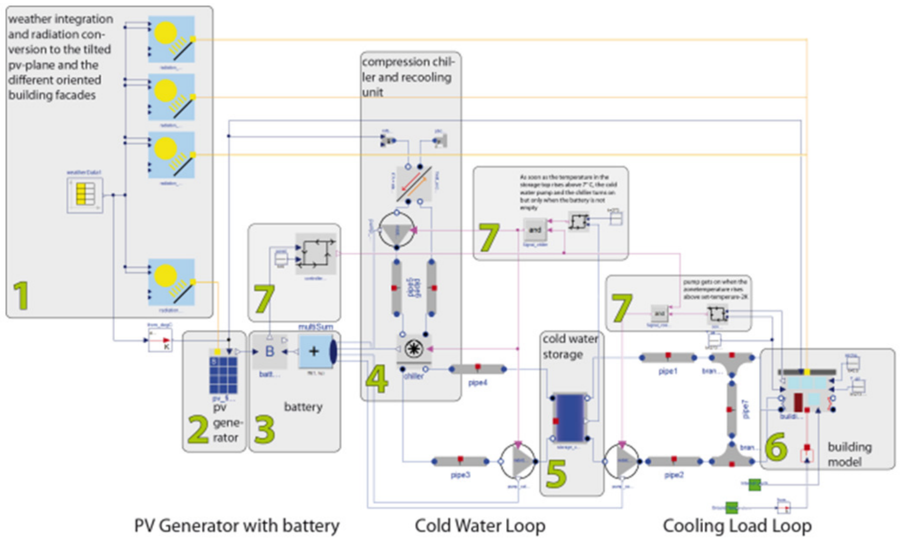

weather integration and radiation conversion to the tilted pv-plane and the different oriented building facades
1
compression chiller and recooling unit
7
2
pv generator
3
battery
4
cold water storage
5
6
building model
PV Generator with battery
Cold Water Loop
Cooling Load Loop

Fig. 9.26: *Modelica system model of the PV cooling system and its sub-models*

Used method and tools for optimization: The GenOpt framework (http://simulationresearch.lbl.gov/GO/) was used for the parameter optimization problem. GenOpt varies predefined parameter sets within the admissible parameter bounds, dependent on the selected optimization algorithm. It then starts the Dymola simulation for one year, and the simulation model writes the cost function value to a file that will be read by GenOpt. Dependent on the progress of the optimization, Genopt varies the parameter set again for the next simulation run until a convergence criterion is satisfied. The GPSHookeJeeves algorithm was used for the optimization problem.

Results of the optimization: GenOpt was started at both climate locations with the same parameter set:

- Number of PV modules: 70,
- capacity of the electric battery: 40 kWh,
- nominal power of the chiller: 1.0 kW,
- volume of the cold water storage: 5.0 m^3.

After 120 simulations, the cost-function for both locations has converged, as shown in Fig. 9.27.

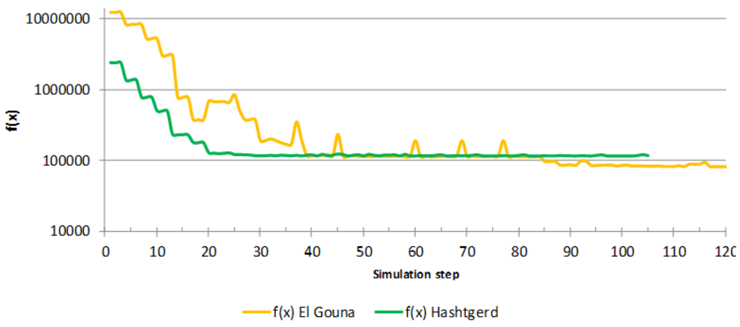


Fig. 9.27: *Final cost-function value and its composition for the locations El Gouna and Hashtgerd*

The cooling load in Hashtgerd leads to higher peak values. However, the yearly cooling energy in El Gouna is higher than in Hashtgerd, because of its higher mean temperature. The maximum temperature during the summer is nearly the same at both locations.

For these reasons, the optimization algorithm has found for Hashtgerd the same number of PV modules (83). This is maximum value of possible modules on the roof area. The location El Gouna with a higher mean temperature and cooling demand needs greater storage capacities for cooling energy. The daily and yearly timeline of the solar irradiation is more balanced in comparison to Hastgerd. Consequently, the optimization algorithm calculated for El Gouna a larger and cheaper cold water storage (6.1 m^3 instead of 5.5 m^3) and reduced the capacity of the electric battery (40 kWh instead of 55 kWh) in comparison to Hashtgerd.

Fig. 9.28 shows the detailed investment costs for both locations. The main costs are from the battery and the PV generator. The costs of the cold water storage and the chiller are less significant.

The PV cooling system causes in Hashtgerd a higher overheating time (16.9 hours) as in El Gouna (10.8 hours) caused by the higher cooling load peaks. Because of the higher alternating external temperature, the compression

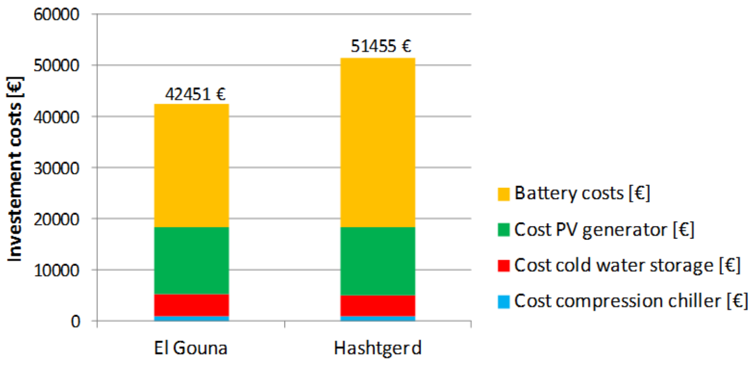


Fig. 9.28: *Investment costs in Euro for both locations*

chiller switches on more often in Hashtgerd (355 times) than in El Gouna (270 times).

9.4.2 Model Predictive Control Framework

Model Predictive Control (MPC) is a control strategy that can be applied to building applications. The idea of MPC is to create an optimization problem that optimizes the control variables (valve positions, heat flow rates) such that the operation point with the lowest cost is obtained that still satisfies comfort constraints and other constraints that may be imposed. An MPC problem estimates future energy use under consideration of weather and occupancy forecasts. Hence, it anticipates future solar gains, thereby possibly reducing the heating power or pre-cooling the building at night when cooling is typically cheaper. Comfort constraints are typically formulated based on zone operative temperatures. Depending on the complexity of the model that is used to estimate energy use, the optimization problem may be categorized into different problem classes such as a Linear Programming (LP), Quadratic Programming (QP), Nonlinear Programming (NLP) etc. Efficient specialized optimization solvers exist depending on the problem class. LP's can for instance be solved very quickly. Each of these solvers however requires that the equalities, inequalities, cost function and input data defining the problem are provided to the solver in a certain format. Obtaining these equations and data requires knowledge of specialists, especially for large problems.

Classical building simulation tools typically do not allow the automated extraction of these model equations or input data. The model equations of individual component models may be publicly available, but using these directly would again require the equations to be recombined manually when individual component models are coupled. Modelica offers some interesting advantages in this respect, which may greatly simplify extracting models for use in MPC. First, open source libraries exist where the users has control over the type of equations, meaning that component models may be simplified to match a certain problem class. For instance, a library of linear building component models may be constructed if the user wants to obtain a LP controller model. Second, Modelica simulation environments allow any variable to be stored in an output file, and therefore the required input data can be exported. Third, Modelica simulation environments typically allow to linearize a model and extract a linear state space formulation of the problem in the form

$$\begin{aligned} \dot{x} &= A\,x + B\,u, \\ y &= C\,x + D\,u, \end{aligned} \tag{9.1}$$

where x are the problem states, $\dot{x}$ is the time derivative of x, u are model inputs and A, B, C, D are constant matrices. When linearizing a model, these matrices are computed. Fourth, specialized Modelica solvers such as JModelica and OpenModelica allow to directly optimize the model equations without needing to export them first. At the time of this writing, their solvers, however, conservatively assume that the problem is a non-linear programming (NLP) problem, which increases the computation time compared to using a solver for linear programming (LP) problem.

Using these properties of the Modelica language and their solvers, an automated way for creating and exporting linear controller models with matching input data may be set up. A first step would be to create a building model in Modelica using a library that contains only linear equations. Second, a state space model of the form (9.1) needs to be extracted. Third, an input data file needs to be generated that matches the inputs u of the state space model. Then, these data can be used by an efficient LP optimization algorithm to compute optimal control results. Note that an interesting feature of Modelica is that the optimization algorithm, which is not implemented in Modelica, may be compiled into C-code that may be called from Modelica. This way, an MPC controller may be embedded into a Modelica model, which removes the need for co-simulation techniques.

This approach was used at the KU Leuven to set up an automated Model Predictive Framework using the IDEAS library. In [PJH15] the linearization approach is outlined. In [JH16] the framework is explained and applied to an office building. In [PSJ16] the framework's performance is compared to a grey box approach.

9.4.3 Optimal Control of Room-Level HVAC and Facade Control

This example demonstrates the use of collocation methods to solve a constrained nonlinear optimal control problem, and compares its computing performance to a gradient free optimization method. The example demonstrates the benefits obtained through the problem manipulation that the Modelica language allows due to its declarative nature.

9.4.3.1 Optimization Problem

The example minimizes sensible cooling, heating and fan energy demand for a thermal zone of a variable air volume flow (VAV) system by adjusting the time profiles of the supply air mass flow rate, the shading device control signal and the reheat power at the terminal box, subject to comfort constraints. Our application requirements are that the optimizations of individual zones are decoupled from each other, and that the central HVAC system can have its own control. For our example, the central HVAC system will supply air at 18°C and at a relative humidity required for humidity control.

Fig. 9.29 shows the simplified thermal model of the zone. Separate models exist for exterior constructions, partitions to adjacent zones and internal mass. These constructions are characterized by thermal capacitors and thermal conductance that account for heat conduction and convection. There is also a thermal conductor for infiltration.

The windows were modeled with a thermal conductance G_{win} and a power source $Q_{sol}^{win}(t) = S_{rad}(t)\, A_{win}\, SHGC\, \chi(u_{win}(t))$, where $S_{rad}(t)$ is the total so-

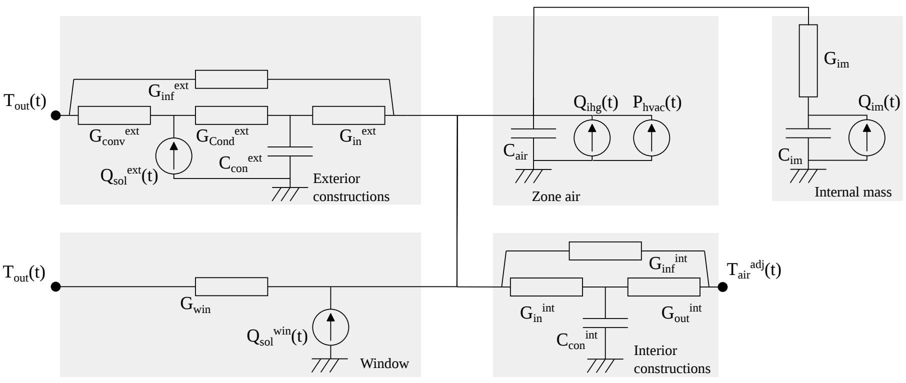


Fig. 9.29: Simplified model of a thermal zone.

lar radiation per unit area towards the window normal direction, A_{win} is the glass area, $SHGC$ is the solar heat gain coefficient of the window, $u_{win}(t) \in [0, 1]$ is the position of the blind, and $\chi\colon \mathbb{R} \to \mathbb{R}$ is a function that models the impact of the blinds on the fraction of solar radiation that enters the zone. The contribution due to solar radiation on the external constructions is $Q_{sol}^{ext}(t) = S_{rad}(t)\, A_{ext}\, \alpha$, where A_{ext} is its area, and α is the solar absorption coefficient of the exterior surface. The thermal conductance G_{im} models the heat transfer between the air and the internal mass. The power sources $Q_{ihg}(t)$ and $Q_{im}(t)$, respectively, model the internal heat gains that contribute to the zone air and internal mass. The internal heat gain is defined as $Q_{ihg}(t) = P_{occ}(t) + P_{lights}(t) + P_{plug}(t)$, where $P_{occ}(t)$ is the internal heat gain due to occupancy, $P_{lights}(t)$ is the internal heat gains due to lights, and $P_{plug}(t)$ is the internal heat gains due to the plug loads. The power delivered by the HVAC system is $P_{hvac}(t) = P_{hea}(t) + P_{sup}(t) + P_{fan}(t)$, where $P_{hea}(t)$ is the power released by the reheating coil in the VAV box, $P_{fan}(t)$ is the fan power, and $P_{sup}(t)$ is the cooling power provided by the supply air from the central HVAC system through the VAV box.

The latter is $P_{sup}(t) = \dot{m}_{air}(t)\, c_p\, (T_{sup}(t) - T_{air}(t))$, where $\dot{m}_{air}(t)$ is the mass flow rate of air passing through the VAV box, c_p is the air specific heat capacity, $T_{sup}(t)$ is the temperature of the supply air entering the VAV box and $T_{air}(t)$ is the temperature of the air in the zone.footnote{Note that the computation of $P_{sup}(t)$ is approximate to decouple the individual optimizations of multiple zones from each other. This was done to reduce the dimensionality of the optimization and to allow them to be solved in a distributed way.

The actual sensible cooling provided by a cooling coil in a system with economizer is $\bar{P}_c(t) = \dot{m}_{air}(t)\, c_p\, (T_{sup}(t) - (T_{mix}(t) + \Delta T_{fan}(t)))$, where $T_{mix}(t)$ is the mixed air temperature after the economizer and $\Delta T_{fan}(t)$ is the temperature raise over the fan. If $y_{out}(t) \in [0, 1]$ is the outside air fraction, this becomes $\bar{P}_c(t) = \dot{m}_{air}(t)\, c_p\, (T_{sup}(t) - T_{air}(t) + y_{out}(t)\, (T_{air}(t) - T_{out}(t)) - \Delta T_{fan}(t))$. As $y_{out}(t)$ and $\Delta T_{fan}(t)$ (through the fan efficiency) depends on the mass flow rates and return air temperatures of other zones, these terms have been neglected in order to decouple the individual zone-level optimizations.} The power of the fan is $P_{fan}(t) = P_{fan}^{nom}(\dot{m}_{air}(t)/\dot{m}_{air}^{nom})^3$, where $\dot{m}_{air}^{nom}$ is the nominal supply air mass flow rate, and P_{fan}^{nom} is the fan power required to supply $\dot{m}_{air}^{nom}$ to the zone. Weather data for Sacramento, CA, have been used.

The described model was implemented in Modelica. The model can be described as an initial-value ordinary differential equation. Thus, it is a special case of the generalized DAE system, in which the algebraic constraints $Y(\cdot,\cdot,\cdot,\cdot,\cdot)$ are absent. Therefore, the state variables, control functions and parameter are

$$\begin{aligned} x(t) &= [T_{air}(t),\ T_{im}(t),\ T_{con}^{int}(t),\ T_{con}^{ext}(t)], \\ u(t) &= [\dot{m}_{air}(t),\ u_{win}(t),\ P_{hea}(t)], \\ \Theta &= [T_{air}(t_0)]. \end{aligned}$$

The energy consumption of the HVAC system is

$$E(t_f) = \int_{t_0}^{t_f} (-P_{sup}(t)\,\eta_{coo} + P_{hea}(t)\,\eta_{hea} + P_{fan}(t))\, dt,$$

where η_{coo} and η_{hea} are coefficients that represents the efficiency of the HVAC system to provide cooling and heating. These typically include the coefficient of performance of chillers or heat pumps, or the efficiency of the furnace, as well as the effect of free-cooling by the economizer. The optimization variables are the time profiles for the control signals of the supply air mass flow rate $\dot{m}_{air}(\cdot)$, the blind position $u_{win}(\cdot)$, the reheat power $P_{hea}(\cdot)$, as well as the initial zone temperature $T_{air}(t_0)$. We modified the equation for $E(t_f)$ to balance the weights of all its terms, obtaining

$$E_\gamma(t_f) = \int_{t_0}^{t_f} (-\gamma^1 P_{sup}(t)\eta_{coo} + \gamma^2 P_{hea}(t)\eta_{hea} + \gamma^3 P_{fan}(t))\, dt,$$

where $\gamma \in \mathbb{R}^3$, with $\gamma^i > 0$ for all $i \in \{1, 2, 3\}$, are constants.

The optimal control problem is formulated as

$$\begin{aligned}
\underset{u(\cdot)\in\mathcal{U}}{\text{minimize}} \quad & E_\gamma(t_f) + \int_{t_0}^{t_f} \kappa\, u_{win}(t)^2\, dt,\\
\text{subject to} \quad & F(t,\, \dot{x}(t),\, x(t),\, u(t),\, y(t),\, \Theta) = 0,\\
& F_0(\dot{x}(t_0),\, x(t_0),\, u(t_0),\, y(t_0),\, \Theta) = 0,\\
& T^l_{air}(t) \leq T_{air}(t) \leq T^u_{air}(t),\\
& 0 \leq u_{win}(t) \leq u^u_{win}(t),\\
& \dot{m}^l_{air} \leq \dot{m}_{air}(t) \leq \dot{m}^u_{air},\\
& P_{hea}(t) \geq 0,\\
& T_{air}(t_0) = T_{air}(t_f),
\end{aligned}$$

for all $t \in [t_0,\, t_f]$, where $\mathcal{U}$ is the set of admissible control functions $u(\cdot)$, $\kappa \in \mathbb{R}^+$ is a constant, $T^l_{air}(t)$ and $T^u_{air}(t)$ are the lower and upper bounds of the comfort region of the air temperature, $u^u_{win}(t)$ is the maximum admissible value for the control signal of the blind, and $\dot{m}^l_{air}$ and $\dot{m}^u_{air}$ are the minimum and maximum supply air mass flow rates for the zone.

The cost function includes an additional term that penalizes excessive control actions of the windows. For example it penalizes deploying the blinds at night. The equality constraint imposes that the temperature of the air at the start and at the end of the optimization period are the same.

9.4.3.2 Used Method and Tools for Optimization

The optimization problem was solved using JModelica *[AGT09]*, using the collocation method described in Section 4.2.3 and solved the finite dimensional NLP problem with IPOPT version 3.11.9, and with the linear solver mumps.

For comparison of the computation time, the optimization problem was also solved using the Nelder-Mead algorithm, a derivative-free algorithm that is implemented in JModelica. For the solution with the collocation method, n_e = 48 elements was used, each being 30 minutes long. For each element, three collocation points were used. The optimization ran for a summer day.

To assess the performances of the collocation-based method, it was compared with the simulation-based Nelder-Mead optimization algorithm. The Nelder-Mead algorithm can be applied to nonlinear optimization problems where derivatives are not available, as is the case with conventional building simulation programs. See [WW04] for a comparison among the Nelder-Mead and other derivative free methods. The optimization variables in the Nelder-Mead optimization problem are the initial temperature of the air in the zone, the control signal for the blinds, and the supply air mass flow rate. To reduce the size of the optimization problem, it was simplified for the Nelder-Mead optimization as follows: The heating power was set to $P_{hea}(t) = 0$ for all $t \in [t_0,\ t_f]$, and both the supply air mass flow rate $\dot{m}_{air}(t)$ and the blind control signal $u_{win}(t)$ was discretized using one-hour rather than 30 minutes intervals. The decision to choose no heating is appropriate for this hot day. Therefore, the optimization variables are

$$\begin{aligned} z = [&T_{air}(t_0),\ \dot{m}_{hvac}(t_0), \dot{m}_{hvac}(t_0 + \Delta t), \dots, \dot{m}_{hvac}(t_0 + 24\,\Delta t), \\ &u_{win}(t_0),\ u_{win}(t_0 + \Delta t), \dots, u_{win}(t_0 + 24\,\Delta t)]. \end{aligned}$$

Hence, the total number of variables to optimize is $n_z = 51$. Let $\tau = \{t_0 + i\,\Delta t\}_{i=0}^{24}$. The cost function minimized by the Nelder-Mead algorithm is

$$\begin{aligned} f_{NM}(z) &= E_\gamma(t_f) + \mu_j\,(\Delta T_{err} + \Delta T_{init})\,, \\ \Delta T_{err} &= \max_{t \in \tau}(0,\ T_{air}(t) - T^u_{air}(t),\ T^l_{air}(t) - T_{air}(t)), \\ \Delta T_{init} &= (T_{air}(t_0) - T_{air}(t_f))^2, \end{aligned}$$

where $\mu_j = 1.5^j/10$, for the iteration counter $j \in \mathbb{N}$, are penalty function multipliers, ΔT_{err} penalizes thermal comfort constraint violations, and ΔT_{init} forces initial and final zone air temperature to be equal. The Nelder-Mead algorithm accepts upper and lower limits for the optimization variables. The limits $T_{air}(t_0) \in [20, 26]$°C, and $\dot{m}_{air}(t) \in [\dot{m}^l_{air}, \dot{m}^u_{air}]$ and $u_{win}(t) \in [0, u^u_{win}(t)]$ for all $t \in \tau$ were used.

9.4.3.3 Results of the Optimization

Both algorithms were run on Ubuntu 14.04 64 bits hosted on a Virtual Box virtual machine with 2 GB of memory and four processors. Both algorithms were initialized with a sub-optimal feasible solution computed by a PI controller that maintains the temperature of the zone at a fixed value of 24°C. The selection of initial conditions of the optimization problem play an important role, in particular for collocation based methods *[MA12]* because the trajectories of the state, input, output and algebraic variables are used to create the initial polynomial approximations and to place the collocation points. Hence, the initial conditions have to be feasible.

Collocation method: Fig. 9.30 to Fig. 9.33 show the results of the collocation method. The optimal cooling power delivered by the HVAC system maintains the temperature of the zone inside the thermal comfort zone. The constraint on the minimum air change causes the temperature to not quite reach the upper comfort bound at night. The optimal blind position follows the maximum admissible value in order to reduce the solar heat gain during the afternoon, when the supply mass flow rate reaches its maximum capacity of seven air changes per hour. Fig. 9.32 shows that the additional term in the cost function that penalizes excessive control actions causes the blinds to be closed at night. The red line in Fig. 9.31 shows that the VAV box does not reheat the air to maintain comfort in the zone. This result is consistent with the high outside air temperatures.

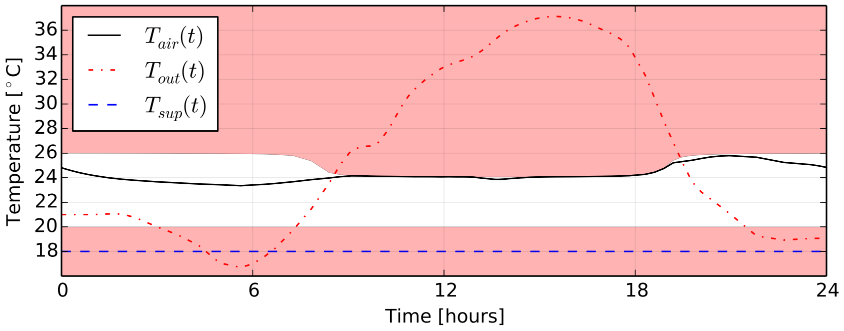


Fig. 9.30: Optimal room air temperature (black line), thermal discomfort zone (red area), outside temperature (red dotted) and supply air temperature (blue dashed).

Nelder-Mead algorithm: The blue lines in Fig. 9.35 show the optimal cooling load computed during successive iterations of the Nelder-Mead algorithm. Each line corresponds to a different penalty function multiplier μ_i, with the bold

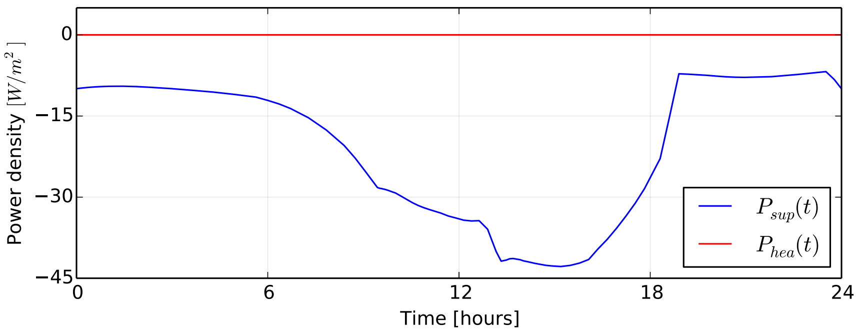


Fig. 9.31: *Optimal cooling and heating power provided by the HVAC system through the VAV box.*

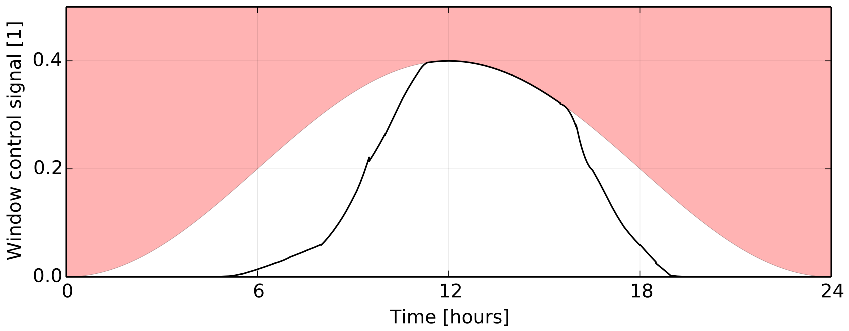


Fig. 9.32: *Optimal control signal for the blind (black line).*

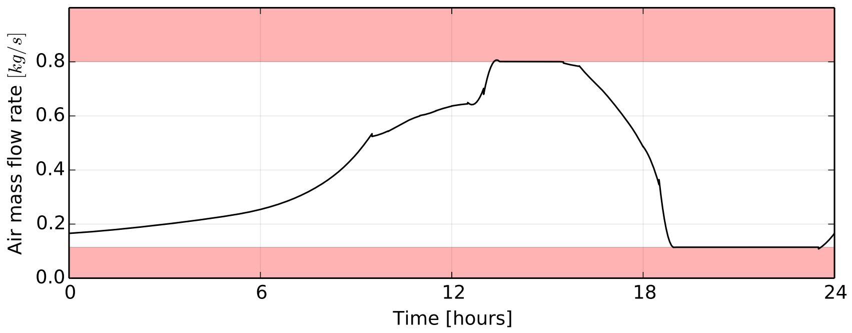


Fig. 9.33: *Optimal supply air mass flow rate. The red areas indicate the infeasible regions.*

line being the final optimal solution. Fig. 9.34 shows a similar plot for the resulting optimal zone air temperature. As μ_i increases, the solutions converge towards an optimal trajectory that satisfy the constraints.

Comparison: Both algorithms converge to solutions that minimize the energy consumption with a similar strategy. The strategy is to maintain the zone temperature close to the upper boundary of the thermal comfort zone during the peak hours. Both algorithms compute an optimal solution that reaches a maximum cooling power density of approximately 45 W/m^2 around 4 PM. The trajectories computed by the two algorithms are different because of the different discretization mechanisms, implementations of the cost and constraint functions, and the numerical methods. The optimized total energy consumption $E(t_f)$ is 15.721 kWh/day and 16.543 kWh/day for the optimization with the collocation and Nelder-Mead method, respectively. Hence, the collocation methods reduces the cost by an additional 5.2%.

Table 9.1 and Table 9.2 show the statistics of the two optimization methods. Despite the fact that the optimization problem solved with the collocation method is larger and has a finer temporal resolution, it was solved approximately 2, 200 times faster than the problem solved using Nelder-Mead.

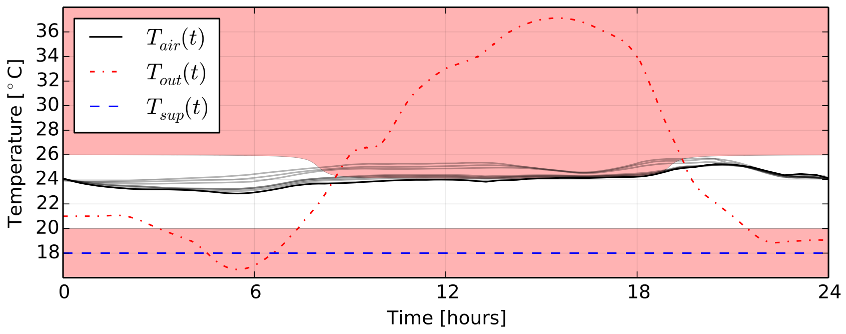


Fig. 9.34: *Resulting optimal zone air temperature (black line) and its successive approximations for different values of the penalty function multiplier μ_i (gray lines).*

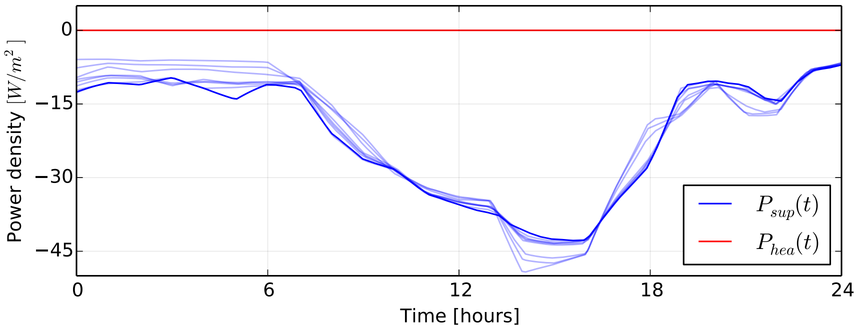


Fig. 9.35: *Optimal cooling power provided by the HVAC system through the VAV box (blue line) and its successive approximations for different values of μ_i (light blue lines), and optimal heating power provided by the VAV box (red line).*

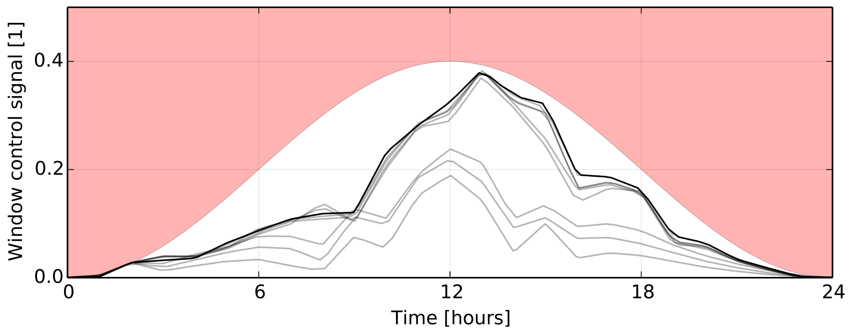


Fig. 9.36: *Optimal control signal for the blind (black line) and its successive approximations for different values of μ_i (gray lines).*

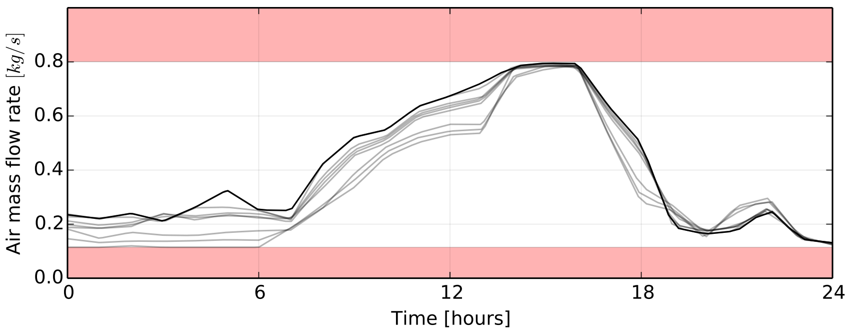


Fig. 9.37: *Optimal supply air mass flow rate (black line) and its successive approximations for different values of μ_i (gray lines). The red areas indicate the infeasible regions.*

Table 9.1: *Statistics of the collocation-based optimization method.*

Number of variables	10385
Number of equality constraints	9953
Number of inequality constraints	1160
Number of Iterations	72
Initialization time	0.93 s
Solution time	6.79 s
Post-processing time	0.04 s
Total computing time	7.75 s

Table 9.2: *Statistics of the simulation-based optimization method for different values of the penalty multiplier u_i and for the cumulative time of the whole optimization.*

μ_i	Computing time in [s]	Function evaluations	Iterations
0.150	2673	25033	447
0.225	1461	13605	242
0.337	1877	16853	300
0.506	2829	25033	447
0.759	2276	20773	370
1.139	1768	16517	294
1.709	1498	13829	246
2.563	1793	15957	284
3.844	1222	10805	192
Total	17401	15840	2822

9.4.4 Building simulation scenario for optimal design

This section describes a case study in which a design parameter optimization was applied to a Modelica building system model. The system model is based on the Annex 60 library and the Modelica Standard Library.

9.4.4.1 Description of the Scenario

The system model represents a two story residential building with 120 m^2 floor area. It has a solar heating system and a backup heater. Figure Fig. 9.38 shows the Modelica model.

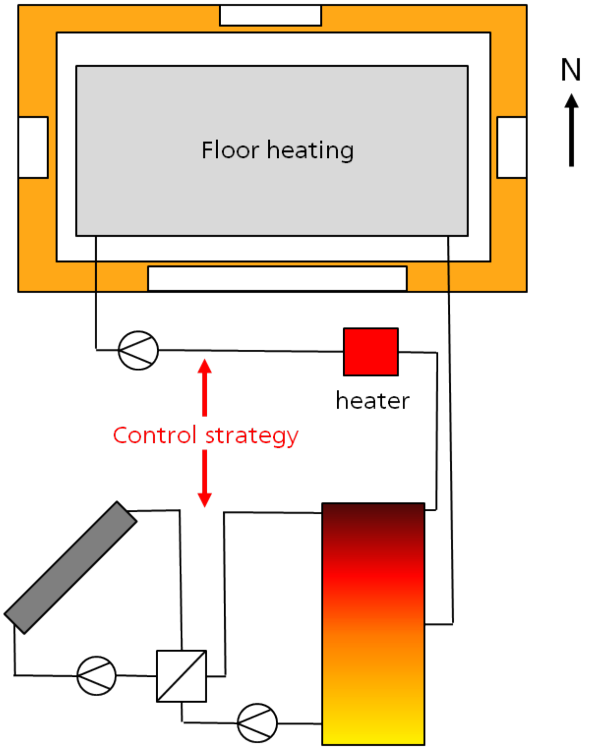


Fig. 9.38: Residential building with its solar heating system

The simulation model has to be fast because the parameter optimization can require hundreds of simulations.

The scenario is specified with following aspects:

9.4.4.2 Climate Locations

Two locations are used, San Francisco (relative warm climate during the heating period) and Chicago (very cold climate during the heating period).

9.4.4.3 User Behavior

A mean air change rate of 0.5 1/h is assumed. The air temperature setpoint profile for heating is 20 ℃. Internal heat sources are considered with a specific mean value of 3 W/m^2. 50 percent is convective, and 50 percent is radiative.

9.4.4.4 Building Geometry and Thermal Properties

The building is a single family house with 120 m^2 floor space. It has two identical stories and a flat roof as shown in Fig. 9.39.

The total thickness of the external walls in the reference case is 0.3 m, also for the base plate and the ceiling between the two stories. All inner walls have a thickness of 0.2 m. The roof thickness is 0.4 m.

The windows in the south façade have a width of 3 m and a height of 1.4 m. All the other facades (east, west, north) have windows with a smaller size of 1 m width and 1.4 m height. The height of the sill of all windows is 0.6 m. Because it is a theoretical model, doors are neglected. All rooms have the same size of 15 m^2.

The materials listed in Table 9.3 were used for the specification of the opaque building elements.

This leads to the U-values shown in Table 9.4.

For the windows, the mean U-value of the panes and frame of the window is 1.0 W/(m^{2}K) and the g-value for perpendicular irradiation is 0.6.

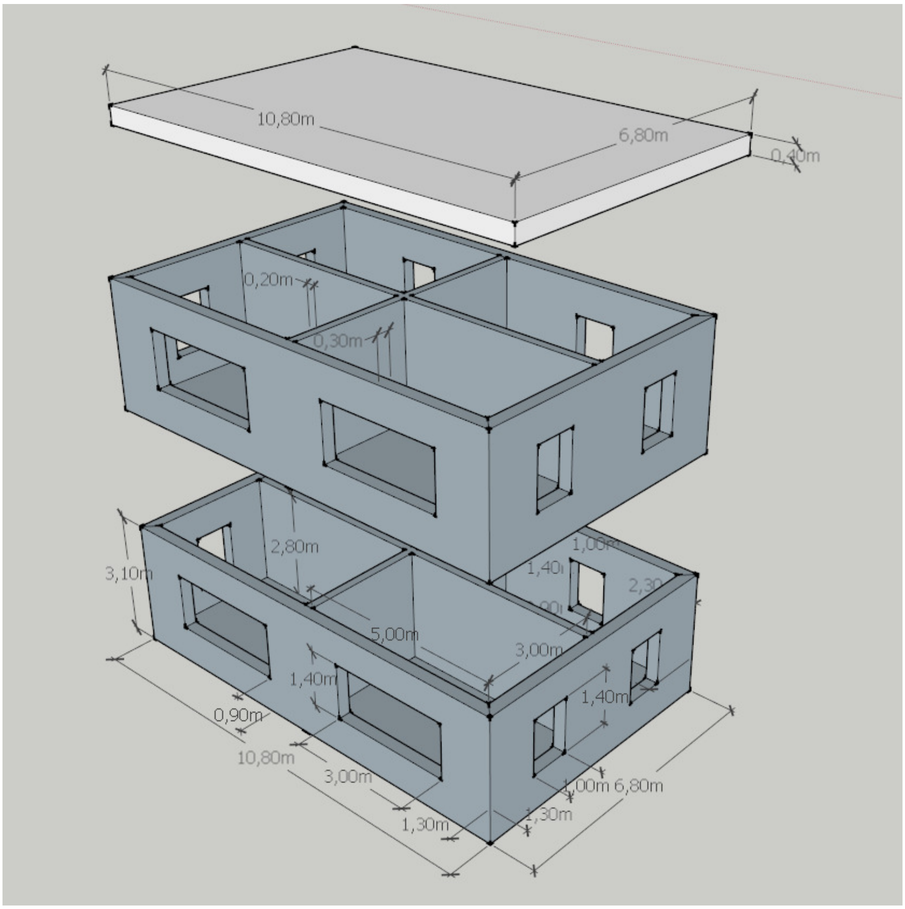


Fig. 9.39: *Geometry of the two stories residential building*

Table 9.3: *Materials of the opaque building elements.*

Material	$\rho\,[\mathrm{kg/m^3}]$	$c\,[\mathrm{J/(kg\,K)}]$	$\lambda\,[\mathrm{W/(m\,K)}]$
Concrete	2,000	1,000	1.35
Insulation	30,0	1,000	0.04
Gravel	1,800	1,000	0.7

Table 9.4: U-values of the opaque building elements.

Construction	U [W/(m^2 K)]	Layers
External walls	0.35	20 cm concrete, 10 cm insulation
Floor	0.35	20 cm concrete, 10 cm insulation
Roof	0.34	20 cm concrete, 10 cm insulation, 10 cm gravel
Inner constructions	3.14	20 cm concrete

9.4.4.5 Energy Supply System

The energy supply system consists of a warm water heating system with floor heating. An additional solar thermal plant with a fixed azimuth angle of 0° and tilt angle of 30° generates thermal energy, which is fed into a solar storage. The specific mass flow rate through the solar collector field is 40 kg/(m^2 h).

The return flow of the heating loop is warmed up by the solar storage. If necessary, the backup system (heater) adds thermal energy to obtain the supply water temperature set point of 45 ℃.

9.4.4.6 Control System

The solar loop is controlled by a simple on/off controller. If the output temperature of the collector is 4 K above the temperature in the lowest layer of the solar storage, and the temperature is under 100 ℃ (in order to avoid boiling), then the solar pump is switched on until the temperature difference falls under 1 K. For the space heating, a two-way valve adjusts the water flow rate.

9.4.4.7 Implementation and Parametrization of the Scenario in Modelica

An implementation of the scenario, based on the Annex 60 library, the Modelica standard library and two models of the BuildingSystems library (a collector model and storage model, which are not present in the Annex 60 library) was realized. Fig. 9.40 shows the system model.

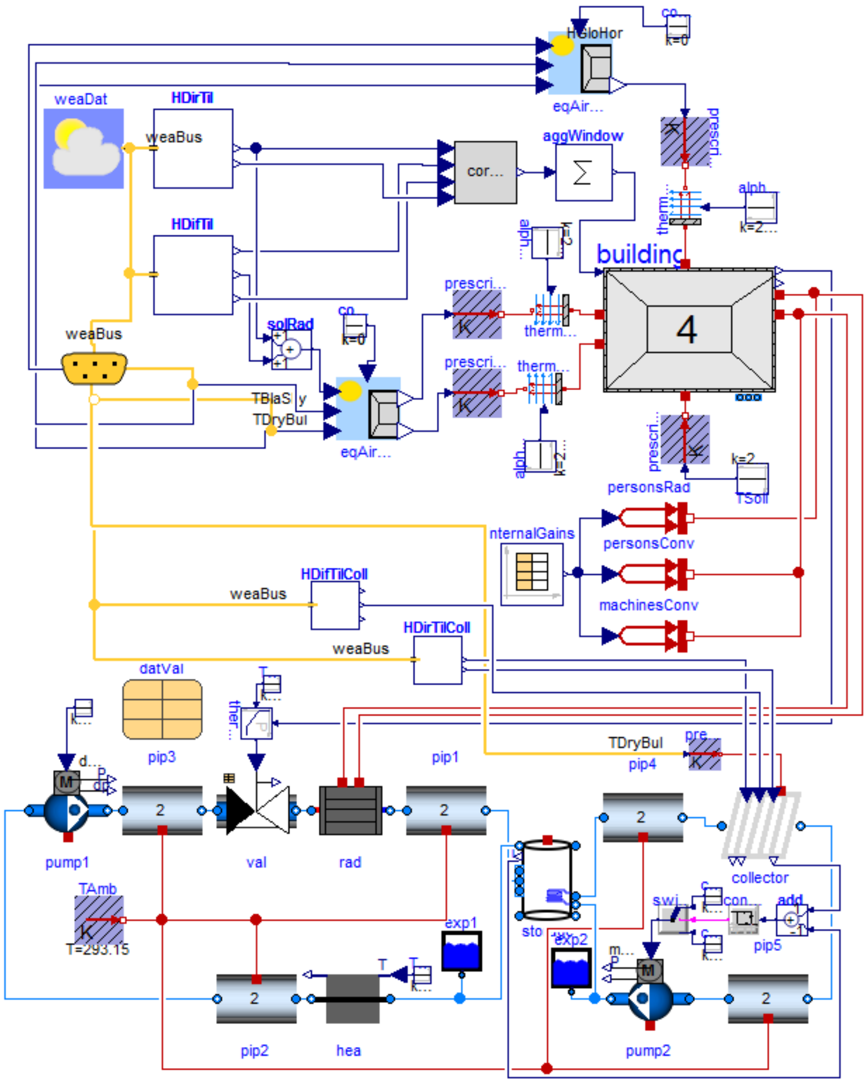


Fig. 9.40: Modelica model of the scenario.

The parameterization of the low-order building model was performed using the TEASER tool (https://github.com/RWTH-EBC/TEASER).

9.4.4.8 Optimization Problem

The following parameters were varied during the optimization:

- The thickness of the insulation of the exterior walls (from 6 cm to 30 cm),
- the volume of the solar storage (from 1 m^3 to 40 m^3), and
- the collector area (from 1 m^2 to 40 m^2).

