BaaS

Building as a Service

FP7-ICT-2011-6: ICT Systems for Energy Efficiency
Small or Medium-scale Focused Research Project
Grant Agreement No. 288409

Deliverable D4.3:

Model simplification for control and monitoring

Deliverable Version:	D4.3, v.1.0
Document Identifier:	BaaS_D43.PDF
Preparation Date:	March 6, 2015
Document Status:	Final
Author(s):	Karel Macek; HON Giorgios Kontes, Giorgios Giannakis, Dimitrios Rovas; TUC Juan Rodríguez Santiago; Fraunhofer Susana Martín, Cesar Valmaseda; CARTIF
Dissemination Level:	PU - Public

Project funded by the European Community in the 7th Framework Programme

ICT for Sustainable Growth

Deliverable Summary Sheet

Deliverable Details	
Type of Document:	Deliverable
Document Reference #:	D4.3
Title:	Model simplification for control and monitoring
Version Number:	1.0
Preparation Date:	March 6, 2015
Delivery Date:	March 6, 2015
Author(s):	Karel Macek; HON Dimitrios Rovas, Giorgios Kontes, Giorgios Giannakis; TUC Juan Rodríguez Santiago; Fraunhofer Susana Martín, Cesar Valmaseda; CARTIF
Document Identifier:	BaaS_D43.PDF
Document Status:	Final
Dissemination Level:	PU - Public

Project Details	
Project Acronym:	BaaS
Project Title:	Building as a Service
Project Number:	288409
Call Identifier:	FP7-ICT-2011-6
Call Theme:	ICT Systems for Energy Efficiency
Project Coordinator:	Fundacion Cartif (CARTIF)
Participating Partners:	Fundation Cartif (CARTIF, ES); NEC Europe Ltd. (NEC, UK); Honeywell, SPOL, S.R.O (HON, CZ); Fraunhofer-Gesellschaft zur Förderung der Angewandten Forschung e.V. (Fraunhofer, DE); Technical University of Crete (TUC, GR); University College Cork, National University of Ireland, Cork (UCC-IRU, IE) Dalkia Energia y Servicios (DALKIA, ES)
Instrument:	STREP
Contract Start Date:	May 1, 2012
Duration:	48 Months

Deliverable D4.3: Short Description

This deliverable addresses the simplification of building models in order to provide faster simulations with sufficient accuracy. The simplified models can be created either in (i) deductive way where the models are simplified in terms of available expert knowledge, or (ii) inductive way where the models are created by learning from the simulated data. Both approaches have been compared to the full-scale simulation models and their accuracy as well as speed has been verified. It is also discussed how the considered models are implemented and integrated within the BaaS platform.

Keywords: Building energy performance simulation modeling; model simplification; geometry reduction; zoning reduction; model selection; cross-validation; Gaussian mixtures; accuracy-speed trade-off.

Deliverable D4.3: Revision History

Version:	Date:	Status:	Comments
0.1	13 Oct. 2014	Draft	HON: draft document design and structure
0.2	7 Feb. 2015	Draft	TUC: changes in formatting, integration of deductive approaches
0.3	21 Feb. 2015	Draft	HON: Changes in structure, simulations by inductive approaches
0.4	1 Mar. 2015	Draft	TUC: Further editing of the deductive part
0.5	3 Mar. 2015	Draft	HON: Extending the discussion on applicability and some implementation notes
0.6	4 Mar. 2015	Draft	Fraunhofer: Adding information on Sierra Elvira and corresponding model simplification. Discussing the link to the rest of WP4.
0.7	4 Mar. 2015	Draft	HON: Adding implementation and integration notes.
0.8	5 Mar. 2015	Draft	CARTIF: Review of the document.
0.9	5 Mar. 2015	Draft	HON: Notation unification.
0.9b	5 Mar. 2015	Draft	TUC: Minor changes in notation.
1.0	6 Mar. 2015	Final	HON: Compiling with cover pages. List of acronyms.

Table Of Contents

List of Figures

List of Tables

Abbreviations and Acronyms

ASCII	American Standard Code for Information Interchange
ASHRAE	American Society of Heating, Refrigerating, and Air-Conditioning Engineers
BAAS	Buiding as a Service
BCVTB	Building Controls Virtual Test Bed
BEMS	Building Energy Management System
BEPS	Building Energy Performance Simulation
ELM	Ensamble Linear Models
FDD	Fault Detection and Diagnosis
FIBP	Fraunhofer-Instituts für Bauphysik
GM	Gaussian Mixture
GP	Evolutionary Algorithm
HVAC	Heating, Ventilation, and Air Conditioning
ICT	Information and Communication Technology
IDD	Input Data Dictionary
IDF	Input Data File
ISO	International Organization for Standardization
LR	Local Regression
NN	Neural Network
PPD	Predicted Percentage of Dissatisfied people
PR	Polynomial Regression
RDD	Report Data Dictionary
SDF	Synchronous Dataflow
SE	Sierra Elvira
STREP	Specific Targeted Research Project
TRNSYS	Transient System Simulation Tool
XML	Extensible Markup Language

Executive Summary

The present document deals with the construction of approximate models of the buildings, including their HVAC systems. These models are intended to be used in Task 5.2 for the diagnostic purposes and in Task 5.3 for the optimization. The key criteria are the low prediction error and low computational time.

These models can be created in two ways (i) inductive - by learning the models from the available data or (ii) deductive - by simplifying the BEPS models (EnergyPlus, TRNSYS). The deductive approach is implemented via geometrical simplification and it takes advantage from available context information. The inductive approach is implemented by means of Gaussian mixture regression. Its major advantage is that there is no need of context information provided. Moreover, it is able to adapt in time with the new data records and it has the potential to run also in embedded devices, similarly to speech recognition.

The document provides also some basic justification of the proposed methods by comparison with complex models. The implementation and integration within the BaaS platform is described as well. The ultimate assessment of the simplified models has to be - however - considered in the context of T5.2 and T5.3 that apply them in end-user-oriented tasks.

1 Introduction

As discussed in deliverable 5.1, the BaaS platform addresses especially (i) monitoring HVAC systems, especially detecting present anomalies and inefficiencies and (ii) efficient control of the HVAC systems. Both functions require support of models of the building behavior.

These models can be constructed by means of BEPS systems (EnergyPlus, TRNSYS). However, these models are constructed by experts. It means numerous details as well as significant high-qualified labor. Moreover, the models are typically very slow in simulation of the building behavior: to simulate 3 days lasts between 70 and 100 seconds. This is a serious drawback especially for the optimization where these simulations are run iteratively in order to determine optimal setting of the control parameters.

This fact has motivated the present deliverable that focuses on the transformation of the BEPS models to models with lower complexity and sufficient precision. As a minor topic, we will discuss also the possibilities how to use the proposed algorithms without the BEPS modeling.

This document is organized as follows: in Section 2, we present the requirements on the simplified models as well as an overview of full models considered. Afterwards, the approach of model simplification are offered - the deductive one in Section 3 and the inductive one in Section 4. Both approaches are illustrated on simulated data from some of the pilot buildings. Consequently, it is described how the simplified models are integrated in the BaaS platform in Section 5. Finally, Section 6 concludes the document.

This deliverable is led by HON with major contribution from TUC, Fraunhofer, and CARTIF as scientific partners.