The cost function consisted of components for labor and material costs for the insulation C_{ins}, component and installment costs for the thermal collectors C_{col}, component cost for the solar storage C_{sol} and energy cost for the needed backup energy C_{bac}. An additional penalty factor C_{pen} considers a desired solar covering rate of the solar heating system:

$$Cf = C_{col} + C_{sto} + C_{ins} + C_{hea} + C_{pen},$$

where

$$C_{col} = \frac{200\,A_{col}}{\tau col},$$

$$C_{sto} = \frac{1649.81\,V_{sto}^{-0.464}\,V_{sto}}{\tau_{sto}},$$

$$C_{ins} = \frac{A_{ext}\,(2.431\,d_{ins}\,100.0 + 87.35)}{\tau_{ins}},$$

$$C_{bac} = r\,Q_{bac},$$

and

$$C_{pen} = p\,(SF_{set} - SF),$$

where τ is the expected life time. Parameter optimizations with a energy prices r of 8 Euro cent per kWh and a penalty factor p of 0 Euro and 1,500 Euro were performed. A desired solar covering rate of $SF_{se} = 0.5$ was assumed. The life time was assumed to be 20 years for the solar collector τ_{col}, 20 years for the solar storage τ_{sto} and 30 years for the insulation τ_{ins}.

Because the optimization of the solar heating system has to be considered as a seasonal problem, each simulation run was performed over 2 years simulation time, where only the second year was evaluated for the cost function. The simulation starts from 1st of July with a start temperature for the solar storage of 60 ℃ for the entire water volume. All optimization runs were initialized with an insulation thickness of 0.1 m, a solar collector area of 20 m^2 and a volume of the solar storage of 10 m^3.

The optimization problem was solved with GenOpt, using the Particle Swarm algorithm in combination with the General Pattern Search algorithm. Dymola 2017 was used to evaluate the cost function.

9.4.4.9 Optimization Results

The colder climate of Chicago leads to a larger insulation thickness of 15 to 20 cm in comparison to San Francisco, where 6 to 7 cm insulation are optimal. If the penalty factor is set to 1,500 Euro, then the thickness of the insulation is increasing at the colder location and decreasing at the warmer location, as shown in Fig. 9.41.

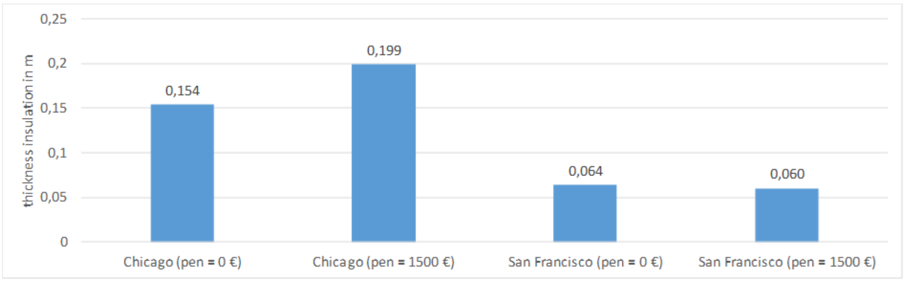


Fig. 9.41: Optimized insulation thickness for Chicago and San Francisco

A pure economic optimization (penalty factor = 0 Euro) leads to a solar coverage rate of 8.5 percent for Chicago and 19.6 percent for San Francisco, as shown in Fig. 9.42. With a penalty factor of 1,500 Euro, for both locations the solar coverage rate is about 50 percent, which minimizes the penalty term.

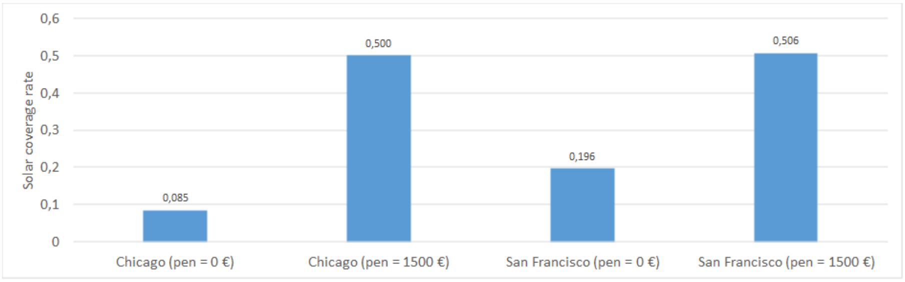


Fig. 9.42: *Optimized solar coverage rate for Chicago and San Francisco.*

The colder climate of Chicago causes a larger collector area than in San Francisco. If the penalty factor set to 1,500 Euro, then the collector area is drastically increased to reach the desired solar coverage rate, as shown in Fig. 9.43.

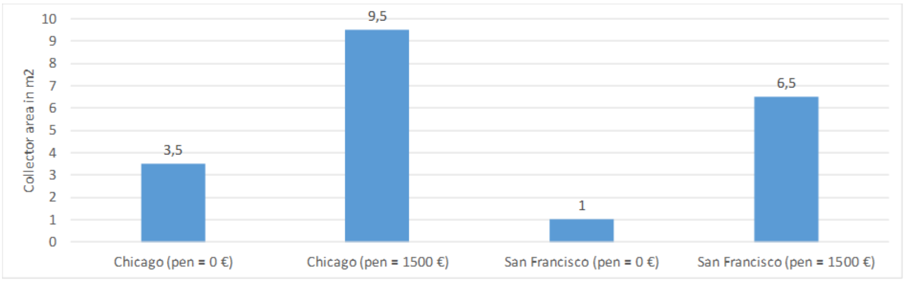


Fig. 9.43: *Optimized volume solar storage for Chicago and San Francisco.*

The optimized volume of the solar storage is 1 m^3, except for Chicago (about 30 m^3) if the penalty factor of 1,500 Euro is applied. A seasonal storage of solar energy from the summer for the heating period with a large storage volume is a precondition to reach the desired solar coverage rate, as shown in Fig. 9.44.

Fig. 9.45 shows that the required backup energy in Chicago is 6 to 8 times higher than in San Francisco.

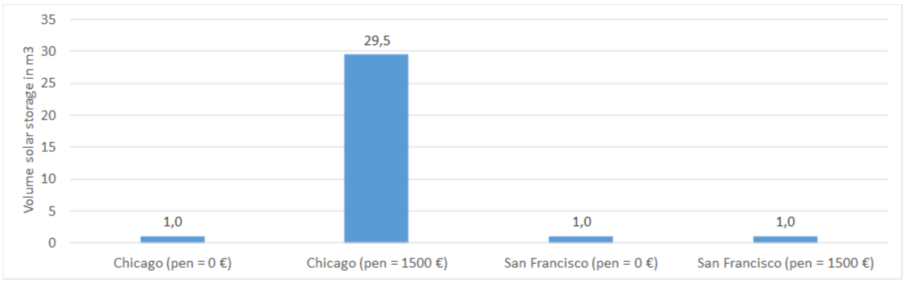


Fig. 9.44: *Optimized volume solar storage for Chicago and San Francisco*

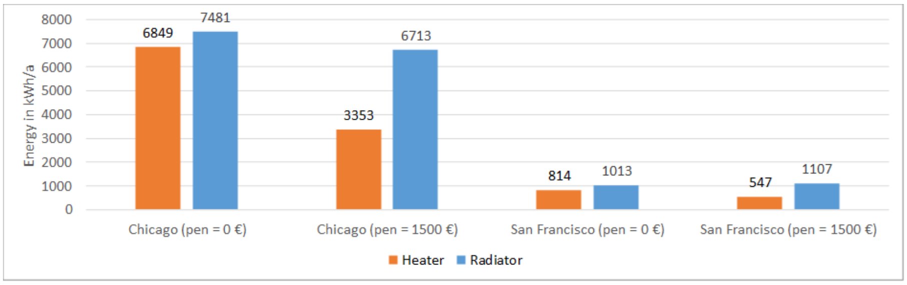


Fig. 9.45: *Optimized thermal energies for Chicago and San Francisco*

9.4.4.10 Computation Time

A single simulation run over two years took 33 seconds. GenOpt required between 25 minutes and 50 minutes.

9.5 Summary

The nine case studies from seven countries of Activity 2.1 have illustrated that Modelica building libraries are able to cover a wide range of building design and building optimization problems.

The authors of the case studies mentioned the following main advantages of Modelica: First, the modular approach of Modelica enabled them to model building systems at various levels of detail. Temporal and spatial resolution could be increased where needed to address a particular design question while abstracting other parts of the model using simplified equations. Second, Modelica's object-orientation and encapsulation through standardized connectors for heat flow and fluid flow allowed collaborative development in which sub-models of complex building systems were configured by different modelers and subsequently combined to form a system model. Third, missing models could be added quickly, either through new implementations, through combinations of existing models, or through use of base classes that implement conservation equations. Fourth, models of different domains such as heat transfer, fluid flow, electrical systems and control systems could be combined graphically to form an integrated model. The main reported drawback of Modelica was that annual simulations can be time-consuming if detailed HVAC models are combined with multi-zone building envelop models. This is largely due to the solvers that are currently implemented in the Modelica simulators. As Modelica is translated to efficient C-code (see Section 5.3.4), this is not an inherent problem of Modelica, but rather a questions of what solvers to use as users simulate ever larger models. To obtain fast simulations, component parameters have to be adjusted and the right solver and its tolerance have to be chosen in the Modelica simulation tools *[JWH15]*. Moreover, detailed control algorithms can slow down the simulations performance, in particular if they generate many *events*. Addressing these problems is an active field of research, see for example Section 5.2.

Two examples could demonstrate the application of optimal design and optimal control problems based on Modelica libraries: A multi-criteria parameter optimization of PV cooling systems for residential buildings and a non-linear optimal control problem for the HVAC and facade control in a thermal zone. For the latter, the computer algebra conducted by the Modelica tool when it constructed the optimization problem, based on the declarative model formulation, led to 2,200 times faster optimization compared to a derivative free optimization method that simply evaluated the cost function iteratively.

A final building simulation scenario for optimal design summarizes the work of Activity 2.1: A parameter optimization method was exemplary applied to a Modelica building system model that couples a solar thermal system to a simplified building model. This building system model is almost entirely composed of components from the Annex 60 library and the Modelica standard library. A two-year simulation of this model took half a minute. Two optimization variants that optimize life cycle cost using weather data for a moderate climate and a very cold climate were performed.

Chapter 10

Activity 2.2: Design of District Energy Systems

10.1 Introduction

Recent developments to reduce the energy use of buildings focus on the integration of renewables and on energy efficiency. European legislation enforces that by 2020 all new buildings are nearly Zero-Energy Buildings (ZEBs) and it requires the deployment of a European Smart Grid. The shift in the energy infrastructure leads to new requirements and technical challenges as the interaction of buildings becomes increasingly important. To quantify the required interaction among buildings and the grid, as well as the restrictions caused by the existing neighborhood topologies and grid configurations, an integrated modeling environment is needed. This activity focuses on the use, verification and demonstration of Modelica libraries from Activity 1.1 and co-simulation tools for multi-scale simulation from Activity 1.2 to assess District Energy Systems (DES). The focus of this activity extends the traditional focus on individual building performance to DES. The analysis of these systems requires an integrated simulation platform that is capable of simulating simultaneously the energy demand of multiple buildings, the thermal and electrical energy distribution grids and the control systems.

The work that was carried out in this activity consisted of three main parts:

1. A literature study focusing on the modeling of DES
2. The development of a common exercise to model DES within Modelica environments
3. The use of co-simulation techniques to model DES

Section 10.2 presents a description of the main components typically found in DES. Section 10.3 addresses the modeling of multi-scale energy flows in districts and presents an overview of simulation tools and environments to model these systems. The overview discusses both Modelica and non-Modelica implementations. Section 10.4 reports on the common exercise that has been set up to analyze the capabilities of the current Modelica libraries of the Annex 60 participants and provides a first step to develop a comparative model validation test for district energy simulations: a DESTEST. In this exercise, a relevant simulation case (including building definition, occupant behavior, grid definition etc.) is developed for two purposes. Firstly, the set can be used to analyze relevant research problems such as how to design district heating systems and how to integrate renewables in districts. Secondly, the information and challenges gathered during the development will be used for setting up a framework to test District Energy System simulation tools. Although the simulation of DES within one software environment has many advantages and makes it a desired solution, scalability of the numerical methods to large models are still a challenge if one solver is used for the whole system of equations. Therefore, in Section 10.6, the opportunities and threads for co-simulation techniques related to DES simulation are analyzed. Finally, Section 10.7 presents conclusions and perspectives.

10.2 Description of District Energy Systems

A DES comprises all components that enable the delivery of energy services in a district. This typically includes electricity, gas, heat or cool, possibly at cascading temperature levels. However, in this section, modeling of gas networks are not discussed.

10.2.1 Electrical Versus Thermal Energy

Although the generation of electrical and thermal power have always been closely related, at least in power plants relying on combustion, their respective carrier grids have experienced a separate evolution. Fig. 10.1 shows an overview of commonly used technologies and their respective energy sources.

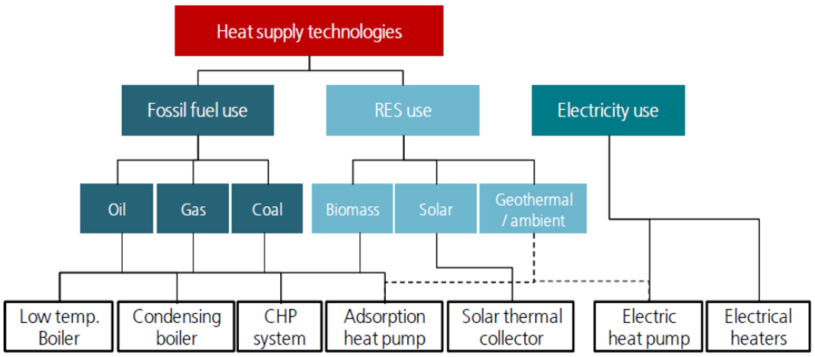


Fig. 10.1: Overview of heating technologies and their respective energy sources. Source [Com16b].

The International Energy Agency reports *[Age14]* that worldwide, electricity has the largest final energy consumption by far. Fig. 10.2 shows the produced and consumed annual electric energy and heat worldwide in 2014. However, note that a substantial part of electric energy is also used to meet heating and cooling demands by use of electric heaters and boilers, heat pumps and chillers. With a trend towards installation of more heat pumps, it can be expected that the electric energy fraction for the building sector will increase compared to the heat fraction.

The previous section considers all end uses of electricity and heat. When looking at the categories of energy use, it can be seen that both forms of energy have approximately the same fraction of energy use in buildings (considering residential and community and public services to correspond to the energy demand of buildings), as can be seen in Fig. 10.3.

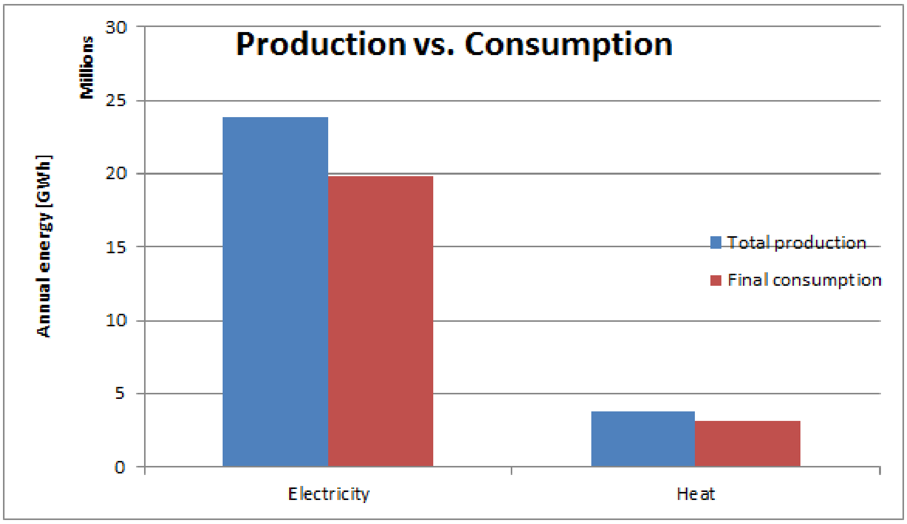


Fig. 10.2: Worldwide electricity and heat production and consumption in 2014. Source: [Age14]

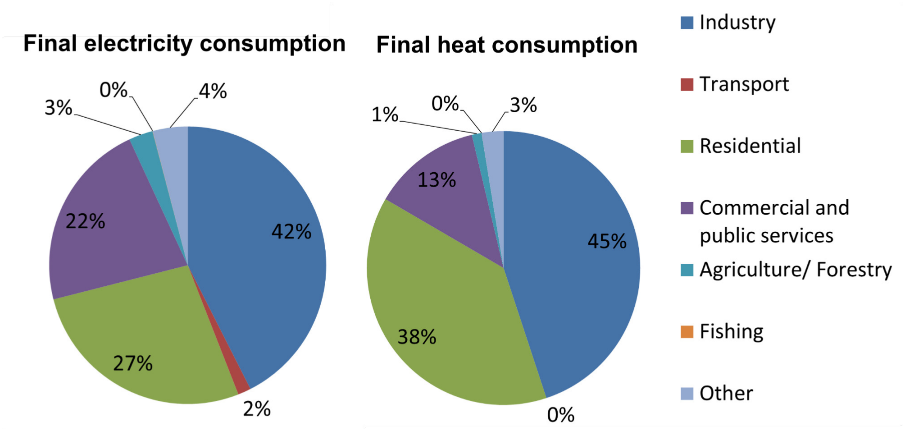


Fig. 10.3: End use fraction of electric and heat energy for different sectors in 2014. Source [Age14]

In the European Union in 2012, 50% of the primary energy use originated from building heating and cooling demand [Com16a], making it the largest energy sector. 18% of the primary energy comes from renewable energy sources. Biomass holds the largest share for heating only, namely 90% of renewable energy sources. [Com16b] further illustrates the end uses and sources of heating.

Part of this heating demand is provided through district heating and cooling networks. The share of district heating in the residential sector in the EU was 9% in 2012, and 8% for the industrial sector [Com16b]. For some EU countries, almost half of the national heat consumption is provided through district heating (Sweden, Denmark, Lithuania, Estonia and Finland).

On the other hand, district cooling systems are mostly used for office buildings. However, district cooling has not gained the same attention as district heating. This can be derived from the projected potential of both concepts in the EU Strategy for Heating and Cooling [Com16b] and the absence of district cooling in the Heat Roadmap for Europe 2050 [CNP13]. The installed capacity is less than 1% of that of district heating systems [Com16b].

There are more than 150,000 km of pipes in the district heating systems in the European Union [Com16b]. Denmark and Sweden have the most extended network, followed by Poland and Germany. However, the average system length varies greatly, signifying a large variation in the number of thermal networks in each country versus its size. Looking at the percentage of the population served by district heating systems, the lead is taken by Denmark and the Baltic states. On the other hand, the final energy use in district heating is by far, with twice as much energy as the runner-up, the largest in Germany. On a European average, almost half of the demand in district heating systems is met by dedicated boilers, while 42% comes from CHP installations. Only 5% is currently provided by directly recycling waste heat. Hence, the list of primary energy sources is still dominated by natural gas (40%) and coal (29%).

10.2.2 State-Of-The-Art in District Energy Networks

In the 1880s, the first district heating systems were installed, powered by steam [LWW+14]. In the meantime, district energy technology has evolved and matured. Three trends can be identified:

- a decreasing supply temperature for heating, and increasing supply temperature for cooling,
- a decreasing size and amount of material used for its components, and
- an increased share of standardization and prefabricated components.

Extending these evolutions, Lund et al. reason that after three generations of district heating systems, the next step must be the fourth generation with even lower heating supply temperatures and an increased focus towards the integration of renewable energy sources. Examples of mainly solar powered DES can be found a.o. in Canada (Drake Landing Solar Community *[SMD+12]*) and Denmark (Marstal district heating system *[PS13]*)

The Swiss Competence Center for Energy Research is exploring an even further step forward with bi-directional thermal networks *[SKS15]*. Much like the electrical grid can convey energy both from a centralized generator to a consumer and back from a rooftop PV into the grid, a bi-directional thermal network uses a single circuit for both heating and cooling. When in net more cooling is needed than heating, the system circulates from a central plant in one direction. When more heating is needed, the system circulates in the opposite direction. In the case of a balanced system, no water flows through the central plant. Thus, in bi-directional systems, buildings can recover heat from each other directly.

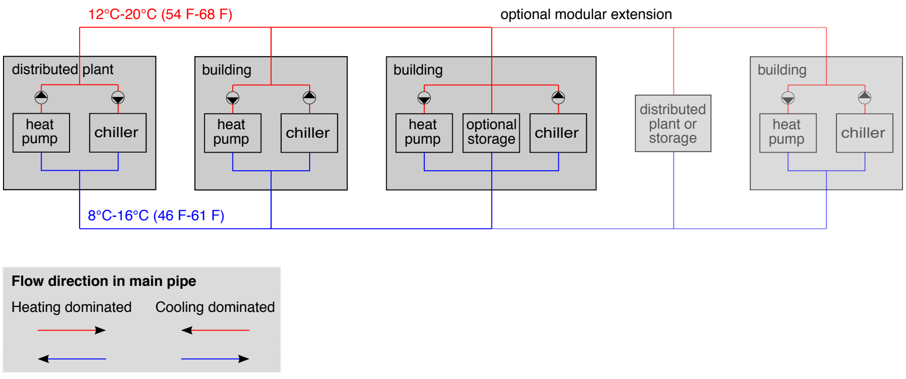


Fig. 10.4: Bi-directional thermal network.

Fig. 10.4 shows an example schematic of a bi-directional thermal network. The plant guarantees that the hot side is kept between 12°C to 20°C, while the cold side is kept between 8°C to 16°C. The system uses near-ambient temperatures to maximize the efficiency of heat pumps. Unlike earlier generations of district heating, the bi-directional system need not be operated to serve the lowest and highest temperature needs. Rather, each individual building is equipped with heat pumps, allowing to locally boost the temperature up or down to the temperature required by the building. The system has the benefit of being modular extensible, allowing adding buildings, heat sources, heat sinks and storage without having to oversize the initial system.

Regarding electrical energy network at the district level, distributed generation by means of PV panels or micro CHP units is becoming more prevalent. However, in the low voltage grid, care must be taken that the voltage stays within an acceptable range. This becomes increasingly difficult with more distributed generation. On the other hand, electrical demand is also expected to go up in the very near future, with increasingly more air conditioning units, heat pumps and electric vehicles.

The integration of all these new technologies necessitate smart control strategies such as demand response, transactive controls or curtailing PV production in case of overproduction, in order to guarantee adequate system operation.

10.2.3 Components Typically Encountered in District Energy Systems

Next, we will describe the components that are typically encountered in thermal and electrical district energy systems

A general overview of thermal components typically encountered in district heating and cooling systems is given in *[FW13]*. Below, a short summary of components is provided. Models for these components must be available in a district energy simulation library.

Pipes transport heat (or cold) from production sites to consumers in the grid.

Substations separate the primary (high pressure) network from the secondary (customer) side hydraulically. This is usually done by using heat exchangers to contain the consequence of leakages at the customer side.

Control valves regulate mass flows in different parts of the network. They regulate the flow that each customer receives and the bypass from the supply to the return pipe. The goal is to maintain the correct temperature in all parts of the network: high enough supply temperature, and a return temperature that is as low as possible *[GW14]*.

Buffer tanks store hot or chilled water for later use and hence limit peak power.

Long term storage allows shifting energy for a long-time, usually seasonally. They include borehole fields or acquifers.

Pumps and pressurization provide a pressure difference such that the water flows from the supply to the return side. For high temperature networks, they also prevent the water from boiling.

At the heat and cold production side, boilers, combined heat and power units, heat pumps or chillers are encountered. Furthermore, solar thermal boilers can be used to directly incorporate solar thermal energy.

For the simulation of electrical consumption in DES, typically only the low voltage grid is considered. Power from the high voltage grid is usually assumed to be available, but local power injection from decentralized production such as photovoltaics is possible. Hence, the components needed for electrical simulations are distribution lines (3-phases and neutral), transformers, power converters such as rectifiers, inverters and DC/DC converters, batteries, and PV panels.

10.2.4 Energy Demand

We will now characterize the thermal and electical energy demand.

The thermal energy demand consists of the demand for domestic hot water (DHW), space heating, process heat and cooling. The conditioning of indoor spaces can be provided with a large variety of systems, such as convectors, radiators, variable or constant air volume units, radiant ceilings and floors. The choice of the emission system determines the temperature at which the heat or cold is required.

The energy demand is largely determined by three aspects: the occupants, the weather conditions and the building characteristics.

Occupants influence the energy demand in the following ways:

- Desired temperature set point or comfort band in the heated and cooled spaces.
- Internal gains directly from occupants.
- Internal gains from appliances used by occupants (e.g., cooking, lighting, electronic devices).
- Domestic hot water demand.

These processes are characterized by high levels of uncertainty. One way to cope with this uncertainty is to model the energy demand stochastically. A software tool to calculate these uncertainties is presented by Baetens and Saelens *[BS15]*.

The weather conditions outside of the modeled buildings determine the heating or cooling demand. Here, mostly solar irradiation and ambient temperatures are prevalent influence factors.

Related to this, the building characteristics such as the level of insulation, aspect ratio and building orientation determine how large the influence from the aforementioned weather conditions on the indoor climate is.

Electricity is needed for lighting and for powering home appliances. It can also be used to power thermal systems, such as heat pumps, cooling machines (air conditioning) or electric heating. Furthermore, electricity is needed to power pumps in the different parts of a thermal network and fans in the ventilation systems. Moreover, electrical cars cause an additional load but also an opportunity for electrical storage.

10.3 Modeling of District Heating and Cooling Systems

10.3.1 Introduction

DES models can be distinguished between physical domains, geographic scales and time scales.

Various modeling tools for DES exist. Many of the existing tools have limited support for modeling integrated energy systems. In tools that have detailed building and HVAC models, the study of electricity systems is idealized, often not allowing to assess the impact of renewable generation on the voltage stability. Other tools focus on medium or high voltage grids, but use simplified building and HVAC models for a very high number of households, sometimes not allowing to determine the required temperature levels.

Fig. 10.5 compares time scales and simulation objectives for district heating and cooling systems. Simulation can be used for time range of seconds for assessing water hammer effects to years for network design and investment planning.

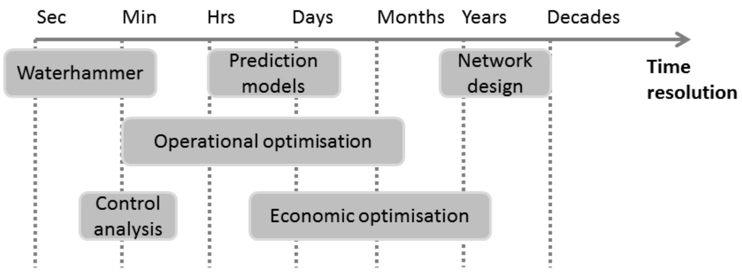


Fig. 10.5: Time considerations depending on the focus of the study.

10.3.2 Modeling Methodology

Modeling and simulation of district energy systems must cover different levels of detail and a large application range from components sizing to network topology design. In this section, a generic methodology for modeling a large scale system is presented in order to assess performances of DES, such as energy demand or efficiency as well as operational and control strategies. Three main steps can be highlight:

Grid and network modeling. The grid topology provided by the operator is aggregated and modeled into the simulation tool. The dynamic thermo-hydraulic behavior of a grid model is carried out with numerical models of the simplified and/or aggregated DES. Grid models include the implementation of the grid topology and pipe models.

Occupancy modeling. In the residential sector, customers needs consist of a combination of space heating or cooling, domestic hot water and electrical demand. Space heating or cooling can be based on models parametrized with physical properties of the buildings or regression functions on the basis of monitoring data. Electrical and domestic hot water demand are usually derived from statistical models.

More details on the determination of typical load profiles can be found in Section 10.2.4. Internal heat gains are deduced from occupancy profiles and activity probability functions. Process heat demand for industrial usage and/or for absorption chillers can be considered, but is not covered in this section.

Supply modeling. Heat plants and decentralized production are identified based on data provided by the grid operator and technical specifications. Operational strategies and control need to be implemented to balance energy and meet specifications.

Fig. 10.6 shows a schematic of all components to be modeled and how they interact.

In order to facilitate both the process of modeling and the post-processing of simulation results, geographic information system (GIS) tools are commonly used. Detailed descriptions of the single components are reported in the following sections.

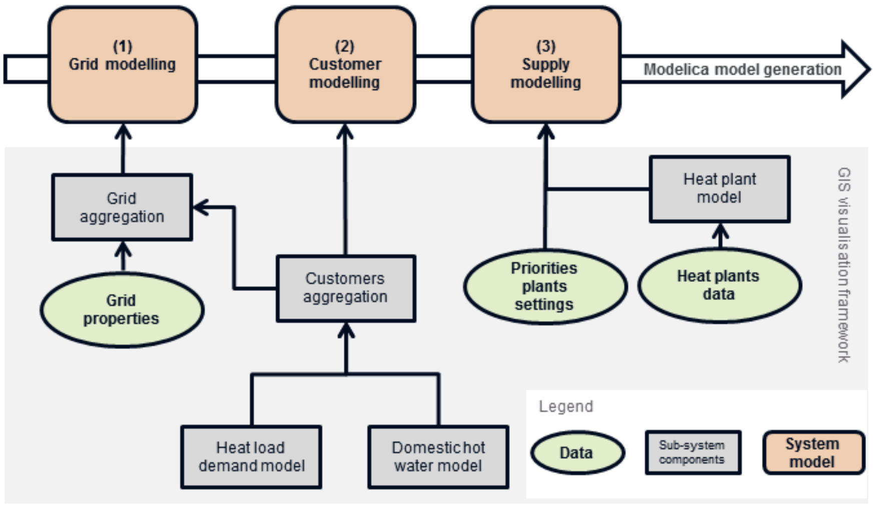


Fig. 10.6: *Methodology flow diagram: example for a large district heating network*

10.3.3 Data Acquisition

An increasing number of cities are creating 3D virtual city models for data integration, harmonisation, storage and visualization. Such an urban data model can provide a range of benefits as it can represent an information hub for advanced applications ranging from urban planning, noise mapping, augmented reality up to energy simulation. To this extent, CityGML is an international standard conceived specifically as information and data model for semantic city models at urban and territorial scale. It models principal urban features both geometrically and semantically, including buildings, land use, terrain and vegetation. For cities, a structured database for district energy system analysis should contain:

1. district network information such as topology, pipes characteristics, storage and production specifications,
2. building attributes such as building id, address, type (e.g. single family house), use, year of construction, year of refurbishment and total heated floor area.

Relevant parameters therefore can be extracted from the city model and aggregated to construct a district energy model.

10.3.4 Problems and Challenges

As part of any modeling work, challenges and problems might be encountered in the whole process, from the digitalization to the final assessment of network performance. The following list is the result of an Annex 60 survey in terms of problems encountered from the participants in their activities.

One of the main interests in modeling district heating networks is to simulate the rate of energy transport through the system. This transport is not only dependent on the water flow through the system, but also on the temperature levels in the network. Variations in temperature in a district heating system are mostly due to three phenomena *[Pal00]*:

- Changes in supply temperature, controlled by a system operator. These variations can be large and fast.
- Return temperature changes due to heat or cold transfer at customer substations. These variations are usually rather slow, but fast variations due to domestic hot water preparation are possible.
- Heat transfer between the pipes and their environment.

Temperature drops due to heat loss or infiltration in pipes are quite stable. However, the changes in supply and return temperatures affect the transient heat losses and the heat storage in the pipes. Heat losses represent the main source of energy losses in high temperature district heating systems. For the latest generation of district heating systems, due to the low supply temperatures, friction losses gain an increasing importance in the overall energy losses *[LS12]*.

There is an important difference between flow and temperature dynamics *[FW13]*. Changes in the flow are quickly transferred to the whole network in form of pressure waves, typically in matter of seconds. Changes in temperature, however, are transferred relatively slowly, in relation to changes in the flow. Thus, the response time from a production plant to a consumer station, with respect to temperature changes, can be up to several hours.

Bi-directional thermal networks require a higher level of physics modeling capability than previous generations of district heating and cooling systems. Because the fluid flow direction can, and likely will, reverse, the associated dynamics need to be captured. This is difficult to accomplish in existing building simulation programs as they are not designed to handle situations where the flow direction changes *[KK12][KKG+14][KSS15]*. In contrast, the Modelica stream connector *[FCO+09a]* is designed to handle flow reversal, and does so in a numerically efficient way. Also, the thermal capacity and location of pipes should be accounted for, in particular if the flow reverses for a few hours daily in which case energy storage in the pipes can become important. Also, the accuracy of modeled system temperatures is more important when heat exchangers are designed to operate over changes of only a few Kelvin. An inaccuracy of 1-2 K could result in a different flow direction and large error in analysing the potential for waste heat recuperation or free cooling potential. The decentralized controls associated with directing flow at junctions, choosing temperature set points, and turning distributed generators on/off are much more intricate than with centralized, uni-directional systems. Furthermore, because the thermal distribution losses are low and the energy input for the heat pumps is low compared to that for a natural gas-fired boiler, the relative importance of flow friction and associated energy use of pumps and fans are more significant.

10.3.5 District Heating and Cooling Modeling Approaches

10.3.5.1 State of the Art

There are two basis methods used for dynamic modeling of district heating networks *[LPBhmR02]*:

1. Physical models that model the physical structure and behavior of the system, and
2. statistical models that use a statistical black-box approach.

Typically, only the energy equations are modeled dynamically, while the pressure and momentum equations are simplified as steady-state.

10.3.5.2 Physical Models for District Heating and Cooling Systems

Physical models represent the physical relations among subsystems. All the important components in a district heating network are explicitly modeled, and the whole structure of the system is taken into account. It is easy to change and add components to an existing model.

Based on the physical approach, Zhao [Zha95] and Benonysson [Ben91] compared methods for quasi- dynamic modeling of district heating networks, which are referred to as the element method and the node method. The element method was found to be inferior to the node method, both with respect to accuracy and computational cost. On the other hand, the node method does not exhibit difficulties due to numerical diffusion. Although the node method has been found to be faster and more accurate in some cases, the element method seems to be more frequently applied in practice [BhmHK+02].

Next, we will describe the element method and the node method.

In the element method, computations are based on a temperature profile along the pipe. The pipe is divided into a number of elements in the axial direction. Each element is typically divided into four sections for the water, the steel pipe, the insulation materials and the cylinder of ground surrounding the buried pipe. Radial heat transfer is computed by assuming a single uniform temperature in the water and the steel, and uniform temperatures are assumed in the insulation and the a cylinder of soil that surrounds the pipe. In this case, the temperature at each time step is determined only from the time step before. When using this method, it is necessary to take the Courant number into account:

$$Cou = \frac{\Delta T\, u}{\Delta x}$$

where ΔT is the time step, Δx is the element length and u is the flow velocity. The Courant number should be less than or equal to one for accurate results. Usually, when the Courant number is equal to one, the model gives the exact result. Large numerical diffusion occurs as the Courant number approaches zero. Numerical diffusion is an artifact of the discretization, and the effect is as if the medium had a higher diffusivity. The temperature of the steel pipe is assumed to be equal to the water temperature since the thermal conductivity of both materials is relatively large compared to the thermal conductivity of the insulation and surroundings. Heat conduction in axial direction is usually neglected.

In the node method *[Ben91]*, only the inlet and outlet of the pipe are taken into account, as opposed to the discretisation in the element method. The overall heat transfer coefficient is based on the method of thermal resistances, but the heat transfer between neighbouring pipes is not taken into account. Computations are based on a time history of temperatures and flows, taking into acount the propagation delay between inlet and outlet for the whole pipe. The thermal capacity of the pipe wall is modelled in an aggregated way for each pipe segment, while that of the insulation and the surrounding ground is not taken into account. This assumption is justified by the lower thermal capacity in these regions and the limited temperature variations as a result of the lower conductivity of the insulation.

10.3.5.3 Statistical Models for District Heating and Cooling Systems

Statistical models based on time-series or neural networks tend to be computationally simpler, but their main drawback is that they have low degree of accuracy and many non-physical relationships. In addition, they require the availability of large amounts of measurements for model estimation and validation *[Pal93]*.

The classical time-series method depends on the assumption of linearity and stationarity, either in the raw data or as a result of some well defined and invertible transformation of the raw data. While those assumptions simplify modeling and enable a coherent use of statistical measures, they also limit the applicability and place demand on proper pre-processing of data. Furthermore, methods based on artificial neural networks can also be used.

10.3.5.4 Modeling of District Electrical Network

An electricity grid consists of different nodes connecting the different generators and loads through cables. For district network application, low-voltage (LV) distribution grids have mostly a radial topology. Thus, there is one feeding transformer for each set of distribution feeders. In the presence of distributed energy resources, bi-directional electricity flows are possible.

Redundancy is possible through particular grid topologies, such as ring and mesh topologies. In these topologies, the loads are supplied by more than one medium voltage distribution grid. In Europe, most LV distribution grids (230/400V) are radial, three-phase grids. In case of a ring or meshed grid, all LV grids are operated in radial configuration.

Different types of grid simulations exist, ranging from electromagnetic to quasy-stationary and steady-state modeling. Electromagnetic effects occur during events such as when a switch opens or closes or lighting strikes the network. In quasi-stationary models, the frequency is modeled as quasi-stationary, assuming a perfect sine wave with no higher harmonics. Voltages and currents are considered as sine waves and just their amplitudes and phase shifts are taken into account during the analysis. With such an assumption, electric quantities can be represented with a phasor, i.e., a vector in the complex plane. A variation to the quasi-stationary models are models with so-called dynamic phasorial representation. The basic idea of the dynamic phasorial representation is to account for dynamic variations of the amplitude and the angle of the phasors. With such an approach, it is possible to analyze faster dynamics without directly representing all the electromagnetic effects and high-order harmonics. For more details, see *[SLA99]*, *[SA00]*. In steady-state models, the time derivatives are eliminated.

In electricity grids, the power flow analysis is performed for each node and line based on the Kirchhoff laws:

1. Kirchhoff's node rule: the sum of currents flowing into a node is zero.
2. Kirchhoff's loop rule: the directed sum of the potential differences in any closed loop is zero.

For fully balanced systems, no current flows in the neutral conductor. Therefore, an equivalent single-phase grid can be used to simulate the balanced three-phase grid.

10.3.6 Applications of Modelica in District Energy System Modeling

As illustrated in previous chapters, Modelica has often been used for modeling DES, in particular for district heating networks. In this chapter, publications are presented which have integrated thermal, electrical and control Modelica models for simulation and/or optimization of DES. The use cases demonstrate the capabilities of Modelica for this purpose, while weaknesses are tackled by using a combination of tools and co-simulation approaches. The majority of Modelica libraries employed in these studies have been introduced in Section 5.2.

10.3.6.1 Buildings

Simulation of a DES usually involves numerous buildings that consist of heterogeneous structures, heating systems, occupancy behavior, etc. First, it introduces modeling difficulties linked to the data availability, the large number of sub-components, bias and validity of the final model. Secondly, it introduces scalability issues, related to the simulation time with respect to the number of buildings. In that sense, component-based and oriented-object modeling languages such as Modelica offer significant possibilities as the model fidelity can rapidly be changed and the symbolic processor can simplify equations and can provide analytic derivatives for the numerical solvers.

Several papers propose the use of simplified building models for simulation and optimization of district energy demand, as they can significantly reduce the computation time. Kim et al. *[KPHR14]* used physical simplifications and model order reduction based on detailed Modelica building models to create simple models for urban energy simulations. The performance of the reduced models was evaluated for annual and hourly heating and cooling loads, showing satisfactory results. Nytsch-Geusen and Kaul *[NGK15]* described a method to simulate district heating and cooling demand for residential buildings, combining low-order thermal models from the `BuildingSystems` Modelica library *[NGHLR12]* with statistical distribution functions for the building stock characteristics. Lauster et al. *[LFT+13]* also presented a tool chain for district heating demand simulation. Typical office building data sets based on statistical data were used to automatically parametrize simplified dynamic building models implemented in Modelica. The method has been applied to simulate the energy demand of a research campus, performing well at an aggregate scale for annual demand, as compared to measurements. For this purpose, they later developed adaptive low order building models, to be used in different levels of required discretization, from single building to districts *[LFH+15]*. These are also automatically generated making use of statistical data. City information models (CIM) and geographical information systems (GIS) have been used to support district energy simulation and optimization. The TEASER tool for automated building heat demand simulations based on GIS-extracted data was developed in RWTH Aachen University. Schiefelbein et al. *[SJL+15]* described the development of a CIM, based on TEASER and the `AixLib` Modelica library *[BM10][FCL+15][RE15]*. Fuchs et al. *[FTL+16]* presented a tool called *uesmodels* to automate district energy simulations using Modelica and Python as a workflow automation tool. Previously, a similar approach was used in *[SJD+14]*, where a city district database was designed and linked to GIS to enable energy optimization of city districts. An interface then automatically generated building models for a simulation environment with Modelica and NEPLAN. The planning tool was verified with German Synthetic Load Profiles (SLP) and measurement data. Nouvel et al. *[NBK+15]* presented the work of an international collaboration, developing the energy Application Domain Extension (ADE) for the CityGML information model. This ADE is used to manage and store data in 3D city models that are related to building energy flows, such as building envelope and energy systems characteristics and occupancy. Coccolo et al. *[CMK15]* showed the connection of the CitySim simulation tool to the CityGML urban information model with the Energy ADE, applied to the EPFL campus.

10.3.6.2 District Heating and Cooling

As explained in Section 10.3.4, the Modelica language allows a higher level of physics modeling capability, especially for bi-directional thermal network. Moreover, a multi-domain language allows considering flow friction and associated energy use of pumps and fans, which are important in networks with small temperature differences.

In that context, many studies employed Modelica to simulate and analyze district heating systems. Basciotti et al. *[BSKD14]* presented a decision support methodology, where substations with different hot water preparation principles are compared based on several performance indicators. In *[BS14]*, they investigated demand and supply-side measures to reduce peak loads in the network. They found large consumer load shifting and utilization of the network as storage to be the most economic feasible measures. Köfinger et al. *[KBS+16]* evaluated the feasibility of low-temperature district heating implementations in Austria using four case studies in which the ecological, economic and energetic performances were assessed. Models based on the `Modelica.Fluid` *[COP+06]*, `Buildings` *[WHMS08][Wet09][WZNP14]*, and the `DisHeatLib` *[BP11]* libraries were used in the above studies.

Giraud et al. *[GBP+15]* assessed control strategies to reduce distribution losses in a virtual district heating network for which they developed and validated Modelica models. A 10% loss reduction was achieved using a dynamic supply temperature algorithm instead of a fixed heating curve. Based on low-order `AixLib` building models and the Modelica Standard Library (`MSL`) pipe models, Fuchs et al. *[FDT+13]* evaluated the impact of building retrofit on supply and return flows of a small network branch. They argue that benefits from building retrofit should be evaluated taking into account also the positive impact at the district level. In *[RTFC+15]*, Modelica was used to simulate and assess potential benefits of seasonal aquifer thermal energy storage in the district heating system of a university campus in Germany, showing promising results. Low-order building models from `BuildingSystems` library, `Annex60` library, `Buildings`, `MSL` and own models were used. Ljubijankic et al. *[LNGU09]* described the application of the `FluidFlow` Modelica library to design a heating and cooling energy supply system in a newly built residential area in Iran. This library was specifically developed to simulate the thermo-hydraulic behavior of complex energy systems. Giraud et al. present a comparison of Modelica with TRNSYS for a solar district heating system *[GPB15]*.

The advantage of Modelica as a modeling language has been illustrated in several applications. Burhenne et al. *[BWE+13]* reviewed Modelica libraries and demonstrated their capabilities for building performance simulation through examples. Simulations of the building envelope, HVAC system or an entire district heating system were shown, and also co-simulation cases were explored. In the district heating simulation example, the authors compared heat losses for different supply strategies. Schwan et al. *[SULK14]* demonstrated the use of the Modelica `GreenBuildings` library *[USM+12]* for district energy system simulations, in an example historical town center. Soons et al. *[STHS14]* modeled a biogas-powered low-temperature district heating system with and without thermal energy storage for the university campus in Eindhoven, illustrating the improved accuracy and level of detail offered by Modelica. On a slightly different topic, Casetta et al. *[CNB+15]* developed models for district cooling systems in Modelica, using `Buildings`, `MSL` and own models. They performed a scenario analysis varying the secondary return temperature of the substations in a simple district cooling network.

Modelica tools also allow the export of models using the FMI standard for subsequent co-simulation with other tools, as described in Section 6. Huber and Nytsch-Geusen *[HNG11]* described the different possible approaches in modeling urban districts. They presented a use case in which a large urban area is modeled, combining EnergyPlus for thermal building simulations with a detailed plant and distribution network in Modelica, coupled within the Building Controls Virtual Test Bed (BCVTB) *[Wet11a]*. Elci et al. *[ENKH13]* also combined Modelica with other software. They presented a district heating case study in which the influence of demand change due to refurbishment is investigated. The study employed simplified building and district heating models in Modelica, and the statistical modeling environment GNU R for heat generation and storage models.

10.3.6.3 Electrical Network

Modelica also provides the possibility to simulate electrical networks. The `IDEAS` library *[BDCJ+15][Ope15]* has been used to study the integration of heat pumps (HP) and photovoltaic (PV) systems in Belgian residential low-voltage feeders. Baetens et al. *[BDCVR+12]* simulated the IEEE 34-Node Test Feeder with net zero-energy dwellings, in order to assess electrical challenges at the neighborhood level. It was shown that the dwelling's self-consumption depended significantly on the grid capacity, as PV generation curtailment would occur due to voltage rise. For the same feeder, the potential of rule-based Demand Side Management (DSM) on domestic hot water production with HP was evaluated, showing promising results *[DCBS+14]*. To analyze the impact of low-energy dwellings on a range of feeders, several grid impact criteria were studied in *[BS13][Bae15]*, evaluating in particular the required grid reinforcements resulting from grid constraint violations. Protopapadaki et al. *[PBS15]* further analyzed correlations between grid impact criteria and building and neighborhood properties, finding significant relationships. Furthermore, `IDEAS` was employed in *[VR15]*, where strategies are evaluated to alleviate the impact of electric vehicles on the distribution grid. For all above studies, buildings, thermal systems, electric vehicles and grid were all simulated within the Dymola environment.

Bonvini et al. *[BWN14]* validated the electrical models of the `Buildings` library using the IEEE four nodes test feeder, and subsequently applied the models to demonstrate how to study the integration and control of renewable energy sources in an electric distribution grid. Bonvini et al. *[BW15]* demonstrated the use of Modelica models for gradient-based optimal control of batteries and HVAC in an electrical district energy system. Wetter et al. *[WBN16]* coupled models from the `Buildings` library for the electrical grid, multiple buildings, HVAC systems and controllers to test a controller that adjusts building room temperatures and PV inverter reactive power to maintain voltage quality in the distribution grid. Chatzivasileiadis et al. *[CBM+16]* exported reduced order building and HVAC models from Modelica as FMUs for co-simulation with the DigSILENT PowerFactory simulator for the low voltage grid and the OMNeT++ simulator for the communication network.

Several other studies also investigated the interaction of low-carbon technologies with the electrical grid at district level. For example, Schlosser et al. *[SSMM14]* analyzed the impact of home energy systems including heat pumps, combined heat and power (CHP) and PV, on a typical German low-voltage grid. This study employed models from the `AixLib` Modelica library to determine thermal loads, and network simulations were performed in MATPOWER. Further, in *[SSP+15]* they assess the potential of control strategies in reducing this impact. Even though a small improvement is achieved, additional measures are needed to eliminate the problems. Baggi et al. *[BRM+14]* modeled a residential neighborhood to assess voltage instabilities and self-consumption for different penetration levels of photovoltaic and storage system into the low voltage grid. They show battery systems can significantly improve grid stability and self-consumption. In this work, few Modelica libraries were combined, namely the `AC` *[Hau08]* and the `ElectricEnergyStorage` libraries *[ECK+11]*. Elci et al. *[EOH+15]* demonstrated that a decentralized solar CHP district heating system has better grid-interactivity compared to a conventional one, based on a simulation with components from the `Buildings` and the internal Modelica library `ISELib` of Fraunhofer ISE.

10.3.6.4 Control and Optimization

Aertgeerts et al. *[ACDCH15]* explored co-simulation methods coupling Modelica with Python to allow optimal control of district energy systems, with promising results showing Model Predictive Control (MPC) outperforming rule-based control methods. Focusing on optimal control involving both thermal and electrical domains, Bonvini and Wetter *[BW15]* presented a tool chain based on JModelica that allows reusing simulation models from the `Buildings` Modelica library to solve optimization problems. They demonstrated that a JModelica-based toolchain is efficient for optimization of optimal control problems with more than 10,000s of variables and that involve both thermal and electrical domains, nonlinear dynamics and constraints. Velut et al. *[VLW+14]* and Runvik et al. *[RLV+15]* used Modelica models to formulate the economic dispatch problem for production optimization of district heating systems. Python and JModelica were employed to solve the optimization problems. With this method they were able to add physical constraints, which significantly affected the optimal control function.

To perform optimization, other approaches utilized co-simulation based on the Functional Mock-up Interface (FMI) specification, but using simulation tools other than Modelica. For instance, Zucker et al. *[ZJB+16]* optimized the operation of a district heating plant towards minimization of CO_2 emissions by optimizing the temperature set points of the building. They used a previously developed co-simulation platform for modeling and simulation of district energy systems with FMI *[WMB+15]*, and automatically generated EnergyPlus building models from GIS data.

10.3.6.5 Advantages of Using Modelica for District Energy System Simulation and Optimization

Numerous use cases are available in the literature, some of which are summarized above, that use the Modelica language. They cover a large range of applications and multiple domains, such as building physics, district heating and cooling, electrical network and control.

The first advantage of using Modelica is the large number of available and well-tested or validated models or sub-components. This allows a quick and fast aggregation of component-oriented models to build large district energy system models. Numerous uses cases also highlight Modelica's potential to integrate multi-physics models, such as hydraulic, thermal and electrical sub-components. It thus allows testing control strategies and optimization of a large district energy systems. Moreover, bi-directional fluid flow is one of the major advantage of the Modelica language for district heating and cooling systems, allowing the analysis of complex inter-meshed networks *[FCO+09a]* *[BP11]*. The above literature review also highlighted numerous examples that include analysis and development of controllers. These can readily be implemented using formulations based on optimal control, finite state machines, discrete time or continuous time control.

Modelica modeling and simulation environments are compatible with numerous external environments. All major Modelica modeling and simulation environments support import and export of FMUs, and they contain application programming interfaces that allow interaction with the model as the simulation progresses. This can be used to tackle scalability issues of large district energy systems. In that perspective, a use-case and preliminary results are presented in the Section 10.6 to assess the potential of co-simulation using Modelica models and FMI export.

10.4 DESTEST

10.4.1 Framework

The modeling exercises in the scope of Activity 2.2 have identified the need for a testing framework for district energy simulation models: a DESTEST. This DESTEST aims at providing the structure for a common reference framework for testing models in a district energy simulation context, just like the BESTEST *[JN95]* does for building simulation models. It also aims to become a benchmark for academic exercises for technological solutions, modeling or method assessment. The latter is reflected in the development of the *Annex 60 Neighborhood Case* that was used to define common exercises for the Activity 2.2 participants. This DESTEST framework will encompass a range of tests that will need to be completed in order to validate models for district energy systems.

The concept of the framework is based on the ASHRAE 140 standard [JN06] for building simulation models. Three types of validation are proposed here:

1. Empirical,
2. analytic and
3. comparative validation,

each of which have their specific advantages and disadvantages. Empirical validation requires measurement data, analytic validation requires test cases for which an analytic solution can be found, and comparative validation focuses on the relative differences between the results from different models or model libraries without knowledge of the true solution. The reason for this division is the (un)availability of measurement data and validated model equations depending for different cases.

The testing starts at the component level, e.g. simple tests of pipes, electricity grid elements etc. The tests are executed in increasing level of complexity and detail, building up from static to dynamic cases and from component level to full system integration.

Focusing on heating and cooling distribution networks, the framework would build up from component tests such as a pipe. Starting with pressure drop validation for a pipe in steady state, levels of complexity can gradually be added, such as temperature drop, changes in mass flow, periods of zero mass flow and mass flow reversal, changes in inlet temperature and thermal capacity of pipe walls and insulation. In a next step, these components are added in a test grid set-up. At first, this is a purely branching network in a steady state (laminar or turbulent flow), thereafter meshed grids can be considered as well as dynamic operation. This case would correspond to the *Annex 60 Neighborhood Case*, which is described in the following section. Lastly, a database with measurements from a real district heating (and/or cooling) network could be built in order to simulate a realistic case and compare the outcome of the simulation with real measurements.

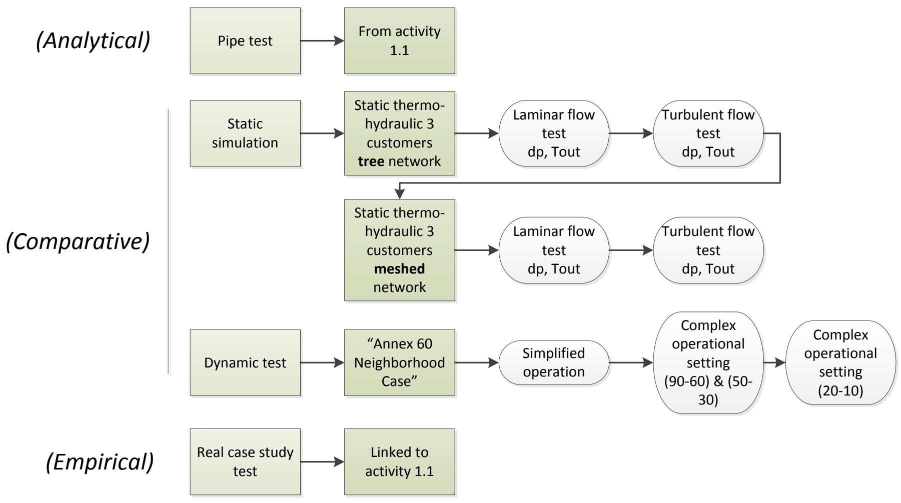

(Analytical)
Pipe test
From activity 1.1
(Comparative)
Static simulation
Static thermo-hydraulic 3 customers **tree** network
Laminar flow test dp, Tout
Turbulent flow test dp, Tout
Static thermo-hydraulic 3 customers **meshed** network
Laminar flow test dp, Tout
Turbulent flow test dp, Tout
Dynamic test
"Annex 60 Neighborhood Case"
Simplified operation
Complex operational setting (90-60) & (50-30)
Complex operational setting (20-10)
(Empirical)
Real case study test
Linked to activity 1.1

Fig. 10.7: *Envisioned testing structure of the DESTEST Framework.*

Finally, a range of key performance indicators has to be defined. These can be divided in different categories. First and foremost, the difference between the results from different models must be considered. Furthermore, the computational speed and memory needed for initialization and simulation are important. For the specific case of Modelica, other numerical parameters such as number of Jacobian evaluations, eigenvalues, number of states, size of non-linear systems etc. can be compared. It is important that for every test, a limited number of relevant key performance indicators is selected in order not to overcomplicate testing.

10.4.2 Definition of Common Exercise: Reference *Annex 60 Neighborhood Case*

10.4.2.1 Introduction

Activity 2.2 primarily focuses on evaluating, testing and demonstrating modeling and simulation approaches developed in Subtask 1. Therefore, the aim of this common exercise is to define a reference *Annex 60 Neighborhood Case* on which various of these models can be tested. The test case provides an environment enabling the analysis of aspects such as the following:

- Interconnection of electrical and thermal grids
- Hydraulic balancing
- Change from consumers to prosumers: influence of distributed energy generation
- Control strategies: their benefits, impact on consumers, modeling options
- Demand Side Management (DSM): how to use flexibility provided by thermal networks and storage

The development of the *Annex 60 Neighborhood Case*, comprises three steps described below. These steps aim at gradually incorporating features which are necessary to analyze the identified issues. During the course of Annex 60 not all steps were completed. Nevertheless this outline will serve as a guide for further development of the DESTEST.

In the first step, the description of the *Annex 60 Neighborhood Case* is elaborated. Considering the aims of this common exercise, a small neighborhood is defined, with typical configuration and providing possibilities to implement a district heating and electrical network. Buildings include residential dwellings and an office building, producing various demand profiles resulting from different use, size, insulation quality and occupancy. Description of a district heating network and an electricity network connecting all buildings is generated as well. The level of complexity of the simulated energy systems is progressively increased, from individual to collective complex systems.

The second step consists of modeling the behavior of buildings and their installations in the defined neighborhood, where buildings are treated as standalone (not coupled). Participants were asked to individually perform this task, freely selecting a building modeling approach, however, within the Modelica environment. Comparison of predefined performance indicators representative for the building energy demand is then carried out. While the main aim of this step is to identify a common library with building models, an additional aim is to demonstrate the consequences of modeling decisions and to illustrate the effect of different levels of complexity. By analyzing the results, the level of understanding in modeling buildings of all participants levels out, while corrections were performed to the developed models.

In the third step, district heating networks were investigated. Participants were asked to individually model the network, using models and simulation methods developed in other activities. For example, direct integration in Modelica may be chosen, or use of FMI and co-simulation. Reporting on the different approaches taken and modeling difficulties encountered, feedback is given to other activities, while knowledge is shared among participants.

10.4.2.2 Description of the Reference *Annex 60 Neighborhood Case*

This section provides an overview of the Annex 60 neighborhood case. A detailed description is available from the reference document at http://www.iea-annex60.org/downloads/Annex60_2_2_ReferenceCaseStudy-v1.0.pdf.

A typical neighborhood is defined including residential buildings and an office building. Mixing of end uses allows studying the potential heat exchange among buildings. Since the simultaneity of heating, cooling and electrical demand is a determining factor in the design and operation of district energy systems, additionally various insulation levels and stochastic user behavior are foreseen.

Neighborhood

The reference neighborhood is based on a small set of reference buildings which are combined to describe a street section. Five typologies are made for the building layouts, i.e. a detached (D), a semi-detached (S) and a terraced (T) dwelling, an apartment block (A) and an office building (O). Two subtypologies are made to distinguish between the levels of thermal insulation. The five dwelling typologies and two levels of thermal insulation are combined into a street design of 24 buildings as shown in Fig. 10.8. All required piping and wiring of district systems (for heating, cooling or electricity) are foreseen on a single side of the street as shown in Fig. 10.8, resulting in asymmetric connections.

Building typology

Detailed plans of all five building typologies are given in the reference document describing all dimensions and interior subdivisions of the buildings. Such detail allows participants to select the number of simulated zones per building. Nonetheless, a minimum of two zones per building is required. Table 10.1

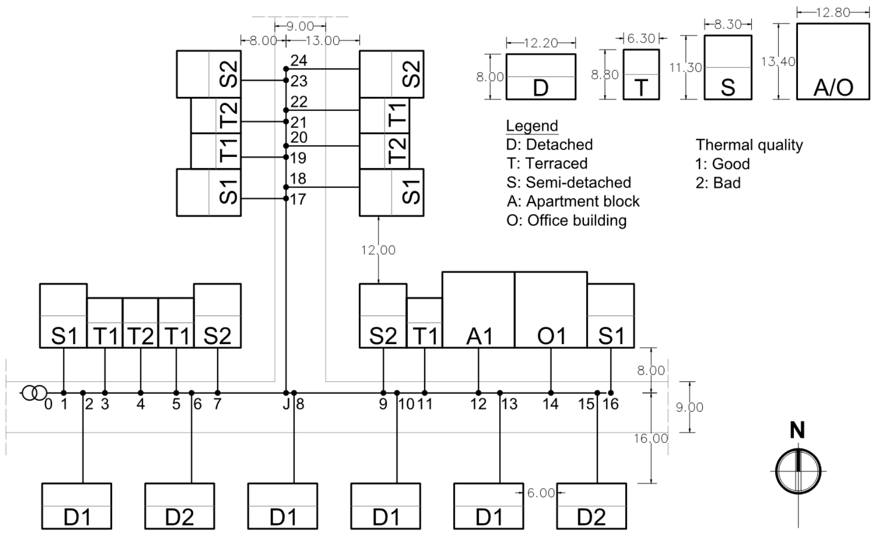


Fig. 10.8: *Street layout of the neighborhood with the selected building typologies.*

summarizes the main geometric parameters of the reference buildings. Each of the single-family residential building comprises two sub-topologies to distinguish between two levels of thermal insulation, the first one referring to the requirements of the EPBD 2010 in Flanders, and the second one representing typical Flemish buildings of the period 1946-1970, both based on the IEE TABULA project typologies. All construction elements are described in detail in the reference document, including material properties, angle-dependent glazing properties, and airtightness.

Occupancy

Initially, a constant set point temperature of 21°C for heating is assumed for all zones. There are no internal gains and no requirements for cooling. This simplification allows for an easier comparison between models and brings the focus on the implemented physics. In a next step, it is proposed to refine this assumption by introducing stochastic user profiles for internal gains, temperature set-points, electricity use, etc. All these are provided as input files for each building, based on generated profiles from the `StROBe` package of

Table 10.1: Summary of the main building parameters (entire building for the dwellings and typical floor for the apartment block and office building, which have a total of 5 floors).

	Detached dwelling (D)	Semi-detached dwelling (S)	Terraced dwelling (T)	Apartment block (A)	Office building (O)	
Usable floor area	102.7	100.9	107.5	107.5	107.5	m^2
Total floor area	185.5	187.8	161.0	161.0	161.0	m^2
Heat loss area	371.1	3051	219.1	104.6	104.6	m^2
Air volume	451.5	455.1	446.5	435.9	435.9	m^3

`openIDEAS` *[BDCJ+15]*.

Space heating, hot water and ventilation systems

For the second step of this exercise, the energy demand is compared. Individual and ideal heating systems are assumed. State #1 residential buildings are heated by an air-to-water heat pump, while state #2 buildings are heated by a condensing gas boiler. There are no cooling systems installed in residential buildings. In the office building, the cooling load is covered by an air-cooled chiller and the heating load by a condensing gas boiler. The same simulation model of the condensing boiler is used for both residential and office buildings. Initially, the distribution and emission systems are not simulated. The condensing boiler, the heat pumps and the chiller are assumed to provide water at respectively $50°C$, $40°C$ and $7°C$. Nominal operation conditions for these systems are provided in the reference document, together with look-up tables for boiler efficiencies for the initial stages.

Reference district heating network

The district heating network topology considers the pipes for supply and return on a single side of the street, as shown in Fig. 10.8, buried at a depth of 1 m. The pipes are buried directly in the ground. Three different scenarios have been considered, using different supply/return temperatures, namely:

- Scenario 90-60 has $90°C$ supply temperature and $60°C$ return temperature.
- Scenario 50-30 has $50°C$ supply temperature and $30°C$ return temperature.
- Scenario 20-10 has $20°C$ supply temperature and $10°C$ return temperature.

The pipeline distances between two nodes and the nominal diameter per scenario are specified, as are the insulation properties of the pipeline. The position of the main heat plant has to be considered at node 0. The temperature level of the different scenarios refers to the supply temperature at node 0 and the return temperatures at the costumers' nodes. The DH design takes into account the maximum heat load requirements based on the space heating demand simulated in the previous steps, assuming a maximum velocity in the pipes of 1.1 m/s.

The supply model can be initially considered as an ideal source supplying unlimited mass flow rate at $90/50/20°C$ respectively for the three scenarios. Therefore, an ideal pump is supposed to be installed, providing at any time step the required mass flow rate (no worst-point control on the pressure drop is considered).

Ground properties

The pipeline is assumed to be buried directly in the ground at 1 m depth. In order to calculate the ground temperature for the heat distribution losses of the pipe model, any ground model can be used. Ground properties and related assumptions are provided for uniform results among participants.

Substation design

Each substation contains a valve to control the water flow from the DH network and a heat exchanger between the primary loop (DH side) and the secondary loop (building side). The valve is controlled in order to meet the heat demand of the building and to maintain a low return temperature on the DH network side. The latter is the most important criterion in limiting the water flow and ambient heat losses in the DH network.

To model the heat exchanger, it is proposed to impose a constant heat transfer value (UA), to neglect the thermal inertia of the heat exchanger and to neglect the heat losses of the heat exchanger to the ambient. A set of equations to describe the substations is given in the reference document, resulting from the previous assumptions, as well as a proposed method to identify the parameters of each substation.

Demand

Participants not modeling the buildings in detail may use the provided heating demand profiles for the district heating simulation. The profiles were generated from detailed building simulations implemented at the KU Leuven, using this present description and the `IDEAS` Modelica library *[BDCJ+15]*.

Model performance criteria

The developed models should be able to simulate a typical annual behavior of the buildings or district heating network. Depending on the models and simulators, not all performance criteria may be available. As a common base, we define the hourly data as the actual value at the beginning of the hour, and the hourly maximum (or minimum) as the maximum value of the above hourly data. The following model performance criteria should be evaluated.

Energy demand and energy use

The energy demand is the energy required by the building to achieve the set points regardless of the heating or cooling system. The energy use of the building takes into account the efficiency of the heating cooling system including conversion efficiency, control efficiency, emission efficiency and distribution efficiency. Initially, predefined system efficiencies are used for calculation of the energy use. The following information is required for model comparison, ideally at an hourly basis:

- Total and peak heating and cooling energy requirements of individual buildings and aggregated.
- Overall average system efficiency for each building.

Thermal behavior

The buildings are simulated in two ways: free floating temperature and with heating and cooling system. For both simulations, peak (maximum and minimum) and average temperatures are compared per dwelling. Ideally, hourly temperature profiles of each building for the free floating case will be available, to allow an additional analysis of the temperature rise and decrease.

District heating network

The following information should be provided:

- Minimum and maximum heat distribution losses
- Total energy for the heat distribution losses
- Minimum and maximum pressure drop
- Total ideal pumping energy

Production plant

The following information should be provided:

- Minimum, maximum and average mass flow rate
- Minimum, maximum and average return temperature
- Peak production
- Total energy production

LV distribution grid

Grid cable parameters

Three different grid strengths will be examined, denoted by strong feeder, moderate feeder and weak feeder. The nominal cross-section area of the cables is given in the reference document for all lines and each grid strength. For strong feeders 120 mm^2 cable is proposed for the entire grid; for the moderate 120 mm^2 for line 0-16 and 95 mm^2 for the branch J-24; and for the weak grid 95 mm^2 for 0-16 and 70 mm^2 for J-24 (see diagram Fig. 10.8). In practice, cables smaller than 70 mm^2 are rarely used.

The lengths of the cables between the LV distribution grid and buildings are shown in Fig. 10.8. The size of these cables is as follows:

- Single-family building (D/S/T): 16 mm^2;
- Apartment building (A): 35 mm^2;
- Office building (O): 50 mm^2.

The cable parameters are based upon the `IDEAS` library *[BDCJ+15]* and are given in the reference document.

Transformer parameters

Transformers are modeled with identical phase impedances for all three phases. The parameters are given in the reference document. For the strong and moderate feeders, a 250 kVA transformer is chosen, while for the weak feeder, a 160 kVA transformer is selected. This choice may vary depending on the load profile of the office building, and/or total grid load.

Single/three-phase connection

Buildings may have a single or three-phase connection to the LV distribution grid. Two options can be assessed:

- All buildings have a three-phase connection leading to balanced conditions.
- Buildings have a single or three-phase connection, depending on the power rating of loads in the building.

In the latter case, all buildings are single-phase connected to the LV distribution grid, except when the power rating of at least one load (e.g., heat pump or photovoltaic system) is larger than 5 kVA. This philosophy is taken from the connection requirements for PV systems in Flanders.

10.4.3 Annex 60 Neighborhood Case: a Comparative Study

10.4.3.1 Introduction

This section presents a comparison of the energy performance results obtained from the different approaches taken by activity 2.2 participants to model the common district case-study described in Section 10.4.2. The modelers (KUL, TUe, EDF, ULg) used the Modelica language and were given flexibility to choose their preferred modeling approach regarding model complexity, used libraries and components, etc. The aim of the exercise was to first predict the annual heating demand (peak and average) of the district, and then to analyze the differences in the results obtained by the different participants, evaluating the potential influence of their selected modeling approach.

10.4.3.2 Case Description

The Annex 60 Neighborhood Case was defined in Definition of Section 10.4.2. The main information is presented here for reader's convenience. The neighborhood contains five typologies of building, i.e. detached (D), semi-detached (S), terraced dwelling (T), apartment (A) and office (O) building. Two levels of thermal insulation are taken into account. Level #1 represents the insulation level required by the EPB 2010 in Flanders and level #2 represents Flemish buildings in 1946-1970 based on the IEE TABULA report *[CRHV11]*. Fig. 10.8 depicts the street layout of the reference neighborhood. Detailed description about building layout, dimensions, thermal properties of building envelope please refer to case definition.

10.4.3.3 Modeling Approaches using Modelica

The models that constitute the district are based on different Modelica components libraries. The level of abstraction used to model the buildings and their sub-division in thermal zones as well as the heat transfer to adjacent buildings was left up to each of the participants. A constant heating set point of $21^\circ C$ was used for all the buildings and no internal gains or shading were modeled. The assumptions and modeling approach adopted by the participants is summarized in Table 10.2.

10.4.3.4 Results and Discussions

Figures 10.9 to 10.15 show the heating demand simulated by the participants for the different building typologies in the reference neighborhood. To better present heating demand for each building typology, we divided the results into a cold and warm season according to the outdoor air temperature from the weather file and the heating set point used. The warm season was defined from the first day when the outdoor air temperature was greater than the heating set point to the first day when every hour was lower than the set point. In this case, this period goes from the 19th of May to the 10th of October. The cold season was assumed to be the remaining time of the year.

Figures 10.9 to 10.15 present a comparison of the thermal performance of the different building typologies for the cold season. The apartment block (Fig. 10.9) and office building (Fig. 10.10) appear to have similar performance. The differences are only 1.4% on average demand and 0.1% on maximum demand for the results from TUe. This is mainly due to the similar geometry and thermal properties used for both building envelops. Note that internal gains were not included in the model. The different modeling approaches by the participants in terms of the number of thermal zones modeled and the heat flow between adjacent buildings (Table 10.2) did not create significant differences. However, for other building typologies, significant differences can be observed, especially for the poorly insulated buildings.

One of the reasons for these differences could be the approaches taken to model the heating system. Fig. 10.16 shows indoor air temperature in the semi- detached dwelling S1_1 from ULg and TUe. The results from ULg show room temperatures varying from 20.4 to $21^\circ C$, with an average of $20.6^\circ C$. From the TUe results, we can observe that room temperature was most of the time at $21^\circ C$, due to the idealized heating control modeled.

Figures 10.17 and 10.18 show annual heating demand and peak demand per

Table 10.2: Summary of Modeling Approaches

Participant	TUe	EDF R&D	KUL	ULg	RWTH AACHEN
Libraries & Versions	Modelica 3.2.1 Buildings 1.6	Modelica 3.2.1; BuildSysPro 2015.12b	Modelica 3.2.1; IDEAS v0.2	Modelica 3.2.1; Buildings 1.5; Thermocycle	Modelica 3.2.1; AixLib v0.3.2
Nb. of Zones	1 per floor (D, T, S); 2 per floor (O); 3 per floor (A)	Mono-zone	1 / floor (unheated attic) (D, T, S); 3 per floor (A,O)	6 (D, T); 5 (S); 5 per floor (A,O)	4 (D, T, S, A, O)
Shading	N	N	N	N	N
Heat transfer through adjacent surface					
Floor to Ground	Y	Y	Y	Y	Y
to adjacent zones	Y	Y	Y	Y	N
to adjacent building	Y	Y	N	N	N
Ground Floor Temperature (oC)	10.3	10	9+periodic effect (ISO 13370)	10.0	10
Weather Location	Uccle, Belgium				

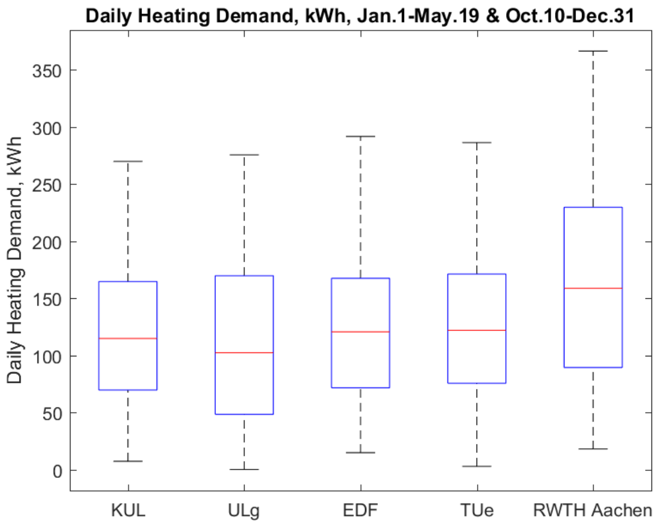


Fig. 10.9: *Simulated daily heating demand apartments*

building of the district. Significant differences can be observed in poorly insulated buildings such as buildings 4, 6, 7, 9, 15, 20, 21, 23 and 24. These results are much more sensitive to the complexity and assumptions adopted in the modeling approach. We can also observe different results for two buildings of the same typology, as for numbers 17 and 18. The difference in heating demand is caused by the way the heat transfer to adjacent buildings is modeled.

Figure 10.19 shows the overall heating demand and peak load from the different participants. As for the building analysis on figures 10.17 and 10.18, we can see that KUL and TUe may have underestimated energy needs with respect to other participants. The results show significant deviations for the overall yearly heating demand up to 20% from the average and 15% for the peak heating demand.

In order to further compare the simulated results from different participants, energy signatures for one building representative of each building typology are generated and shown in Figures 10.21 to 10.24. The energy signature shows

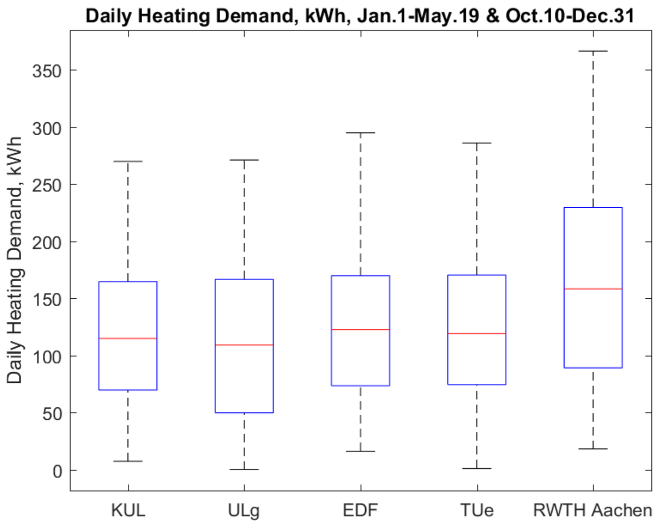


Fig. 10.10: Simulated daily heating demand for offices

the relationship between the outside air temperature and heating demand for a given building. A linear regression of daily average values was performed for every building typology.

Taking the energy signature of the detached dwelling (Fig. 10.20) as an example, the relationship between daily heating demand and daily average outdoor temperature can be observed. This relationship consists of two parts, one where it is approximately constant (flat) for all temperatures above the balance temperature (when the set-point is reached without the need for heating) and a part with certain slope for outdoor temperatures when heating is required. A linear regression for these temperatures above the balance temperature is shown in Fig. 10.21. R-squared for the three regression curves range from 0.72 to 0.78.

The energy signatures for all other building types of thermal insulation level 1# are shown in figures 10.22 to 10.24. As depicted from figures 10.9 and 10.10, heating demands in the apartments building (A) and office building (O) have similar distributions. As stated before, this is mainly due to not including inter-

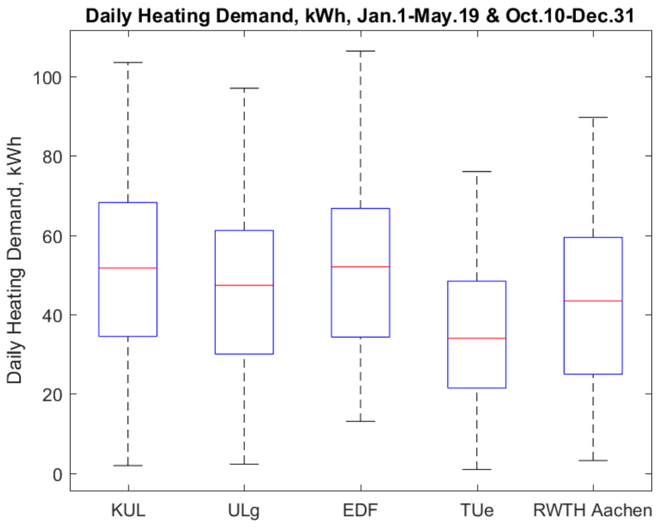


Fig. 10.11: *Simulated daily heating demand for detached dwellings*

nal gains and a similar set point schedule. For this reason, only the energy signature of the apartments building is shown (Fig. 10.22). Only one building per typology is illustrated, in this case, terraced house no.11 and semi-detached house no.1. The terraced dwellings have the largest observed difference between participants. Both slopes and balanced temperatures vary significantly. This could be explained by the way heat transferred is modeled across adjacent buildings, since terraced house no.11 has a larger heat transfer surface to adjacent buildings compared to its own volume.

In addition, Figures 10.25 to 10.28 show the energy signature generated from the simulated data obtained by the participants for buildings with thermal insulation level #2. Compared to Fig. 10.20, the linear relationship between daily heating demand and daily average outdoor temperature is more obvious and, as expected, the balance temperature of poorly insulated buildings is significantly higher, as shown in Fig. 10.25. The energy signature of each building typology with thermal insulation level #2 are shown in figures 10.26 to 10.28.

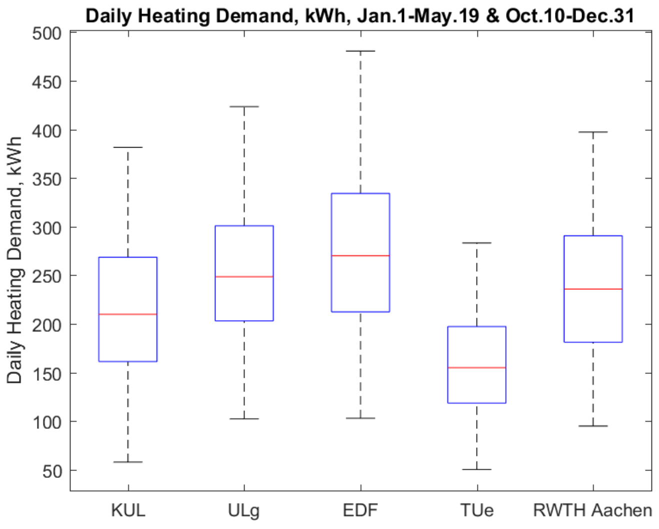


Fig. 10.12: Simulated daily heating demand for semi-detached dwellings S1_1

10.4.3.5 Summary and Discussion of the Common Results

Five participants from different institutions modeled and simulated the Annex 60 Neighborhood Case using different approaches and Modelica libraries. As illustrated and discussed in previous sections, the largest relative differences on heating demand among the participants occurred in poor insulated buildings. A preliminary analysis has been conducted to explore the possible causes, which can be concluded as follows:

- Relative to the selected modeling approaches, e.g. zoning methods, heat storage for internal partitions, heat transfer through surfaces (from adjacent zones and buildings), soil temperature, thermal models;
- Relative to the parameters selected, e.g., geometrical dimensions were provided as detailed floor plan with thickness of walls. However, modelers may have used different measuring approaches regarding outside/inside/center of the walls;
- Undefined inputs, e.g. infiltration rate for non-conditioned zones, thick-

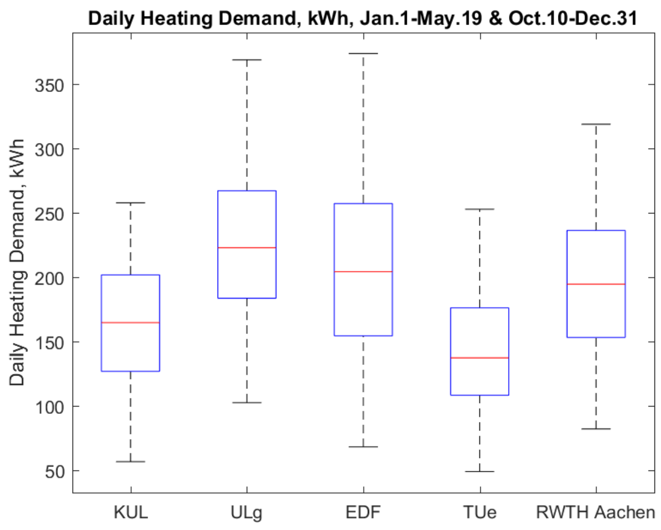


Fig. 10.13: Simulated daily heating demand for semi-detached dwellings S2_7

ness of glass pane and air space.

However, within this project, we could not conclusively determine where the discrepancies origin from. Therefore, one suggestion for further research is to conduct a systematic evaluation, e.g., a one-at-a-time analysis. It would be interesting to additionally perform a sensitivity analysis with respect to the mentioned parameters. Further evolution of the DESTEST to compare the energy performance results obtained from different modeling approaches will require that the case descriptions avoid ambiguous specification of parameters.

10.4.4 Examples of the Use of the Annex 60 Neighborhood Case

In this section, we briefly present two examples of the use of Annex 60 Neighborhood Case that extends the above performance studies. They were directed toward bioclimatic design and retrofit solutions for DHS extension re-

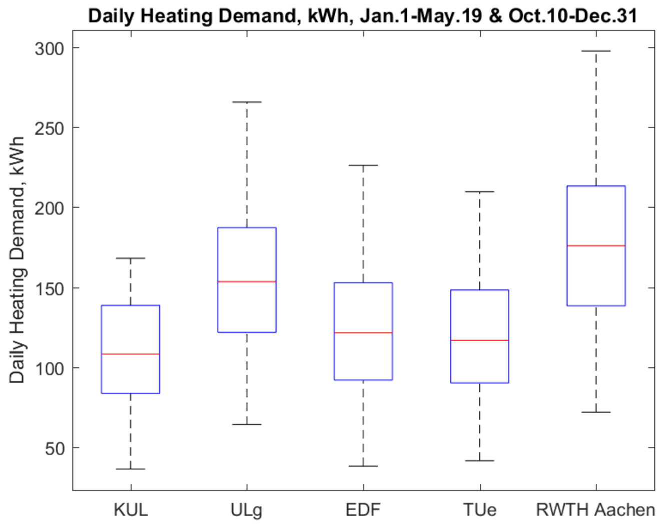


Fig. 10.14: *Simulated daily heating demand for terraced dwellings T1_3*

spectively. The following results are mainly preliminary and will be further described and analyzed in future work.

10.4.4.1 Towards Urban and Bioclimatic Design: Taking the Building Surroundings into Account

The ANR VALERIE collaborative project (2009-2012) adopted a bioclimatic approach of building analysis to understand the possible energy exchanges (like infra-red radiation) of the building with its neighbors, shading effects or urban heat island effects *[Duf12]*. Results showed that current design methods under-use these available resources. Thus, the source of performance brought by the use of free renewable energy available in the environment is more significant than “more insulation” for new buildings *[CDRR12]*. Urban densification is also a major factor of urban micro climate degradation, and specialists of urban issues seek more information on the influence of buildings. In order to study these new research questions, the ANR MERUBBI

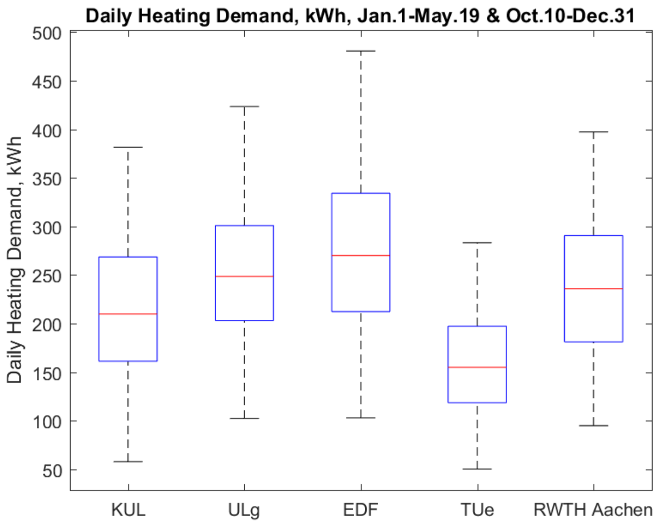


Fig. 10.15: *Simulated daily heating demand for terraced dwellings T2_4*

project (2014-2018) is developing a methodology that takes the urban aspect into consideration by dealing with the integration of a new building into an existing district.

The new tool chain developed in order to answer these research questions has been applied to the Annex 60 Neighborhood Case. Starting from the common neighborhood description, the district was modeled using SketchUp. Then, the 3D architectural model is transformed into an energy model in two steps. The first one consists of a transformation of the 3D architecture model into a pivot XML file. Then, a Python tool has been developed to translate the gbXML file into a building energy model based on the BuildSysPro open-source Modelica library *[PKL14]*. Raytracing calculations, based on the open-source Embree library *[WWB+14]*, are used to calculate the form factors in the district, and the amount of direct, diffuse and reflected solar radiation for each surface, as well as the ground albedo.