Table 1: Summary of Partner Contribution

Partner	Deliverable Focus
HON	Deliverable lead; Requirement specification; Implementation of the inductive approach
TUC	Overview of the full models; Implementation of the deductive approach
Fraunhofer	Linking to the rest of WP4; inputs related to the FIBP building and SE model.
CARTIF	Document review. Inputs related to the CARTIF building: generation of historical datasets.

1.1 Relation to other activities in the project

This deliverable provides important subroutines for the assessment as well as control tasks. The following table is a brief summary of such relationships.

Table 2: Relationship with other Work Packages and tasks

D4.3 Section	Relationship
Section 2	Discusses the drawbacks of full models as described in D4.2.
Section 3	Provides accelerated models for model-based control in D5.3.
Section 4	Provides models for context-free FDD in D5.2 and for control, especially the data-driven one, in D5.3.
Section 5	Provides information about integration of the simplified models in the Application layer kernel that is subject of WP5.

1.1.1 Role in WP4

The scope of BaaS WP4 is to deliver a number of simulation models able to be used by different services and in different situations.

BEPS models need nowadays a considerable amount of time to create accurate representations of the simulated objects. Steps forward are being done to speed up the various steps needed to build the model from a geometrical or structural point of view, as well as to integrate the different facilities installed and couple simulation outputs with potential sensors that allow behavioural comparisons. This kind of work, seems to be necessary in control cases where the complexity of building systems and the utilization of forecasted weather conditions, need of highly accurate methods that fully exploit the available inputs to reach nearly optimal control conditions in terms of energy demand and building internal comfort. On the other hand, in some cases the use of this simulation engines is not justified, either because the simulation accuracy is not the scope sought or because the simulations would continuously be run in time gaps shorter than the running time needed for each simulation trial. In this case it is fully justified the use of less accurate methods that provide a quantitative value useful to for example, run fault detection tests, evaluate major parameter deviations in determinate time periods, or maybe flag building/facility states needed for further building/facility control and evaluation.

All this examples represent a brunch of cases where simplification methods enrich BaaS system without making use of high computational loads needed by specialized software, permitting them to be run in simpler hardware architectures.

2 From Full-Scale to Simplified Models

In this section,we motivate the need of simplified models in more detail. Subsection 2.1 introduces the problem at a general level ans summarizes used nomenclature. Subsection 2.2, describes the nature of existing full-scale models and is concluded by discussion on their limitations. These limitations are motivate the requirements on the simplified models in Section 2.3. Finally, some hierarchies form simple to more complex models are discussed in Subsection 2.4.

2.1 Basic Notation

We understand a model to be a mapping that describes the evolution of the quantities in the considered system. We assume that this evolution runs in discrete time steps $t = 1, 2, \ldots$ Some of the variables denoted as internal states x_t. They evolve with respect to their previous values, actions a_t and external factors d_t where all x_t, a_t, d_t are real vectors of given dimensions. This dependency can be expressed either as a deterministic mapping M where

$$x_{t+1} = M(x_t, a_t, d_t) \tag{2.1}$$

or as a conditional probability density function f with $f(x_{t+1}|x_t, a_t, d_t)$. We will prefer the latter notation since it describes the uncertainty of the model explicitly. The external factors evolve independently of the internal states x_t and actions a_t. We assume that they can be predicted for n_{hor} steps ahead. For the forecasting of the internal states $x_{t+1}, x_{t+2}, \ldots x_{t+n_{\text{hor}}}$, we will assume that the actions $a_t, a_{t+1}, \ldots a_{t+n_{\text{hor}}-1}$ are known – since they are controlled inputs to the system – and the external factors $d_t, d_{t+1}, \ldots d_{t+n_{\text{hor}}-1}$ are either known or predicted externally (e.g. weather forecast).

Table 3: General Notation

Symbol	Explanation
k	discrete time
x_t	$\in \mathbb{R}^{n_x}$ internal state at time t
$x_{t,i}$	ith component of x_t
a_t	actions (control inputs) at time t
d_t	external disturbances at time t
$f(\cdot\|\cdot)$	conditional probability density function
$M(\cdot)$	a general mapping
n_{hor}	prediction horizon
n_{data}	number of available data records
n_x	number of internal states

The most important components of the state x are temperatures T and the heat fluxes q.

Table 4: Specific Notation - Deductive Approach

Symbol	Explanation
T	temperature
q	heat flux
$\overline{\Delta T}$	mean temperature difference
s	sample standard deviation
∇T	temperature gradient
$\mathbf{n}$	normal vector to a surface
δ	calculated distance
G	data matrix of simulated results
C	companion matrix
T	Vandermonde matrix
V	Koopman modes matrix
λ	eigenvalue
$\varepsilon_1, \varepsilon_2$	Koopman modes tolerance
$rmse$	root mean square error

Table 5: Specific Notation - Inductive Approach

Symbol	Explanation
r	$\in \mathbb{N}^{n_r}$ model complexity configuration vector
D	data
$D_{[i]}$	ith cross validation block
n_{blc}	number of blocks in cross validation
w_i	relative weight of ith state component
σ_i	deviation normalization constant of ith state component
n_{ord}	autoregressive order
n_{gau}	number of Gaussian components

2.2 Full-Scale Models and Their Limitations

According to building thermal simulation zonal-models, a full-scale building consists of one or more thermal zones which are coupled with each other and with the environment. A zone consists of an air volume of a uniform temperature and all surfaces bounding or inside that air volume. The basis for the zone air temperature estimation is the formulation of energy and moisture balances for the zone air and the solution of the resulting ordinary differential equations.

Using a Predictor/Corrector approach, the simulation engine [1] calculates the actual zone mean air temperature. The fundamental idea is to use an iterative method: first, predict the air system load needed to reach the desired temperature (predictor); second, simulate the air system to determine its actual capability of achieving this temperature (corrector); and, finally, reconfigure the zone air heat balance to calculate the achieved zone mean air temperature. An identical process is performed to calculate the actual zone mean air humidity.

A consequential aspect of the overall solution scheme, is the definition of the heat balance equation at each face of a building construction component (wall, roof, floor, etc.). The heat balance on interior faces, which are not exposed to outdoor environmental conditions, is expressed for all time steps $k = 1, \ldots$ as follows:

$$q_{\mathrm{lwi},k} + q_{\mathrm{swi},k} + q_{\mathrm{soli},k} + q_{\mathrm{ki},k} + q_{\mathrm{ci},k} = 0; \tag{2.2}$$

where, $q_{\mathrm{lwi},k}$ is the sum of the longwave radiant exchange transfer between zone surfaces and the longwave radiation transfer from equipment in zone; $q_{\mathrm{swi},k}$, the shortwave radiation transfer to surface from lights; $q_{\mathrm{soli},k}$, the transmitted solar radiation flux absorbed at surface; $q_{\mathrm{ki},k}$, the conduction transfer through the wall; and, $q_{\mathrm{ci},k}$, the convective heat transfer to the zone air. External surfaces boundary conditions (external boundary conditions) are derived by Equation 2.3 and have the major impact to unsteady heat transfer through building elements, since they are affected by variable weather data.

Similarly, the heat balance on exterior faces, exposed to environmental conditions, is expressed as follows:

$$q_{\mathrm{lwo},k} + q_{\mathrm{solo},k} + q_{\mathrm{ko},k} + q_{\mathrm{co},k} = 0; \tag{2.3}$$

where, $q_{\mathrm{lwo},k}$ is the longwave radiant exchange with the air and surroundings; $q_{\mathrm{solo},k}$, is the absorbed solar shortwave radiation flux; $q_{\mathrm{ko},k}$ is the conduction transfer into the wall; and, $q_{\mathrm{co},k}$, the convective transfer to the outside air[1]. Each simulation engine incorporates the appropriate mathematical models to quantify each term of Equations 2.2 and 2.3. These models, use available (geometric and weather) data, to derive proper boundary conditions. Every simulation model uses Equations 2.2 and 2.3 to provide realistic boundary conditions at each simulation time step. Note that these equations are considered for all components of the building. The following paragraphs provide more information about the complexity of models in particular pilot buildings.

2.2.1 FIBP Building

The FIBP building is the office building of the Centre for Sustainable Building in Kassel, situated at the University of Kassel, Germany. The building consists of 26 rooms, spanned in three floors and each physical room has at least one corresponding thermal zone in the full scale model.

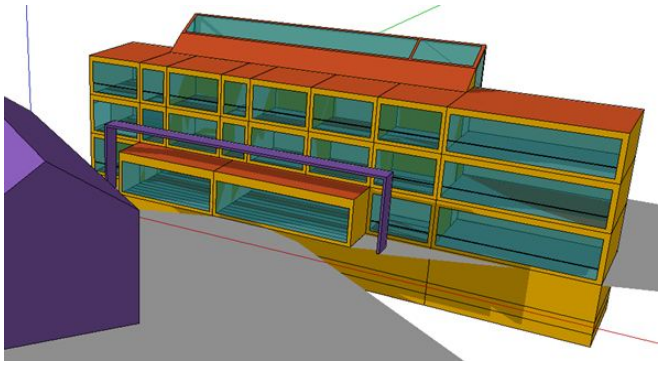

Figure 1: FIBP building – full-scale simulation model

The full-scale model of the FIBP building has been set up in Trnsys 17 using Google SketchUp with the purpose to hold the option of employing the detailed radiation mode that is provided with Trnsys 17. To meet the requirements for the detailed radiation mode, all zones have to be convex, i.e. every wall of a zone has to be in the line of sight with all other surfaces of the zone. This can only be accomplished by slightly simplifying and modifying the original geometry of the model.

[1]Note that there is no counterpart to $q_{\mathrm{swo},k}$ since there is no shortwave radiation from lights considered for the exterior faces.

2.2.2 TUC Building

TUC building is of triangular shape and comprises 10 office rooms, an open meeting space, two corridors (one in each floor), the main entrance, an equipment room, a toilet and a basement that is used as a storage area. In both floors (ground floor and first floor) there is a central corridor running the length of the building with offices on either side. In the middle of the corridor there is an open meeting space of semicircular form. The plan of the first floor is similar to the ground floor, with the only difference the presence in the first floor of a semicircular atrium to connect the ground floor meeting space with a large glazing on the roof.

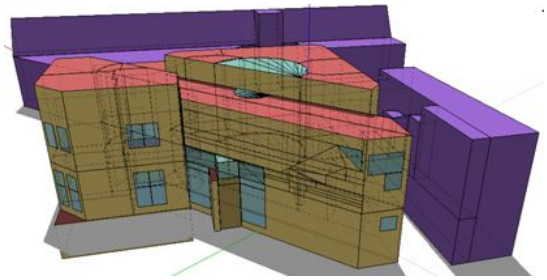

Figure 2: TUC building – full-scale simulation model

For energy performance tasks, a full-scale thermal simulation model has been developed in Energy-Plus. The representation of the building geometry was created according to the floor plans using the Google SketchUp plugin, Openstudio – see Figure 2.

The presence of the solar atrium, along with the radiation model used, gives an uneven heat distribution between the simulation and reality. For this reason, each corridor is split into more zones, so that the solar gain and radiant exchange effects can be correctly captured, increasing the number of zones by eight.

2.2.3 CARTIF Building

CARTIF Building has a rectangular-plant with 3 floors (basement, ground floor and first floor). In the basement there are service spaces, like electrical controls and storage rooms. Boiler room and thermal-solar facilities are situated in two rooms close to the garage and exterior. All spaces are nor heated nor cooled. In ground floor there are two main zones, offices and conference rooms are situated on the north side (administrative area). Research laboratories and the warehouse are located in the southern side. On first floor there are research laboratories and head's offices. Energy Division laboratory is placed on South-East side.

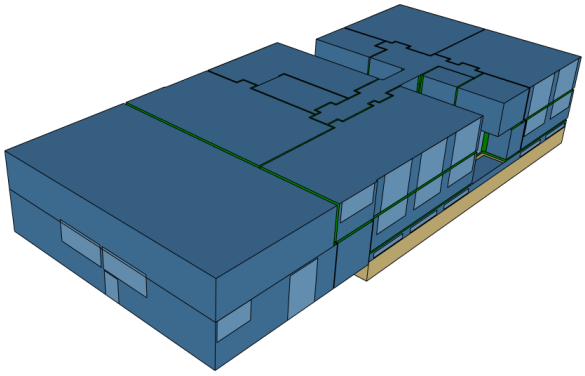

Figure 3: CARTIF building – full-scale simulation model

For energy performance tasks, a full-scale thermal simulation model has been developed in TRNSYS. The representation of the building geometry was created according to the floor plans using the Google SketchUp plugin, Openstudio – see Figure 3.

For CARTIF Building, the categorization of rooms to thermal zones was performed according to its HVAC zones definition. An HVAC zone consists of one or more rooms and a thermostat assigned to that zone. Such a categorization led to a full-scale thermal simulation model comprising 25 thermal zones.

For a detailed description of FIBP and TUC Buildings' full-scale thermal simulation models, refer to [2, 3], while the CARTIF Building's full-scale thermal simulation model is described thoroughly in [3].

2.2.4 Sierra Elvira Building

The Sierra Elvira School in Granada (Spain) is a complex consisting of three separate buildings. Only the building occupied by the primary school is to be studied in BaaS. It has a rectangular plan with four floors, the three highest ones are occupied by teaching classes while the ground floor homes the storage rooms, boiler room, bathrooms and a library.

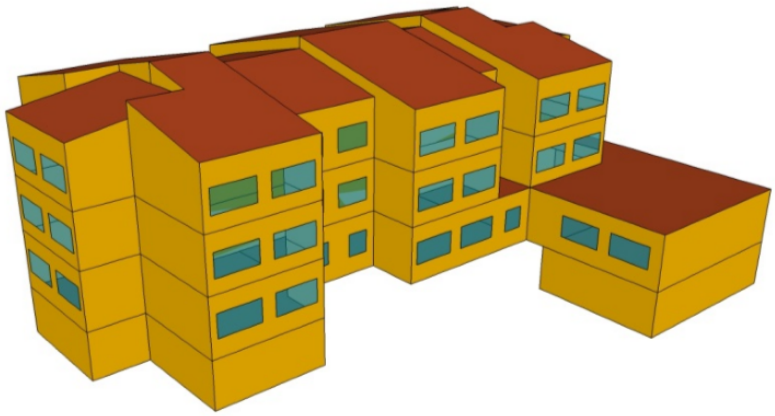

Figure 4: Skechup model of school (IDF file)

The full-scale model of Sierra Elvira represents every single space of the building, and it was

done in combination of Google Sketch Up and Trnsys17 avoiding the use of complicated radiation models that highly delay the simulation times without adding the simulation much accuracy when compared to simplified models or existing real measurements[2].