At the time of this writing, only preliminary results could be established and analyzed. These show that, for instance, that certain houses are negatively

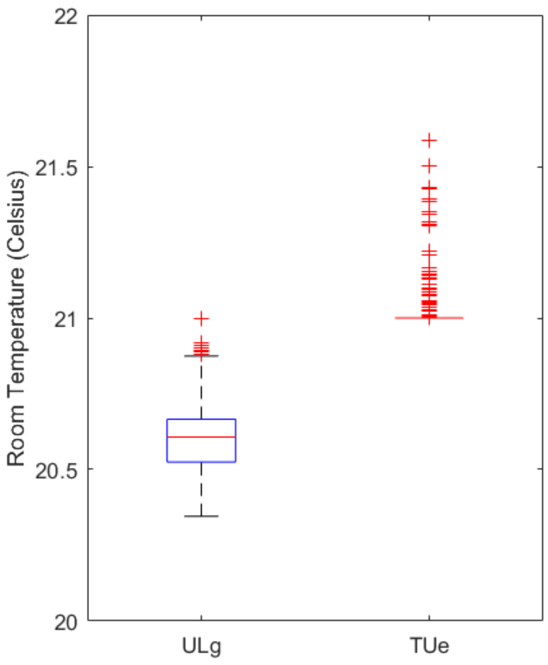


Fig. 10.16: *Simulated room temperature for semi-detached dwelling S1_1*

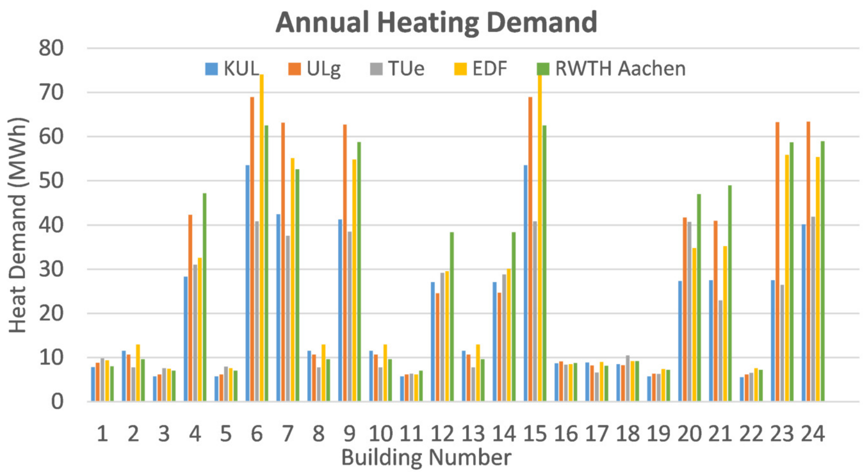


Fig. 10.17: *Simulated annual heating demand for each building from the different participants*

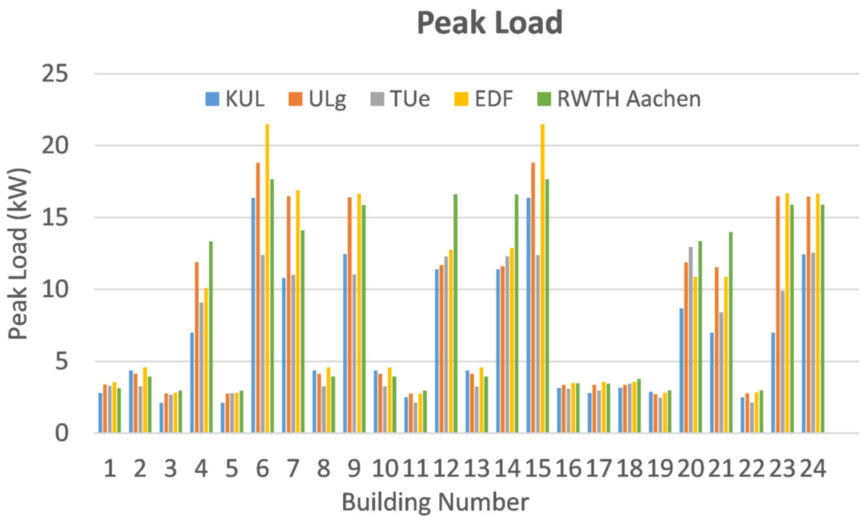


Fig. 10.18: *Simulated annual heating peak load for each building from the different participants*

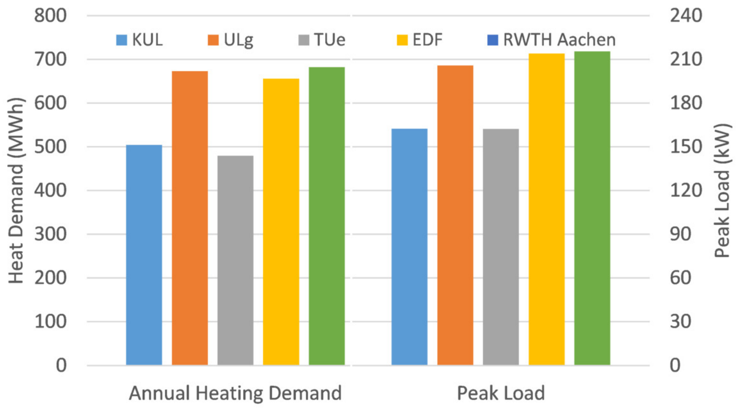


Fig. 10.19: *Overall yearly annual heating demand and peak load on the Annex 60 Neighborhood Case*

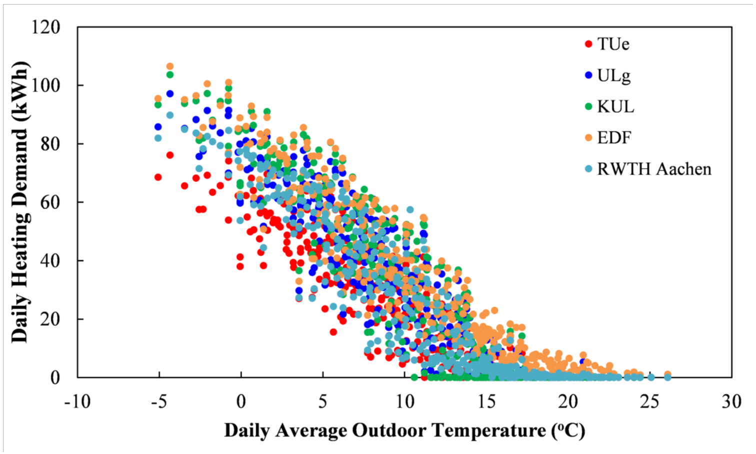


Fig. 10.20: *Relationship of simulated daily heating demand for D1 and outdoor temperature by different participants*

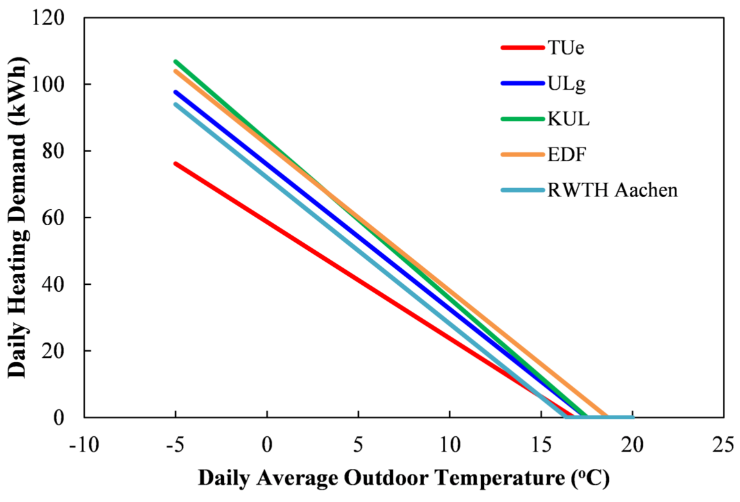


Fig. 10.21: *Energy signature for building type D1*

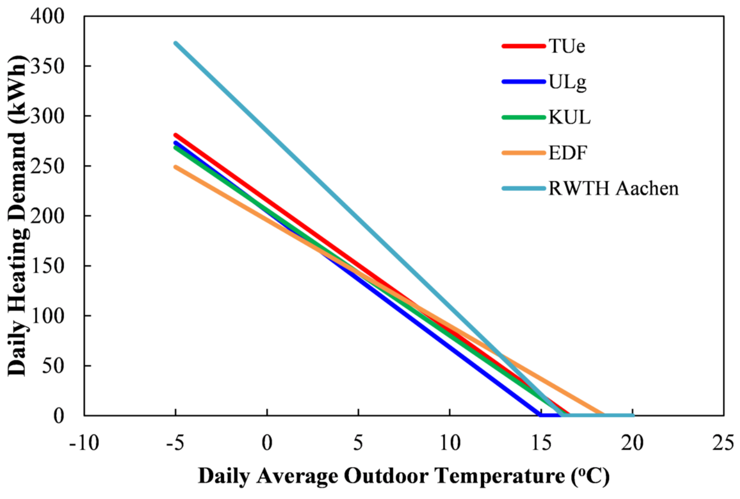


Fig. 10.22: *Energy signature for building type A*

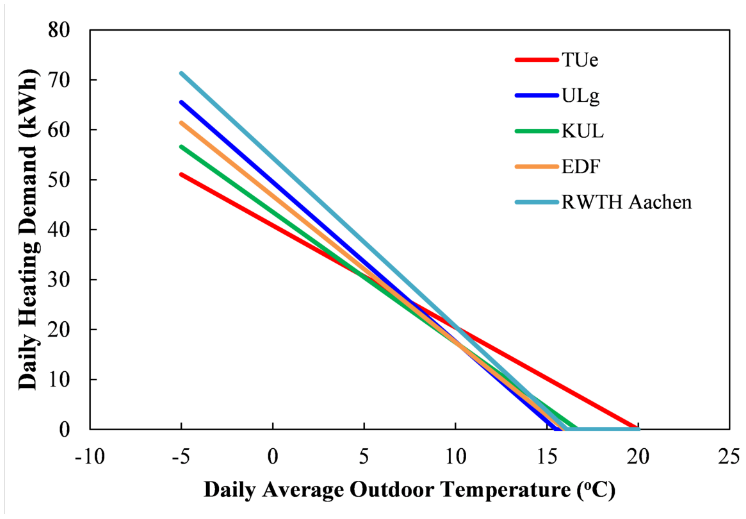


Fig. 10.23: Energy signature for building type T1

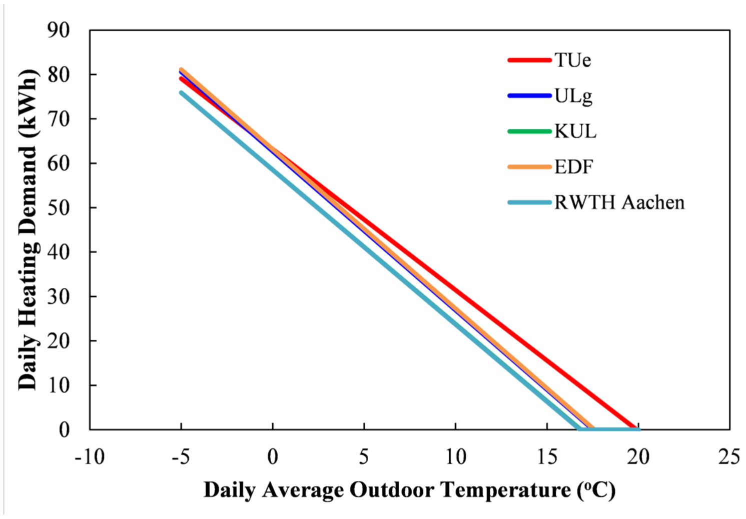


Fig. 10.24: Energy signature for building type S1

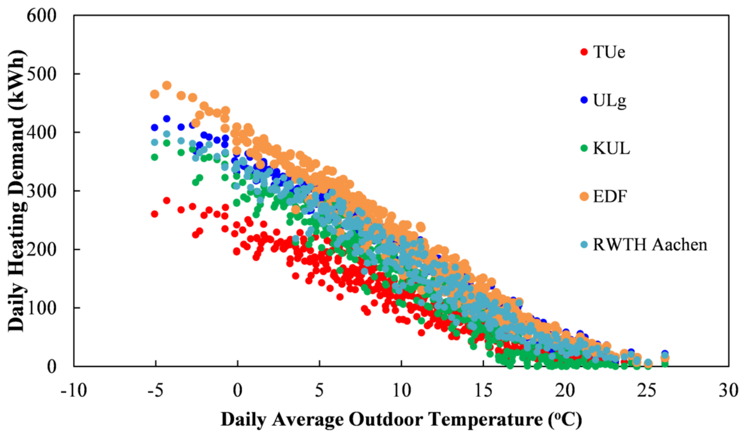


Fig. 10.25: *Relationship of simulated daily heating demand for D2 and outdoor temperature by different participants*

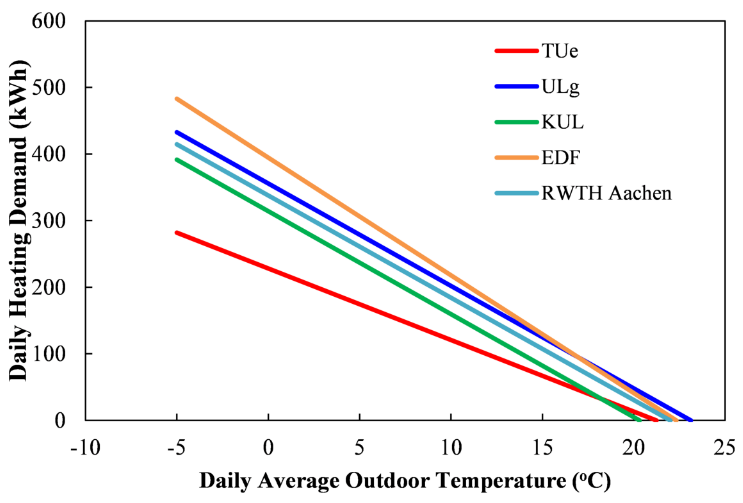


Fig. 10.26: *Energy signature for building type D2*

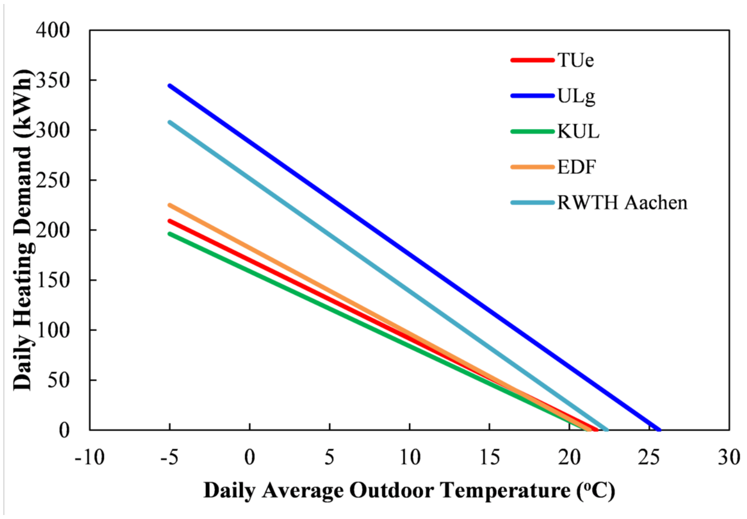


Fig. 10.27: Energy signature for building type T2

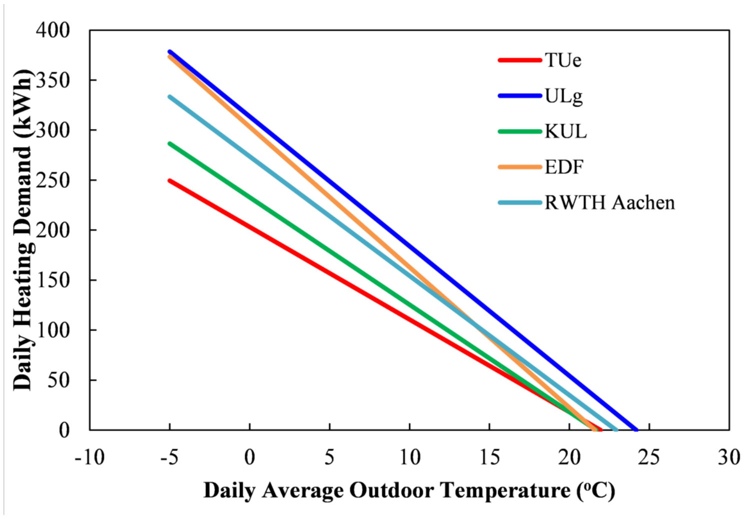


Fig. 10.28: Energy signature for building type S2

impacted by the presence of nearby apartment blocks, showing an increase of 20% of their heating energy consumption. This preliminary work will be detailed further when the tool chain of the ANR MERUBBI project is fully validated.

The impact of shadow effects is small for the configuration of the Annex 60 Neighborhood Case. Hence, it is not necessarily the best test case to assess the effect of local micro climate effects. Nevertheless, it was found useful to perform preliminary tests. It however demonstrates that in order to develop a meaningful DESTEST, different use cases should be carefully analyzed to make sure that the developed DESTEST is suitable to assess the analyzed problem. This example showed that it the development of more than one neighborhood case description may be necessary to cover the variety of problems that may be encountered and assessed in district energy simulations.

10.4.4.2 Model-Based Assessment of Cost-Effective Retrofit Solutions for a District Heating System Extension

Numerous research has been conducted on DHS, early examples go as far as the 14th century. However, few studies can be found on the evolution and the adaptation of older DHS in today's world. Therefore, the present study adopted the Annex 60 Neighborhood Case to define a fictional neighborhood for the investigation of the long-term adaptability of such an existing DHS. In particular, we considered reusing the return water of the existing network to heat a new branch of the DHS, as shown in Fig. 10.29.

The upper part indicates the initial DHS (red and orange pipes), which is directly fed by the heat source. The lower part shows the extension branch to be added to the initial DHS (orange and blue pipes), which will be fed by the return water of the initial DHS. In this study, some promising retrofit opportunities were evaluated on an extended DHS. In particular, the following stand-alone retrofit measures were applied in the original study :

0. No retrofit measure.
1. Retrofit of the envelope of the municipality buildings.
2. Decentralized storage in each of the new buildings.
3. Centralized storage on the municipality buildings.
4. Limitation of the internal set point temperature.

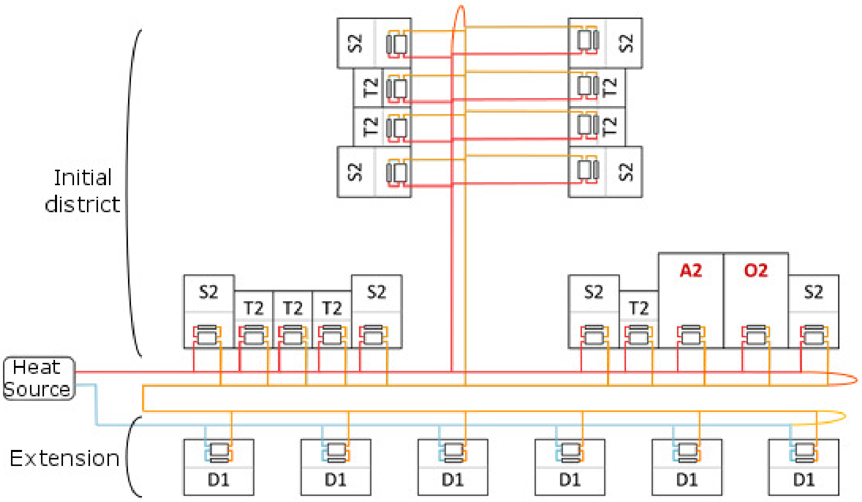


Fig. 10.29: *Diagram of the studied district based on the Annex 60 Neighborhood Case. Red lines represent the supply flow, orange lines the intermediate flow, and blue lines the return flow. Buildings A2 and O2, in red, are assumed to belong to the municipality. The curves connecting supply and return lines represent the bypass valves.*

Studies were done to compare the results to the new DHS without any retrofit measures (solution #0) during the coldest week. Solution #1 allows significant energy savings, and thus presents better results to extend the network up to a certain point. Installing individual and centralized water tanks (retrofits #2 and #3) present a global increase of the energy demand of the DHS. Preliminary results show that, during the peak demand, the DHS is unable to supply enough heat to the customers, regardless of the retrofit measure. As a consequence, during the peak laod, we observe a maximum deviation of 1.12 Kelvin with respect to the set point temperature.

This study is an example of how the Annex 60 Neighborhood Case has been used as a case to analyze possibilities for retrofitting of an existing DHS.

10.5 Multi-Physics Simulation of DES in Modelica

This example demonstrates how Modelica allows to couple models of different physical domains for voltage stabilization of the distribution grid through demand response and PV inverter control. The example analyzes both the thermal and electrical dynamics of a small neighborhood with a high PV penetration. It requires the coupling of models for building energy simulation, electrical system simulation and feedback control loops. The models interact with each other through the electrical load imposed by the building on the grid, and the feedback control that adjusts the building room temperature set point and that increases the reactive power of the PV inverter in case of violation of the power quality.

Fig. 10.30 shows a net zero energy (NZE) neighborhood. By coupling models of buildings, HVAC, electrical systems and controls we can assess the effect of building load onto the electrical grid and assess the efficacy of control measures. The neighborhood contains seven small office buildings. Each building represents a small office that is part of the EnergyPlus commercial reference building models *[DFS+11]*. Each model has one floor and is divided into five thermal zones plus one attic. The floor area of each building is about 500 m^2.

The building model comprises four different components, the thermal part, the schedules, the HVAC system and the electrical models. The thermal part accounts for the heat transfer through the envelope and energy storage in

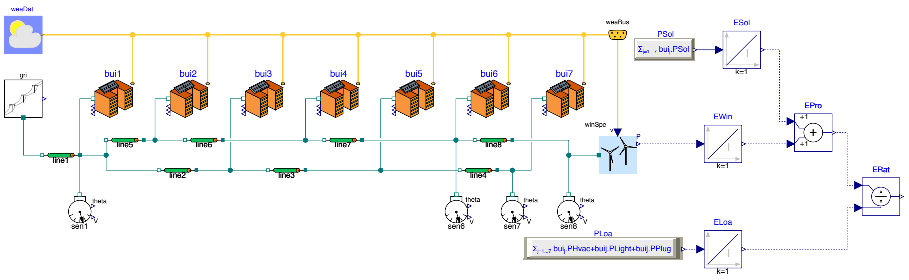


Fig. 10.30: *Neighborhood model with renewable energy sources. The yellow lines are weather data, green lines are electrical lines, and dashed blue lines are for post-processing.*

building constructions. The schedules represent the building internal loads due to occupants, as well as plug loads and lighting systems. The HVAC model represents the mechanical system and the control loops that maintain the thermal comfort in the building. The electrical models represent the interaction between the building and the electric network. They include an inductive load model that represents the electric load of the building and PV panels. All models are implemented in Modelica using the `Buildings` library *[WZNP14][BWN14]*. The thermal model was available as an EnergyPlus model. To integrate it with the rest of the building model, it was automatically converted to Modelica. The automatic conversion program leverages the OpenStudio API to identify the thermal zones and the components of the building fabrics. The conversion program converts the models by instantiating and connecting components of the Modelica Buildings library.

The building electricity consumption is modeled using the inductive load

$$P_{bui}(t) = P_{hvac}(t) + P_{plug}(t) + P_{light}(t),$$
$$Q_{bui}(t) = P_{bui}(t)\tan(\phi),$$

where $P_{bui}(\cdot)$ is the total active power, $Q_{bui}(\cdot)$ is the total reactive power, ϕ is the phase angle of the apparent power for the power factor p_f, with $\phi = \arccos(p_f)$, $P_{hvac}(\cdot)$ is the power consumed by the HVAC system, $P_{plug}(\cdot)$ are the plug loads and $P_{light}(\cdot)$ is the lighting power. In addition, the neighborhood has a wind turbine that supplements the energy provided by the PVs. TMY3 weather data for San Francisco, CA, were used. For more information about the electrical models and their implementations see [BWN14].

The nominal voltage of the neighborhood is V_{nom} = 1.2 kV and the nominal load of each building is P_{nom}^{load} = 18.6 kW. The total nominal power of the PVs installed in the neighborhood is P_{nom}^{PV} = 130 kW and hence twice the sum of the nominal load of the office units. The PVs are unevenly distributed among the different buildings. The nominal power of the wind turbine is P_{nom}^{wind} = 93 kW and therefore it is five times the nominal load of an office building. The buildings are connected through annealed aluminum cables of size AWG 1/0 that are 300 m long. The neighborhood produces annually about 25% more energy than it consumes.

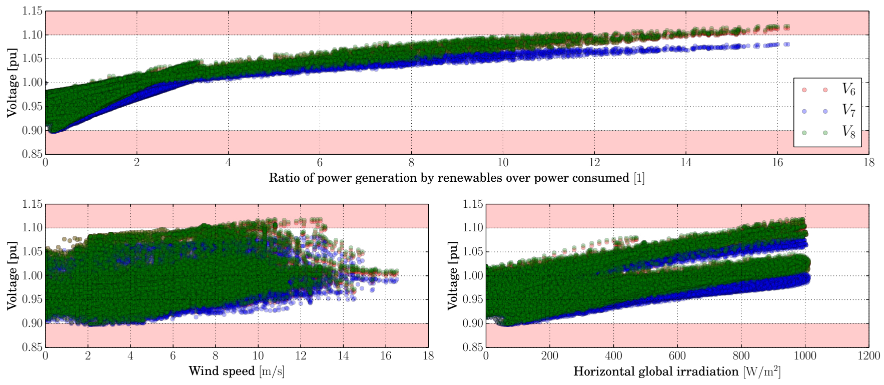


Fig. 10.31: Voltage levels in three different nodes of the neighborhood as a function of power generated by renewables, wind speed and horizontal global irradiation without feedback control.

We assumed the electrical system to be balanced because the analysis does not focus on possible asymmetries caused by the connection of the loads and sources on different phases, but rather on their impact on the voltage quality. In particular, we aim to keep the voltage within an admissible region of V = [0.9, 1.1] pu. To obtain diversity, the PV efficiency, orientation and tilt angles have been varied. Also, the power factor of the inductive load varies among the buildings between 0.8 and 0.95.

We will now analyze the simulation results for the case where there is no con-

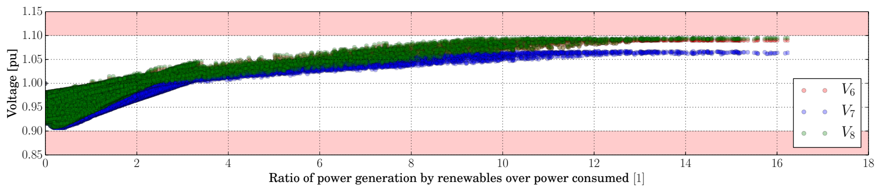


Fig. 10.32: Voltage levels in three different nodes of the neighborhood as a function of power generated by renewables with feedback control that adjust room air temperature setback and reactive power control at all PV inverters.

trol to ensure that the voltage remains within the admissible region. Fig. 10.31 shows how the voltages in three different nodes of the neighborhood vary with respect to the power generated by the renewables over the power consumed. The lower two plots show the voltage at the three nodes as a function of wind speed and horizontal global irradiation. As expected, the voltage increases as the power generated by the renewables increases. The highest voltage is V_8, measured by the sensor `sen8` in Fig. 10.30, which is at the end of the line where the concentration of PVs is higher and where the wind turbine is located. During 17 hours of the year, the voltage is above or below the admissible region of $V = [0.9,\ 1.1]$ pu.

To keep voltages within the admissible region, we will now add two control measures to the above example. During low voltages, we increase the set point temperature of the buildings by 2 Kelvin. During high voltages, we add reactive power at the PV converters *[TSBC11]*. Fig. 10.33 shows the section of the Modelica model that comprises of the building load, the PV, and a reactive load. Based on the voltage at the PV connection, a feedback controller injects reactive load in order to not exceed a voltage set point of 1.09 pu. Fig. 10.32 shows that this control measure keeps the voltage within the admissible region.

10.6 Co-Simulation of DES

Co-simulation defines a technique that allows simulators to be executed simultaneously while exchanging data during run-time. It allows a coherent integration of a decomposed system by assigning a specific solver to each

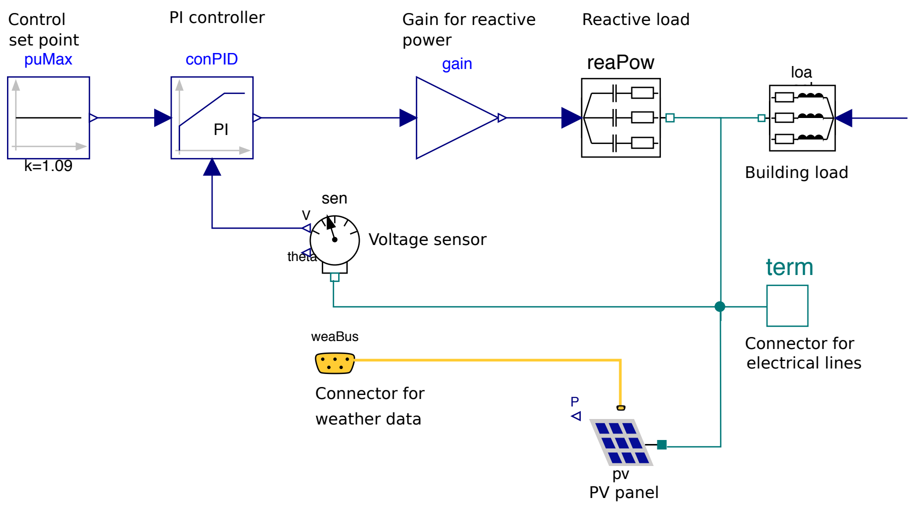


Fig. 10.33: *Electrical circuit with reactive power control at the PV inverter.*

sub-component. Thus, co-simulation methods can provide a solution to address performance limitations that may be encountered when simulating the overall system in one tool. Furthermore, co-simulation allows reuse of domain-specific tools.

In this section, we propose a practical application of co-simulation and tools described in Section 6, by applying it to a simple DES test case using a limited number of dwellings. The model has been implemented using Dymola and the OpenIDEAS library. Neither the electrical grid, the heating systems nor the building envelop as been simplified. The main purpose of this section is thus to propose a proof of concept of co-simulation and highlight advantages, good practices and the main drawbacks of such approach for DES.

10.6.1 Modeling for FMU-Based Co-Simulation

To focus on scalability for district energy system simulation, a simple district model using a changing number of buildings has been implemented in Modelica and acts as a reference to compare the co-simulation results against. This comparative study involves 3 and 6 dwellings, connected to a low voltage distribution grid. For each dwelling, we consider 2 thermal zones (day and night zones) and one heat-pump (HP) connected to a 3-phase linear feeder. Occupancy profiles, such as temperature set points, electrical and hot-water demands are heterogeneous and derived from a stochastic model *[BS15]*. We consider complex quasi-stationary equations of the grid in order to study the influence of the HP demand on the voltage fluctuation *[PBS15]*. Therefore, we used an output time step of 300 s. An example of 3 grid-connected buildings with scalar time-dependent connections is presented in Fig. 10.34.

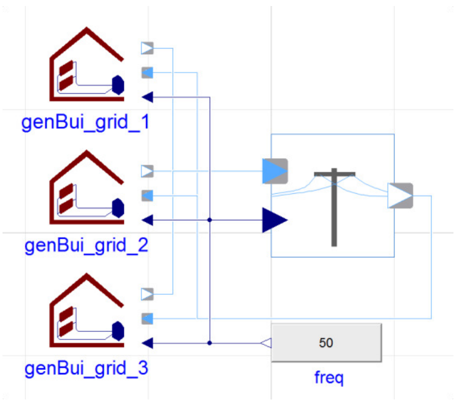


Fig. 10.34: Full simulation test case for 3 grid-connected buildings in Dymola.

Decomposition and graph dependence

Decomposing the model into sub-model for co-simulation is an important question, but it would be far beyond the subject to discuss theoretical aspects. However, in the scope of DES co-simulation through FMI and Modelica models, some good practices and advice can be highlighted. For more details, see also Section 6.7. The model decomposition is usually a trade-off between:

- exchange of quantities that vary slowly in order to use a large communication time step,
- decouple equations to make sub-models simpler to solve,
- create equivalent-size sub-systems to optimize the CPU usage in case of parallel computing.

The Modelica language usually make the decomposition straightforward since it is a component-oriented modeling language that explicitly declares decomposition, inputs and outputs of each sub-components. Moreover, DES typically consist of many similar components that are repeated and linked to each other. As a consequence, one could consider creating communications between clusters of buildings, buildings or, deeper, between heating systems, thermal envelope, and the network. As a starting point, we consider dwellings as several FMUs, and the electrical network as another one.

In our case, on the building side, we consider steady state complex voltage as inputs (supplied by the grid) and current as outputs. After defining the decomposition, one may need to add adapters to convert acausal ports to causal ports such as shown in the code snippet below. This adapter is then used to connect buildings to the grid as shown in Fig. 10.34.

```
model Pin2FMI_current
"Adapts an acausal connector to input/output signals"

  import QS = Modelica.Electrical.QuasiStationary;

  Modelica.ComplexBlocks.Interfaces.ComplexOutput i;
  Modelica.ComplexBlocks.Interfaces.ComplexInput v;
  Modelica.Blocks.Interfaces.RealInput freq(start=50);
  QS.SinglePhase.Interfaces.PositivePin positivePin;

protected
  QS.SinglePhase.Basic.Ground ground;
  QS.SinglePhase.Sensors.CurrentSensor iSensor;
  QS.SinglePhase.Sources.VariableVoltageSource vSource;

equation
  connect(iSensor.y, i);
  connect(v, vSource.V);
  connect(vSource.f, freq);
  connect(vSource.pin_n, ground.pin);
  connect(iSensor.pin_n, positivePin);
  connect(iSensor.pin_p, vSource.pin_p);

end Pin2FMI_current;
```

Concerning external resources, on the one hand, weather data should be delegated to a single FMU because of inconsistency risks. On the other hand, occupancy, hot water demand, and all the dwelling-related data could be implemented in the FMU of the associated house. Indeed, those data usually come from stochastic modeling [BS15] and logically differs among dwellings. Three possibilities are available:

1. create one FMU data-reader for all the dwellings,
2. create one FMU data-reader for each dwelling,
3. embed data reading into each FMU-dwellings.

The first solution is straightforward but introduces numerous communications that could affect computation time. Solutions 1 and 2 are not compliant with the boundary condition management of the used IDEAS library in which the information manager is embedded in each building. Therefore, those solutions would have introduced a lot of structural modifications. Embedded data reading in each dwellings implies a specific management of the resources. First, the declaration of all the possible data profiles as resources for each FMU may lead to increased memory requirements. This solution would lead to load all available profiles for each dwelling. Secondly, exporting each dwelling using its own data profile path would be a tedious work, and we thus loose the possibility to change the data profile after the compilation. The solution we developed is to automatically change the resource path during the instantiation of the FMU by casting an integer parameter to the resources' path as follows:

```
model GenBui_grid
  "General 2-zone building standalone with dynamic resource␣
↪path"
[...]

parameter Integer idOcc = 36
"id-number for occupant behavior on external data references
↪";

inner IDEAS.Occupants.Extern.StrobeInfoManager strobe(
FilNam_P="P/P_"+String(idOcc)+".txt",
filDir = Modelica.Utilities.Files.fullPathName("C:/Data/
↪Inputs/"));
[...]
```

In this example, the resource path points to *C:/data/Inputs/P/P_36.txt*. This way, the resulting FMU is still generic, i.e. we can change the data profile by changing the tunable parameter *idOcc* and the instantiation of the FMU would load only one data profile, limiting the memory usage.

The final model dedicated to co-simulation is then a simple combination of FMUs. The dependency graph for 3-connected building is shown on Fig. 10.35.

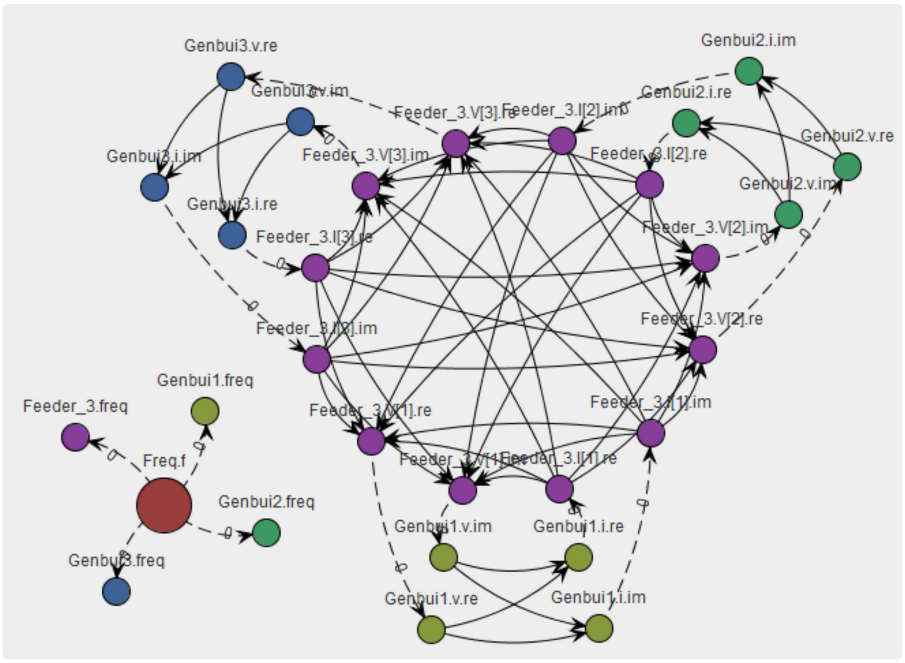


Fig. 10.35: Dependency graph of the 3 grid-connected dwellings.

10.6.2 Computational Experiments

We compared six different simulations for 3 and 6 grid-connected dwellings. This gives us an understanding of the scalability of the simulation. The six

Table 10.3: Simulations information

Sim.	Environment	Solver tolerance
'ref'	Dymola	1×10^{-8}
1	Dymola	1×10^{-6}

Table 10.4: Co-simulations information

Co-sim.	Environ-ment	Solver tol.	Master tol.	Communication Time Step (s)
2	Dymola	1×10^{-6}	1×10^{-6}	60
3	Dymola	1×10^{-6}	1×10^{-6}	120
4	Dymola	1×10^{-6}	1×10^{-6}	300
5	Daccosim	1×10^{-6}	1×10^{-6}	120

simulations reflect changing the solver tolerance, the master algorithm tolerance and the communication time-step in the case of co-simulation. Detailed characteristics of all simulations can be found in Tables 10.3 and 10.4.

Note that the simulation number 6 uses the Daccosim environment and parallel computing *[GVD+15]* (see Section 6.5.3). Parallel computing using Dymola for simulation and co-simulation shows irrelevant results in term of CPU time and is not reported in this analysis. Only fixed time-step communication was used in the master algorithm.

Table 10.5 shows the results for the 3 grid-connected buildings, including the CPU time, in absolute format for one year simulation (hour/minutes), in relative format with respect to the reference simulation and the Mean Absolute Error (MAE) of main physical quantities.

One can see that all the co-simulation results are worse in terms of computation time with respect to the complete simulation with the same tolerance (i.e. simulation no. 1). Moreover, MAE for the current might not be acceptable for $300s$ communication time-step (i.e. simulation no. 4) since it is higher than 2%. We noted that the delay on the voltage is determined by the communication time-step, whereas no delay is noted in the output of the current. This is due to the master-algorithm itself and might be an issue in the context of fast-control and safety, but has no impact on peak or mean values.

Table 10.5: Simulations results for 3 buildings

Sim.	*CPU Time*	*MAE*	
	(h, min) / ratio	I_{re}	V_{re}
'ref'	$(7, 28)$		
1	0.56	3.12×10^{-3}	1.35×10^{-5}
2	2.18	5.37×10^{-3}	4.10×10^{-5}
3	1.28	5.31×10^{-3}	6.36×10^{-5}
4	0.74	2.04×10^{-2}	1.39×10^{-4}
5	0.78	5.30×10^{-3}	1.01×10^{-4}

Note the good accuracy between simulation no. 1 and co-simulation no. 3 and 5 on Fig. 10.36.

The results of the 6 grid-connected dwellings are summarized in Table 10.6. Concerning accuracy, the same analysis as the 3-dwellings simulations can be maid, delay on the voltage and a good accuracy for communication time-steps smaller than 300s can be observed. As a result in this case, we consider a communication step higher than 120s not adapted for grid-connected building simulation. However, this value could increase for district heating systems with smoother dynamics.

In terms of computation time, we observe three main results :

1. Without parallel computing, we note lower simulation times due to the co-simualtion itself. The relative computation time decrease from 1.28 (3 grid-connected dwellings) to 0.72 (6 grid-connected dwellings) for simulation no. 3. That means that the gain in computation time increases if the number of buildings is higher. Although, co-simulation no. 3 still is slower than the equivalent simulation, but the gap is decreasing.
2. Doubling the number of connected dwellings increases the simulation time by a factor of 3.45 (i.e. simulations no. 1 and 2), whereas for co-simulation, this factor is about 1.9. This indicates that co-simulation is more scalable and potentially allows increasing the number of buildings in an easier way.
3. Simulation no. 6, using parallel computing, shows the best result with a relative computation time of 0.37, which is even faster than simulation no. 1 using comparable tolerance. Parallel computing on a two physical

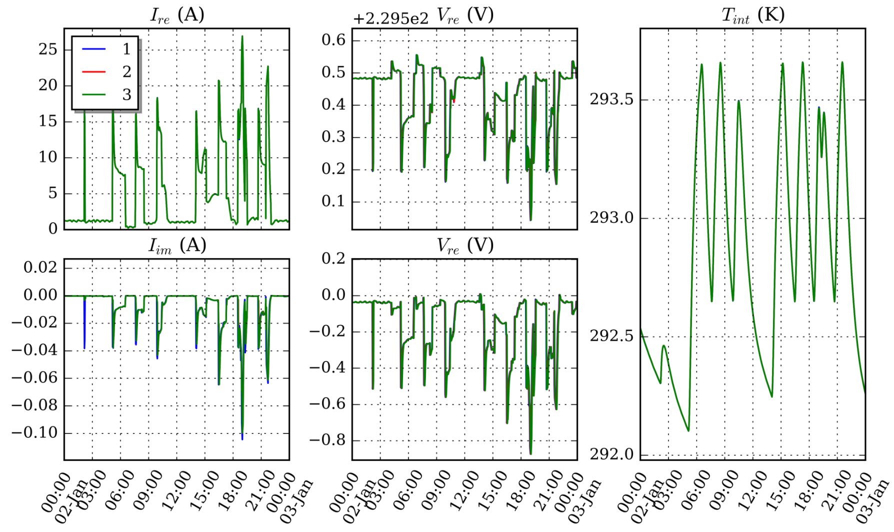


Fig. 10.36: *Comparison between simulations no. 1, 3 and 5 : Inner temperature, current and voltage of the building no. 1 for one day.*

Table 10.6: *Simulations results for 6 buildings*

Sim.	*CPU Time*	*MAE*	
	(d, h, min) / ratio	I_{re}	V_{re}
'ref'	$(1, 1, 49)$		
1	0.56	2.97×10^{-5}	5.62×10^{-7}
2	1.21	2.84×10^{-3}	5.13×10^{-5}
3	0.72	3.33×10^{-4}	9.26×10^{-5}
4	0.41	1.35×10^{-2}	2.72×10^{-4}
5	0.37	2.72×10^{-4}	1.23×10^{-4}

core CPU divides the computation time by 1.65 to 1.94 compared to an equivalent co-simulation.

10.6.3 Conclusion and Perspective on Co-Simulation of DES

Conclusion

This section tackled practical aspect of co-simulation of District Energy System using Modelica and FMI decomposition. It involved decomposition and adapting methods for future FMUs and introduced a method to deal with external data inputs using casting to generate heterogeneous occupancy data and save memory during the FMU instantiation. To analyze the possibilities that co-simulation offers, a comparative study involving 3 and 6 grid-connected dwellings, heterogeneous input data, different communication step-size and two environments has been performed. This comparison takes into account the quality of the main outputs, to ensure a good accuracy of results, and the CPU usage. Co-simulation shows a better scalability comparing simulations for 3 and 6 buildings since the computation time was proportional to the number of buildings if used with a fixed time-step communication master algorithm. Moreover, we note a significant time saving using parallel computing. In our experiments, the CPU time was 1.9 faster using a dual-core CPU. Although the speed-up cannot be attributed to the co-simulation methods itself, co-simulation generally facilitates the implementation of parallel computing.

Perspectives

Co-simulation for DES offers many perspectives. From a methodological point of view, one could improve the time simulation considering adaptive communication time-steps. In this case, the scalability of the approach might be decreased. Considering the implementation, parallel computing and an FMU containing inner co-simulation and parallel computing schemes, offers good perspectives in terms of CPU usage and interoperability. From the application point of view, test case using a higher number of buildings could be considered to assess those results for a larger number of dwellings. The effect of the heterogeneity of the structure and the occupancy data profile on the co-simulation performances could be also analyzed for grid-connected dwellings or other DES.

10.7 Conclusions and Outlook

10.7.1 Conclusion

Activity 2.2 “Design of district energy systems” looked into the use of Modelica and co-simulation techniques to analyze DES. The activity consisted of mainly three parts. A first part looked into the existing simulation tools to analyze the behavior of DES. In the second part a common exercise was set-up to develop and test a simple District Energy System and to initiate a framework to develop a validation test called DESTEST. Finally, in the third part of this activity, a preliminary assessment of co-simulation techniques for District Energy Simulations was performed.