2.2.5 Limitations of Full-Scale Models

When accuracy and computational effort constraints are present, simulation speed-up techniques like the ones discussed in this section, can yield an good trade-off between accuracy and computational complexity. Despite the fact that full-scale building simulation models have been developed for the demosites, they are not suitable for computationally demanding tasks, such as model assisted control design, for which a purpose the present simulation models will be created. Even though for small buildings, full scale models have been used successfully for computationally demanding tasks [4, 5], field studies [6] have shown that, for large buildings a very detailed model is impractical to simulate, since it is too complex and takes too long to run. Simulation is predominantly slowed down by the increasing number of zones, windows and surfaces [6], and as such simulation speedup techniques should focus on geometry reduction, zoning reduction and/or order reduction approaches.

Based on the above summary of full-scale models, we can remark the following typical drawbacks of the BEPS modeling:

- Labor related issue: BEPS modeling uses full description of the considered system based on first principles. This description has to be entered by experts which can be time-consuming and expensive. This is limiting especially for the monitoring tasks where the added value is not as high as in case of control and where only approximate models can be used.
- Efficient computing related issue: BEPS modeling works with relatively short time steps in order to approximate dynamics described by means of differential equations which is computationally complex and time-demanding. The situation is even more complicated by high dimensionality of the simulation. This is limiting especially for the optimization tasks where the evolution of the system has to be simulated many times.

2.3 Requirements on Simplified Models

The simplified models shall be assessed in terms of the following parameters:

- Sampling - length of one time step, e.g. 1 s, 15 minutes, 1 day.
- Horizon - considered number of prediction steps n_{hor}, e.g. $n_{\text{hor}} = 96$ for one day ahead simulation with 15 minutes sampling.

[2] In Deliverable D4.2 it is described how the model was simplified to obtain acceptable results with shorter simulation times. In a first approximation, the studied building was replicated in sketch up room by room, assigning to each space the real occupancy profiles, and assuming due to the lack of installed sensors at the beginning of the project, that if this simulation fits the measurements obtained from the boiler demand and temperatures existing in some “test rooms”, all the building model was identified. In a second step, due to the orientation of the building and the heating distribution topology, the zoning was reduced, merging all the rooms of a single floor with same orientation and no sensor installed in them, reducing the number of zones an 80% (This reduction left only 4 zones by floor plus boiler room and library. Test cells are still single zones in this model to control the parity with real measurements). In a third step the building was divided in only three zones related to the hot water distribution, building oriented to the south, north and boiler room plus library. The results obtained in all the simulations were compared against the accurate one, obtaining that simulation time decreases up to 50% with inaccuracies of around a 1.5% for the total building energy demand. Thus, the simplification of models for Sierra Elvira School has been addressed in alternative ways already in D4.2 and therefore it is not discussed in this deliverable as much as the other pilot sites.

- Computational time of learning - time needed to create the model.
- Memory requirements - bytes needed to store the model.
- Adaptation - whether the model shall be determined once for ever or it is needed to update it by new data.
- Gradient - whether the model has well defined and easily calculable gradient.
- Uncertainty - whether the model is able to describe the uncertainty in the system and whether this description is explicit or implicit only.

And the following criteria:

- Prediction error (mean square error, mean absolute error, mean absolute scale error),
- Simulation time - use, i.e. how fast the states for the next n_{hor} steps shall be returned.

Table 6: Requirements on simplified models

Task	Monitoring and anomaly detection (D5.2)	Control design (D5.3)
Sampling	<15 minutes	$<$ 15 minutes
Horizon n_{hor}	Single step	$>$ 2 days
Computational time (learning)	$<$ 10 min	$<$ 10 min
Memory requirements	$<$ 10 kB	$<$ 10 kB
Adaptation	Yes, if possible	Yes, if possible
Gradient	Not necessarily	Yes, if possible
Uncertainty	Not necessarily	Not necessarily
Simulation time	Possibly minimal	Possibly minimal
Prediction error	Possibly minimal	Possibly minimal

The reduced modeling for control design seems to be more complex than for monitoring, especially with respect to the long prediction horizon.

2.4 Hierarchy of Modeling and Simulation Approaches

In Table 6, we have listed several requirements on the reduced models. It is obvious that some trade-offs between several criteria might arise. It can be especially considered that the increased precision will increase also the complexity of the models. Since the practical use of the models might put emphasis in this trade-off on different levels of precision and complexity, it is necessary to consider not only one reduced model, but a hierarchy of models ranging from simple and inaccurate to complex and precise. Some of possible hierarchies are:

- Polynomials: constant, linear, quadratic, cubic, ...
- Gaussian mixtures: number of Gaussian components.
- Neural networks: number of neurons.
- Regression trees: tree of depth one, tree of depth two, ...
- Ensembles of local models: number of local models.
- Autoregressive order: one delayed value, two delayed values, ...

- Spacial: zone, zone group, column of zone groups/floor, building...

Thus, we can see that these hierarchies relate to the adopted modeling approach. In Section 3, the spacial hierarchy is either used where a group of zones are either considered as representation of the whole building, or it is merged into a virtual new zone. Section 4 uses the number of Gaussian components as well as the autoregressive order.

3 Deductive Approach

This chapter provides methods how models can be reduced by special treating of the contextual information. The first one is the geometry reduction approach that is based on decomposition of the building into some repeating patterns. The second approach based on zoning reduction where the ways of aggregation of zones are explored.

3.1 Geometry Reduction Approaches

Since high complexity and prohibitive simulation runtime are predominantly due to the full-scale, detailed, geometry representation of the demonstration buildings, geometry reduction approaches are required. Leveraging the building geometry's periodicity, a commonly used approach is to define a representative building block, cut-out from the full-scale geometry representation and able to reproduce the whole building. Its walls where actually other zones adjoin are defined as "boundary walls".

A significant difficulty of this approach is to determine proper boundary conditions at each boundary wall, as they are naturally derived by the heat balance equation on a boundary wall. The heat balance equation on exterior faces, described by Equation 2.3, remains valid. However, it is the heat balance on interior faces, corresponding to a boundary wall, that deviates from the one formulated by Equation 2.2.

For these faces, two types of boundary conditions could be defined:

- *Adiabatic boundary condition* - is a special case of *Neumann* conditions, where the surface is assumed to be perfectly insulated. Thus, any heat transfer through boundary walls is excluded for all $k = 1, \ldots$:

$$-\kappa(\nabla T_k \cdot \mathbf{n}) = 0; \tag{3.1}$$

  where κ is a scalar expressing the wall's thermal conductivity, ∇T_k the temperature gradient of the given wall at the time k, and $\mathbf{n}$ is a transposed normal vector of the wall. For instance, if we consider a floor, the transposed normal vector would be $\mathbf{n} = (0,0,-1)$ and the Equation 3.1 states that there is no heat flux to the ground.

- *Dirichlet boundary condition* - describes a situation for which the surface is maintained at a fixed temperature profile $T_{\text{sch},k}$ at each simulation time step k. The boundary condition is formulated as:

$$T_{\text{so},k} = T_{\text{sch},k} \tag{3.2}$$

  for all $k = 1, \ldots$.

The efficiency of two Geometry Reduction Approaches is investigated for two buildings: the FIBP Building, simulated using TRNSYS; and the TUC Building, simulated using EnergyPlus. The proposed Geometry Reduction Approaches are evaluated on these buildings, with respect to the accuracy of the simulation results and their impact (positive or negative) to the computational effort. Towards investigating the accuracy, both the detailed and reduced (speedup) model of the two buildings are simulated for a certain time interval and the temperature deviation is selected as the accuracy measure. For the simulation speedup on the other hand, scenarios with different run periods are investigated and the respective speedup times are presented. In all cases, the speedup models exhibit significantly reduced execution times.

3.1.1 FIBP Building – Leveraging Geometry's Periodicity

As detailed simulation of indoor temperatures of the full-scale model increases computational complexity, a "tower" of three offices upon each other is cut-out, since the whole building can be viewed as a parallel expansion of the simple "tower" sub-model in one dimension. Here, the outer surfaces are actually adjoining rooms' surfaces, which are defined as "boundary walls", and the adjacent constructions that have a shading effect on the tower, but are not included in the thermal simulation, are modeled as shading groups. The tower is shown in Figure 5.

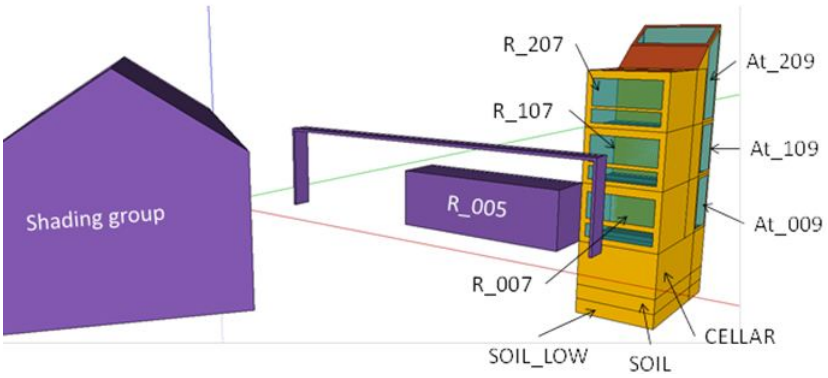

Figure 5: FIBP Building – tower simulation model

The tower concept is preferred against a horizontal cut-out to keep the essential effect of vertical temperature stratification.

Initially, a *Dirichlet* boundary condition with variable temperature and fixed at each simulation timestep is supplied as boundary condition for each wall. In this case, the boundary condition at the outer plane of contact surfaces is defined by Equation 3.2, where the temperature schedule for the outer plane of a boundary wall (T_{sch}) is the actual room temperature of adjoining rooms available from measurements. This approach has limited perspective from a practical point of view, since forecast simulations would require predictions of the temperatures in the adjoining rooms, which there is no chance to be available. Since room temperature trajectories are to be simulated by the model this would be a vicious circle.

Using an *Adiabatic* boundary condition, the aforementioned hindrance can be overcome; however, there are cases, such as the tower operating differently from the rest building, that the assumption of a zero heat transfer could lead to an over- or underestimation of the zone air temperatures.

3.1.2 TUC Building – Co-Simulation Approach

For the TUC building on the other hand, its triangular geometry does not define a building block, like the concept of "towers" above, being able to reproduce the whole building, so the periodicity/adiabatic technique can no longer be applied. In this case, an alternative option would be to divide the building into three sub-buildings, each of which can be simulated separately, establishing the connection by using compatible *Dirichlet* boundary conditions. Such splitting of the building, along with the parallel simulation in existing multi-core computer architectures, can yield a sizable reduction in simulation run-time. The use of co-simulation, described next, can yield an effective method, for the transfer of boundary conditions between the sub-buildings, establishing their thermal interaction. For the particular case at hand, the TUC building, since the two main corridors (that include the attrium) are connected through horizontal holes, the need for correctly

computing radiant gains in the attrium, forces including both corridors in one of the sub-buildings. With this prerequisite, the whole building is divided to three sub-buildings, shown in Figure 6.

Regarding the boundary conditions, a first approach would be to use similar *Dirichlet* boundary conditions to that applied to the "tower" of FIBP building (adjacent rooms temperature). A more complex approach would be to force as boundary condition for each sub-building contact surface the temperature of the corresponding surface, resulting from the adjacent sub-building's simulation. In other words, suppose that wall A is a common surface of sub-buildings 1 (A_1) and 2 (A_2). Then, at the end of sub-building:1 simulation, the temperature profile of surface A_1 is applied as boundary condition to surface A_2 so as to simulation of sub-building:2 run. Thus, a set of *Dirichlet* boundary conditions needs to be defined.

Aiming at control the temperature of a surface that is adjacent to an area that is not included to the simulation model (part of a boundary wall), "OtherSideCoefficients" feature in EnergyPlus [7] is used.

Other side coefficients affect the other side of a surface for all time steps $k = 1, \ldots$ as follows:

$$T_{\mathrm{so},k} = c_1 T_{\mathrm{z},k} + c_2 T_{\mathrm{sch},k} + c_3 T_{\mathrm{a},k} + c_4 T_{\mathrm{g},k} + c_5 T_{\mathrm{a},k} v_{\mathrm{w},k} \tag{3.3}$$

where $T_{\mathrm{so},k}$ refers to the surface temperature, $T_{\mathrm{z},k}$ to the temperature of the zone being simulated, $T_{\mathrm{sch},k}$ to the dry-bulb temperature of the outdoor air, $T_{\mathrm{a},k}$ to the temperature schedule for the outer plane of the surface, , $T_{\mathrm{g},k}$ to the temperature of the ground and $v_{\mathrm{w},k}$ to the outdoor wind speed, respectively. Here, a temperature profile ($T_{\mathrm{so},k}$) is provided as boundary condition.

In the present work, the boundary condition at the outer plane of each boundary wall is defined by equation 3.3, with the following coefficients' combination: $c_1 = 0$; $c_2 = 1$; $c_3 = 0$; $c_4 = 0$; $c_5 = 0$; where the temperature schedule for the outer plane of a sub-building boundary surface (T_{sch}) is the temperature profile of the inner plane of the corresponding surface of the adjacent sub-building. This situation corresponds to Equation 3.2 and is illustrated in Figure 7.