Section 10.2 gives a description of all components that enable the delivery of energy services in DES. Because of the research interest of this activity, the focus lies on the modeling of electrical, heating and cooling networks at the neighborhood scale. The literature review of District Energy Simulation models (Section 10.3) focused on Modelica and non-Modelica implementations. It revealed that already many models and tools have been developed for building and district simulations. These models cover a large range, such as building physics, district heating and cooling, electrical network and control. As the analysis of DES requires an integrated whole system approach, often these tools have some drawbacks and shortcomings towards integrated energy system modeling. In many simulation environments, not all domains can be treated at the required level of detail. The many use cases demonstrate the suitability of the Modelica approach to model DES. The following advantages highlight this: numerous available and well-tried models or sub-components can be used to build large scale DES models including multi-physics options that are able to take into account specific phenomena such as bi-directional flows in thermal networks. Modelica is compatible with numerous external simulation environments through FMI exports. The created models can be used as testbed for controllers and as basis to optimize the design and operation of DES.

Section 10.4 reports on the common exercise that has been set up to analyze the capabilities of the Modelica libraries of the Annex 60 participants. It provides a first step to develop a test environment for District Energy Simulations: a DESTEST. A generic Annex 60 Neighborhood Case with 24 buildings was developed. It includes a mix of residential buildings and an office building. The description includes a full description of the building geometry, properties and occupant behavior and defined a heating network and an electricity network connecting all the buildings in the neighborhood. Five Annex 60 participants from different institutions modeled and simulated the heating demand of the neighborhood, using different approaches and Modelica libraries. Surprisingly, the results of the different participants showed deviations for the overall yearly heating demand up to 20% from the average and 15% for the peak heating demand. Exact identification of the cause of the discrepancies was not possible within the time frame of this project. This demonstrates that the many possible modeling approaches and the implementation by different users, each with their own interpretation of the data, indeed may easily result in important deviations.

The purpose of the Annex 60 Neighborhood Case is twofold.

1. It serves as a reference case to analyze relevant research problems such as how to design district heating systems and how to integrate renewables in districts. This was illustrated with two case studies. A first study analyzes to what extent the inclusion of the surroundings in DES simulations influence the result. For the Annex 60 Neighborhood Case, preliminary results showed that taking into account shadowing effects

increased the heating demand of buildings with up to 20%. A second study assessed cost-effective retrofit solutions for a DES extension. In both cases, the Annex 60 Neighborhood Case proofed to be useful to execute the analysis.

2. The information and challenges gathered during the development of the DESTEST will be used for setting up a framework to test District Energy System simulation tools. This framework will focus on developing different tests for testing models in a DES context. It also aims to provide benchmarks for academic and industrial analysis for technological solutions, modeling or method assessment.

As an alternative for full Modelica simulations, Section 10.6 analyzed the possibilities that co-simulation of DES using Modelica and FMI decomposition offers. It involved model decomposition and introduced a method to deal with external data inputs using casting to generate heterogeneous occupancy data and save memory during the FMU instantiation. By performing a comparative study involving 3 and 6 grid-connected dwellings, heterogeneous input data, different communication step-size and two environments, it was shown that co-simulation shows better scalability. Furthermore, for the tools used, the co-simulation techniques facilitated the implementation of parallel computing, further decreasing the computation time. Finally, the co-simulation case was an interesting example demonstrating the benefits of the interaction between Activities 1.2 and 2.2 of this Annex.

10.7.2 Outlook

The transition towards zero energy buildings and neighborhoods requires a constant evolution of the tools to design and assess the opportunities and challenge. Already many simulation environments and models exist to simulate the system behavior of the different components that DES consist of. It is expected that this number will rapidly increase and form a comprehensive framework that can be used to design and assess existing and upcoming DES-technologies in a fully integrated and systematic manner. The massive integration of renewable energy sources and reuse of excess energy from buildings and industrial processes demands flexibility in terms of supply and demand. Furthermore, different storage technologies should be added to future intelligent energy systems. In order to adequately apply demand side management techniques and make use of different energy storage options, state-of-the art controllers and algorithms are needed. Future research should look into how to implement these controllers more easily in the simulation environment to test and improve them. Lastly, multi-critria optimization tools should be further improved to facilitate the automated design of DES.

The implementation of data into DES simulation software is often a very elaborate and error-prone process. Moreover, the necessary data are often hard to find or even completely lacking. In short, there is a need for methods to easily implement existing data and reliable data to construct models of existing neighborhoods and to construct representative data sets for a generic analysis. To easily implement real neighborhoods, existing technologies such as BIM and GIS applications have to be further exploited and transformed into a toolchain capable of an automated generation of neighborhoods models. Especially challenging is finding reliable manners to implement missing data. Once such an automated toolchain is developed, it can be used to statistically analyze many different neighborhoods and even cities to create relevant reference neighborhood cases with statistically realistic behavior which is one of the aims of the DESTEST framework continuation in IBPSA Project 1. The question however remains how to verify the results of such neighborhood models. Still, only a limited amount of high quality measurement data are available to perform validation tests. The availability of such a toolchain will facilitate the development of neighborhood models, but will inevitably further increase the computation time and raise scalability issues. This emphasizes the importance of further analysis and development of numerical solvers, co-simulation techniques and parallel computing algorithms.

Chapter 11

Activity 2.3: Model use During Operation

11.1 Introduction

This chapter provides insights into using Modelica models to solve real-world problems in real-time. Three applications are presented in this chapter: Model predictive control (MPC); Fault detection and diagnosis (FDD) and Hardware-in-the-Loop (HIL).

11.1.1 Model Predictive Control

Model Predictive Control (MPC) uses a model of a studied system to predict its future evolution. This eliminates many drawbacks of traditional control approaches, such as laborious control gain tuning, weak prediction capability, difficult implementation of supervisory control, and need for re-tuning after operating conditions have changed. In addition, MPC is also able to handle constraints on control inputs and system states *[MBH+12]*.

11.1.2 Fault Detection and Diagnosis

In general terms, a fault is considered as any issue or state that causes a degradation of performance *[RWF05]*, even if it is not perceived immediately by humans. Detecting a fault is the process by which, using available information, there is a realization of this degradation of performance. Diagnosing a fault is determining the root cause of the degradation of performance *[Str08]*. Fault Detection and Diagnosis (FDD) is the field within control engineering that studies the automated detection and diagnosis of faults *[Ise97]*.

11.1.3 Hardware in the Loop

Testing and evaluating controller hardware performance is crucial for scalable deployment of control logics such as load controls and distributed control solutions. A Hardware-in-the-loop (HIL) method connects the controller hardware with a system model in lieu of the actual system. Therefore, controller hardware and control strategies (e.g., load control algorithms) can be tested and optimized in conjunction with a real-time emulator that is connected to the a real controller. Models running on the real-time emulator would represent building or grid energy equipment and systems. This method enables a closed control loop in a partially virtual system with real controllers.

11.1.4 Overview of Projects that Contributed to the Annex 60 Activity 2.3

Activity 2.3 focuses on the use of Modelica models to augment monitoring, control and fault detection and diagnostics methods. This section presents an overview of the case studies that contributed to the activity.

Table 11.1: Overview of projects participating in Activity 2.3

Area	Title	Brief description
MPC	MPC for energy control of underground public spaces - Universita Politecnica Delle Marche, Italy	This case study concerns the SEAM4US EU FP7 project pilot that has been deployed in the Passeig de Gracia (PdG) Line 3 metro Station in Barcelona, Spain. The pilot is currently operating, and is aimed at demonstrating the effectiveness of MPC applied to the ventilation, lighting and passenger movement systems in underground subway stations. The development of the MPC component required modeling the energy dynamics of the underground stations. The modeling of the environmental dynamics included the passenger flow, ventilation, lighting systems, and outdoor and indoor thermal, fluid and pollutant dynamics.
	MPC for heat pumps - KU Leuven, Belgium	In this case study, an MPC approach was developed to optimize the heat production of two identical heat pumps and a gas boiler. The MPC was implemented and tested using the Modelica environment at the headquarters office building of 3E, which is located in the center of Brussels, Belgium.

Continued on next page

Table 11.1 – continued from previous page

Area	Title	Brief description
	MPC for chiller plants - University of Miami, USA	In this case study, am MPC approach was established for chiller staging. The studied case is a chiller plant with three identical chillers, three identical chilled water pumps, three identical condenser water pumps, and three identical cooling towers. Each chiller has one dedicated chilled water pump, one dedicated condenser water pump, and one dedicated cooling tower.
FDD	Model-based FDD for District Cooling Systems - Lawrence Berkeley National Laboratory, USA	The project aims to improve the way current Energy Management Systems (EMS) operate by extending their capabilities with optimization and fault detection techniques. Such techniques are based on physics-based models that represent district cooling systems (DCS) and their components (e.g., chillers, pumps and cooling towers). The DCS object of this study is located at the US Naval Academy in Annapolis (MD). The system is characterized by a central loop where more than 20 buildings utilize the chilled water (CHW) for air conditioning. The building types range from data centres and gyms to swimming pools and other facilities. The CHW is provided to the central loop by two separate plants located in different zones of the campus. Each plant has three centrifugal liquid chillers (with both single and double stage compressors) and four cooling towers.

Continued on next page

Table 11.1 – continued from previous page

Area	Title	Brief description
	Fault detection through qualitative models of air handling unit components - Fraunhofer ISE, Germany	For demonstrating fault detection with qualitative models, different faults have been simulated with the Modelica fault triggering library, developed by the German Aerospace Center. An application example of a heat exchanger of a HVAC&R system shows the applicability of the qualitative modeling approach.
	Fault diagnosis using qualitative models of air handling units - National University of Ireland, Galway, Ireland	This case study, comprises a constant air volume air handling unit (AHU). The AHU serves a facility consisting of an audio laboratory of around 50 m^2. In this audio laboratory, strict conditions of temperature and humidity should be maintained. The building is located in Cork city in the Republic of Ireland.
	Quantitiative model-based diagnosis of AHUs - KU Leuven, Belgium	This case study focuses on the application of an open and easy to replicate fault detection and diagnosis method. The method was used for component failure detection in an AHU, focusing on a prospective malfunction of the dampers, the heat recovery system and the fans.

Continued on next page

Table 11.1 – continued from previous page

Area	Title	Brief description
HIL	Hardware in-the-loop - University of Alabama, USA	The study was performed at the integrated building energy and control laboratory at The University of Alabama, Tuscaloosa, AL, USA. A room served by a VAV terminal unit is the study object, where the room and VAV terminal unit are modeled in Modelica and downloaded to the HIL machine that is connected to a real VAV box controller. The objective of this case study is to research different control algorithms including fault tolerant controls for VAV boxes.

11.2 Background on Areas for Model Use During Operations

11.2.1 Model Predictive Control

A recent review of control for HVAC systems with an emphasis on MPC can be found in *[AJS14]*. MPC eliminates many drawbacks of conventional control in large-scale applications, including difficult parameter tuning, limited prediction capability, difficult implementation of supervisory control and limited adaptability to varying operating conditions. From the operational point of view, the following aspects are relevant in the engineering of MPC for energy efficiency applications *[MBH+12]*:

- The selection of the cost function: MPC can minimize for operating cost, energy use, peak demand, greenhouse gas emission, discomfort, etc.
- The decision whether uncertainty should be part of the cost function, which leads to a stochastic MPC problem.
- The handling of thermal comfort: various metrics for thermal comfort may be used in the cost function, such as the predicted mean vote in *[Fan73]*, *[BKC+12]*, *[HK14]*. Otherwise, other descriptive comfort met-

rics might be used in a set of linear constraints, such as zone temperatures, COtextsubscript{2} concentrations, and relative humidity *[ASH04]*, *[FOM08]*, *[KMB11]*, *[OPJ+10]*, *[MBH+12]*.

- The modeling technology: the three main approaches are detailed modeling *[HKLF05]*, *[CHMK10]*; simplified modeling and grey-box models *[Bra90]*, *[KMB11]*, *[OPJ+10]*, *[AMH00]*, *[BM11] [RDS14]*, and black box models *[CJL06]*, *[CPV+12]*, *[LH06a][LH06b]*.
- The implementation of the control actions: two major implementation methods can be found, either computing the control signals in real-time *[KMB11]*, *[OPJ+10]*, or using look-up tables for accessing solutions pre-computed off-line *[AB09]*, *[Cof11]*, *[BKC+12]*, *[DZMJ11]*.
- The implementation technology and, consequently, the development and deployment framework of the MPC solution. TRNSYS, MATLAB and Modelica appear at present time to be the most used tools for developing scalable and site-specific solutions for optimized control. The Modelica language has some key features that provides substantial advantages over the TRNSYS environments facing the complexity of large non-linear MPC applications *[WBN16]*, *[BWE+13]*. A specific MPC library for linear problems provide integrated control system design in Modelica *[HA09]*. Modelica models can be directly used in the main MPC loop *[IKS08]* unless the size of the model makes it computationally impractical. In those cases, they can be used as the data source for model reduction processes *[BWE+13]*.

The design of building models for MPC is not a trivial task. MPC models must provide sufficiently accurate predictions of future states, while also being computationally efficient for on-site deployment using cost-effective computational resources. At the same time, MPC models must provide results in a time frame compatible with operational time constraints. Furthermore, MPC models must be embedded in systems that, for cost reasons, will not include all of the sensing/actuating capabilities desired.

11.2.2 Fault Detection and Diagnosis

Fault detection and diagnosis in building operations can be seen as part of the building optimization process. A large amount of energy is wasted because many HVAC&R systems do not operate as designed *[KB05b]*, *[KB05a]*, *[BRK+14]*. Malfunctioning or faults are often not detected or only detected when they manifest themselves at the system level, such as when occupants complain *[BRK+14]*. It is estimated that FDD methodologies can reduce energy waste by 5% to 40% *[PKH01]*, *[WRB03]*, *[RLW+05]*, *[BRA+12]*, *[LKH05]*, in particular when faulty operations are timely rectified for the most frequent and high-impact faults types *[Age06]*, *[Hei12]*, *[LY10]*. Apart from leading to energy waste and reduction of equipment life, faults can also lead to health problems for the building's occupants *[MI03]*. However, the building sector is lagging behind other industrial sectors regarding development and application of FDD since operational optimisation of building operations has received little attention. In current practice faults are mostly identified manually during routine inspections, due to persistent alarms, or as a result of a noticeable degradation of performance. The problem with this approach is that many faults can be undetected for long periods of time, thus leading to considerable energy and monetary waste *[HDN+09]*. Automated FDD can help with this problem by providing timely indications of the existence and root cause of the fault, and possibly also suggest correctives actions.

Automated FDD requires a-priori knowledge of the normal and faulty behavior of the systems to be embedded in the methodology. In this sense, FDD techniques can be classified as rule-based or model-free methodologies *[Don10]*, model-based methodologies and history-based methodologies *[Ste15]*. In the Annex 60 project, the focus lies specifically in using Modelica and FMI for automated FDD in order to test and demonstrate the applicability of this technology.

Modelica models can be used for FDD in two different aspects: directly, by using simulation results as a reference for the monitored data, and indirectly, by using simulation data as training data for black box models. In the latter case, the results of the black box model are then used as a reference for the monitored data. The direct use of Modelica models for FDD, based on fault models, has been reported in *[BIFM09]*, *[LLM06]*, *[CMZ11]*. However, in the building sector, Modelica models have been rarely used for FDD, partly due to the effort that is typically involved in the development and calibration of a building and HVAC&R model. In the past few years, a step forward has been made with respect to the development of standardized building and HVAC&R libraries with the publication of the various Modelica libraries specifically for building energy modeling, which facilitates the setup of simulation models.

The possibility to import and export Modelica models as Functional Mock-up Units (FMUs) enables the integration of models using a standardized, tool-independent API into existing FDD routines. This allows coupling tools for data analysis, simulation, FDD and optimization in one single environment. An integral solution can be realized, for example, using the Building Controls Virtual Test Bed (BCVTB) *[Wet11a]* or JModelica with the Python module PyFMI *[AAG+10]*.

11.2.3 Hardware in the Loop

HIL is a process used for product development and testing in industries such as automotive and aerospace. Example applications can be found in *[WG06]* and *[ECP07]* where Modelica models were used for the development of hybrid electric vehicles. Specifically, implementations of a HIL test platform were used for simulations of drive cycles during testing of battery and fuel cell energy storage systems respectively. In *[ZLD+09]*, a HIL simulation system of a civil aircraft thrust reverser using a Modelica-based simulation platform was presented.

Although HIL with Modelica is a common process in different industries, it has not yet found wide applicability in the buildings community. In *[KRD13]*, the authors present HIL-based design and validation approaches of a home energy system using Modelica. In *[NKWM12]*, HIL was used for the development of a model-based controller of a window blind. In this process, Modelica models from the `Buildings` library were used to construct a model of a physical test cell with a controllable window blind. This model was used in real-time, together with Radiance and the BCVTB, to determine the blind position that minimized the energy consumption of the test cell. This blind position was then converted into an actuation signal that was used to control the blind of the physical test cell.

11.2.4 Modelica Interfacing Options

Modelica models can either be interfaced directly, or exported as a Functional Mockup Unit. Next, we describe these two approaches.

11.2.4.1 Direct Interfacing of Modelica

The Modelica standard does not specify the application programming interface (API) of compiled models. However, some Modelica simulation environment compile the model and generate a text file that contains parameters of the models that may be changed prior to simulating the model. This, however, is tool-dependent, and the format of the text file may vary as tools are updated.

Furthermore, different Modelica modeling environments provide their own API to manipulate models, such as setting parameter values, simulate the model, and obtain outputs. For example, Modelica has been integrated in the scientific computing environments Maple and Mathematica so that models can be manipulated and used directly from these high-level languages. Furthermore, JModelica and Dymola both have a Python interface. Dymola contains specific tools to couple Dymola and Matlab/Simulink.

The BCVTB is another free tool for interfacing Modelica models with different software and hardware tools specially oriented at building simulation applications.

Since tools and interfacing options are tool dependent and can change with time, we refer the interested reader to the documentation of the particular Modelica modeling environment.

11.2.4.2 Functional Mock-Up Interface

The Functional Mock-up Interface (FMI) is used to exchange models between different simulation tools by encapsulating tool-specific models in Functional Mockup Units (FMU). See also Section 6. Recent years witnessed important developments and adoption of the FMI technology. Now, over 90 tools support the standard, including all Modelica simulation environments. The use of FMI has the advantage that the FDD application can be developed independent of a specific simulation tool choice, and, conversely, the user can provide models for use in the application from a variety of model development environments. Moreover, the API is defined by a standard, rather than determined by a on-off-a-kind project that may not likely be supported through the life cycle of the building.

11.3 General Overview on Modeling Approaches for Buildings

Building energy modeling encompases the modeling of several interrelated processes such as building physics, energy systems, electrical elements, controls, occupancy, etc. The performance of the models during operation is linked not only with the quality of the model used, but also with its development and implementation costs and its reliability. Such aspects remain the largest bottleneck for efficient implementation of models in real-life scenarios. To date, a comprehensive overview of approaches for building energy modeling can be found in *[HL12]*.

The value of building energy modeling during operations is determined by the availability of real data that can be used either to develop a model or to calibrate a model. Three modeling approaches can be defined for building energy modeling:

- *Whitebox models* model a building by considering as much of the underlying physics as reasonably possible. Advantages of these models are the generalization capabilities, and the ability to directly relate physical phenomena to model equations. The disadvantages of this approach are the time and effort it takes to construct such a model and the complexities associated with calibrating the model with measurement data.
- *Blackbox models* are the complete opposite of whitebox models and neglect all physical insights by solely using the information of measurement data. The biggest advantage of this approach is that the model is configured automatically based on data. The disadvantages are that a good model needs a large amount of data and the resulting model might not be valid outside the range of the data that was used to build it. Furthermore, if used for FDD applications, if a black-box model is trained against faulty data, it cannot be used to identify operational faults.
- *Greybox models* use a predefined model structure which relies on physical insights. Measurement data are used to determine unknown parameter values. Compared to blackbox models, these models can have a wider range of applicable results and require less data. However, they can suffer from poorly chosen model structures for different buildings and systems. Compared to whitebox models, the calibration procedure

can be easier to implement due to the reduced set of parameters, while still representing the physical behavior in the structure.

11.4 Application Areas and Case Studies

In the following sections, an overview of how Modelica was integrated in the processes of the three application areas concerned in this chapter is presented. For MPC, applications to heat pumps, chiller plants, and energy control of underground public spaces will be discussed. For FDD, applications to district cooling systems and air-handling unit components will be discussed. For hardware in the loop, an application to a VAV system will be discussed.

11.4.1 Model Predictive Control

In the design of MPC, the structure of the system model is critical. The models should be able to provide reasonably accurate predictions of future states, while also having low computational demand, so that they are compatible with cost-effective computational resources for on-site deployment. These competing requirements for the system modeling can be fulfilled by Modelica. Advantages of Modelica for MPC include the ability to model multiple physical phenomena (such as heat transfer, fluid distribution, and electrical systems) *[HZS16]* with different system dynamics (continuous, discrete, or hybrid time and discrete event) *[WZNP14]* and different system sizes, ranging from single equipment to a building to communities with district energy systems and electrical grids *[WBN16]*.

The following examples demonstrate these benefits of using Modelica for MPC.

11.4.1.1 MPC for Heat Pumps

This case study presents the development of an MPC approach to optimize the heat production of two identical heat pumps and a gas boiler. The MPC has been implemented and tested [DCH16] at the headquarters office building of 3E, which is located in the center of Brussels, Belgium.

The MPC ensures thermal comfort while trying to minimize the heating costs. Comfortable temperatures in the conditioned zones lie between 20°C and 24°C during office hours. The heating system operates from September through May. The produced heat is distributed by three parallel hydraulic circuits through fan coil units (FCU), radiators and an air handling unit (AHU) to directly or indirectly transfer the heat to the zones.

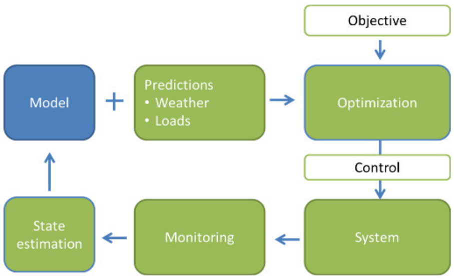


Fig. 11.1: *MPC architecture.*

The MPC runs online repeatedly through the loop presented in Fig. 11.1. With a system model and predictions of weather and internal load, an optimal control problem is solved to find the control variables which minimize the objective function over the prediction horizon. Since neither the model nor the predictions are perfect, a feedback loop is implemented. The heating system water temperature and the zone air temperature measurements from the building monitoring system are used to correct erroneous predictions through a simple moving horizon state estimation algorithm, which updates the states of the model based on measurements. With the new model state the next optimization loop starts.

The optimal control problem in the MPC is solved using JModelica [AAG+10], based on a toolbox written in Python to implement optimization problems using Modelica models. The reason for using Modelica and JModelica in the approach of MPC is twofold. First, the models implemented to represent the thermal behavior of the building and heating system are constructed in Modelica. The object-oriented implementation allowed constructing a library of low order resistance-capacitance models to represent the thermal system. A python toolbox implemented in JModelica allows for identification of model parameter values by solving an optimization problem that fits the model building temperature outputs to real measurements *[DCMAH15]*. Parameters of several model structures were identified, and from these, the best fitted model was selected. Secondly, JModelica was used to set up the optimal control problem. Since JModelica uses gradient-based methods for solving an optimal control problem, the equation based Modelica models are particularly well-suited as they allow symbolic differentiation and analysis of the model structure.

11.4.1.2 MPC for Chiller Plants

This case study presents the implementation of MPC for a chiller plant in order to improve the operating efficiency of the plant. To facilitate the implementation of the MPC, a software environment was developed consisting of three modules as shown in Fig. 11.2: Dynamic Optimization, Pre-processing and Post-processing. As the core of the framework, the Dynamic Optimization module is composed of an optimization engine and a system model. Raw measured data from the plant is processed in the Pre-processing module, which cleans it for input into the Dynamic Optimization module. The optimization results are then processed in the Post-processing module. Modelica is used to build the system model for the entire chiller plant (the primary chilled water loop and the condenser water loop).

In Fig. 11.2, the Dynamic Optimization module is designed to perform the optimization of the control inputs for the chiller plant. The objective function of the optimization is the energy consumption of the chiller plant during the optimization period. The optimized control inputs are the condenser water set point and the thresholds for chiller staging. The input variables for the optimization are the cooling load, the outdoor wet bulb temperature and the state of the controller. The Pre-processing module contains two components: Initializer and Input File Generator. The Initializer generates the Initial Data based on the Final Data from the previous optimization period. The Input File Generator converts the raw data, such as cooling load and weather data, into the Input Data, which can be directly read by the System Model. In the Post-processing module, the System Model reads the Optimization Output Data and generates

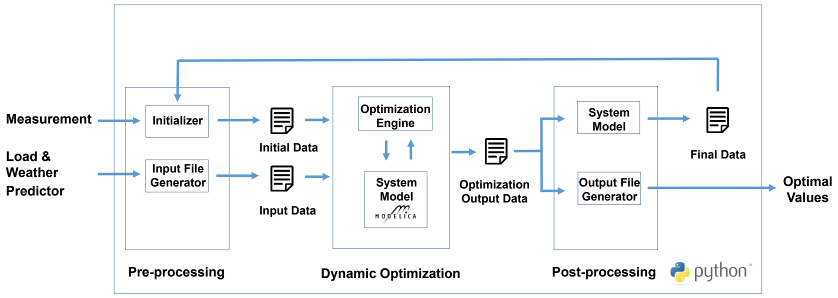


Fig. 11.2: MPC for chiller plants workflow.

the Final Data, which is then used for the next optimization period. The Final Data module includes the state vector. In addition, a component called Output File Generator processes the raw data in the Optimization Output Data and exports the data for later use, such as for plotting the results and generating the control signals.

For our experiments, we used the Dymola simulation environment. Dymola, as other Modelica environments, allow easy reseting of state variables which is needed when solving the optimal control problem, and when advancing time during the MPC algorithm. Of further benefit was the adaptive time step solver, which adjusted the time step automatically, selecting smaller time steps in times when the system changes rapidly. This lead to fast simulation time, while properly representing the dynamics of the process.

11.4.1.3 MPC for Energy Control of Underground Public Spaces

This case study describes the MPC implementation on the forced ventilation system in the Passeig De Gracia Metro Station in Barcelona (Spain). Given the complexity of the environment and the safety requirements specified, an MPC engineering framework was designed to minimize the impact of the system development on the station operation. Within this framework, Modelica was used to develop the whole building metro station model, including thermal and fluid dynamics, ventilation and lighting systems, and train and passenger flows. This system model was then used as an emulator to train a Bayesian Network that was used inside the MPC. Despite the size of the resulting model, amounting to about 77,000 unknowns, the adopted Modelica development environment, Dymola, still provided manageable simulation times with respect to the overall MPC engineering requirements.

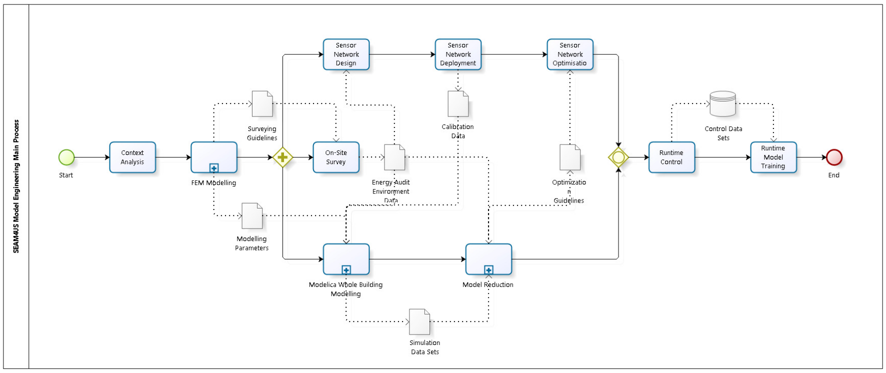


Fig. 11.3: *MPC for energy control of underground public spaces workflow.*

The MPC engineering workflow is depicted in Fig. 11.3. Context Analysis was aimed at surveying the site geometry, the plants technology and at assessing the original design. The characterization of the outdoor environment was accomplished through Finite Element modeling (FEM). It aimed at establishing the boundary conditions that influence the indoor dynamics. FEM was used to develop a preliminary model of the indoor environment as well, which informed a preliminary sensor network design. At this point, three tasks were started concurrently: on-site survey, the preliminary sensor network design and the development of the Modelica multi-physics station model. Despite the FEM modeling phase providing a number of insights regarding the behavior of the environment airflow and pollutants under different environmental conditions, it was limited to steady state analysis because of the size of the environment. This strongly limited its applicability to the development of complex MPC algorithms in the following phases, where optimality is achieved through accurate management of the transitory conditions. To this aim, Modelica provided a good compromise between expressiveness and computational efficiency. It endowed the abstraction and modularization that are required to manage the modeling complexity of such large domains. Knowledge encapsulation, topological interconnection, hierarchical modeling, and object orientation, as well as the availability of validated libraries granted operational effectiveness, and, at the same time, reduced the development effort. As soon as the first set of the monitoring data was available, the Modelica model was calibrated according to ASHRAE guidelines [ASH02]. This was a complex task and the computational stability and efficiency of the Modelica solvers were major factors in the success of this phase. Once calibrated, the Modelica station model was used to generate the data-set to train the Bayesian Network that was embedded in the deployed MPC loop and to fine tune the algorithms of the MPC systems without affecting the real station (Model Reduction). The flexibility of the Modelica environment was a major successful factor in this stage. The large station model was embedded as an FMU component in the MATLAB environment, and allowed for the development of optimal control algorithms in an emulated environment, without affecting the real station at all.

In this case study, Modelica proved effective as a fundamental part of a larger MPC engineering process. Modelica allows for the development of multiphysics domains made of thousands of components and evolving in a mixed continuous-discrete time model. Modelica is also an efficient language, as the solver speeds are adequate and the possibility of exporting FMUs opens possibilities of system integration. From an engineering perspective, the major limitations were associated with lack of a native support for model calibration, for which we used sensitivity and cluster analysis.

11.4.2 Fault Detection and Diagnosis

In building applications, the implementation of fault detection and diagnosis systems is usually programmed by experts for each specific plant, exploiting knowledge about the structure of the plant and the behavior of its components during correct and, if necessary, faulty conditions. These implementations, traditionally called *expert systems*, are typically structured as a set of rules that link potential symptoms and the faults possibly causing them, and an algorithm that applies the rules to given observations about the behavior of the system. The problem with this approach is not so much a technical one, but lies in the inevitably high efforts required to adapt or re-write the diagnostic program for a new plant or to reflect changes in an existing plant. The code (or the set of rules) has to be inspected in order to determine which part still applies and which must be modified or newly produced. The reason why these, often prohibitive, efforts are inevitable is that such programs capture the application of expert knowledge to a specific plant and, hence, leave both the structure of the plant and the knowledge about the physics of its components implicit in the code. A further drawback is that the set of rules is limited to symptoms and faults that have been experienced previously *[Ste15]*.

Model-based fault detection and diagnosis overcomes these limitations by using models that provide an explicit representation of the knowledge about the components and the information about the plant structure, which determines how the components interact with each other. Based on a library of generic component models and the representation of the HVAC system topology, a system model (possibly covering both the nominal and faulty behaviors) can be obtained. The context-free component models allow for their re-use, the automated generation of system models, and, hence, cost-effective creation of new applications and easy adaptation to variants and modifications *[Ste15]*. The following examples demonstrate these benefits of using Modelica for FDD.

11.4.2.1 FDD for District Cooling Systems

The goal of this case study was to provide FDD and fault identification for a chilled water plant, including the increase in energy use or operational costs due to the fault. This data is provided to the operator in order to prioritize the urgency of correcting a fault. The project was conducted at the US Naval Academy in Annapolis (MD), which has a district cooling system with two plants that serve 20 buildings.

Fig. 11.4 visualizes the approach used to implement the fault detection algorithm in the district cooling plant. The FDD algorithm is based on a state and parameter estimation algorithm that uses an Unscented Kalman Filter *[BSG+14]*. This requires repeated simulation of the model, subject to different parameters, initial states and input trajectories. The cooling plant and its components have been modeled using the Modelica `Buildings` library *[WZNP14]* and calibrated to measured data. The models were exported as

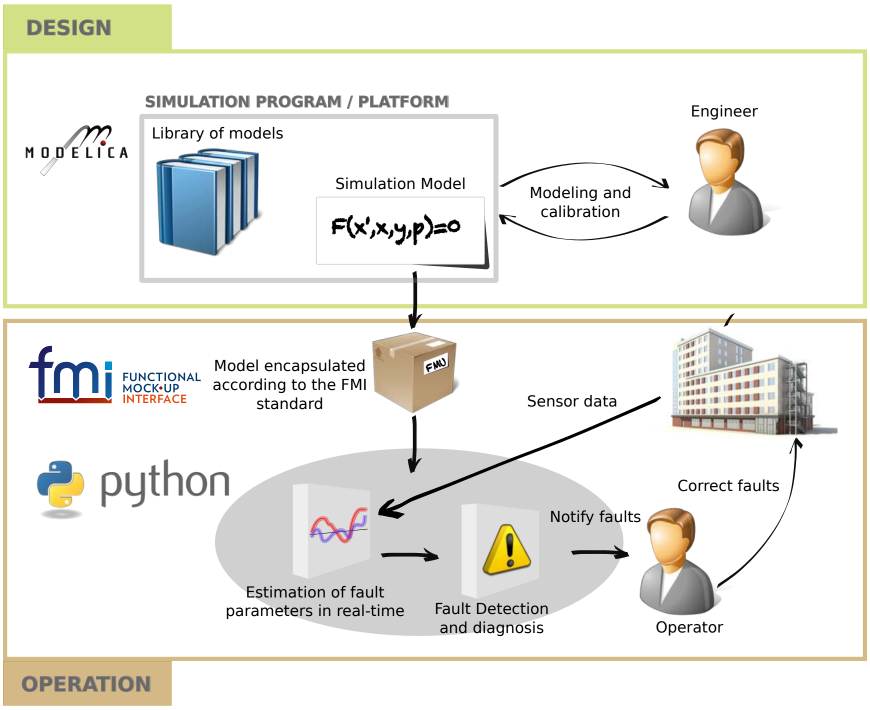

DESIGN
MODELICA
SIMULATION PROGRAM / PLATFORM
Library of models
Simulation Model
$F(x',x,y,p)=0$
Modeling and calibration
Engineer
fmi FUNCTIONAL MOCK•UP INTERFACE
Model encapsulated according to the FMI standard
FMU
Sensor data
python
Correct faults
Notify faults
Estimation of fault parameters in real-time
Fault Detection and diagnosis
Operator
OPERATION

Fig. 11.4: *Model-based fault detection using state estimation techniques compliant with the FMI standard.*

an FMU for co-simulation and used during the operation of the plant within the FDD module, for representing the expected operation, and for computing the differences in energy and cost between faulty and correct operation. Python was used for the workflow automation, which consists of reading measured data from a data base, executing the state and parameter identification, which in turn executes the FMU for different inputs, initial states and parameters, and writing the results to a database for display to the operator in a web-based GUI. Fig. 11.5 shows parts of the GUI with monitored and estimated Coefficient of Performance (COP) of the chiller. The results of the state and parameter estimation algorithm are compared to the COP of the calibrated model. One of the advantages of this approach is that it provides a statistical description of the performance, and hence it allows to set fault thresholds based on a probability.

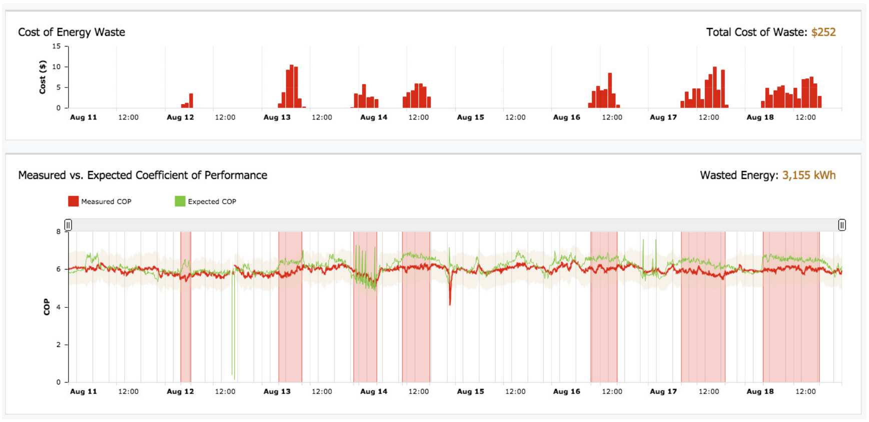


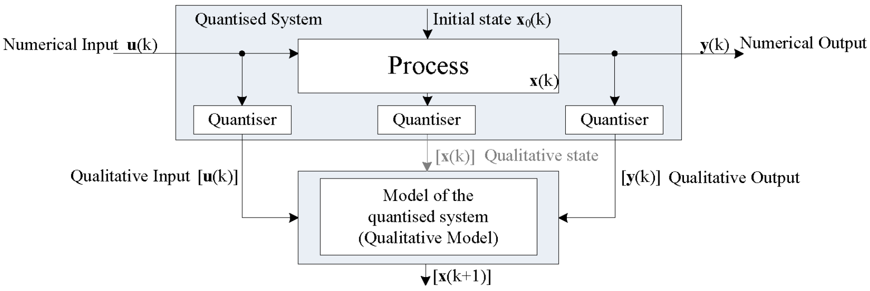
Fig. 11.5: Outputs of the fault identification algorithm, with costs caused by the fault (top graph) and comparison between measured and expected COP (bottom graph). The red rectangles are time intervals that have been identified as faulty periods in which the performance of the chiller was outside the tolerance.

Using Modelica allowed reusing a variety of component models from the Modelica `Buildings` library to build subsystem models of the chilled water plant, such as for the chillers and the cooling towers, which were then calibrated with measured data. After the calibration, these models were exported as FMUs in order to provide a model for the FDD algorithm. The use of FMUs has advantages in terms of compatibility and functionality. First, as FMI is a vendor-independent application programming interface for simulators, it allows the use of models authored by various modeling environments, thereby making the FDD framework independent of the model authoring tool. Second, the use of FMU allows to easily set values for the states, parameters and inputs for the repetitive simulations required by the FDD algorithm, as well as storing of final states for reuse as initial states for a new simulation.

11.4.2.2 FDD for Air Handling Units Components Using Qualitative Approaches

In contrast to quantitative models, qualitative models describe the behavior of a system only roughly. Instead of numerical values, qualitative models can deal with a symbolic representation of the system inputs, outputs and states, i.e. they can be represented by arguments like "high" or "low"

The process shown in Fig. 11.6 is a dynamic discrete-time, continuous-variable system whose qualitative behavior is described by a quantized system.



Quantised System
Initial state $\mathbf{x}_0(k)$
Numerical Input $\mathbf{u}(k)$
Process
$\mathbf{x}(k)$
$\mathbf{y}(k)$ Numerical Output
Quantiser
Quantiser
Quantiser
$[\mathbf{x}(k)]$ Qualitative state
Qualitative Input $[\mathbf{u}(k)]$
$[\mathbf{y}(k)]$ Qualitative Output
Model of the quantised system (Qualitative Model)
$[\mathbf{x}(k+1)]$

Fig. 11.6: Quantized System [S03].

We will now describe the qualitative approach that uses a stochastic automaton (SA) as the qualitative model. The qualitative model describes the qualitative behavior of the quantized system (see Fig. 11.6). Important work in this field has, e.g., been done by *[Lun94]*. Because the complexity of the so-called behavior tensor which stores the conditional probabilities of the state transitions of the automaton increases rapidly with a rising number of inputs, outputs and state signals, solutions for reducing the computational efforts and storage amounts are required. As shown in *[MKLR15]*, the complexity of the behavior tensor of the SA can be reduced by exploiting the underlying tensor structure of the behavior relation. Therefore, a non-negative canonical polyadic (CP) tensor decomposition can be used to make qualitative models applicable to large discrete-time systems. Usually, qualitative models can be generated by an abstraction from a quantitative model or by stochastic qualitative identification. In our application, we will focus on the latter which has been developed by *[Lic98]*. The stochastic qualitative identification allows the generation of a qualitative model directly from measurement data or from simulation data. For both cases, for Fault Detection (FD) nominal data of the faultless system behavior is needed. Fault Detection and Diagnosis (FDD) requires also fault models or faulty measurement data. *[LS96]* used a qualitative observer for fault detection and diagnosis. Qualitative observation means the prediction of the possible system states at discrete time $k+1$ based on the measured input and output combination at time k. In general, the qualitative observer is the qualitative equivalent of e.g. the Luenberger observer known from continuous-time systems. The algorithm yields for each time step a probability vector describing the possible behavior of the system states for a given input.

If the probability vector contains only zeros, the measured input-output combination is inconsistent with the qualitative model and a fault can be structurally detected, *[Lic98]*. For demonstrating FD with qualitative models, different faults have been simulated with the Modelica fault triggering library, developed by the German Aerospace Center (DLR), *[vdL14]*. In the following, an application example of a heat exchanger of a HVAC&R system shows the applicability of the qualitative modeling approach.

We will now present an example. The considered system is a Heat Exchanger (HX) that cools air. It has been simulated with Modelica for the input signals shown in Fig. 11.7 for nominal and faulty conditions.

The simulated fault describes a malfunction of the pump that leads to the problem of a fully opened valve, because the controller tries to track the set point of the air outlet temperature. Fig. 11.8 shows the faulty behavior during the time interval $2420 \leq t \leq 2708$. As the figure shows, the air outlet temperature of the HX equals the air inlet temperature because no water circulates.

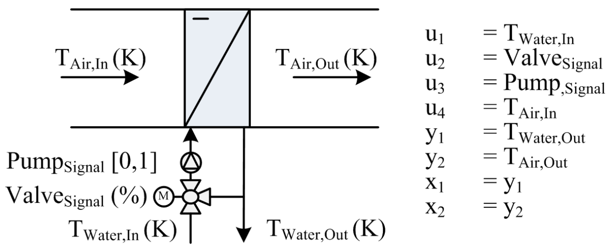


Fig. 11.7: *Generic HX scheme.*

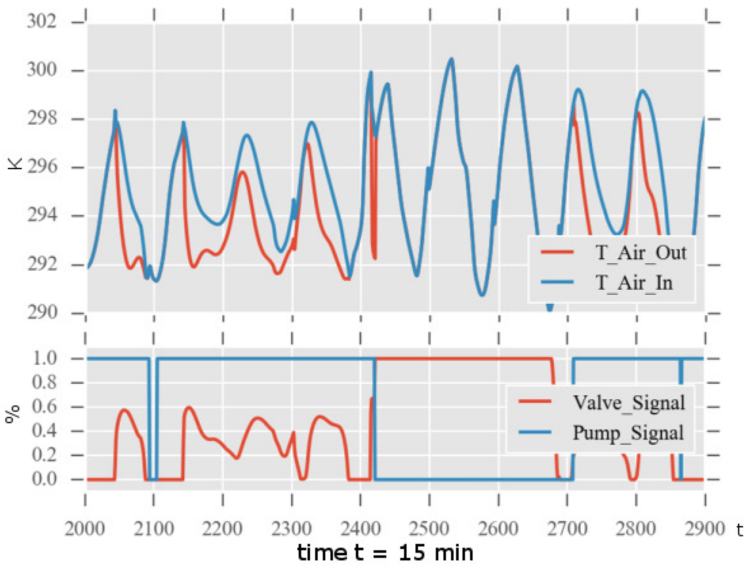


Fig. 11.8: *Simulation Data.*

Next, the qualitative model was trained using the nominal behavior of the HX as produced by the simulation model. Fig. 11.9 illustrates the qualitative state trajectory of the state variable $T_{air,out}$ for a selected time range.

The state trajectory computed by the qualitative observation algorithm and it shows the probability distribution of the state signal for each discrete time step k. The different grey shades of the bars denote the probabilities. In the example, the state signal $T_{air,out}$ was quantized into five intervals that can be seen by the horizontal separation of the black and grey bars.