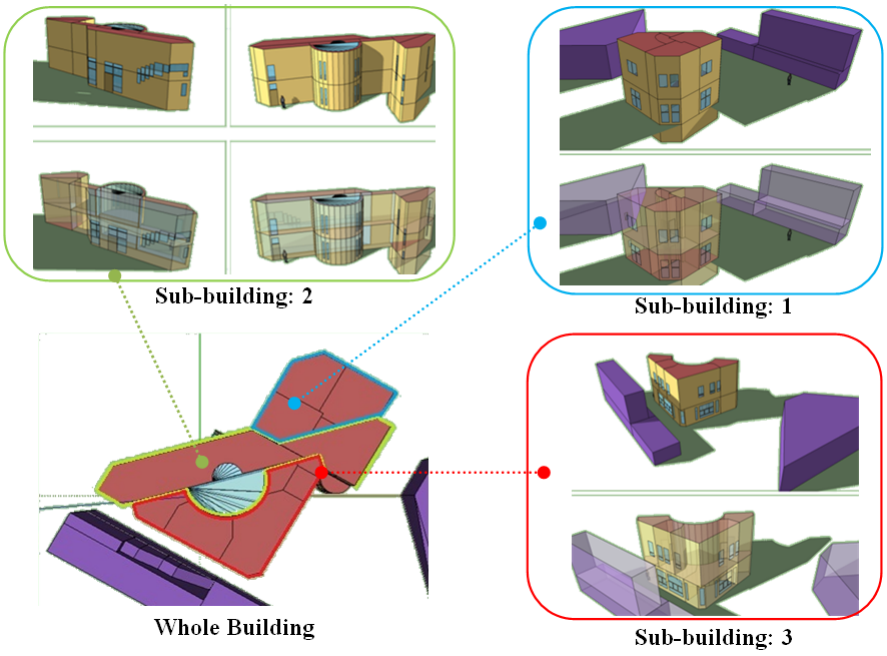

Figure 6: Dividing the whole building to 3 sub-buildings

In order to exchange these boundary conditions in a proper way, a dynamic communication between the three sub-buildings is required and can be achieved by using EnergyPlus with External Interfaces and especially with the Building Controls Virtual Test Bed (BCVTB) [8].

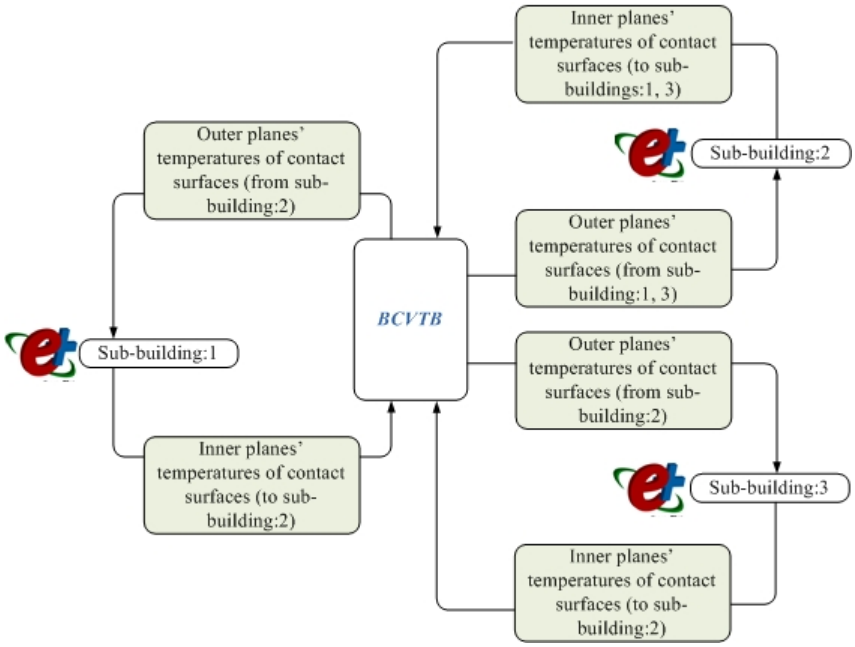

Figure 7: Data exchange between the sub-building through BCVTB

Through such a dynamic communication, a co-simulation between three different thermal models, developed in EnergyPlus, is achieved and the boundary conditions on "boundary walls" (contact surfaces) are defined as follows:

- The state variables x_1 contain the temperatures of inner planes of all that surfaces which belong to sub-building:1 and which are temperatures of outer planes of the corresponding surfaces which belong to sub-building:2.
- The state variables x_2 contain the temperatures of inner planes of all that surfaces which belong to sub-building:2 and which are temperatures of outer planes of the corresponding surfaces which belong to sub-building:1 and sub-building:3.
- The state variables x_3 contain the temperatures of inner planes of all that surfaces which belong to sub-building:3 and which are temperatures of outer planes of the corresponding surfaces which belong to sub-building:2.
- Suppose that the function f_1, f_2 and f_3 that computes the next state variable of the simulator 1 and 2 and 3, respectively. The simulator 1 computes, for each time step k, the sequence: $x_{k+1,1} = f_1(x_{k,1}, x_{k,2}, x_{k,3})$ and similarly, simulators 2 and 3 compute the sequences $x_{k+1,2} = f_2(x_{k,1}, x_{k,2}, x_{k,3})$ and $x_{k+1,3} = f_3(x_{k,1}, x_{k,2}, x_{k,3})$ respectively, with initial conditions on $x_{0,1}$, $x_{0,2}$ and $x_{0,3}$.
- To advance from time k to $k+1$, each simulator uses its own time integration algorithm. At the end of the time step, the simulator 1 sends the new state $x_{k+1,1}$ through the BCVTB and receives the state $x_{k+1,2}$ through the BCVTB. The same procedure is done with the simulators 2 and 3. The BCVTB synchronizes the data in such a way that it does not matter which of the three simulators is called first.

3.1.3 Simulation Results

Towards investigating the accuracy of the simulation results, both the full-scale and reduced (after the implementation of a speedup technique) model are simulated for a predefined time interval (run period), and standard deviation of the difference between the resulting zone air temperatures are calculated according to the following equations.

To start, the temperature difference between the full-scale and the reduced model simulation results is given for the ith temperature and kth time step by:

$$\Delta T_{k,i} = T_{\text{full-scale},k,i} - T_{\text{reduced},k,i}. \tag{3.4}$$

Consecutively, the mean is calculated according to:

$$\overline{\Delta T}_i = \frac{1}{n_{\text{data}}} \sum_{k=1}^{n_{\text{data}}} \Delta T_{k,i} \tag{3.5}$$

while, the sample standard deviation is calculated as follows:

$$s_i = \sqrt{\frac{1}{n_{\text{data}} - 1} \sum_{k=1}^{n_{\text{data}}} \left(\Delta T_{k,i} - \overline{\Delta T}_i\right)^2}. \tag{3.6}$$

The confidence bound is given as $\overline{\Delta T}_i \pm s_i$. A large $\overline{\Delta T}_i$ points in the direction of a general failure, whereas a large s_i indicates a high uncertainty for the reproducibility. The reason for a general failure can be either a wrong model or wrong parameterizations.

For the simulation speedup on the other hand, the simulation runtimes for a number of cases, performed to record the reduction in simulation runtime due to the implementation of a speed-up technique, are presented.

FIBP Building

The required simulation time of a *basic* building simulation in TRNSYS scales up with increasing number of thermal zones according to the polygon shown in Figure 8 — the relevant calculation time in *seconds* is on the left axis. The right axis refers to the simulation time in minutes required to simulate the full-scale and the simplified (tower) FIBP building model. The simulation time of the full-scale model including all thermal zones (25), is approximately 800 minutes for one year. The tower (9 zones) takes approximately 50 minutes for the same simulation interval. Since the models were designed for the purpose of co-simulation the following considerations are relevant.

Co-simulation requires repeated simulations for the prediction horizon in connection with a stochastic optimization algorithm; the number of zones increases the dimension of the parameter space and hence prolongs the convergence time and the number of iterative simulations required by the algorithm. A prediction horizon of 72 hours including a one day settling phase requires approximately 7 minutes simulation time for the full-scale model. Assuming 100 iterations — which is a conservative estimation for a 54 dimensional parameters space — to find an optimum, a controller design would take approximately 11 hours. This allows for only two complete controller design runs a day. Using the tower model requiring only 24 s for the relevant prediction horizon, at least 24 runs per day are possible.

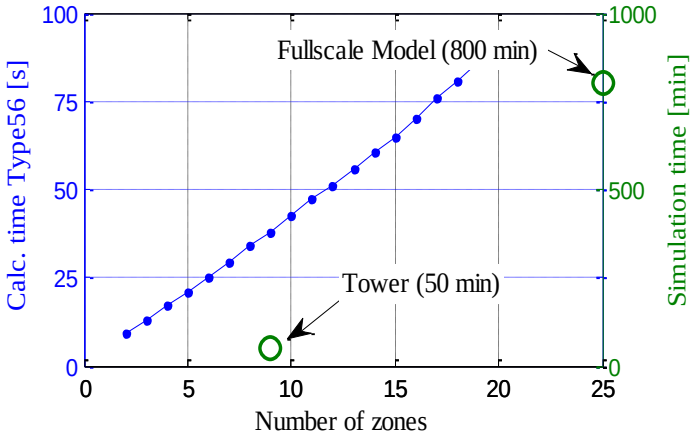

Figure 8: Upscaling of the simulation runtime for increasing number of thermal zones in TRNSYS Type 56 — base interval is one year. Left axis: general building model, right axis: fullscale FIBP & tower building model

In order to compare the impact on the results for this simplification, two simulations for a winter month (January) and two simulations for a summer month (June) are conducted. The setpoint temperatures (blue dashed line) — available from real measurement data — are applied as room set temperatures for the three zones R007, R107 and R207. Figure 10 shows that, for winter months, although a phase shift is clearly visible, temperature patterns are very similar.

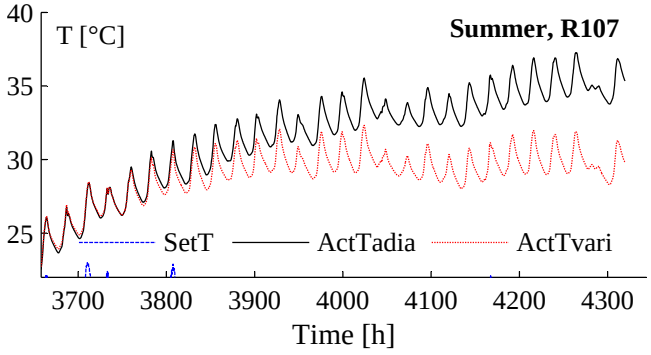

Figure 9: FIBP tower building model – different boundary conditions, for a simulation interval of a summer month

However, in summer we have a clear overestimation of the temperatures which is due to the fact, that neighboring rooms receive some sort of cooling e.g. night ventilation, while for the tower zones no window opening or any other user influence was considered. In addition, the summer interval shows the significance of the thermal stratification in vertical direction; the difference between the bottom and the top zone is approximately 5 K.

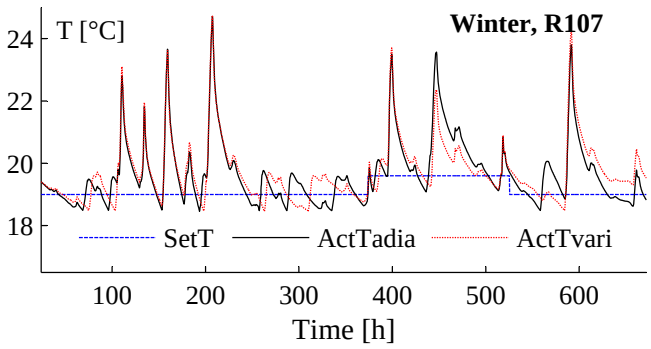

Figure 10: FIBP tower building model – different boundary conditions, for a simulation interval of a winter month

Hence, if rooms are operated similarly as is the case for winter, the two optional boundary condition lead to nearly the same results. But if the neighboring room is operated differently, as is the case for the winter month, the thermal coupling leads to systematic over- or underestimation.

The mean and the standard deviation of the expression $\Delta T_k = \text{ActTvaria}_k - \text{ActTadia}_k$, characterize the similarity of the temperature trajectories for different boundary conditions. Variable boundary conditions (ActTvaria_k) are considered providing the same conditions for the tower as would be prevalent for these zones embedded in the full-scale model. Results of the deviation are presented in Table 7.

Table 7: Mean and standard deviation results for rooms R207, R107 and R007 of the FIBP tower building model

Thermal Zone i	Winter month		Summer month	
	$\overline{\Delta T}_i$	s_i	$\overline{\Delta T}_i$	s_i
R207	-0.020	0.145	-2.14	1.91
R107	0.055	0.476	-2.51	1.92
R007	-0.005	0.527	-1.96	1.40

TUC Building

Moving to the TUC Building simulation experiments setup, a whole year simulation time interval is selected to investigate the deviation, proving that a parallel connection between the sub-buildings is able to assimilate zone air temperature trends of a full-scale model, regardless of the neighboring zones thermal conditions (temperature, humidity, etc.). Thus, assuming all zones to be free floating, the conditions during this period are as follows: shading devices are completely open; internal gains are equal to zero, since lights and electric equipment are switched off; there is no HVAC system available to control the temperature of the zone; and natural ventilation is considered at night, during summer months only.

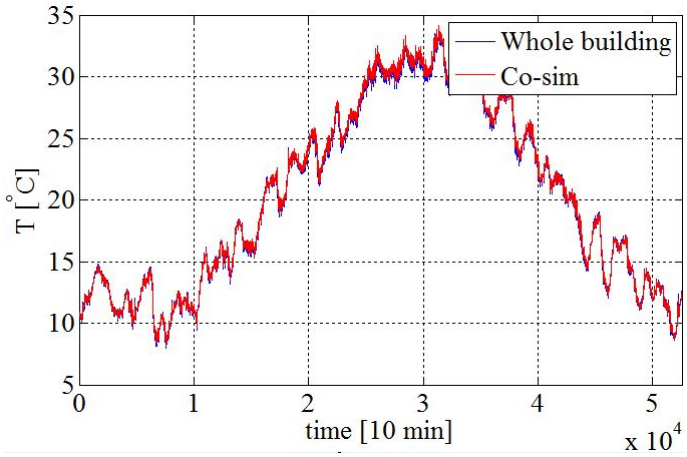

Figure 11: Simulated Air Temperature values in an office room (office 11) — whole building model and sub-building:1 model (Co-sim)

Observing the temperature results in Figure 11, it is obvious the similarity of trends (almost identical) for a randomly selected office room (office 11 – part of sub-building:1). Although the temperature differences are near to zero during the winter months, slight differences, which are within acceptable limits, arise during the summer months, proving that the above types of boundary conditions are efficient enough to describe the dynamic behavior of the full scale building. The mean and the standard deviation for the office rooms are presented in Table 8.