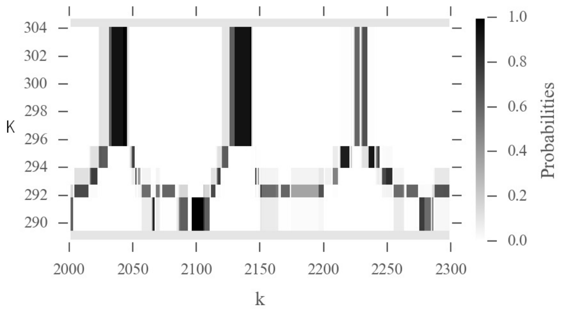


Fig. 11.9: *Qualitative state trajectory of the air outlet temperature $T_{air,out}$ (nominal condition).*

Fig. 11.10 shows the qualitative state trajectory for the faulty condition. During the faulty operation, the measured input-output pair is inconsistent with the qualitative model, leading to components of the probability vector being close to zero. This is visualized by the white space in the figure for $2420 \leq k \leq 2708$.

For a better visualization, Fig. 11.11 displays 1 for non-faulty conditions and 0 for faulty conditions.

While this example shows how fault detection with qualitative models can be realized, fault diagnosis is also possible. To achieve this, a set of qualitative models, where each model is trained with a different faulty condition has to be generated. Note that even for this simple example, the behavior relation of the SA contains over $3.9 \cdot 10^6$ values, leading to a significant calculations for each

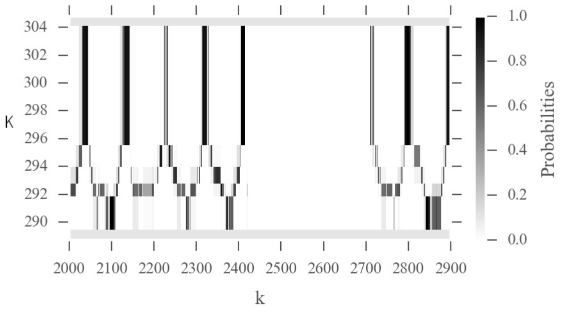


Fig. 11.10: *Qualitative state trajectory of the air outlet temperature $T_{air,out}$ (faulty condition).*

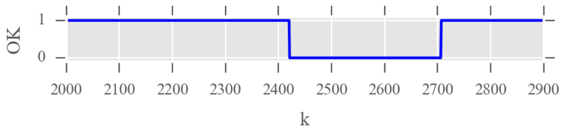


Fig. 11.11: *Probability of the air outlet temperature* x_2 *(faulty condition).*

discrete time step k. For building automation systems (BAS), such high computing demands may not be possible. We therefore used a CP-decomposition, which reduced the size of the behavior relation by a factor of 3200 from $3.9\dot{1}0^6$ to 1200. This met the computing capabilities of the BAS for real-time application.

We will now turn the discussion to another case study. This case study demonstrates the use of Modelica to support a qualitative model-based fault detection and diagnosis approach based on first-order logic and a generic diagnosis algorithms. Model-based diagnosis is based on an explicit representation of the knowledge about the components and the information about the plant structure, which determines how the components interact with each other. Based on a Modelica library of generic component models representing first-principles of energy and mass transfer between components of the AHU and the representation of the AHU topology, a system model, possibly covering both the nominal and faulty behaviors, can be obtained. This model is then used by a generic diagnosis algorithm, which is neither plant- nor domain-specific. The diagnosis models used in qualitative model-based diagnosis are obtained by using the Modelica model to capture acceptable deviations of variables from their respective nominal behavior. Instead of using a stochastic automata, this method uses an inferencing system based on first-order logics to generate the diagnosis.

In Fig. 11.12, a complete workflow and system modules are presented that are required to build a diagnostic solution for a class of plants, such as HVAC systems, and to deploy and run it for a single plant. This process is referred to as qualitative model-based diagnosis. Here, we give an overview of the steps and modules. The steps are described in detail in *[SPF+14]* and are summarized as follows:

- Producing the general solution (top row) involves:
- The production of a library of Modelica models of the components (e.g. first principle models) *[FSTK13]* and
- its transformation into a qualitative diagnostic model library.
- Producing an application system (middle-row), based on the general solution, which requires:
- The configuration and calibration of the Modelica models of the correct

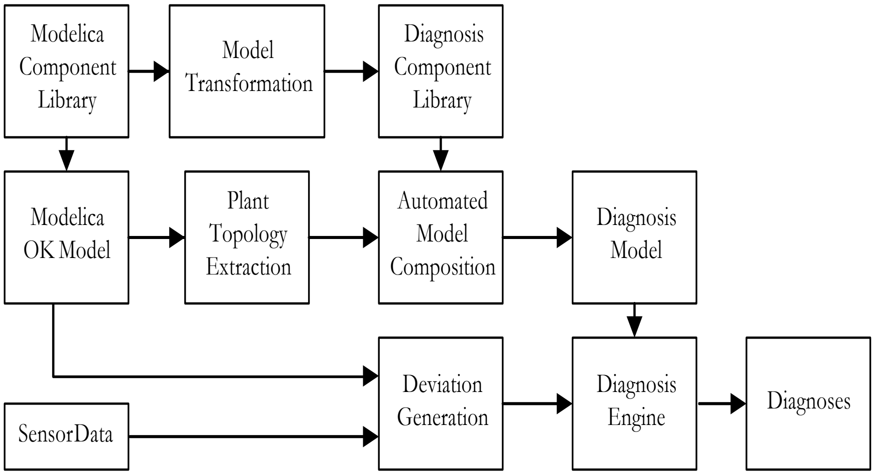


Fig. 11.12: Fault Detection of Air Handling Unit Components in a Qualitative Approach Workflow.

behavior *[FSK15]*, and

- the composition of the diagnostic model based on the diagnostic library and the topology of the plant, which can be extracted from the Modelica system model.

- For on-line diagnosis (bottom row):

- Computing the difference between the real data and the predictions generated by the Modelica model of the plant, and comparing the difference with given thresholds *[SPF+14]*, and
- diagnosing the differences, using a runtime diagnosis engine such as Raz'r *[OCCM14]*, which is an implementation of consistency-based diagnosis *[MS96]*. The output of the consistency-based diagnosis engine is a minimum set of combinations of component faults.

In this example, models were developed and applied to support model-based diagnosis. Use of Modelica led to a clear understanding of the underlying physical equation system since the equations within the component model and also on the level of the system model (connect equations) are clearly defined and can be easily reconstructed by the user of the model.

The main drawback of Modelica tools for this case study was the lack of native tools to interact with simulations during run-time. A python-Modelica interface needed to be developed. However, this need can nowadays be overcome by the use of FMI, for which Python interfaces are available.

11.4.2.3 FDD for Air Handling Unit Systems Using a Quantitative Approach

This case study focuses on the application of an open and replicable FDD method. The method consists of a combination of a Modelica model-based fault detection and a classifier-based diagnosis. It analyzes the monitored data on a real-time basis where a calibrated Modelica model evaluates the outputs from the outdoor conditions and the control signal and compares the estimation against the measurement to detect errors.

The method was used for component failure detection in an AHU, focusing on a prospective malfunction of the dampers, the heat recovery system and the fans. The AHU is part of a comprehensive test facility located on the KU Leuven Technology Ghent campus in Belgium. The facility is equipped with its own weather station measuring global solar irradiation, relative humidity, temperature, precipitation, wind speed and wind direction. A set of embedded sensors keep track of the AHU parameters such as the air volume flow, air temperature and relative humidity in the supply air and in the return air. The damper position signals are monitored as well. The monitored data are centralized and stored with a one minute frequency in a Soft-PLC based monitoring system where the BMS and the FDD algorithm is integrated on the same industry PC hardware.

As depicted in Fig. 11.13, a semi-automated BIM based process has been utilized to generate the Modelica model used for fault detection. It relies on a Python framework to extract and convert BIM data into a Modelica component compatible format. In this context, the BIM-based modeling approach is facilitated by the component-oriented feature of Modelica, which in turn allows an easy mapping between BIM objects and Modelica components. Data from the building delivery manual are used as input for the remaining parameter values not available in the BIM. An optimization-based calibration process is used to calibrate the Modelica model where a data set obtained throughout the initial

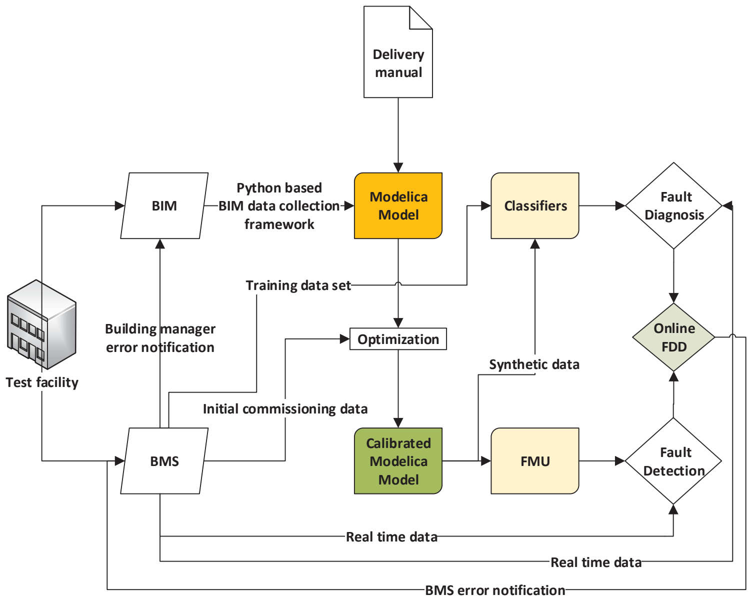


Fig. 11.13: Fault Detection of Air Handling Unit Systems in a Quantitative Approach Workflow.

commissioning process and assumed as fault free is utilized.

The Modelica model is exported as an FMU to be integrated into the BMS for a near-real time estimation of the outputs (e.g. supply air temperature). A discrepancy between the model estimation and the measurements superior to a predefined threshold is a sign of a likely issue on the system. To diagnose the error cause, the classifiers estimate the set of control signals which could provide the current output value as outcome. The cause of an error is isolated in a component if a discrepancy between the estimated and the actual control signal within this specific component is detected. In addition to the fault-free initial commissioning data, the classifiers are trained using synthetic data which represent several operating scenarios generated from the calibrated Modelica model.

An error notification is sent to the BMS if the algorithm detects the fault and its root cause is determined. An error notification is sent to the building manager if the occurrence of the error spans over a predefined period of time (e.g. 2 days). A BIM based error notification has been developed where the error location is pinpointed within the virtual representation of the building.

In this project, the use of Modelica was motivated by its ability to model the dynamic behavior of the components of the AHU. In addition, as an open language and considering the availability of open-source and validated libraries dedicated to buildings, Modelica fits well with developing an open FDD method.

11.4.3 Hardware in the Loop

Lastly, this section describes a hardware in the loop system. In this study, the Modelica `Buildings` library *[WZNP14]* is utilized to model an HVAC system that was used in a HIL setup to test the performance of closed loop control algorithms. During this project, the interface used to run Dymola models on the dSpace HIL simulator was still in development and not yet sufficiently reliable for utilization in this project. This was circumvented by importing the Dymola-generated code for the model in Simulink, and then using Simulink's code generation functionality.

The pieces of equipment chosen to complete the loop in this study include a

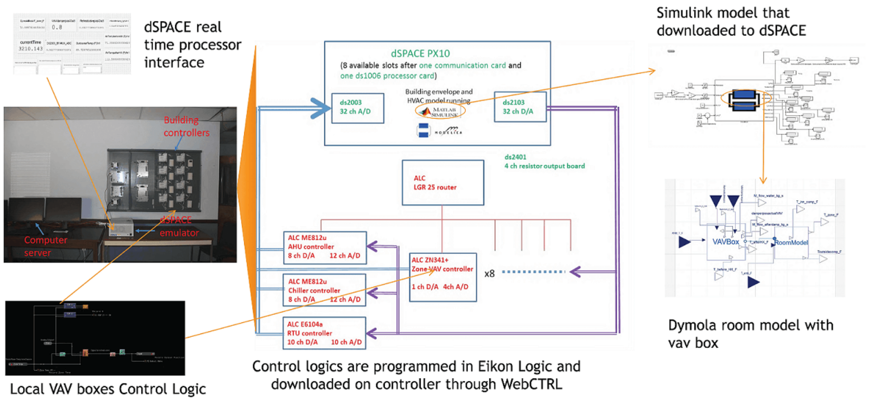

dSPACE real time processor interface
Building controllers
dSPACE emulator
Computer server
Local VAV boxes Control Logic
dSPACE PX10
(8 available slots after one communication card and one ds1006 processor card)
ds2003 32 ch A/D
Building envelope and HVAC model running
ds2103 32 ch D/A
ds2401 4 ch resistor output board
ALC LGR 25 router
ALC ME812u AHU controller 8 ch D/A 12 ch A/D
ALC ME812u Chiller controller 8 ch D/A 12 ch A/D
ALC E6104a RTU controller 10 ch D/A 10 ch A/D
ALC ZN341+ Zone VAV controller 1 ch D/A 4ch A/D
x8
Control logics are programmed in Eikon Logic and downloaded on controller through WebCTRL
Simulink model that downloaded to dSPACE
VAVBox
RoomModel
Dymola room model with vav box

Fig. 11.14: Hardware in the loop workflow.

dSpace processor to execute models and a set of programmable VAV terminal controllers from Automated Logic Corporation (ALC). Additionally, digital-to-analog (D/A) and analog-to-digital (A/D) boards are used to establish communications between the dSpace and the controllers.

The model workflow shown in Fig. 11.14 mimics a room served by a VAV box with a reheat valve and a damper. This study focuses only on sensible heat transfer in buildings. Therefore, there is no humidity control and the air is assumed to be dry air. An AHU provides simple dry air at a prescribed temperature and sufficient pressure to overcome losses due to the damper and heat exchanger. The air is given a nominal flow rate, which serves as the flow rate of air with the damper fully open. From the AHU, the supply air flows through a heat exchanger block (i.e., reheat coil) with a constant effectiveness. Reheat is provided using a water-to-air heating coil, controlled by a linear two-way valve. Finally, the VAV damper is modeled using an exponential damper module from the Modelica `Buildings` library. The nominal supply airflow rate is set inside this module, and varies exponentially down to zero as the damper is closed. External signals are used to control both valve position on the reheat water loop and supply air damper position, as these signals will be received from the ALC controllers in the HIL simulation through an A/D board. Ideal flow and temperature sensors are placed throughout the model to monitor properties and ensure correct performance. Such sensor information was sent to ALC controllers through a D/A board. Finally, parameters that are not yet defined are set as inputs to the system. These will be declared later in Simulink, allowing them to be changed during real-time testing.

The room model in Fig. 11.14 describes a room using a thermal resistance-capacitance (RC) model and is modified from a RC model in the Modelica `Annex60` library *[WvTH13]*. A capacitor for each surface represents thermal energy storage, followed by conduction and internal convection resistors. The ambient temperature is an input to the system, while the ground temperature is held at a constant value. Fluid ports allow the air from the VAV box to enter the room and return the air to an exit, maintaining the mass balance. Therefore, a fixed volume of air is maintained in the room at all times. There are a variety of assumptions made in this model, including the neglecting of infiltration and handling of outdoor convection as constant.

The combined thermal zone and VAV box model was downloaded to dSpace for the real-time execution. As previously mentioned, the model was first imported in Simulink for the code generation. This is possible using a Dymola-to-Simulink interface called DymolaBlock. This feature allows the model to be inserted into Simulink as a functional block whose outputs are calculated from defined inputs. In order to complete the loop, some parameters must be associated with D/A and A/D boards, and therefore either read from or sent to the real controllers. In this study, the room temperature is sent to the controller and the reheat valve position is returned for the heating model simulation. A dSpace blockset allows these communications to be defined in Simulink. Before building the model to code, a fixed step integrating method was selected. The model is then compiled to C code using Matlab, and the variable description files are downloaded onto dSpace for the real-time testing.

This HIL setup was used to test a conventional VAV box control called single maximum logic *[TSPC12]*. The outdoor air temperature was set to be very cold and the VAV box always stayed in the heating loop. Therefore only reheat valve position had to be controlled. The control was accomplished by using a PI block. Zone temperature and setpoint were compared to determine an appropriate reheat valve position, which was then sent to the Modelica model. The VAV box changed the model accordingly and solved for zone temperature at each time step, writing this value (sensor information) to the controllers to complete the loop. Both zone air temperature and reheat value position were

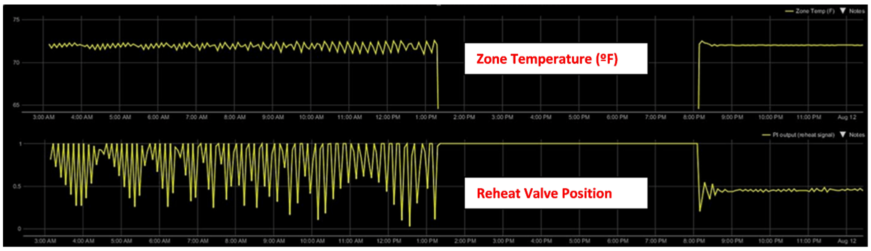


Fig. 11.15: Hardware in the loop results.

trended and recorded in the BEMS (Fig. 11.15).

As can be seen in Fig. 11.15, the desired setpoint of the room was 72 Fahrenheit. The simulation was stopped around 1 pm and later resumed, with different PI loop tuning to illustrate the difference this can make. Before the re-tuning, wild fluctuations in the reheat valve were experienced. If this control were implemented in a real system, the constant opening and closing of the valve would lead to noise, wasted energy, discomfort, mechanical wear and eventually failure.

After a more optimal tuning was applied to the PI block, a classic underdamped oscillator curve can be seen. The reheat valve position ceases to undergo oscillatory changes, and finds a steady value to maintain room temperature at the setpoint. Reversals in signal can still be seen, but they are minimal compared to the prior tuning and are practically unavoidable.

During this study, sometimes compatibility issues have been encountered between the building code for dSpace and the Dymola-to-Simulink interface. This only occured with certain models, such as MixedAir from the Modelica `Buildings` library, and we have not yet located the root cause of these issues. These models work fine in Simulink but return a Simstruct Mex error during code generation. FMI usage should avoid such issues in future implementations.

11.5 Discussion and Conclusions

The use of Modelica to support the operation of building energy systems is promising as buildings involve multiple physical phenomena (e.g., heat transfer, fluid dynamics, electricity, etc.) and are complex in terms of their dynamics (e.g., coupling of continuous time physics with discrete time and discrete event control). Some clear advantages in the use of Modelica for supporting model use during operations include:

- The open source characteristic combined with the existence of several freely available Modelica libraries *[BDCVR+12]*, *[LTF+14]*, *[NGHLR13]*, *[WZNP14]* allows for models to be modified and extended depending on the application requirements;
- Object-orientation not only enhances modularity and reusability of the models but also allows models to be developed following the physical system structure, which makes them easier to understand. The object-orientation enables extension and reuse of components and the use of standardized interfaces enables collaboration across physical domains and disparate developer groups.
- The Modelica object-oriented approach also allows for the development and the management of large and complex models. In such large-scale applications, the translators, the modeling language and environment are significantly stressed, and their robustness proved an enabling factor of the overall modeling process. This makes the Modelica toolchain well suited for handling highly complex models from the end user and the engineering perspective.
- The possibility to import and export Modelica models as Functional Mock-up Units (FMUs) enables the integration of models using a standardized, tool-independent API into existing FDD routines or the development of integral solutions that couple tools for data analysis, simulation, FDD and optimization in one single environment. An integral solution can be realized, for example, using the Building Controls Virtual Test Bed (BCVTB) *[Wet11a]* or JModelica with the Python module PyFMI *[AAG+10]*.
- Modelica allows for a seamless use of the models developed in the design phase during the operational phase, for example, by exporting models as FMUs.

Finally, it is important to note that the advantages of Modelica can turn against the unexperienced developer. For example, because of the object-orientation employed in many libraries, it can be difficult to predict the depth to which a change in one component can have an effect in other components in the library. However, regression tests as are setup for the Modelica `Annex60` library can detect such unintended side effects. In addition, the current capabilities of Modelica IDEs are still developing means to provide better debugging information. Also, training is highly recommended for novice users as the type of model verification and debugging done in equation-based languages differs from what users may be accustomed to when writing procedural code.

Chapter 12

Management and Dissemination

The IEA EBC Annex 60 was conducted from 2012 to 2017 through the collaboration of 42 institutes from 16 countries. The research phase of Annex 60 started on July 1, 2013. The Annex benefited from a strong organizational framework in order to coordinate work between the various researchers worldwide.

12.1 Coordination and Meetings

12.1.1 Semi-Annual Expert Meetings

During the research phase, eight international, semi-annual expert meetings were successfully held and most of the expert meetings were followed by technical workshops over multiple days:

- on March 11-12, 2013 at RWTH Aachen, Germany, during the planning phase (50 attendees),
- on August 23-24, 2013, in Aix-les-Bains, France (54 attendees),
- on March 8-9, 2013, in Lund, Sweden (50 attendees),

- on September 15, 16 and 17, 2014, in Berkeley, USA (50 attendees),
- on April 21 and 22, 2015, in Galway, Ireland (50 attendees),
- on September 16-18, 2015, in Leuven, Belgium (50 attendees),
- on May 9-11, 2016, in Miami, USA (30 attendees),
- on October 22-23, 2016, in Porticcio, France (65 attendees).

12.1.2 Coordination Meetings

In order to synchronize work packages and activities, more than 120 coordination meetings and web conferences were conducted within the activities and among the leaders since the start of Annex 60. Minutes of these activities and meetings were continuously organized and disseminated using the Annex-internal bitbucket repository system. The repository system, which included a wiki, was a key resource for organizing collaboration.

12.1.3 Shared Content Management System

In order to support joint collaboration, documentation, report and code management, a cross-platform, cloud-based repository system, called Bitbucket, was utilized and maintained by the operating agents. The system provides version control using git and is recommended for managing future Annex projects.

12.2 Publications and Outreach

12.2.1 Annex 60 Joint Publications

Besides this final report, Annex 60 published 11 journal articles, 38 conference papers and a Modelica library, which are all accessible at http://www.iea-annex60.org/pubs.html.

12.2.2 Outreach

Several special scientific tracks at national and international conferences were organized to disseminate and promote results of the entire project. In order to represent the Annex in a consistent manner, a common text for the acknowledgement section of papers was formulated and distributed to all participants. The Annex 60 project gained a high international visibility through, among others, the following outreach activities:

- multiple announcements in the Newsletters of the International Building Performance Simulation Association (IBPSA),
- announcements in the Newsletter of its US-Affiliate IBPSA-USA,
- announcements in the Newsletter of the Modelica Association,
- presentation at the SimBuild 2012 conference in Madison, WI, USA,
- presentation of the Annex at the International Symposium “EnTool 2013 Symposium, Workshop & Summer School” in Dresden, Germany,
- presentation at the Modelica North-America User Meeting in Ann Arbor, Michigan, in May 2013,
- presentation at the IBPSA-USA meeting in Dallas, TX, in January 2013,
- presentation at a special conference track at the international Building Simulation conference in August 2013 in France,
- implementation of a special conference track at the BauSIM 2014 conference in Aachen, Germany,
- receiving the Best Paper Award for an Annex 60 paper out of 100 presentations at BauSIM 2014,
- implementation of a special conference track at the ASHRAE/IBPSA-USA Building Simulation Conference in 2014,
- presentations at the SIAM Annual Meeting in the USA in 2014,
- organization of special tracks at the Building Simulation 2015 conference in Hyderabad, India, where 28 joint papers were presented from the Annex 60 framework,
- presentations at a public workshop sponsored by the Netherlands and Flanders chapter of the IBPSA,
- presentations in 2016 at a public Annex 60 workshop with hands-on training by IBPSA-France,
- presentations at the Clima 2016 conference in Aalborg, Denmark,
- presentations at the ASHRAE and IBPSA-USA SimBuild 2016 confer-

ence in Salt Lake City, Utah, USA,
- presentations within a special track at the BauSIM 2016 conference in Dresden, Germany,
- invited presentation about Annex 60 and IBPSA Project 1 at the IBPSA-England BSO 2016 conference in Newcastle, England,
- presentation at the Modelica North America Users' Group meeting in Troy, Michigan in 2016,
- invited presentation at the international conference of the European Energy Research Alliance (EERA) in 2016,
- presentations at the Modelica conference in Prague in 2017, and
- presentation of the Annex 60 / IBPSA Project 1 continuation framework at the Building Simulation 2017 conference in San Francisco, USA.

Furthermore, progress of the Annex 60 framework was disseminated in a number of issues of the IEA EBC News.

12.2.3 Annex 60 Web Site

The public Annex 60 web site is accessible at http://www.iea-annex60.org. This page also host papers that resulted from the Annex 60 collaboration, lists participants and their respective contact information.

Chapter 13

Conclusions

13.1 Technology Development

The technology development in Annex 60 was organized around three standards:

1. IFC for data modeling,
2. Modelica for multi-domain, multi-physics modeling, and
3. FMI for run-time interoperability of simulators.

Basing the work on these standards was critically important to enable joint technology development among multiple participants, many of whom brought into Annex 60 their existing code, further developed it within the project, and then integrated it back into their tools. A prime example of technology developed around these standards is the Modelica `Annex60` library developed in Activity 1.1. In this work, multiple institutes started a collaborative development of a free, open-source Modelica library for building and district energy systems. This work harmonized the previously fragmented and duplicative development of libraries, and resulted in a jointly developed library that is now used by four major Modelica libraries for building systems. As part of this process, the four libraries not only grew in their functionality, but also improved their robustness, validation and documentation. Getting to this point required the developers of previous libraries, each having a considerable code base, to mutually agree upon common processes for development and quality control. These processes needed to allow for rapid experimentation, as is often done in University settings, as well as ensure robust and stable development, which is more important for commercial software companies and government laboratories. The joint development also required the developers to agree upon various design decisions and conventions for coding, documentation and validation, to jointly work on implementation and vetting of a core of a library, to refactor their existing libraries, and to open-source previously proprietary code. This is the first international collaboration for a library with free, open-source models for buildings and district energy systems that are built using an open-standard modeling language. It initiated a larger open-source development that will be further supported though IBPSA, whose vision includes providing a standard library with fundamental model descriptions that will be supported by manufacturers and integrated in various building performance simulators.

A second example of technology developed using the previously mentioned standards is the development of building and district energy simulation tools based on the FMI standard. A key advantage of using the FMI interface is that it decouples the model authoring from the simulation run-time environment, allowing quite different approaches to be used to integrate the models in time. Over the course of the project, different FMI simulators not only emerged from within Annex 60, but also from within the significantly larger FMI community. Within Annex 60, using the FMI standard as an Application Programming Interface (API) to different simulators allowed run-time coupling of various simulators, prototyping of an interface that allows energy and control models authored in FMI for co-simulation to be integrated with a commercial building automation system for real-time control or real-time energy monitoring, and hardware-in-the-loop testing of building control sequences. From the technology development point of view, such a decoupling between model authoring and simulation, and work with an open standard, has the key advantage of being able to develop, prototype, and test different simulators on common models, while keeping the time and cost to adapt a new simulator low. This is compared to what is done in most modeling and simulation tools, which have a tight integration of models and solvers. While it has been shown that the FMI standard is well-suited for the classes of problems encountered in building and district energy simulation, certain limitations were encountered, as is common with any standard that covers such a complex use case, and improvements have been recommended. Some of these recommendations have already been addressed by the FMI standards committee. Overall, using the well-developed FMI standard, together with its development process that ensures stability, gave rise to the ecosystem of building and district energy simulation tools developed as part of this project.

A third example of technology development using these open standards is that which was done to support processes that transform digital planning and design to simulation. In Annex 60, a mechanism was developed to transform a digital model of a building and its energy systems to Modelica code, which can then be readily used for advanced building performance simulation. This was accomplished through the use and extension of the Open BIM data formats defined by the Industry Foundation Classes (IFC) as well as through the use of other BIM-standards, such as the Information Delivery Manual (IDM) and Model View Definitions (MVD). Annex 60 thoroughly addressed the prevailing tedious, cumbersome and error-prone process of manual data conversion and model generation by providing a methodology and software framework for automatically, or at least semi-automatically, transforming a digital model into an object-oriented acausal model. The software framework developed in Annex 60 supports the Open BIM format IFC. Models are checked for integrity in terms of geometric consistency and HVAC definition. Using a flexible module for schema parsing, models are transformed into the intermediate data format SimXML. To manage these SimXML data, a dynamic schema parser in C++ with an API between C++ and Python was developed to interact with the data model. An object and parameter mapping mechanism as well as respective mapping rules were defined to formulate engineering knowledge in a rule-based methodology. These rules are processed by the framework. Furthermore, an IFC MVD was published in order to specify the subset of IFC data relevant to building performance simulation. To support multiple Modelica libraries, we selected a template-based approach and implemented it in Python. As a result, the software framework of the translator from BIM to Modelica can generate models using the four different Modelica libraries, using the same software stack with small adaptations. The developed methodology was tested for a set of use cases. Annex 60 thereby followed a bottom-up approach which was based on these use cases. The framework supports the whole model checking and model transformation process but is currently limited to these use cases. The result is a modular framework which is open for further development and dissemination.

In addition to the use of open standards, workflow automation also played an important role in technology development, both for model development and also for case studies. Due to the large ecosystem of free, open-source Python packages, Python-based tools were further developed and used in the case studies, and documentation and workshops for novice users were developed. An example for workflow automation during model development is the regression testing that we setup for the *Annex60* Modelica library. This library consists of $2,400$ files. During any change to a model, more than $120,000$ result points are compared to verify that they are either identical to or expectedly different from results computed with an earlier version, within a tolerance of 10^{-3}. Furthermore, daily tests are run to compare all of these points among the two simulators Dymola and JModelica to make sure that the library produces the same result with different simulators. Regarding workflow automation for case studies, many case studies used optimization, stochastic simulations, or coupled multiple simulation tools. For such applications, automating the workflow was critical to produce reliable and repeatable results.

Subtask 2 demonstrated how these technologies can be used for the design and operation of building and district energy systems. In many of these applications, models from Activity 1.1 were combined with FMI-tools from Activity 1.2 and workflow automation scripts from Activity 1.4, and with open-source and commercial software that has been developed outside of Annex 60, to solve problems related to the design and operation of building and district energy systems. The case studies showed that very low energy systems and increased grid integration imposes structural changes to building and community energy modeling and simulation tools and processes. For example,

- Models of different physical domains, control systems, occupant behavior, and occupancy need to be combined for dynamic, multi-physics simulations that involve electrical systems, thermal systems, flow distribution, controls and possibly communication systems. Such models typically evolve at vastly different time scales.
- Tools need to properly handle control sequences and hybrid systems in which the states evolve in time based on both continuous and discrete time semantics that arise from physics and digital control, respectively.
- Subsystem models need to be extractable in order to export and execute

models in a self-contained form in a building automation system, or a hardware-in-the-loop setup.

- Model equations need to be accessible in order to perform model order reduction, model linearization, extraction of a linear model in state-space representation, and to solve optimal control problems.

The case studies showed Modelica is well-suited to address the above problems. Various case studies combined thermal, electrical and control models for projects at the building and district scales. While Modelica tools generate very efficient C-code for simulation, simulating large systems can still be a challenge. Research in translation of large models and in multi-rate solvers with adaptive time step control is ongoing and needed to apply this technology to large systems.

As part of analyzing thermal district energy systems, it becomes evident that a standardized validation test, similar to ANSI/ASHRAE BESTEST 140, but for district systems, would be immensely valuable. Currently, there is neither a test, nor a common understanding of what is meant by a district energy simulation. Approaches in industry range from simple spread-sheet based analysis that disregard temperature levels, energy storage and pressure drops, to fully coupled simulation of buildings, thermal distribution networks, flow friction and feedback control. In absence of a test procedure for district models, it is hard to judge how credible the tool and approach that was selected by the modeler is. A first start for such a test was done in Annex 60, and it is planned to continue this effort within the IBPSA Project 1.

Annex 60 participants who worked on model use during operation reported that the object-orientation of Modelica not only enhances reusability of the models, but also allowed models to be developed following the physical system structure, which makes them easier to understand. Moreover, the possibility to import and export Modelica models as FMUs enabled the integration of models using a standardized, tool-independent API into independent or combined solutions for data analysis, simulation, fault detection and diagnosis, and optimization now and in the future.

In summary, had our work not been based on such open standards, such a tight collaboration and integration of software among widely ranging use cases would not have been possible. Moreover, analyzing multi-physics problems such as the integration of thermal and electrical systems, overlaid with controls that coordinates the two, would have been very difficult without Modelica's support for multi-physics modeling, and FMI's support for co-simulation using domain-specific tools.

13.2 Governance and Dissemination

The Annex 60 benefited from a strong organizational framework to coordinate work between the various researchers world-wide. Eight semi-annual international expert meetings were conducted, most of them followed by technical workshops over multiple days. To synchronize work packages and activities, more than 120 online coordination meetings and web-conferences were held within the activities and among activity leaders. A Bitbucket repository system was used, including a wiki, as a key resource to organize collaboration and manage shared documents. Several special scientific tracks at national and international conferences were organized to disseminate and promote results of the entire project, which helped to gain a high international visibility. Annex 60 published 11 journal articles, 38 conference papers, a Modelica library and diverse source code packages, all managed through a Git code repository.

13.3 Continuation Framework

Although the Annex has delivered a solid basis of tools and demonstrated their application, tool development is a continuous effort. It continuously needs to provide support, respond to new system technologies and apply advances in computer science and applied mathematics. Upcoming research and development issues that remain to be solved after completion of the Annex will therefore be coordinated under the umbrella of the network of the International Building Performance Simulation Association (IBPSA) as a living and supporting dissemination framework.

Chapter 14

Glossary

This section defines various technical terms that are used throughout the report.

Building Information Model (BIM) A Building Information Model (BIM used as a noun) is an instance of a data model that describes a building and its components unambiguously, e.g. a digital footprint of a building and its properties, represented over its entire life cycle.

Building Information Modeling (BIM) Building Information Modeling (BIM used as a verb) is a collaboration method to consistently acquire, manage, exchange and hand-over information of a building which are relevant over its entire life cycle, using digital models and targeting a consistent communication between parties.

Business Process Modeling Notation (BPMN) Graphical notation to structure and display processes to formally and descriptively describe tasks, responsibilities and informations within IDM. So-called swim-lanes are used to graphically group elements, which belong to an actor respectively.

Industry Foundation Classes (IFC) Object oriented data format for digital description of building information models. Open BIM data exchange standard developed by buildingSMART published as ISO 16739. Physical data exchange as Step (Express) or ifcXML data file.

Information Delivery Manual (IDM) Method standardized as ISO 29481 to capture and specify processes and to define information exchange requirements in BIM. Usually as formal description of planning processes using Busines Process Modeling Notation (BPMN).

Model View Definition (MVD) Part or subset of the IFC scheme which is defined in order to fulfil a single or multiple domain-specific exchange requirements. Typically, IDM serves as method to define these exchange requirements.

package In Modelica, a package is a collection of models, functions, or other packages that are used to hierarchically structure a Modelica library.

class In Modelica, a *class* is a term that includes models, blocks, functions and packages.

connector In Modelica, a connector is an instance of a class that is used to connect models and blocsk with each other. Connectors can contain constants, parameters and variables, but no equations.

co-simulation Co-simulation refers to a simulation in which different simulation programs exchange run-time data at certain synchronization time points. A master algorithm sets the current time, input and states, and request the simulator to advance time, after which the master will retrieve the new values for the state. Each simulator is responsible for integrating in time its differential equation. See also *model-exchange*.

block In Modelica, a block is a special case of a model in which all connectors are either inputs or outputs.

events An event is either a *time event* if time triggers the change, or a *state event* if a test on the state triggers the change.

model In Modelica, a model is a special class in which the causality of its connector need not be specified.

model-exchange Model-exchange refers to a simulation in which different simulation programs exchange run-time data. A master algorithm sets time, inputs and states, and requests from the simulator the time derivative. The master algorithm integrates the differential equations in time. See also *co-simulation*.

zero-crossing function A zero crossing function is a function that is used by solvers to indicate when variables cross a threshold. For example, suppose a thermostat should switch off a heater when the room temperature $T_r(t)$ crosses a set point $T_s(t)$. For this case, a zero crossing function is $f(t) = T_r(t) - T_s(t)$.

Zero-crossing functions are used by numerical solvers to detect where such discrete changes in a model occur. Specifically, at $t = t_1$, they search for $\tau \in \arg\min\{t > t_1 \mid f(t) = 0\}$.

direct dependency A variable is said to directly depend on another variable if there is an algebraic constraint between these variables. For example, consider the model shown in Fig. 14.1. If the input u changes, then the output y_3 immediately changes, whereas y_1 and y_2 only change after time is advanced. Hence, the output y_3 is said to have a direct dependency on u. Similarly, the state derivative $\dot{x}_1$ directly depends on the input u, and $\dot{x}_2$ directly depends on the state x_1. The outputs y_1 and y_2 directly depend on the states x_1 and x_2, respectively. Hence, unless time is advanced, if u is updated, only y_3 needs to be updated, whereas if y_1 and y_2 need not be updated. Such information is useful to detect the existence of algebraic loops, and to implement certain numerical time integration methods such as the QSS methods.

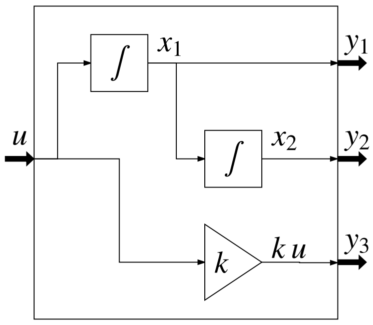


Fig. 14.1: Signal flow diagram that illustrates direct dependency.

Functional Mockup Interface The Functional Mockup Interface (FMI) standard defines an open interface to be implemented by an executable called *Functional Mockup Unit* (FMU). The FMI functions are called by a simulator to create one or more instances of the FMU, called models, and to run these models, typically together with other models. An FMU may either be self-integrating (co-simulation) or require the simulator to perform the numerical integration.

Functional Mockup Unit Compiled code or source code that can be executed using the application programming interface defined in the *Func-*

tional Mockup Interface standard.

rollback We say that a simulator is doing a rollback if its model time is set to a previous time instant, and all its state variables are set to the values they had at that previous time instant.

time event We say that a simulation has a time event if its model changes based on a test that only depends on time. For example,

$$y = \begin{cases} 0, & \text{if } t < 1, \\ 1, & \text{otherwise,} \end{cases}$$

has a time event at $t = 1$.

state event We say that a simulation has a state event if its model changes based on a test that depends on a state variable. For example, for some initial condition $x(0) = x_0$,

$$\frac{dx}{dt} = \begin{cases} 1, & \text{if } x < 1, \\ 0, & \text{otherwise,} \end{cases}$$

has a state event when $x = 1$.

Chapter 15

References

[316] AEC 3. BIM*Q. 2016. URL: http://aec3.de/kompetenzen/BIM-Q-Database.htm.

[ACDCH15] Arnout Aertgeerts, Bert Claessens, Roel De Coninck, and Lieve Helsen. Agent-based control of a neighborhood: a generic approach by coupling Modelica with Python. In *BS2015, 14th Conference of International Building Performance Simulation Association*. 2015.

[AJS14] Abdul Afram and Farrokh Janabi-Sharifi. Theory and applications of HVAC control systems - a review of model predictive control (MPC). *Building and Environment*, 2014. doi:10.1016/j.buildenv.2013.11.016.

[ANHB13] A. Afshari, G. R. Norouzi, G. Hultmark, and C. N. Bergsoe. Two-pipe chilled beam system for both cooling and heating of office buildings. In *Proceedings of CLIMA 2013: 11th REHVA World Congress and the 8th International Conference on Indoor Air Quality, Ventilation and Energy Conservation in Buildings*. Prague, Czech Republic, June 2013.

[Age06] International Energy Agency. Iea annex 34: computer aided evaluation of HVAC sytem performance. Technical Report, International Energy Agency, Hertfordshire, UK, 2006.

[Age14] International Energy Agency. World Electricity and Heat for 2014. 2014. URL: http://www.iea.org/statistics/statisticssearch/report/?country=WORLD&product=electricityandheat&year=2014.

[Alc16] Digital Alchemy. Digital Alchemy - Simergy Pro. 2016. URL: http://www.digitalalchemypro.com/html/products/DAProducts_Simergy.html.

[AB09] Alessandro Alessio and Alberto Bemporad. A survey on explicit model predictive control. In Lalo Magni, DavideMartino Raimondo, and Frank Allgöwer, editors, *Nonlinear Model Predictive Control*, volume 384 of Lecture Notes in Control and Information Sciences, pages 345–369. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-01094-1_29.

[AMH00] Klaus Kaae Andersen, Henrik Madsen, and Lars H. Hansen. Modelling the heat dynamics of a building using stochastic differential equations. *Energy and Buildings*, 31(1):13–24, 2000. doi:10.1016/S0378-7788(98)00069-3.

[And13] Joel Andersson. *A general-purpose software framework for dynamic optimization*. PhD thesis, KU Leuven, Leuven, Belgium, October 2013. URL: https://lirias.kuleuven.be/bitstream/123456789/418048/1/thesis_final2.pdf.

[ASH02] ASHRAE. Ashrae guideline 14. measurement of energy and demand savings. Technical Report, ASHRAE, 2002.

[ASH04] ASHRAE. *ASHRAE 90.1 - 2004 Energy Standard for Buildings Except Low-Rise Residential Buildings*. American Society of Heating, Refrigerating and Air-conditioning Engineers, si edition edition, 2004.

[ARHZ15] Emira El Asmi, Sylvain Robert, Benjamen Hass, and Khaldoun Zreik. A standardized approach to BIM and energy simulation connection. *International Journal of Design Sciences and Technology*, 21(1):59–82, 2015.

[Aut15] Autodesk. Revit Family - Overview. 2015. URL: http://www.autodesk.com/products/revit-family/overview.

[Aut16] Autodesk. AutoCAD - Overview. 2016. URL: http://www.autodesk.com/products/autocad/overview.

[BM11] Peder Bacher and Henrik Madsen. Identifying suitable models for the heat dynamics of buildings. *Energy and Buildings*, 43(7):1511–1522, 2011. doi:10.1016/j.enbuild.2011.02.005.

[BM10] Azadeh Badakhshani and Dirk Müller. Dynamische exergetische Analyse von Gebäuden anhand der Gesamtsystemsimulation in der objektorientierten Programmiersprache Modelica. In *BauSIM 2010 - 3rd German-Austrian IBPSA Conference*, 409–413. International Building Performance Simulation Association, 2010. URL: http://www.ibpsa.org/proceedings/bausimPapers/2010/409.pdf.

[BDCVR+12] R. Baetens, R. De Coninck, J. Van Roy, B. Verbruggen, J. Driesen, L. Helsen, and D. Saelens. Assessing electrical bottlenecks at feeder level for residential net zero-energy buildings by integrated system simulation. *Applied Energy*, 96:74–83, 2012. doi:10.1016/j.apenergy.2011.12.098.

[Bae15] Ruben Baetens. *On Externalities of Heat Pump-Based Low-Energy Dwellings at the Low-Voltage Distribution Grid*. PhD thesis, KU Leuven, 2015.

[BDCJ+15] Ruben Baetens, Roel De Coninck, Filip Jorissen, Damien Picard, Lieve Helsen, and Dirk Saelens. OPENIDEAS - An Open Framework for Integrated District Energy Simulations. In *Proceedings of Building Simulation 2015*. Hyderabad, India, 2015.

[BS13] Ruben Baetens and Dirk Saelens. Multi-criteria grid impact evaluation of heat pump and photovoltaic based zero-energy dwellings. In *BS2013, 13th Conference of International Building Performance Simulation Association*, 3529–3536. 2013.

[BS15] Ruben Baetens and Dirk Saelens. Modelling uncertainty in district energy simulations by stochastic residential occupant behaviour. *Journal of Building Performance Simulation*, 1493(September):1–17, 2015. doi:10.1080/19401493.2015.1070203.

[BRM+14] Shalako Baggi, Davide Rivola, Vasco Medici, Gianluca Corbellini, Davide Strepparava, and Roman Rudel. Modeling and simulation of a residential neighborhood with photovoltaic systems coupled to energy storage systems. In *EU PVSEC 2014, 29th European Photovoltaic Solar Energy Conference and Exhibition*. 2014.

[BS10] Maria Bernardete Barison and Eduardo Toledo Santos. An overview of BIM specialists. In *Proceedings of the International Conference on Computing in Civil and Building Engineering icc-cbe2010*, 141. 2010. URL: http://www.engineering.nottingham.ac.uk/icccbe/proceedings/pdf/pf71.pdf.

[BP11] Daniele Basciotti and Olivier Pol. A theoretical study of the impact of using small scale thermo chemical storage units in district heating networks. In *International Sustainable Energy Conference*. 2011.

[BS14] Daniele Basciotti and R R Schmidt. Peak reduction in district heating networks: a comparison study and practical considerations. In *14th International Symposium on District Heating and Cooling*. 2014.

[BSKD14] Daniele Basciotti, R R Schmidt, M Köfinger, and C Doczekal. Simulation-based analysis and evaluation of domestic hot water preparation principles for low-temperature district heating networks. In *14th International Symposium on District Heating and Cooling*. 2014.

[Bay14] Michael Bayer. Mako Templates for Python. http://www.makotemplates.org/, 2014. Accessed: 2015-05-13.

[Baz04] Vladimir Bazjanac. Virtual building environments - Applying information modeling to buildings. In *5th European Conference on Product and Process Modelling in the Building and Construction Industry - ECPPM 2004*, 41–48. Istanbul, Turkey, August 2004. Attila Dikbas and Raimar Scherer, Taylor & Francis 2004. doi:10.1201/9780203023426.ch7.

[Baz10] Vladimir Bazjanac. Space boundary requirements for modeling of building geometry for energy and other performance simulation. In *27th CIB-W78 Conference*. Cairo, Egypt, 2010.

[BK07] Vladimir Bazjanac and Arto Kiviniemi. Reduction, Simplification, Translation and Interpretation in the Exchange of Model Data. In *24th CIB_W78*, 163–168. University of Maribor, CIB W78, 2007. URL: http://cic.vtt.fi/projects/vbe-net/data/2007_Data_Simplification_@_CIB-W78_Maribor.pdf.

[BMNG16] Vladimir Bazjanac, Tobias Maile, and Christoph Nytsch-Geusen. Generation of building geometry for energy performance simulation using Modelica. In *CESBP/ BauSIM 2016 conference*. Dresden, Germany, 2016.

[BMOD+11] Vladimir Bazjanac, Tobias Maile, James O'Donnell, Cody Rose, and Natasa Mrazovic. Data environments and processing in semi-automated simulation with EnergyPlus. In *CIB W078-W102: 28th International Conference*. Sophia Antipolis, France, 2011. CIB.

[BMR+11] Vladimir Bazjanac, Tobias Maile, Cody Rose, James O'Donnell, Natasa Mrazovic, Elmer Morrissey, and Benjamin Welle. An assessment of the use of building energy performance simulation in early design. In *IBPSA Building Simulation 2011. Sydney, Australia*. Sydney, Australia, 2011. IBPSA.

[Ben91] A. Bennonysson. *Dynamic Modelling and Operation Optimization of District Heating Systems*. PhD thesis, Technical University of Denmark (DTU), 1991.

[Ben15] Bentley. Modeling, Documentation and Visualization Software. 2015. URL: https://www.bentley.com/en/products/product-line/modeling-and-visualization-software/microstation.

[BBCK15] Federico Bergero, Mariano Botta, Esteban Campostrini, and Ernesto Kofman. Efficient compilation of large scale dynamical systems. In Peter Fritzson and Hilding Elmqvist, editors, *11-th International Modelica Conference*, 449–458. Paris, France, September 2015. Modelica Association. doi:10.3384/ecp15118449.

[BRC16] Federico Bergero, Akshay Ranade, and Francesco Casella. QSS and multi-rate simulation of object-oriented models. In *Proceedings of the 7th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, EOOLT '16, 69–77. New York, NY, USA, 2016. ACM. doi:10.1145/2904081.2904091.

[Bie10] Lorenz T. Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. volume 10. SIAM, 2010. doi:10.1137/1.9780898719383.

[BBE+99] Niclas Björsell, Axel Bring, Lars Eriksson, Pavel Grozman, Magnus Lindgren, Per Sahlin, Alexander Shapovalov, and Mika Vuolle. IDA indoor climate and energy. In N. Nakahara, H. Yoshida, M. Udagawa, and J. Hensen, editors, *Proc. of the 6-th IBPSA Conference*, 1035–1042. Kyoto, Japan, September 1999. URL: http://www.ibpsa.org/conferences.htm.

[BOA+11] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf. The functional mockup interface for tool independent exchange of simulation models. In *Proc. of the 8-th International Modelica Conference*. Dresden, Germany, March 2011. Modelica Association. doi:10.3384/ecp11063105.

[BSG+14] Marco Bonvini, Michael D. Sohn, Jessica Granderson, Michael Wetter, and Mary Ann Piette. Robust on-line fault detection diagnosis for HVAC components based on nonlinear state estimation techniques. *Applied Energy*, 124(0):156 – 166, 2014. doi:10.1016/j.apenergy.2014.03.009.

[BW15] Marco Bonvini and Michael Wetter. Gradient-based optimal control of batteries and HVAC in district energy systems. In *BS2015, 14th Conference of International Building Performance Simulation Association*. 2015.

[BWN14] Marco Bonvini, Michael Wetter, and Thierry Stephane Nouidui. A Modelica package for building-to-electical grid integration. In *5th BauSim Conference*, 6–13. Aachen, Germany, September 2014. IBPSA-Germany. URL: http://simulationresearch.lbl.gov/wetter/download/2014-BauSim-BonviniWetterNouidui.pdf.

[BWS14] Marco Bonvini, Michael Wetter, and Michael D Sohn. An FMI-based framework for state and parameter estimation. In *10th International Modelica Conference, Lund, Sweden*. 2014. doi:10.3384/ECP14096647.

[BKC+12] James E Braun, Donghun Kim, Eugene Cliff, John a Burns, and Bill Henshaw. Whole building control system design and evaluation : simulation-based assessment. Technical Report, GPIC Energy Efficient Buildings Hub, February 2012.

[Bra90] JE Braun. Reducing energy costs and peak electrical demand through optimal control of building thermal storage. *ASHRAE transactions*, 1990.

[BCB17] Willi Braun, Francesco Casella, and Bernhard Bachmann. Solving large-scale Modelica models: new approaches and experimental results using OpenModelica. In *Proc. of the 12-th International Modelica Conference*, 557–563. Prague, Czech Republic, May 2017. Modelica Association. doi:10.3384/ecp17132557.

[BBG+13] David Broman, Christopher Brooks, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Determinate composition of FMUs for co-simulation. In *Embedded Software (EMSOFT), 2013 Proceedings of the International Conference on*, 1–12. 2013. doi:10.1109/EMSOFT.2013.6658580.

[BGL+15] David Broman, Lev Greenberg, Edward A. Lee, Michael Massin, Stavros Tripakis, and Michael Wetter. Requirements for hybrid cosimulation standards. In *18th International Conference on Hybrid Systems: Computation and Control*. ACM Press, April 2015. URL: http://simulationresearch.lbl.gov/wetter/download/2015-BromanEtAl_HybridCosimulation_HSCC.pdf.

[BLL+15] Christopher Brooks, Edward A. Lee, David Lorenzetti, Thierry S. Nouidui, and Michael Wetter. Demo: CyPhySim – a cyber-physical systems simulator. In *18th International Conference on Hybrid Systems: Computation and Control*. ACM Press, April 2015. URL: http://simulationresearch.lbl.gov/wetter/download/2015-BrooksEtAl_CyPhySimDemo.pdf.

[Bro90] Gösta Brown. The BRIS simulation program for thermal design of buildings and their services. *Energy and Buildings*, 14(4):385–400, 1990. doi:10.1016/0378-7788(90)90100-W.

[Bal13] K. Brunix and al. Short-term demand response of flexible electric heating systems: the need for integrated simulations. In *Proceedings of the 10th international Conference on the European Energy Market*. 2013.

[BRA+12] Ken Bruton, Paul Raftery, Niall Aughney, Marcus M. Keane., and Dominic O'Sullivan. Development of an automated fault detection and diagnosis tool for AHU‘s. In *Proceedings of the Twelfth International Conference for Enhanced Building Operations*. Manchester, UK, 2012.

[BRK+14] Ken Bruton, Paul Raftery, Barry Kennedy, Marcus M. Keane, and D. T. J. O'Sullivan. Review of automated fault detection and diagnostic tools in air handling units. *Energy Efficiency*, 7(2):335–351, November 2014. doi:10.1007/s12053-013-9238-2.

[bSI14] bSI. Open BIM Collaboration Format (BCF). 2014. URL: http://www.buildingsmart-tech.org/specifications/bcf-releases/bcf-intro.

[bSI15a] bSI. Core Purpose. 2015. URL: http://buildingsmart.org/about/vision-mission/core-purpose/.

[bSI15b] bSI. IFC4 Add1 Release - Welcome to buildingSMART-Tech.org. 2015. URL: http://www.buildingsmart-tech.org/specifications/ifc-releases/ifc4-add1-release.

[bSI15c] bSI. Web service specification for BIM Collaboration Format. 2015. URL: https://github.com/BuildingSMART/BCF-API.

[BEWS93] W.F. Buhl, A.E. Erdem, F.C. Winkelmann, and E.F. Sowell. Recent improvements in SPARK: strong component decomposition, multivalued objects,and graphical interface. In *Proc. of the 3-rd IBPSA Conference*, 283–291. Adelaide, Australia, August 1993.

[Bui16] BuildingSMART. Model View Definition Releases. 2016. URL: http://www.buildingsmart-tech.org/specifications/ifc-view-definition.

[bia] buildingSMART international. Concepts for the MVDs. online. URL: http://www.buildingsmart-tech.org/ifc/IFC4/Add2/html/link/chapter-4.htm.

[bib] buildingSMART international. IfcDOCTool summary page. online. URL: http://www.buildingsmart-tech.org/specifications/specification-tools/ifcdoc-tool.

[BIFM09] Peter Bunus, Olle Isaksson, Beate Frey, and Burkhard Münker. RODON - a model-based diagnosis approach for the DX diagnostic competition. *Proc. DX‘09*, pages 423–430, 2009.

[BJH10] Sebastian Burhenne, Dirk Jacob, and Gregor P. Henze. Uncertainty analysis in building simulation with monte carlo techniques. In *SimBuild 2010 Fourth National Conference of IBPSA-USA*. 2010.

[BWE+13] Sebastian Burhenne, Dominik Wystrcil, Mehmet Elci, Sattaya Narmsara, and Sebastian Herkel. Building performance simulation using Modelica: analysis of the current state and application areas. In *BS2013, 13th Conference of International Building Performance Simulation Association*. 2013.

[BhmHK+02] Benny Bøhm, Seung-kyu Ha, W Kim, Bong-kyun Kim, T Koljonen, Helge V. Larsen, Michael Lucht, Yong-soon Park, Kari Sipilä, Michael Wigbels, and Magnus Wistbacka. Simple models for operational optimisation. Technical Report, Technical University of Denmark, 2002.

[CMO+14] Jun Cao, Tobias Maile, James O'Donnell, Reinhard Wimmer, and Christoph van Treeck. Model transformation from SimModel to Modelica for building energy performance simulation. In *5th BauSim Conference*, 22–24. Aachen, Germany, September 2014. IBPSA-Germany.

[CWT+15] Jun Cao, Reinhard Wimmer, Matthis Thorade, Tobias Maile, James O'Donnel, Jörg Rädler, Jérôme Frisch, and Christoph van Treeck. A flexible model transformation to link bim with different modelica libraries for building energy performance simulation. In *Proceedings of the 14th IBPSA Conference*. 2015.

[Cas15] Francesco Casella. Simulation of large-scale models in modelica: state of the art and future perspectives. In Peter Fritzson and Hilding Elmqvist, editors, *11-th International Modelica Conference*, 459–468. Paris, France, September 2015. Modelica Association. doi:10.3384/ecp15118459.

[COP+06] Francesco Casella, Martin Otter, Katrin Proelss, Christoph Richter, and Hubertus Tummescheit. The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks. In *5th International Modelica Conference*, 631–640. 2006.

[CR16] Francesco Casella and Akshay Ranade. Efficient modelling and simulation of multi-domain smart grids using Modelica and multi-rate integration algorithms. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 6285–6291. October 2016. doi:10.1109/IECON.2016.7794050.

[CNB+15] Damien Casetta, Cynthia Nerbollier, Guillaume Brecq, Pascal Stabat, and Dominique Marchio. Dynamic modelling of a district cooling network with Modelica. In *BS2015, 14th Conference of International Building Performance Simulation Association*. 2015.

[CEO95] François E. Cellier, Hilding Elmqvist, and Martin Otter. *The Control Handbook.*, chapter Modeling from Physical Principles, pages 99–108. CRC Press, Boca Raton, FL, 1995.

[CK06] François E. Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer, 2006. doi:10.1007/0-387-30260-3.

[CBM+16] Spyros Chatzivasileiadis, Marco Bonvini, Javier Matanza, Rongxin Yin, Thierry S. Nouidui, Emre C. Kara, Rajiv Parmar, David Lorenzetti, Michael Wetter, and Sila Kiliccote. Cyber-physical modeling of distributed resources for distribution system operations. *Proceedings of the IEEE*, 104(4):789–806, 2016. doi:10.1109/JPROC.2016.2520738.

[CJL06] Kuentai Chen, Yue Jiao, and E. Stanley Lee. Fuzzy adaptive networks in thermal comfort. *Applied Mathematics Letters*, 19(5):420–426, 2006. doi:10.1016/j.aml.2005.06.013.

[CDRR12] Lou Chesné, Thierry Duforestel, Jean-Jacques Roux, and Gilles Rusaouën. Energy saving and environmental resources potentials: toward new methods of building design. *Building and Environment*, 58:199–207, 2012. doi:10.1016/j.buildenv.2012.07.013.

[CLW13] Tim Chipman, Thomas Liebich, and Matthias Weise. mvdXML - Specification of Standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules. Technical Report, Model Support Group (MSG) of buildingSMART International Ltd., 2013. URL: http://www.buildingsmart-tech.org/downloads/accompanying-documents/formats/mvdxml-documentation/mvdxml-schema-documentation-v1-1-draft.

[Cig13] Jiř Cigler. *Model Predictive Control for Buildings*. PhD thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, 2013. URL: https://support.dce.felk.cvut.cz/mediawiki/images/5/5f/Diz_2013_cigler_jiri.pdf.

[CPV+12] Jiří Cigler, Samuel Prívara, Zdeněk Váňa, Eva Žáčeková, and Lukáš Ferkl. Optimization of predicted mean vote index within model predictive control framework: computationally tractable solution. *Energy and Buildings*, 52:39–49, 2012. doi:10.1016/j.enbuild.2012.05.022.

[CH15] J. A. Clarke and J. L. M. Hensen. Integrated building performance simulation: progress, prospects and requirements. *Building and Environment*, 91(0):294 – 306, 2015. Fifty Year Anniversary for Building and Environment. doi:10.1016/j.buildenv.2015.04.002.

[Cla15] Joe Clarke. A vision for building performance simulation: a position paper prepared on behalf of the IBPSA Board. *Journal of Building Performance Simulation*, 8(2):39–43, 2015. doi:10.1080/19401493.2015.1007699.

[CMK15] Silvia Coccolo, D Mauree, and Jérôme Kämpf. Urban energy simulation based on a new data model paradigm: the CityGML application domain extension energy. a case study in the EPFL campus of Lausanne. In *CISBAT2015, International Conference on Future Buildings & Districts Sustainability from Nano to Urban Scale*. 2015.

[CL55] Earl A. Coddington and Norman Levinson. *Theory of ordinary differential equations*. McGraw-Hill Book Company, Inc., New York-Toronto-London, 1955.

[Cod14] CodeSynthesis. Xsd: XML data binding for C++ [Computer Software]. In *Retrieved from http://www.codesynthesis.com/products/xsd/*. 2014.

[Cof11] Brian Coffey. *Using Building Simulation and Optimization to Calculate Lookup Tables for Control*. PhD thesis, University of California at Berkeley, 2011. URL: http://escholarship.org/uc/item/1202p562.

[CHMK10] Brian Coffey, Fariborz Haghighat, Edward Morofsky, and Edward Kutrowski. A software framework for model predictive control with GenOpt. *Energy and Buildings*, 42(7):1084–1092, July 2010. doi:10.1016/j.enbuild.2010.01.022.

[Com16a] European Commission. Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions on an eu strategy for heating and cooling. Technical Report, European Commission, Brussels, 2016. URL: https://ec.europa.eu/energy/sites/ener/files/documents/1_EN_ACT_part1_v14.pdf.

[Com16b] European Commission. Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions on an EU Strategy for Heating and Cooling - Review of available information. Technical Report, European Commission, Brussels, 2016. URL: https://ec.europa.eu/energy/sites/ener/files/documents/1_EN_autre_document_travail_service_part1_v6_0.pdf.

[CNP13] David Connolly, Steffen Nielsen, and Urban Persson. *Heat Roadmap Europe 2050*. Department of Development and Planning, Aalborg University, Aalborg, 2013. ISBN 9788791404481.

[CSM14] A. Constantin, R. Streblow, and D. Mueller. The Modelica HouseModels library: Presentation and evaluation of a room model with the ASHRAE Standard 140. In *Proceedings of the 10th International Modelica Conference*, 293–299. Lund, Sweden, March 2014. Modelica Association. doi:10.3384/ECP14096293.

[CRR+11] Anthony Cormier, Sylvain Robert, Pierrick Roger, Louis Stephan, and Etienne Wurtz. Towards a BIM-based service oriented platform: application to building energy performance simulation. In *Proceedings of the 12th International IBPSA Conference*. Sydney, Australia, 2011. URL: http://www.ibpsa.org/proceedings/BS2011/P_1930.pdf.

[CLP+99] Drury B. Crawley, Linda K. Lawrie, Curtis O. Pedersen, Richard J. Liesen, Daniel E. Fisher, Richard K. Strand, Russell D. Taylor, Frederick C. Winkelmann, W. F. Buhl, A. E. Erdem, and Y. J. Huang. EnergyPlus, a new-generation building energy simulation program. In *Proc. of Renewable and Advanced Energy Systems for the 21st Century*. Lahaina, Maui, Hawaii, April 1999.

[CLW+96] Drury B. Crawley, Linda K. Lawrie, Frederick C. Winkelmann, Curtis Pedersen, Richard Liesen, and Dan Fisher. Next-generation building energy simulation tools – a vision of the future. In *Proceedings of the ACEEE 1996 Summer Study on Energy Efficiency in Buildings*, volume 4 of Commercial Buildings: Technologies, Design, and Performance Analysis, 4.69–4.75. Asilomar, Pacific Grove, CA, August 1996. ACEEE.

[CMZ11] Xiaolong Cui, Jiming Ma, and Shengkui Zeng. The fault modeling methodology of actuator system based on Modelica. In *Reliability, Maintainability and Safety (ICRMS), 2011 9th International Conference on*, 997–1002. IEEE, 2011.

[CRHV11] W. Cyx, N. Renders, M. Van Holm, and S. Verbek. Ee tabula - typology approach for building stock energy assessment. Technical Report, VITO - Vision on technology, 2011.

[CH03] Krzysztof Czarnecki and Simon Helsen. Classification of model transformation approaches. In *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, volume 45, 1–17. USA, 2003.

[CH06] Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3):621–645, 2006. doi:10.1147/sj.453.0621.

[DCBS+14] Roel De Coninck, Ruben Baetens, Dirk Saelens, A Woyte, and Lieve Helsen. Rule-based demand-side management of domestic hot water production with heat pumps in zero energy neighbourhoods. *Journal of Building Performance Simulation*, 7:271–288, 2014. doi:10.1080/19401493.2013.801518.

[DCH16] Roel De Coninck and Lieve Helsen. Practical implementation and evaluation of model predictive control for an office building in Brussels. *Energy and Buildings*, 111:290–298, 2016. doi:10.1016/j.enbuild.2015.11.014.

[DCMAH15] Roel De Coninck, Fredrik Magnusson, Johan Åkesson, and Lieve Helsen. Toolbox for development and validation of grey-box building models for forecasting and control. *Journal of Building Performance Simulation*, pages 1–16, 2015. doi:10.1080/19401493.2015.1046933.

[DFS+11] Michael Deru, Kristin Field, Daniel Studer, Kyle Benne, Brent Griffith, Paul Torcellini, Bing Liu, Mark Halverson, Dave Winiarski, Michael Rosenberg, Mehry Yazdanian, Joe Huang, and Drury Crawley. U.S. Department of Energy commercial reference building models of the national building stock. Technical Report NREL/TP-5500-46861, National Renewables Energy Laboratory, 1617 Cole Boulevard, Golden, Colorado 80401, February 2011.

[DEP16] W Stuart Dols, Steven J Emmerich, and Brian J Polidoro. Coupling the multizone airflow and contaminant transport software contam with energyplus using co-simulation. In *Building simulation*, volume 9, 469–479. Springer, 2016. doi:10.1007/s12273-016-0279-2.

[DZMJ11] Alexander Domahidi, Melanie N. Zeilinger, Manfred Morari, and Colin N. Jones. Learning a feasible and stabilizing explicit model predictive control law by robust optimization. *Proceedings of the 50IEEE Conference on Decision and Control and European Control Conference*, pages 513–519, 2011. doi:10.1109/CDC.2011.6161258.

[Don10] Gheorghe Donca. Modelling and simulation fault detection and diagnostics of HVAC systems. *Analele Universitaatii din Oradea, Fascicula: Ecotoxicologie, Zootehnie si Tehnologii de Industrie Alimentara 2010*, pages 381–387, 2010.

[DER89] Iain S. Duff, Albert M. Erisman, and John K. Reid. *Direct Methods for Sparse Matrices*. Monographs on Numerical Analysis. Oxford Science Publications, Oxford, 1989.

[Duf12] Thierry Duforestel. Le projet habisol-valerie. Technical Report, ANR, 2012.

[ETSL11] Chuck Eastman, Paul Teicholz, Rafael Sacks, and Kathleen Liston. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. John Wiley & Sons, second edition, 2011. ISBN 978-0-470-54137-7.

[EBC14] EBC. Modelica AixLib library]. In *Retrieved from https://github.com/RWTH-EBC/AixLib*. Institute for Energy Efficient Buildings and Indoor Climate EBC, Energy Research Center E.ON, RWTH Aachen University, Germany, 2014.

[EvB13] Sebastian Ebertshäuser and Petra von Both. Ifcmodelcheck - a tool for configurable rule-based model checking. In *eCAADa 2013: Computation and Performance Vol.2., Delft*. 2013.

[EvB16] Sebastian Ebertshäuser and Petra von Both. Prüfung und Konvertierung von TGA IfcXML-Modellen zu SimModel. In *IBPSA BauSIM 2016 proceedings, Dresden*. 2016.

[ECP07] Arno Ebner, Fiorentino Valerio Conte, and Franz Pirker. Rapid validation of battery management system with a Dymola hardware-in-the-loop simulation energy storage test bench. *The World Electric Vehicle Association Journal*, 1:205–207, 2007.

[EHLP13] Martin Egger, Kerstin Hausknecht, Thomas Liebich, and Jakob Przybylo. BIM-Guide for Germany Information und guidebook BIM guide for Germany. Technical Report, Federal Ministry for Transport, 2013.

[ECK+11] M Einhorn, F V Conte, C Kral, C Niklas, H Popp, and J Fleig. A Modelica library for simulation of elecric energy storages. In *8th International Modelica Conference*, 436–445. 2011. doi:10.3384/ecp11063436.

[ENKH13] Mehmet Elci, Sattaya Narmsara, Florian Kagerer, and Sebastian Herkel. Simulation of energy conservation measures and its implications on a combined heat and power district heating system : a case study. *Proceedings of BS2013:13th Conference of Building Performance Simulation Association*, pages 104–111, 2013.

[EOH+15] Mehmet Elci, Axel Oliva, Sebastian Herkel, Konstantin Klein, and Alexander Ripka. Grid-interactivity of a solar combined heat and power district heating system. *Energy Procedia*, 70:560–567, 2015. doi:10.1016/j.egypro.2015.02.161.

[EOC95] H. Elmqvist, M. Otter, and F. Cellier. Inline integration: a new mixed symbolic/numeric approach for solving differential– algebraic equation systems. In *Keynote Address, Proc. ESM'95*, xxiii–xxxiv". Prague, Czech Republic, June 1995. European Simulation Multiconference. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.127.3787.

[Elm78] Hilding Elmqvist. *A structured model language for large continuous systems*. PhD thesis, Lund Institute of Technology, Lund, Sweden, 1978. URL: http://lup.lub.lu.se/record/8524888.

[Elm14] Hilding Elmqvist. Modelica evolution – from my perspective. In *10-th International Modelica Conference*, 17–26. Lund, Sweden, March 2014. Modelica Association. doi:10.3384/ECP1409617.

[EMO14] Hilding Elmqvist, Sven Erik Mattsson, and Hans Olsson. Parallel model execution on many cores. In *10-th International Modelica Conference*, 363–370. Lund, Sweden, March 2014. Modelica Association. doi:10.3384/ECP14096363.

[EO94] Hilding Elmqvist and Martin Otter. Methods for tearing systems of equations in object-oriented modelling. In *European Simulation Multiconference*, 326–332. Barcelona, Spain, June 1994. URL: http://www.robotic.de/fileadmin/control/shared/publications/1994/elmqvist_esm.ps.gz.

[EAWP13] Atiyah Elsheikh, Muhammad Usman Awais, Edmund Widl, and Peter Palensky. Modelica-enabled rapid prototyping of cyber-physical energy systems via the functional mockup interface. In *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2013 Workshop on*, 1–6. IEEE, 2013.

[EWP+13] Atiyah Elsheikh, Edmund Widl, Peter Pensky, Florian Dubisch, Markus Brychta, Daniele Basciotti, and Wolfgang Müller. Modelica-enabled rapid prototyping via TRNSYS. In *The 13th International Conference of the International Building Performance Simulation Association*. 2013.

[Fan73] P. O. Fanger. *Thermal Comfort*. McGraw-Hill, 1973. ISBN 8757103410.

[FSK15] Jesús Febres, Raymond Sterling, and Marcus Keane. Automated calibration of air handling unit models using a modified Preisach model. In *Proceedings of BS2015: 14th Conference of International Building Performance Simulation Association*, 1547 – 1554. 2015.

[FSTK13] Jesús Febres, Raymond Sterling, Ignacio Torrens, and Marcus M. Keane. Heat ventilation and air conditioning modelling for model-based fault detection and diagnosis. In *Building Simulation 2013*, 3513 – 3520. 2013.

[FAB+02a] F. Felgner, S. Agustina, R. Cladera Bohigas, R. Merz, and L. Litz. Simulation of thermal building behaviour in Modelica. In Martin Otter, editor, *Proceedings of the 2nd Modelica conference*, 147–154. Oberpfaffenhofen, Germany, March 2002. Modelica Association and Deutsches Zentrum fur Luft- und Raumfahrt. URL: https://modelica.org/events/Conference2002/papers/p19_Felgner.pdf.

[FK14] Joaquín Fernández and Ernesto Kofman. A stand-alone quantized state system solver for continuous system simulation. *SIMULATION*, 90(7):782–799, 2014. doi:10.1177/0037549714536255.

[FBC+14] X. Floros, F. Bergero, N. Ceriani, F. Casella, E. Kofman, and F. E. Cellier. Simulation of smart-grid models using quantization-based integration methods. In *10-th International Modelica Conference*, 787–797. Lund, Sweden, March 2014. Modelica Association. doi:10.3384/ECP14096787.

[FCO+09a] Rüdiger Franke, Francesco Casella, Martin Otter, Michael Sielemann, Hilding Elmqvist, Sven Erik Mattsson, and Hans Olsson. Stream connectors – an extension of Modelica for device-oriented modeling of convective transport phenomena. In Francesco Casella, editor, *Proc. of the 7-th International Modelica Conference*. Como, Italy, September 2009. Modelica Association. doi:10.3384/ecp09430078.

[Fra14] Rüdiger Franke. Client-side Modelica powered by Python or JavaScript. In *10-th International Modelica Conference*, 1105–1112. Lund, Sweden, March 2014. Modelica Association. doi:10.3384/ECP140961105.

[FCO+09b] Rüdiger Franke, Francesco Casella, Martin Otter, Katrin Proelss, Michael Sielemann, and Michael Wetter. Standardization of thermo-fluid modeling in Modelica.Fluid. In Francesco Casella, editor, *Proc. of the 7-th International Modelica Conference*. Como, Italy, September 2009. Modelica Association. doi:10.3384/ecp09430077.

[FW13] Svend Frederiksen and Sven Werner. *District heating and cooling*. Studentlitteratur AB, Lund, first edition, 2013.

[FOM08] Roberto Z. Freire, Gustavo H C Oliveira, and Nathan Mendes. Predictive controllers for thermal comfort optimization and energy savings. *Energy and Buildings*, 40(7):1353–1365, 2008. doi:10.1016/j.enbuild.2007.12.007.

[Fri11] Peter Fritzson. *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*. Wiley-IEEE Press, September 2011. ISBN 978-1-1180-1068-6. doi:10.1002/9781118094259.

[Fri15] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3*. John Wiley & Sons, 2015. doi:10.1002/9781118989166.

[FAB+02b] Peter Fritzson, Peter Aronsson, Peter Bunus, Vadim Engelson, Levon Saldamli, Henrik Johansson, and Andreas Karström. The open source Modelica project. In Martin Otter, editor, *Proceedings of the 2nd Modelica conference*, 297–306. Oberpfaffenhofen, Germany, March 2002. Modelica Association and Deutsches Zentrum fur Luft- und Raumfahrt. URL: https://modelica.org/events/Conference2002/papers/p37_Fritzson2.pdf.

[FCL+15] Marcus Fuchs, Ana Constantin, Moritz Lauster, Peter Remmen, Rita Streblow, and Dirk Müller. Structuring the building performance modelica library AixLib for open collaborative development. In *14-th IBPSA Conference*, 331–338. International Building Performance Simulation Association, December 2015. URL: http://www.ibpsa.org/proceedings/BS2015/p2202.pdf.

[FDT+13] Marcus Fuchs, Thomas Dixius, Jens Teichmann, Moritz Lauster, Rita Streblow, and Dirk Müller. Evaluation of interactions between buildings and district heating networks. In *BS2013, 13th Conference of International Building Performance Simulation Association*. 2013.

[FTL+16] Marcus Fuchs, Jens Teichmann, Moritz Lauster, Peter Remmen, Rita Streblow, and Dirk Müller. Workflow automation for combined modeling of buildings and district energy systems. *Energy*, 2016. doi:10.1016/j.energy.2016.04.023.

[VIB15] BMVi Bundesministerium für Verkehr und digitale Infrastruktur (BMVi). Stufenplan Digitales Planen und Bauen. 2015.

[GW14] Henrik Gadd and Sven Werner. Achieving low return temperatures from district heating substations. *Applied Energy*, 136:59–67, 2014. doi:10.1016/j.apenergy.2014.09.022.

[GVD+15] Virginie Galtier, Stephane Vialle, Cherifa Dad, Jean-Philippe Tavella, Jean-Philippe Lam-Yee-Mui, and Gilles Plessis. FMI-based distributed multi-simulation with DACCOSIM. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, 39–46. Society for Computer Simulation International, 2015.

[GT14] Mike Gathercole and Niraj Thurairajah. The influence of BIM on the responsibilities and skills of a project delivery team. In *International Conference on Construction in a Changing World*. Heritance Kandalama, Sri Lanka, 2014. Birmingham School of the Built Environment.

[GbX16] GbXML. About Green Building XML Schema. 2016. URL: http://www.gbxml.org/About_GreenBuildingXML_gbXML.

[GLG+15] Georgios I. Giannakis, Georgios N. Lilis, Miguel Angel Garcia, Giorgos D Kontes, Cesar Valmaseda, and Dimitrios V. Rovas. A Methodology to Automatically Generate Geometry And Material Inputs for Energy Performance Simulation from Ifc Bim Models. In *14th International Conference of the International Building Performance Simulation Association*. Hyderabad, India, 2015. IBPSA.

[GPB15] Loic Giraud, Cedric Paulus, and Roland Baviere. Modeling of solar district heating: a comparison between TRNSYS and Modelica. *Proceedings of the EuroSun 2014 Conference*, pages 1–11, 2015. doi:10.18086/eurosun.2014.19.06.

[GBP+15] Loïc Giraud, Roland Bavière, Cédric Paulus, Mathieu Vallée, and Jean-François Robin. Dynamic modelling , experimental validation and simulation of a virtual district heating network. In *Proceedings of ECOS 2015 - The 28th international conference on Efficiency, Cost, Optimization, Simulation and the Environmental Impact of Energy Systems*. Pau, 2015.

[Gra11] Granlund. BSPro Homepage. 2011.

[Gra16] Graphisoft. About ARCHICAD - A 3D architectural BIM software for design & modeling. 2016. URL: http://www.graphisoft.com/archicad/.

[GKRT10] M. Gräber, K. Kosowski, C. Richter, and W. Tegethoff. Modelling of heat pumps with an object-oriented model library for thermodynamic systems. *Mathematical and Computer Modelling of Dynamical Systems*, 16(3):195–209, 2010. doi:10.1080/13873954.2010.506799.

[HW96] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*. Springer series in computational mathematics. Springer-Verlag, Berlin, second edition, 1996. ISBN 3-540-60452-9. doi:10.1007/978-3-642-05221-7.

[Hau08] Anton Haumer. Quasi-Stationary Modeling and Simulation of Electrical Circuits using Complex Phasors. In *6th International Modelica Conference*, 229–236. 2008.

[HDN+09] Philip Haves, Joseph J. Deringer, Massieh Najafi, Brian Coffey, and Daniel McQuillan. Automated rooftop air conditioning fault detection in retail stores and extension of Learn HVAC. Technical Report, Lawrence Berkeley National Laboratory, Berkeley, 2009.

[Hei12] Kristin Heinemeier. Rooftop HVAC fault detection and diagnostics: technology and market review energy and demand savings estimates. Technical Report October, Western Cooling Efficiency Center, California, USA, 2012.

[HL12] J L M Hensen and R Lamberts. *Building Performance Simulation for Design and Operation*. Taylor & Francis, 2012. doi:10.4324/9780203891612.

[HKLF05] Gregor P Henze, Doreen E Kalz, Simeng Liu, and Clemens Felsmann. Experimental analysis of model-based predictive optimal control for active and passive building thermal storage inventory. *HVAC&R Research*, 11(2):189–213, 2005. doi:10.1080/10789669.2005.10391134.

[Hie06] Jiri Hietanen. IDM/MVD - ITM Summit #34 Washington DC. Technical Report, Tampere University of Technology, 2006. URL: http://cic.vtt.fi/projects/vbe-net/data/20061030_IDM-MVD_@_Washington_DC.pdf.

[HHTM05] Alexander Hoh, Timo Haase, Thomas Tschirner, and Dirk Müller. A combined thermo-hydraulic approach to simulation of active building components applying modelica. In Gerhard Schmitz, editor, *4-th International Modelica Conference*. Hamburg, Germany, 2005. Modelica Association.

[HH11] Christina J. Hopfe and Jan L. M. Hensen. Uncertainty analysis in building performance simulation for design support. *Energy and Buildings*, 43(10):2798–2805, 2011. doi:10.1016/j.enbuild.2011.06.034.

[HK14] Jianjun Hu and Panagiota Karava. Model predictive control strategies for buildings with mixed-mode cooling. *Building and Environment*, 71:233–244, 2014. doi:10.1016/j.buildenv.2013.09.005.

[HZS16] Sen Huang, Wangda Zuo, and Michael D. Sohn. Amelioration of the cooling load based chiller sequencing control. *Applied Energy*, 168:204–215, 2016. doi:10.1016/j.apenergy.2016.01.035.

[HNG11] Jörg Huber and Christoph Nytsch-Geusen. Development of Modeling and Simulation Strategies for Large-Scale Urban Districts. In *Building Simulation 2011*, 1753–1760. IBPSA, 2011.

[HA09] Sebastian Hölemann and Dirk Abel. Modelica predictive control - an MPC library for Modelica. In *Automatisierungstechnik*, volume 57, 187–194. 2009. doi:10.1524/auto.2009.0766.

[IC90] M. D. Ilic and M. Crow. The waveform relaxation method for systems of differential/algebraic equations. In *Proceedings of the 29th IEEE Conference on Decision and Control*. IEEE, 1990. doi:10.1109/CDC.1990.203640.

[IKS08] L. Imsland, P. Kittilsen, and T. S. Schei. Model-based optimizing control and estimation using Modelica models. In *Proceedings of the Modelica Conference 2008*, 301–310. 2008. doi:10.4173/mic.2010.3.3.

[Ise97] R. Isermann. Supervision, fault-detection and fault-diagnosis methods - an introduction. *Control Engineering Practice*, 5(5):639–652, 1997. doi:10.1016/S0967-0661(97)00046-4.

[JHB17] Lennart Ochel Jan Hagemann, Patrick Täuber and Bernhard Bachmann. Smart processing of function calls to achieve efficient simulation code. In *Proc. of the 12-th International Modelica Conference*, 581–588. Prague, Czech Republic, May 2017. Modelica Association. doi:10.3384/ecp17132581.

[JKC+16] WoonSeong Jeong, Jong Bum Kim, Mark J. Clayton, Jeff S. Haberl, and Wei Yan. A framework to integrate object-oriented physical modelling with building information modelling for building thermal simulation. *Journal of Building Performance Simulation*, 9(1):50–69, January 2016. doi:10.1080/19401493.2014.993709.

[JS16] WoonSeong Jeong and JeongWook Son. An algorithm to translate building topology in Building Information Modeling into object-oriented physical modeling-based building energy modeling. *Energies*, 9(1):50, January 2016. doi:10.3390/en9010050.

[JH16] Filip Jorissen and Lieve Helsen. Towards an Automated ToolChain for MPC in Multi-zone Buildings. In *International High Performance Buildings Conference*, number 4. West-Lafayette, Indiana, 2016.

[JWH15] Filip Jorissen, Michael Wetter, and Lieve Helsen. Simulation speed analysis and improvements of Modelica models for building energy simulation. In Peter Fritzson and Hilding Elmqvist, editors, *11-th International Modelica Conference*, 59–69. Paris, France, September 2015. Modelica Association. doi:10.3384/ecp1511859.

[JN95] R. Judkoff and J. Neymark. *International Energy Agency building energy simulation test (BESTEST) and diagnostic method*. National Renewable Energy Laboratory (NREL), February 1995. URL: http://www.osti.gov/scitech/servlets/purl/90674, doi:10.2172/90674.

[JN06] Ron Judkoff and Joel Neymark. Model validation and testing: The methodological foundation of ASHRAE Standard 140. *ASHRAE Transactions*, 112 PART 2:367–376, 2006.

[KH09] D. E. Kalz and S. Herkel. The impact of auxiliary energy on the efficiency of the heating and cooling system: monitoring of low-energy buildings. *Energy and Buildings*, 41(10):1019–1030, 2009. doi:10.1016/j.enbuild.2009.05.004.

[KRD13] Chen Kan, Streblow Rita, and Mueller Dirk. Simulation based design and validation of home energy system. In *13th Conference of International Building Performance Simulation Association*. 2013.

[KB05a] Srinivas Katipamula and Michael R Brambley. Methods for fault detection, diagnostics, and prognostics for building systems - a review, part ii. *HVAC&R Research*, 11(2):169–187, 2005. doi:10.1080/10789669.2005.10391133.

[KB05b] Srinivas Katipamula and Michael R Brambley. Methods for fault detection, diagnostics, and prognostics for building systems - a review, part i. *HVAC&R Research*, 11(1):3–25, 2005. doi:10.1080/10789669.2005.10391123.

[KMB11] Anthony Kelman, Yudong Ma, and Francesco Borrelli. Analysis of local optima in predictive control for energy efficient buildings. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, 5125–5130. 2011. doi:10.1109/CDC.2011.6161498.

[KPHR14] Eui-Jong Kim, Gilles Plessis, Jean-Luc Hubert, and Jean-Jacques Roux. Urban energy simulation: Simplification and reduction of building envelope models. *Energy and Buildings*, 84:193–202, 2014. doi:10.1016/j.enbuild.2014.07.066.

[Kle93] S. A. Klein. Development and integration of an equation-solving program for engineering thermodynamics courses. *Computer Applications in Engineering Education*, 1(3):265–275, 1993. doi:10.1002/cae.6180010310.

[KDB76] S. A. Klein, J. A. Duffie, and W. A. Beckman. TRNSYS – A transient simulation program. *ASHRAE Transactions*, 82(1):623–633, 1976.

[Kof03] Ernesto Kofman. Quantization-based simulation of differential algebraic equation systems. *SIMULATION*, 79(7):363–376, 2003. doi:10.1177/0037549703038881.

[KJ01] Ernesto Kofman and Sergio Junco. Quantized-state systems: a DEVS approach for continuous system simulation. *Transactions of The Society for Modeling and Simulation International*, 18(1):2–8, 2001.

[KKG+14] Philipp Kräuchi, M. Kolb, Thomas Gautschi, Urs-Peter Menti, and Matthias Sulzer. Modellbildung für thermische Arealvernetzung mit IDA-ICE. In *BauSIM 2014*. Aachen, Germany, September 2014. URL: http://www.ibpsa.org/proceedings/bausimPapers/2014/p1130_final.pdf.

[KK12] Philipp Kräuchi and Matthias Kolb. Simulation thermischer Arealvernetzung mit IDA-ICE. In *BauSIM 2012*. Berlin, Germany, September 2012. URL: http://www.ibpsa.org/proceedings/bausimPapers/2012/BauSIM2012_139.pdf.

[KSS15] Philipp Kräuchi, Thomas Schluck, and Matthias Sulzer. Modelling of low temperature heating networks with ida-ice. In *Proceedings of International Conference CISBAT*, 827–832. Lausanne, Switzerland, September 2015. doi:10.5075/epfl-cisbat2015-827-832.

[KBS+16] M Köfinger, Daniele Basciotti, R.R. Schmidt, E. Meissner, C. Doczekal, and A. Giovannini. Low temperature district heating in Austria: Energetic, ecologic and economic comparison of four case studies. *Energy*, 2016. doi:10.1016/j.energy.2015.12.103.

[LOK10] Tuomas Laine, Olof Granlund Oy, and Antti Karola. Energy and thermal performance management through utilisation of building information models. 2010.

[LPBhmR02] Helge V. Larsen, Halldór Pálsson, Benny Bøhm, and Hans F. Ravn. Aggregated dynamic simulation model of district heating networks. *Energy Conversion and Management*, 43(8):995–1019, 2002. doi:10.1016/S0196-8904(01)00093-0.

[LTF+14] M. Lauster, J. Teichmann, M. Fuchs, R. Streblow, and D. Mueller. Low order thermal network models for dynamic simulations of buildings on city district scale. *Building and Environment*, 73:223–231, 2014. doi:10.1016/j.buildenv.2013.12.016.

[LFH+15] Moritz Lauster, Marcus Fuchs, Max Huber, Peter Remmen, Rita Streblow, and Dirk Müller. Adaptive thermal building models and methods for scalable simulations of multiple buildings using Modelica. In *BS2015, 14th Conference of International Building Performance Simulation Association*. 2015.

[LFT+13] Moritz Lauster, Marcus Fuchs, Jens Teichmann, Rita Streblow, and Dirk Müller. Energy simulation of a research campus with typical building setups. In *BS2013, 13th Conference of International Building Performance Simulation Association*. 2013.

[LBN13] LBNL. Simergy: Simergy Homepage. October 2013. URL: https://simergy-beta.lbl.gov/.

[LKH05] Thoi H Le, Gottfried Knabe, and Gregor P Henze. Fault detection and diagnosis of control strategies for air-handling units. In *Building Simulation*, 609–616. Montréal, 2005.

[LNNW15] Edward A. Lee, Mehrdad Niknami, Thierry S. Nouidui, and Michael Wetter. Modeling and simulating cyber-physical systems using CyPhySim. In *EMSOFT*. October 2015. URL: http://simulationresearch.lbl.gov/wetter/download/2015-LeeEtAl_CyPhySim_EMSOFT.pdf, doi:10.1109/EMSOFT.2015.7318266.

[LY10] S.H. Lee and F.W.H. Yik. A study on the energy penalty of various air-side system faults in buildings. *Energy and Buildings*, 42(1):2–10, January 2010. doi:10.1016/j.enbuild.2009.07.004.

[LRSV82] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli. The waveform relaxation method for time-domain analysis of large scale integrated circuits. 1982. doi:10.1109/TCAD.1982.1270004.

[Lel82] Ekachai Lelarasmee. The waveform relaxation method for time domain analysis of large scale integrated circuits: theory and applications, memorandum no. ucb/erl m82/40. 1982.

[LS12] Hongwei Li and Svend Svendsen. Energy and exergy analysis of low temperature district heating network. *Energy*, 45(1):237–246, 2012. doi:10.1016/j.energy.2012.03.056.

[Lic98] Gerwald Lichtenberg. *Theorie und Anwendung der qualitativen Modellierung zeitdiskreter dynamischer Systeme durch nichtdeterministische Automaten*. volume 686 of Fortschrittberichte VDI, Reihe 8: Mess-, Steuerungs- und Regelungstechnik. VDI Verlag GmbH, 1998. ISBN 3-18-368608-2. URL: http://d-nb.info/953082466/about/html.

[LS96] Gerwald Lichtenberg and Andrew Steele. An Approach to Fault Diagnosis using parallel qualitative Observers. In *International Workshop on Discrete Event Systems*, 290 – 295. Institution of Electrical Engineers, 1996.

[LSW+13] Thomas Liebich, Konrad Stuhlmacher, Matthias Weise, Romy Guruz, Peter Katranuschkov, and Raimar J Scherer. HESMOS D+ Additional Deliverable: Information Delivery Manual Work within HESMOS. Technical Report, © HESMOS Consortium, Brussels, Belgium, 2013. URL: http://hesmos.eu/downloads/hesmos_additional-del_idm_final.pdf.

[LH06a] Simeng Liu and Gregor P. Henze. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: part 1. theoretical foundation. *Energy and Buildings*, 38(2):142–147, 2006. doi:10.1016/j.enbuild.2005.06.002.

[LH06b] Simeng Liu and Gregor P. Henze. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: part 2: results and analysis. *Energy and Buildings*, 38(2):148–161, 2006. doi:10.1016/j.enbuild.2005.06.001.

[LNGU09] Manuel Ljubijankic, Christoph Nytsch-Geusen, and Steffen Unger. Modelling of complex thermal energy supply systems based on the Modelica-Library FluidFlow. *7th Modelica Conference*, pages 335–340, 2009. doi:10.3384/ecp09430070.

[Lof04] J. Lofberg. Yalmip: a toolbox for modeling and optimization in matlab. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium*. IEEE, 2004.

[LS82] D.W. Low and E.F. Sowell. ENET, a PC-based building energy simulation system. In *Energy Programs Conference*, 2–7. Austin, Texas, September 1982. IBM Real Estate and Construction Division.

[LWW+14] Henrik Lund, Sven Werner, Robin Wiltshire, Svend Svendsen, Jan Eric Thorsen, Frede Hvelplund, and Brian Vad Mathiesen. 4th Generation District Heating (4GDH). Integrating smart thermal grids into future sustainable energy systems. *Energy*, 68:1–11, 2014. doi:10.1016/j.energy.2014.02.089.

[LLM06] Karin Lunde, Rüdiger Lunde, and Burkhard Münker. Model-based failure analysis with RODON. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence*, 647–651. IOS Press, 2006.

[Lun94] J. Lunze. Qualitative modelling of linear dynamical systems with quantized state measurements. *Automatica*, 30(3):417–431, 1994. doi:10.1016/0005-1098(94)90119-8.

[MBH+12] Y Ma, F Borrelli, B Hencey, B Coffey, S Bengea, and P Haves. Model predictive control for the operation of building cooling systems. *IEEE*

Transactions on Control Systems Technology, 20(3):796–803, 2012. doi:10.1109/TCST.2011.2124461.

[MWA+17] Alessandro Maccarini, Michael Wetter, Alireza Afshari, GÃ¶ran Hultmark, Niels C. BergsÃ¸e, and Anders Vorre. Energy saving potential of a two-pipe system for simultaneous heating and cooling of office buildings. *Energy and Buildings*, 134:234 – 247, 2017. doi:10.1016/j.enbuild.2016.10.051.

[MA12] Fredrik Magnusson and Johan Åkesson. Collocation methods for optimization in a Modelica environment. In *Proceedings of the 9th International Modelica Conference*, 649–658. 2012. doi:10.3384/ecp12076649.

[MHS15] Tobias Maile, Philip Haves, and Richard See. Integrating a rule based code compliance software platform into a building simulation front end. In *14-th IBPSA Conference*, 1909–1915. International Building Performance Simulation Association, December 2015. URL: http://www.ibpsa.org/proceedings/BS2015/p2914.pdf.

[MS96] Andreas Malik and Peter Struss. Diagnosis of dynamic systems does not necessarily require simulation. In *7th Int. Workshop on Principles of Diagnosis*. 1996.

[MS93] Sven Erik Mattsson and Gustaf Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing*, 14(3):677–692, 1993. doi:10.1137/0914043.

[MBZF08] N. Mendes, Rogerio Marcos Barbosa, Freire Roberto Zanetti, and Oliveira R. C. L. F. *A Simulation Environment for Performance Analysis of HVAC Systems*. Building Simulation Springer, 2008. doi:10.1007/s12273-008-8216-7.

[MOG03] N. Mendes, R.C.L.F. Oliveira, and Henrique G. Domus 2.0: a whole-building hygrothermal simulation program. In *Eighth IBPSA International Building Performance Simulation Association*, 863–870. Eindhoven, Netherlands, 2003. IBPSA International Building Performance Simulation Association.

[Mer02] Rolf Mathias Merz. *Objektorientierte Modellierung thermischen Gebäudeverhaltens*. PhD thesis, Universität Kaiserslautern, Kaiserslautern, Germany, September 2002. URL: https://kluedo.ub.uni-kl.de/files/1345/kluedo-1500.pdf.

[ME09] Boris Michaelsen and Joerg Eiden. HumanComfort modelica-library thermal comfort in buildings and mobile applications. In Francesco Casella, editor, *Proc. of the 7-th International Modelica Conference*, 403–412. Como, Italy, September 2009. Modelica Association. doi:10.3384/ecp09430082.

[MBKC13] Gustavo Migoni, Mario Bortolotto, Ernesto Kofman, and François E. Cellier. Linearly implicit quantization-based integration methods for stiff ordinary differential equations. *Simulation Modelling Practice and Theory*, 35(0):118 – 136, 2013. doi:10.1016/j.simpat.2013.03.004.

[MHB+15] R.J. Miranda, S. Huang, G.A., Barrios, D. Li, and W. Zuo. Energy efficient design for hotels in the tropical climate using modelica. In *Proceedings of the 11th International Modelica Conference*. Versailles, France, 2015. doi:10.3384/ecp1511871.

[MI03] SA Mumma and Application Issues. Detecting system degradation. *ASHRAE IAQ Applications*, pages 14–15, 2003.

[MKLR15] Thorsten Müller, Kai Kruppa, Gerwald Lichtenberg, and Nicolas Réhault. Fault detection with qualitative models reduced by tensor decomposition methods. *IFAC-PapersOnLine*, 48(21):416–421, 2015. doi:10.1016/j.ifacol.2015.09.562.

[MW15] W. Müller and E. Widl. Using FMI components in discrete event systems. In *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2015 Workshop on*, 1–6. April 2015. doi:10.1109/MSCPES.2015.7115397.

[Nem16] Nemetschek. Allplan Architecture Neuerungen 2016. 2016. URL: https://www.allplan.com/en/software/architecture/allplan-architecture-2017.html.

[NIB13] NIBS. *National BIM Standard-United State - V2 eBook*. National Institute of Building Science and buildingSMARTalliance, second edition, 2013. URL: https://www.nibs.org/store/ViewProduct.aspx?id=2038899.

[NIB16] NIBS. About the National BIM Standard-United States. 2016. URL: https://www.nationalbimstandard.org/about.

[NKWM12] Thierry Stephane Nouidui, Phalak Kaustubh, Zuo Wangda, and Wetter Michael. Validation and application of the room model of the Modelica Buildings library. In *Proc. of the 9th International Modelica Conference*. 2012. doi:10.3384/ecp12076727.

[NW14] Thierry Stephane Nouidui and Michael Wetter. Linking simulation programs, advanced control and FDD algorithms with a building management system based on the Functional Mock-Up Interface and the building automation Java architecture standards. In *ASHRAE/IBPSA-USA Building Simulation Conference*, 49–55. Atlanta, GA, September 2014. IBPSA-USA. URL: http://simulationresearch.lbl.gov/wetter/download/2014-IBPSA-USA-NouiduiWetter.pdf.

[NWZ14] Thierry Stephane Nouidui, Michael Wetter, and Wangda Zuo. Functional mock-up unit for co-simulation import in EnergyPlus. *Journal of Building Performance Simulation*, 7(3):192–202, 2014. doi:10.1080/19401493.2013.808265.

[NBK+15] Romain Nouvel, Jean-Marie Bahu, Robert Kaden, Jérôme Kämpf, Piergiorgio Cipriano, Moritz Lauster, Karl-Hainz Haefele, Esteban Munoz, Olivier Tournaire, and Egbert Casper. Development of the CityGML application domain extension energy for urban energy simulation. In *BS2015, 14th Conference of International Building Performance Simulation Association*, 560–564. 2015.

[NRE10] NREL. NREL: OpenStudio. 2010. URL: http://openstudio.nrel.gov/.

[NGHLR12] Christoph Nytsch-Geusen, Jörg Huber, Manuel Ljubijankic, and Jörg Rädler. Modelica BuildingSystems - Eine Modellbibliothek zur Simulation komplexer energietechnischer Gebäudesysteme. In *Proceedings of the BAUSIM 2012 Conference*. Berlin, Germany, September 2012. URL: http://www.ibpsa.org/conferences.htm.

[NGHLR13] Christoph Nytsch-Geusen, Jörg Huber, Manuel Ljubijankic, and Jörg Rädler. Modelica BuildingSystems - eine Modellbibliothek zur Simulation komplexer energietechnischer Gebäudesysteme. *Bauphysik*, 35(1):21–29, 2013. doi:10.1002/bapi.201310045.

[NGHN13] Christoph Nytsch-Geusen, Jörg Huber, and Yue Nie. Simulation-based design of pv cooling systems for residential buildings in hot and dry climates. In *Proceedings of the 13th International IBPSA Conference*. Aix-les-Bains, France, August 2013. International Building Performance Simulation Association.

[NGK15] Christoph Nytsch-Geusen and Werner Kaul. Generation of dynamic energetic district models from statistical relationships. In *BS2015, 14th Conference of International Building Performance Simulation Association*, 324–330. 2015.

[NGNHH05] Christoph Nytsch-Geusen, Thierry Nouidui, Andreas Holm, and Wolfram Haupt. A hygrothermal building model based on the object-oriented modeling language Modelica. In Ian Beausoleil-Morrison and Michel Bernier, editors, *Proceedings of the Ninth International IBPSA Conference*, volume 1, 867–876. Montreal, Canada, August 2005. International Building Performance Simulation Association and Ecole Polytechnique de Montreal.

[OMR+13] James O'Donnell, T. Maile, C Rose, N Mrazovic, E. Morrissey, Kristen Parrish, C Regnier, and V. Bazjanac. Transforming BIM to BEM: Generation of Building Geometry for the NASA Ames Sustainability Base BIM. Technical Report LBNL-6033E, Lawrence Berkeley National Laboratory, Berkeley, CA, January 2013. URL: http://buildings.lbl.gov/sites/all/files/LBNL-6033E.pdf.

[OSM+11] James T. O'Donnell, Richard See, Tobias Maile, Vladimir Bazjanac, and Philip Haves. SimModel: A domain data model for whole building energy simulation. In *IBPSA Building Simulation 2011. Sydney, Australia*. Sydney, Australia, 2011. IBPSA.

[OCCM14] OCC'M. Raz'r. 2014.

[OPJ+10] Frauke Oldewurtel, Alessandra Parisio, Colin N Jones, Manfred Morari, Dimitrios Gyalistras, Markus Gwerder, Vanessa Stauch, Beat Lehmann, and Katharina Wirth. Energy efficient building climate control using stochastic model predictive control and weather predictions. In *Proceedings of the American Control Conference (ACC), 2010*, 5100–5105. 2010. doi:10.1109/ACC.2010.5530680.

[OOME08] Hans Olsson, Martin Otter, Sven Erik Mattsson, and Hilding Elmqvist. Balanced models in Modelica 3.0 for increased model quality. In Bernhard Bachmann, editor, *Proc. of the 6-th International Modelica Conference*, volume 1, 21–33. Bielefeld, Germany, March 2008. Modelica Association and University of Applied Sciences Bielefeld. URL: http://www.modelica.org/events/modelica2008/Proceedings/sessions/session1a3.pdf.

[Ope15] OpenIDEAS. Integrated District Energy Assessment Simulations. 2015. URL: https://github.com/open-ideas.

[OE17] Martin Otter and Hilding Elmqvist. Transformation of differential algebraic array equations to index one form. In *Proc. of the 12-th International Modelica Conference*, 565–579. Prague, Czech Republic, May 2017. Modelica Association. doi:10.3384/ecp17132565.

[PD00] Peter Jan Pahl and Rudolf Damrath. *Mathematische Grundlagen der Ingenieurinformatik*. Springer-Verlag Berlin Heidelberg, 2000. doi:10.1007/978-3-642-57013-1.

[Pal93] Olafur Petur Palsson. *Stochastic modeling, control and optimization of district heating systems*. PhD thesis, Ph. D. thesis, IMSOR, DTU, 1993.

[PWBH11] Xiufeng Pang, Michael Wetter, Prajesh Bhattacharya, and Philip Haves. A framework for simulation-based real-time whole building performance assessment. *Building and Environment*, 54(54):100–108, August 2011. doi:10.1016/j.buildenv.2012.02.003.

[Pan88] Constantinos C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing*, 9(2):213–231, March 1988. doi:10.1137/0909014.

[PS13] Thomas Pauschinger and Thomas Schmidt. Solar unterstützte Kraft-Wärme-Kopplung mit saisonalem Wärmespeicher. *Euro Heat & Power*, 2013.

[PBR+12] Matthias Pazold, Sebastian Burhenne, Jan Radon, Sebastian Herkel, and Florian Antretter. Integration of Modelica models into an existing simulation software using FMI for co-simulation. In *9th International Modelica Conference*. 2012. doi:10.3384/ecp12076949.

[PH14] Damien Picard and Lieve Helsen. A new hybrid model for borefield heat exchangers performance evaluation. In *ASHRAE Annual Conference*, volume 120, 1–8. Seattle, WA, 2014. ASHRAE.

[PJH15] Damien Picard, Filip Jorissen, and Lieve Helsen. Methodology for Obtaining Linear State Space Building Energy Simulation Models. In *11th International Modelica Conference*, 51–58. Versailles, France, 2015. doi:10.3384/ecp1511851.

[PSJ16] Damien Picard, Maarten Sourbron, and Filip Jorissen. Comparison of Model Predictive Control Performance Using Grey-Box and White-Box Controller Models of a Multi-zone Office Building. In *International High Performance Buildings Conference*, 4. West-Lafayette, Indiana, 2016.

[PKH01] Mary Ann Piette, Sat Kartar Kinney, and Philip Haves. Analysis of an information monitoring and diagnostic system to improve building operations. *Energy and Buildings*, 33 (8):783–791, 2001. doi:10.1016/S0378-7788(01)00068-8.

[PWM+16] Sergio Pinheiro, Reinhard Wimmer, Sergej Muhic, Tobias Maile, James O'Donnell, Vladimir Bazjanac, Jérôme Frisch, and Christoph van Treeck. Model View Defintion for Advanced Building Energy Performance Simulation. In *BauSim Conference*. Dresden, Germany, 2016.

[PKL14] Gilles Plessis, Aurélie Kaemmerlen, and Amy Lindsay. BuildSysPro: a Modelica library for modelling buildings and energy systems. In *10-th International Modelica Conference*, 1161–1169. Lund, Sweden, March 2014. Modelica Association. doi:10.3384/ECP140961161.

[Pol97] Elijah Polak. *Optimization – Algorithms and Consistent Approximations*. volume 124 of Applied Mathematical Sciences. Springer Verlag, 1997. doi:10.1007/978-1-4612-0663-7.

[PW06] Elijah Polak and Michael Wetter. Precision control for generalized pattern search algorithms with adaptive precision function evaluations. *SIAM Journal on Optimization*, 16(3):650–669, 2006. doi:10.1137/040605527.

[PH13] Atul Porwal and Kasun N. Hewage. Building Information Modeling (BIM) partnering framework for public construction projects. *Automation in Construction*, 31:204–214, May 2013. doi:10.1016/j.autcon.2012.12.004.

[PBS15] Christina Protopapadaki, Ruben Baetens, and Dirk Saelens. Exploring the impact of heat pump-based dwelling design on the low-voltage distribution grid. In *BS2015, 14th Conference of International Building Performance Simulation Association*, 1–8. 2015.

[Pto14] Claudius Ptolemaeus, editor. *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014. URL: http://ptolemy.org/books/Systems.

[Pal00] Hálldór Pálsson. *Methods for planning and operating decentralized combined heat and power plants*. Risø & DTU-Department of Energy Engineering (ET), 2000.

[RRDW15] Abbass Raad, Vincent Reinbold, B. Delinchant, and F. Wurtz. Energy building co-simulation based on the WRM algorithm for efficient simulation over FMU components of web service. In *Proceedings of Building Simulation 2015*. IBPSA, 2015. URL: http://www.ibpsa.org/proceedings/BS2015/p2387.pdf.

[RK11] Paul Raftery and Marcus Keane. Visualizing patterns in building performance data. In IBPSA International Building Performance Simulation Association, editor, *Proceedings of 12th Building Simulation Conference*. 2011.

[RES10] RESNET. Commercial Buildings Energy Modeling Guidelines and Procedures. August 2010. URL: http://www.comnet.org.

[RDS14] G. Reynders, J. Diriken, and D. Saelens. Quality of grey-box models and identified parameters as function of the accuracy of input and observation signals. *Energy and Buildings*, 82:263–274, 2014. doi:10.1016/j.enbuild.2014.07.025.

[RTFC+15] Carles Ribas Tugores, Henning Francke, Falk Cudok, Alexander Inderfurth, Stefan Kranz, and Christoph Nytsch-Geusen. Coupled modeling of a District Heating System with Aquifer Thermal Energy Storage and Absorption Heat Transformer. In *11th International Modelica Conference*. 2015. doi:10.3384/ecp15118197.

[Ron14] Armin Ronacher. Jinja2 Templates for Python. http://jinja.pocoo.org/, 2014. Accessed: 2015-05-13.

[RB13] Cody M. Rose and Vladimir Bazjanac. An algorithm to generate space boundaries for building energy simulation. *Engineering with Computers*, 29(4):1–10, 2013. doi:10.1007/s00366-013-0347-5.

[RGCJ+16] Sergi Rotger-Griful, Spyros Chatzivasileiadis, Rune Hylsberg Jacobsen, Emma M. Stewart, Javier Matanza Domingo, and Michael Wetter. Hardware-in-the-loop co-simulation of distribution grid for demand response. In *Proc. of the 19th Power Systems Computation Conference (PSCC)*. Genoa, Italy, June 2016. doi:10.1109/PSCC.2016.7540828.

[RLW+05] Kurt Roth, Patricia Llana, Detlef Wetphalen, James Brodrick, Energy-savings Potential, and Market Factors. Automated whole building diagnostics. *ASHRAE journal*, 47(5):82–84, 2005.

[RWF05] KW Roth, D Westphalen, and MY Feng. *Energy impact of commercial building controls and performance diagnostics: market characterization, energy impact of building faults and energy savings*. National Technical Information Service, U.S. Department of Commerce, Springfield, VA, 2005.

[RLV+15] Håkan Runvik, Per-Ola Larsson, Stéphane Velut, Jonas Funquist, Markus Bohlin, Andreas Nilsson, and Sara Modarrez Razavi. Production planning for distributed district heating networks with JModelica.org. In *11th International Modelica Conference*, 217–223. 2015. doi:10.3384/ecp15118217.

[RE15] RWTH-EBC. Aixlib - a Modelica model library for building performance simulations. 2015. URL: https://github.com/RWTH-EBC/AixLib.

[Sah96] Per Sahlin. *Modelling and Simulation Methods for Modular Continuous Systems in Buildings*. PhD thesis, KTH, Stockholm, Sweden, May 1996. URL: http://www.equa.se/dncenter/thesis.pdf.

[SEG+04] Per Sahlin, Lars Eriksson, Pavel Grozman, Hans Johnsson, Alexander Shapovalov, and Mika Vuolle. Whole-building simulation with symbolic DAE equations and general purpose solvers. *Building and Environment*, 39(8):949–958, August 2004. doi:10.1016/j.buildenv.2004.01.019.

[SS89] Per Sahlin and Edward F. Sowell. A neutral format for building simulation models. In *Proceedings of the Second International IBPSA Confer-*

ence, 147–154. Vancouver, BC, Canada, June 1989. URL: http://www.ibpsa.org/conferences.htm.

[SHK14] A. Samuel, J. Hand, and N. Kelly. High resolution modelling for performance assessment of future dwellings. In *Proceedings of the Second National IBPSA-England Conference*. London, UK, June 2014.

[SN14] Jibonananda Sanyal and Joshua New. Building simulation modelers – are we big data ready? September 2014.

[SJD+14] J Schiefelbein, A Javadi, M Diekerhof, R Streblow, D Müller, and A Monti. GIS supported city district energy system modeling. In *9th International Conference on System Simulation in Buildings*, number 1, 1–17. Liège, 2014.

[SJL+15] Jan Schiefelbein, Amir Javadi, Moritz Lauster, Peter Remmen, Rita Streblow, and Dirk Müller. Development of a city information model to support data management and analysis of building energy systems within complex city districts. In *CISBAT2015, International Conference on Future Buildings & Districts- Sustainability from Nano to Urban Scale*. 2015.

[SKS15] Thomas Schluck, Philipp Kräuchi, and Matthias Sulzer. Non-linear thermal networks - how can a meshed network improve energy efficiency? In *Proceedings of International Conference CISBAT*, 779–785. Lausanne, Switzerland, September 2015. doi:10.5075/epfl-cisbat2015-779-784.

[SSMM14] Tim Schlösser, Sebastian Stinner, Antonello Monti, and Dirk Müller. Analyzing the impact of home energy systems on the electrical grid. In *2014 Power Systems Computation Conference*, 1–7. IEEE, 2014. doi:10.1109/PSCC.2014.7038378.

[SSP+15] Tim Schlösser, Sebastian Stinner, Ferdinanda Ponci, Rita Streblow, Antonello Monti, and Dirk Müller. Impact of control approaches for building energy systems on distribution grid state. *2015 IEEE Eindhoven PowerTech*, pages 1–6, 2015. doi:10.1109/PTC.2015.7232791.

[S03] Jochen Schröder. *Modelling, State Observation and Diagnosis of Quantised Systems*. volume 282 of Lecture Notes in Control and Information Sciences. Springer-Verlag, 2003. ISBN ISBN 978-3-540-44075-8. doi:10.1007/3-540-46086-1.

[SWF+15] Joseph Schuchart, Volker Waurich, Martin Flehmig, Marcus Walther, Wolfgang E. Nagel, and Ines Gubsch. Exploiting repeated structures and vectorization in Modelica. In Peter Fritzson and Hilding Elmqvist, editors, *11-th International Modelica Conference*, 265–272. Paris, France, September 2015. Modelica Association. doi:10.3384/ecp15118265.

[SULK14] Torsten Schwan, René Unger, Christian Lerche, and Christian Kehrer. Model-based design of integrative energy concepts for building quarters using Modelica. In *Proceedings of the 10th International Modelica Conference*, 97–106. March 2014. doi:10.3384/ecp1409697.

[SHS+11] Richard See, Philip Haves, Pramod Sreekanthan, Mangesh Basarkar, James O'Donnell, and Kevin Settlemyre. Development of a user interface for the EnergyPlus whole building energy simulation program. In *IBPSA Building Simulation 2011. Sydney, Australia*. Sydney, Australia, 2011. IBPSA.

[SML13] Muhammad Tariq Shafiq, Jane Matthews, and Steve Lockley. A study of BIM collaboration requirements and available features in existing model collaboration systems. *Journal of Information Technology in Construction (ITcon)*, 18:148–161, April 2013. URL: http://www.itcon.org/2013/8.

[SMD+12] Bruce Sibbitt, Doug McClenahan, Reda Djebbar, Jeff Thornton, Bill Wong, Jarrett Carriere, and John Kokko. The performance of a high solar fraction seasonal storage district heating system - five years of operation. *Energy Procedia*, 30:856–865, January 2012. doi:10.1016/j.egypro.2012.11.097.

[Sim15] EQUA Simulation. IDA-ICE. March 2015. URL: http://www.equa.se/.

[STHS14] F F M Soons, J Ignacio Torrens, J L M Hensen, and R A M De Schrevel. A Modelica based computational model for evaluating a renewable district heating system. *9th International Conference on System Simulations in Buildings*, 2(1):1–16, 2014.

[STLL84] E.F. Sowell, K. Taghavi, H. Levy, and D.W. Low. Generation of building energy system models. *ASHRAE Transactions*, 90(1B):573–586, 1984.

[SBN89] Edward F. Sowell, W. Fred Buhl, and Jean-Michel Nataf. Object-oriented programming, equation-based submodels, and system reduction in SPANK. In *Proceedings of the Second International IBPSA Conference*, 141–146. Vancouver, BC, Canada, June 1989. URL: http://www.ibpsa.org/conferences.htm.

[SBEW86] Edward F. Sowell, Walter F. Buhl, Ahmet E. Erdem, and Frederick C. Winkelmann. A prototype object-based system for HVAC simulation. Technical Report LBL-22106, Lawrence Berkeley National Laboratory, September 1986.

[SH01] Edward F. Sowell and Philip Haves. Efficient solution strategies for building energy system simulation. *Energy and Buildings*, 33(4):309–317, 2001. doi:10.1016/S0378-7788(00)00113-4.

[SA00] Aleksandar M. Stankovic and Timur Aydin. Analysis of asymmetrical faults in power systems using dynamic phasors. *Power Systems, IEEE Transactions on*, 15(3):1062–1068, 2000. doi:10.1109/59.871734.

[SLA99] Alex M. Stankovic, Bernard C. Lesieutre, and Timur Aydin. Modeling and analysis of single-phase induction machines with dynamic phasors. *Power Systems, IEEE Transactions on*, 14(1):9–14, 1999. doi:10.1109/59.744460.

[Ste15] Raymond Sterling. *Self-aware buildings: an evaluation framework and implementation technologies for improving building operations*. PhD thesis, National University of Ireland, Galway, 2015.

[SPF+14] Raymond Sterling, Gregory Provan, Jesús Febres, Dominic O Sullivan, Peter Struss, and M Keane. Model-based fault detection and diagnosis of air handling units : a comparison of methodologies. *Energy Procedia*, 62(0):686–693, 2014. doi:10.1016/j.egypro.2014.12.432.

[Str08] Peter Struss. Model-based problem solving. In V. Lifschitz F. van Harmelen and B. Porter, editors, *Handbook of Knowledge Representation, Elsevier*, volume 6526, pages 395 – 465. Elsevier B.V., 2008. doi:10.1016/S1574-6526(07)03010-6.

[SWI15] SWIG. SWIG: simplified wrapper and interface generator [Computer Software]. In *Retrieved from http://www.swig.org*. 2015.

[THN13] Roshana Takim, Mohd Harris, and Abdul Hadi Nawawi. Building Information Modeling (BIM): a new paradigm for quality of life within Architectural, Engineering and Construction (AEC) industry. *Procedia - Social and Behavioral Sciences*, 101:23–32, November 2013. doi:10.1016/j.sbspro.2013.07.175.

[TCT+16] Jean-Philippe Tavella, Mathieu Caujolle, Charles Tan, Gilles Plessis, Mathieu Schumann, Stéphane Vialle, Cherifa Dad, Arnaud Cuccuru, and Revol Sébastien. Toward an hybrid co-simulation with the FMI-CS standard. Technical Report, EDFLab Paris-Saclay, Saclay, France, RISEGRID Institute, Saclay, France, EDFLab Les Renardières, Moret, France, Centrale Supélec, Saclay, France, CEA LIST, Saclay, France, April 2016. URL: https://hal-centralesupelec.archives-ouvertes.fr/hal-01301183/document.

[TSPC12] Steven T. Taylor, Jeff Stein, Gwelen Paliaga, and Hwakong Cheng. Dual maximum VAV box control logic. *ASHRAE Journal*, 54(12):16–24, 2012.

[TRR+15] Matthis Thorade, Jörg Rädler, Peter Remmen, Tobias Maile, Reinhard Wimmer, Jun Cao, Moritz Lauster, Christoph Nytsch-Geusen, Dirk Müller, and Christoph van Treeck. An open toolchain for generating Modelica code from building information models. In Peter Fritzson and Hilding Elmqvist, editors, *11-th International Modelica Conference*, 383–391. Paris, France, September 2015. Modelica Association. doi:10.3384/ecp15118383.

[Til01] Michael M. Tiller. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publisher, 2001. doi:10.1007/978-1-4615-1561-6.

[TEP05] Hubertus Tummescheit, Jonas Eborn, and Katrin Prölss. AirConditioning - a Modelica library for dynamic simulation of AC systems. In Gerhard Schmitz, editor, *Proceedings of the 4th Modelica conference*, 185–192. Hamburg, Germany, March 2005. Modelica Association and Hamburg University of Technology. URL: http://www.modelica.org/events/Conference2005/online_proceedings/Session3/Session3a1.pdf.

[TSBC11] Konstantin Turitsyn, P. Sulc, Scott Backhaus, and Michael Chertkov. Options for control of reactive power by distributed photovoltaic generators. *Proceedings of the IEEE*, 99(6):1063–1073, June 2011. doi:10.1109/JPROC.2011.2116750.

[UBTB14] UDK-Berlin and TU-Berlin. Solar Dechatlon 2014. Technical Report, University of Arts Berlin and Technical University Berlin, German Federal Ministry for economic Affairs and Energy, 2014.

[USM+12] René Unger, Torsten Schwan, B. Mikoleit, Bernard Bäker, Christian Kehrer, and Tobias Rodemann. “green building” – modelling renewable building energy systems and electric mobility concepts using Modelica. In *Proc. of the 9-th International Modelica Conference*, 897–906. Munich, Germany, September 2012. Modelica Association. doi:10.3384/ecp12076897.

[vdL14] FLJ van der Linden. General fault triggering architecture to trigger model faults in Modelica using a standardized blockset. In *10th International Modelica Conference*, number Section 5, 427–436. 2014. doi:10.3384/ECP14096427.

[VR15] Juan Van Roy. *Electric vehicle charging integration in buildings Local charging coordination and DC grids*. PhD thesis, KU Leuven, 2015.

[vTR06] Christoph van Treeck and Ernst Rank. Dimensional reduction of 3D building models using graph theory and its application in building energy simulation. 2006. doi:10.1007/s00366-006-0053-7.

[vTWM15] Christoph van Treeck, Reinhard Wimmer, and Tobias Maile. *BIM für die Energiebedarfsermittlung und Gebäudesimulation.*, chapter 18, pages 293–303. VDI Buch, Springer Verlag, 2015. doi:10.1007/978-3-658-05606-3_18.

[VLW+14] Stéphane Velut, Per-Ola Larsson, Johan Windahl, Linn Saarinen, and Katarina Boman. Short-term production planning for district heating networks with JModelica.org. In *10th International Modelica Conference*. 2014. doi:10.3384/ECP14096959.

[VSS14] Rebekka Volk, Julian Stengel, and Frank Schultmann. Building Information Modeling (BIM) for existing buildings – literature review and future needs. *Automation in Construction*, 38:109–127, March 2014. doi:10.1016/j.autcon.2013.10.023.

[WWB+14] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst. Embree: a kernel framework for efficient cpu ray tracing. *ACM Transactions on Graphics (TOG)*, 33(4):143, 2014. doi:10.1145/2601097.2601199.

[WRB03] Detlef Westphalen, Kurt W Roth, and James Brodrick. System & component diagnostics. *ASHRAE Journal*, pages 58–59, 2003.

[Wet05] Michael Wetter. BuildOpt – a new building energy simulation program that is built on smooth models. *Building and Environment*, 40(8):1085–1092, August 2005. doi:10.1016/j.buildenv.2004.10.003.

[Wet06a] Michael Wetter. Multizone airflow model in Modelica. In Christian Kral and Anton Haumer, editors, *Proc. of the 5-th International Modelica Conference*, volume 2, 431–440. Vienna, Austria, September 2006. Modelica Association and Arsenal Research. URL: http://www.modelica.org/events/modelica2006/Proceedings/sessions/Session413.pdf.

[Wet06b] Michael Wetter. Multizone building model for thermal building simulation in Modelica. In Christian Kral and Anton Haumer, editors, *Proc. of the 5-th International Modelica Conference*, volume 2, 517–526. Vienna, Austria, September 2006. Modelica Association and Arsenal Research. URL: http://www.modelica.org/events/modelica2006/Proceedings/sessions/Session5b4.pdf.

[Wet09] Michael Wetter. Modelica-based modeling and simulation to support research and development in building energy and control systems. *Journal of Building Performance Simulation*, 2(2):143–161, June 2009. doi:10.1080/19401490902818259.

[Wet11a] Michael Wetter. Co-simulation of building energy and control systems with the building controls virtual test bed. *Journal of Building Performance Simulation*, 3(4):185–203, 2011. doi:10.1080/19401493.2010.518631.

[Wet11b] Michael Wetter. The future of building system modeling and simulation. In Jan L. M. Hensen and Roberto Lamberts, editors, *Building Performance Simulation for Design and Automation*, pages 481–504. Taylor & Francis, Oxon, OX, 2011.

[WBN16] Michael Wetter, Marco Bonvini, and Thierry S. Nouidui. Equation-based languages - a new paradigm for building energy modeling, simulation and optimization. *Energy and Buildings*, 2016. doi:10.1016/j.enbuild.2015.10.017.

[WFG+15] Michael Wetter, Marcus Fuchs, Pavel Grozman, Lieve Helsen, Filip Jorissen, Moritz Lauster, Dirk Müller, Christoph Nytsch-Geusen, Damien Picard, Per Sahlin, and Matthis Thorade. IEA EBC Annex 60 Modelica library – an international collaboration to develop a free open-source model library for buildings and community energy systems. In *14-th IBPSA Conference*, 395–402. International Building Performance Simulation Association, December 2015. URL: http://www.iea-annex60.org/downloads/p2414.pdf.

[WFN15] Michael Wetter, Marcus Fuchs, and Thierry Stephane Nouidui. Design choices for thermofluid flow component and systems that are exported as Functional Mockup Units. In Peter Fritzson and Hilding Elmqvist, editors, *11-th International Modelica Conference*, 31–41. Paris, France, September 2015. Modelica Association. doi:10.3384/ecp1511831.

[WH06] Michael Wetter and Christoph Haugstetter. Modelica versus TRNSYS – A comparison between an equation-based and a procedural modeling language for building energy simulation. In *Proc. of SimBuild*. Cambridge, MA, August 2006. IBPSA-USA. URL: http://ceae.colorado.edu/ibpsa/ocs/viewpaper.php?id=162&cf=2.

[WHMS08] Michael Wetter, Philip Haves, Michael A. Moshier, and Edward F. Sowell. Using SPARK as a solver for Modelica. In *Proc. of SimBuild*. Berkeley, CA, August 2008. IBPSA-USA. URL: http://ibpsa.us/simbuild2008/technical_sessions/SB08-DOC-TS03-1-Wetter.pdf.

[WNL+15] Michael Wetter, Thierry S. Nouidui, David Lorenzetti, Edward A. Lee, and Amir Roth. Prototyping the next generation energyplus simulation engine. In *14-th IBPSA Conference*, 403–410. International Building Performance Simulation Association, December 2015. URL: http://www.iea-annex60.org/downloads/p2419.pdf.

[WvTH13] Michael Wetter, Christoph van Treeck, and Jan Hensen. New generation computational tools for building and community energy systems. Technical Report, International Energy Agency, 2013.

[WW04] Michael Wetter and Jonathan Wright. A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. *Building and Environment*, 39(8):989–999, August 2004. doi:10.1016/j.buildenv.2004.01.022.

[WZNP14] Michael Wetter, Wangda Zuo, Thierry S. Nouidui, and Xiufeng Pang. Modelica Buildings library. *Journal of Building Performance Simulation*, 7(4):253–270, 2014. doi:10.1080/19401493.2013.765506.

[WMB+15] E. Widl, W. Müller, D. Basciotti, S. Henein, S. Hauer, and K. Eder. Simulation of multi-domain energy systems based on the Functional Mock-up Interface specification. In *Smart Electric Distribution Systems and Technologies (EDST), 2015 International Symposium on*, 510–515. September 2015. doi:10.1109/SEDST.2015.7315261.

[WJEP15] Edmund Widl, Florian Judex, Katharina Eder, and Peter Palensky. Fmi-based co-simulation of hybrid closed-loop control system models. In *Complex Systems Engineering (ICCSE), 2015 International Conference on*, 1–6. IEEE, November 2015. doi:10.1109/ComplexSys.2015.7385981.

[WCR+15] Reinhard Wimmer, Jun Cao, Peter Remmen, Tobias Maile, James O'Donnell, Jérôme Frisch, Rita Streblow, Dirk Müller, and Christoph van Treeck. Implementation of advanced BIM-based mapping rules for automated conversion to Modelica. In *Proceedings of the 14th IBPSA Conference*. 2015.

[WMO+14] Reinhard Wimmer, Tobias Maile, James O'Donnell, Jun Cao, and Christoph van Treeck. Data-requirements specification to support BIM-based HVAC-definitions in Modelica. In *BauSIM 2014, Aachen, Germany, September, 2014*. 2014.

[WG06] Dietmar Winkler and Clemens Gühmann. Hardware-in-the-loop simulation of a hybrid electric vehicle using Modelica/Dymola. In *The 22nd International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium and Exposition*. 2006.

[WK10] Jeffrey Wix and Jan Karlshoej. Information Delivery Manual - Guide to Components and Development Methods. Technical Report, BuildingSMART International, 2010.

URL: http://web.stanford.edu/group/narratives/classes/08-09/CEE215/ReferenceLibrary/Industry Foundation Classes (IFC)/IDM/IDM2_Methodology.pdf.

[WB06] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. doi:10.1007/s10107-004-0559-y.

[ZL98] Bernard Zeigler and Jong Sik Lee. Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment. In *SPIE Conference on Enabling Technology for Simulation Science*, volume SPIE Vol. 3369, 49–58. Orlando, 1998. doi:10.1117/12.319354.

[Zha95] Hongping Zhao. *Analysis, Modelling and Operational Optimization of District Heating Systems*. Centre for District Heating Technology, Technical University of Denmark, 1995.

[ZLD+09] Jianjun Zhao, Ziqiang Li, Jianwan Ding, Liping Chen, Qifu Wang, Qing Lu, WangHongxin, and Shuang Wu. HIL simulation of aircraft thrust reverser hydraulic system in Modelica. In *7th International Modelica Conference*. 2009. doi:10.3384/ecp09430040.

[Zim13] Dirk Zimmer. Using artificial states in modeling dynamic systems: Turning malpractice into good practice. In *Proceedings of the 5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, 77–85. 2013. URL: http://www.ep.liu.se/ecp/084/009/ecp13084009.pdf.

[ZJB+16] Gerhard Zucker, Florian Judex, Max Blöchle, Mario Köstl, Edmund Widl, Stefan Hauer, Aurelien Bres, and Jyoti Zeilinger. A new method for optimizing operation of large neighborhoods of buildings using thermal simulation. *Energy and Buildings*, 125:153 – 160, 2016. doi:10.1016/j.enbuild.2016.04.081.

[ZC10] Wangda Zuo and Qingyan Chen. Simulations of air distributions in buildings by FFD on GPU. *HVAC&R Research*, pages 785–798, 2010. doi:10.1080/10789669.2010.10390934.

[ZWT+16] Wangda Zuo, Michael Wetter, Wei Tian, Dan Li, Mingang Jin, and Qingyan Chen. Coupling indoor airflow, HVAC, control and building envelope heat transfer in the Modelica Buildings library. *Journal of Building Performance Simulation*, 9:366–381, 2016. doi:10.1080/19401493.2015.1062557.

[AECUKC15] AEC (UK) Committee. AEC (UK) BIM Technology Protocol v2.1 - Practical implementation of BIM for the UK Architectural, Engineering and Construction (AEC) industry. Technical Report, AEC (UK) Inititative, 2015. URL: https://aecuk.files.wordpress.com/2015/06/aecukbimtechnologyprotocol-v2-1-1-201506022.pdf.

[BIMGW12] BIM Guide Workgroup. Singapore BIM Guide Version 2. Technical Report, Building and Construction Authority, 2012.

[BDA12] BLIS Consortium and Digital Alchemy. IFC Solutions Factory - The Model View Definition site. 2012. URL: http://www.blis-project.org/IAI-MVD/.

[GFMAE14] German Federal Ministry for economic Affairs and Energy. Rooftop Project Manual - Project Drawings. In *Solar Decathlon Europe 2014*. Team Rooftop, Universität der Künste Berlin and Technische Universität Berlin, 2014.

[MC14] MODELISAR Consortium. *Functional Mock-up Interface for Model-Exchange and Co-Simulation version 2.0*. 2014. URL: https://www.fmi-standard.org/downloads.

[SOCP11] Jan Široký, Frauke Oldewurtel, Jiří Cigler, and Samuel Prívara. Experimental analysis of model predictive control for an energy efficient building heating system. *Applied Energy*, 88(9):3079–3087, 2011. doi:10.1016/j.apenergy.2011.03.009.

[AGT09] Johan Åkesson, Magnus Gäfvert, and Hubertus Tummescheit. JModelica – an open source platform for optimization of modelica models. In *Proceedings of MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*. Vienna, Austria, February 2009. TU Wien.

[AAG+10] Johan Åkesson, Karl-Erik Årzén, Magnus Gäfvert, Tove Bergdahl, and Hubertus Tummescheit. Modeling and optimization with Optimica and JModelica.org - languages and tools for solving large-scale dynamic optimization problem. *Comput-*

ers and Chemical Engineering, 34(11):1737–1749, January 2010. doi:10.1016/j.compchemeng.2009.11.011.

Index

www.iea-ebc.org

EBC

Energy in Buildings and
Communities Programme

EBC is a programme of the International Energy Agency (IEA)

iea
Energy Technology
Network