Table 8: Mean and standard deviation results for each office room of TUC building

Thermal Zone i	Whole year	
	$\overline{\Delta T}_i$	s_i
office 1	0.168	0.028
office 2	0.238	0.033
office 3	0.153	0.025
office 4	0.251	0.027
office 5	0.132	0.075
office 8	0.251	0.060
office 9	0.225	0.051
office 10	0.251	0.053
office 11	0.251	0.082
office 13	0.173	0.065

Three scenarios are investigated to prove the effectiveness of the parallel connection, regarding the simulation runtime for different run periods. In Table 9, Full is considered to be the full-scale TUC Building model, Sub:1, Sub:2 and Sub:3 are the simulation models of the sub-buildings presented in Figure 6, runing independently with boundary conditions described by Equation 3.3 and coefficients $c_1 = 0.5, c_2 = 0.5, c_3 = c_4 = c_5 = 0$; and Co-sim describes the parallel simulation which was presented above.

While one might expect that the runtime of the parallel simulation should coincide with the maximum value comparing Sub:1, Sub:2 and Sub:3 simulation runtimes at each run period (one day, one week and one month), Table 9 shows that, when the run period of simulation is growing,

the resulting runtime of the dynamic connection is always greater than the corresponding runtime of sub-buildings. This is due to the large size of data exchanged through the co-simulation.

Table 9: Simulation runtimes for each sub-building (Sub:1, Sub:2 and Sub:3), the parallel simulation (Co-sim) and the whole building (Full)

Run Period	sub:1 (sec)	sub:2 (sec)	sub:3 (sec)	Co-sim (sec)	Full (sec)
1 day	13.6	9.9	13.2	15.4	98.9
1 week	20.5	21.9	17.3	37.8	201.1
1 month	47.4	75.5	33.5	120.9	612.9

Nevertheless, it is estimated that, dividing the whole building to three sub-buildings leads to a reduction of simulation runtime by 80 %. Such a result was expected since the division of the original building has significantly reduced the number of thermal zones, walls and windows, as shown in Table 10.

Table 10: Number of zones, walls and windows for each sub-building (Sub:1, Sub:2 and Sub:3) and the whole building (Full-scale)

Model	Full-scale	Sub:1	Sub:2	Sub:3
Zones	30	8	14	8
Walls	281	101	107	99
Windows	105	33	51	21

3.2 Zoning Reduction Approaches

In common practice, a full-scale thermal simulation model treats each room of a building as an individual thermal zone. Such an assumption increase significantly the simulation runtime, since computational effort is more than proportional to the number of zones; as mentioned earlier, increased number of zones corresponds to increased number of ordinary differential equations to be solved. Hence in many cases, building simulation modelers incorporate the HVAC zones definition, where each zone consists of one or more rooms and a thermostat assigned to that zone. At this level of detail, the thermal simulation model, where each HVAC zone is a thermal zone, can be still expensive for computationally demanding tasks. Concerning a further zoning reduction, building simulation experts are able to reduce the number of HVAC-thermal zones, but such a reduction is usually based on some similarity between the regions being combined (e.g. similar internal loads).

Towards an automatic methodology to reduce the number of zones, utilizing simulation results of a full-scale, validated, thermal simulation model, in this Section, two approaches are presented. The first approach utilizes the Hierarchical Clustering theory [9], while the second approach adopts the Koopman modes theory [10]. The Koopman modes, as a systematic approach to zoning and model reduction, has recently been proposed in [11], where motivational results are presented for a real building.

Implementations of these approaches are illustrated in the full-scale model of CARTIF Building and their results are presented.

3.2.1 Hierarchical Clustering Analysis

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Hierarchical clustering strategies are divided into agglomerative and divisive. Given a data-set consisting of a number of objects — for example, time-series of thermal zones' air temperatures, derived from a thermal model simulation for a predefined simulation run-period — the agglomerative technique considers that each object starts in its own cluster and pairs of clusters are merged at each step, until all objects are placed in one cluster (bottom-up). The divisive technique works vice versa; assuming that all objects belong to a cluster, it divides this cluster until each cluster contains a unique object (top-down).

Here, the agglomerative strategy for hierarchical clustering is adopted, since considering a large number of thermal zones, a reduced number of thermal zones must be obtained.

In a broad sense, the agglomerative technique is performed as follows [9]:

1. A single cluster is defined for each object of the data-set; for instance, if there is a set of n_x objects to be merged, there are n_x clusters, where each cluster contains an object of n_x;
2. The similarity or dissimilarity between every pair of objects in the data set is estimated; here, the distance between every objects' pair is calculated. Some commonly norms (metric) to compute this distance are: Euclidean, Squared Euclidean, and Maximum distances.
3. The pair with the minimum distance is selected; this pair is merged into one cluster, leading to newly formed clusters.
4. The newly formed clusters are grouped into larger clusters until a hierarchical tree is formed. To calculate the distance between clusters that include more than one objects a linkage criterion is selected. The linkage criterion determines the distance between sets of objects as a function of the pairwise distances between objects. Some commonly used linkage criteria between two sets of objects are: Maximum, Minimum and Average linkages.
5. Finally, the point to cut the hierarchical tree into clusters is determined. In this step, branches off the bottom of the hierarchical tree are pruned, and all objects below each cut are assigned to a single cluster.

The choice of an appropriate metric (in step 2.) will influence the shape of the clusters, as some elements may be close to one another according to one distance and further away according to another.

To explain the aforementioned metrics and linkage criteria, suppose that we have a data-set represented as a matrix $G \in \mathbb{R}^{n_x, n_{\text{data}}}$ where consisting of n_x stands for number of objects (e.g. n_x zones' operative temperature timeseries) and n_{data} stands for the number of time steps when the values were measured. Moreover, two clusters A and B are considered. Cluster A consists of n_A objects where each of them is element of $\{1, 2, \ldots, n_x\}$, while cluster B consists of n_B objects $\{B_1, B_2, ..., B_{n_B}\}$, where $n_A + n_B \leq n_x$. The distance $\delta_{i,j}$ between objects $i \in A$ and $j \in B$ can be calculated by one of the following equations:

- According to the Euclidean distance:

$$\delta_{i,j} = \sqrt{\sum_{k=1}^{n_{\text{data}}} (G_{i,k} - G_{j,k})^2}; \tag{3.7}$$

- or according to the squared Euclidean distance:

$$\delta_{i,j} = \sum_{k=1}^{n_{\text{data}}} (G_{i,k} - G_{j,k})^2 \tag{3.8}$$

- according to the maximum distance:

$$\delta_{i,j} = \max_k |G_{i,k} - G_{j,k}|. \tag{3.9}$$

The distance between cluster A and cluster B, δ_{AB}, is defined as follows:

- According to the Maximum linkage criterion:

$$\delta_{AB} = \min_{i,j} \delta_{i,j} \tag{3.10}$$

- according to Minimum linkage criterion:

$$\delta_{AB} = \max_{i,j} \delta_{i,j} \tag{3.11}$$

- according to Average linkage criterion:

$$\delta_{AB} = \frac{\sum_{i \in A} \sum_{j \in B} \delta_{i,j}}{n_A \cdot n_B}. \tag{3.12}$$

In the present work, the choice of linkage criterion is not of high importance, since the point to cut the hierarchical tree into clusters is defined as the point where the zones' pair with the minimum distance are linked (see Figure 12).

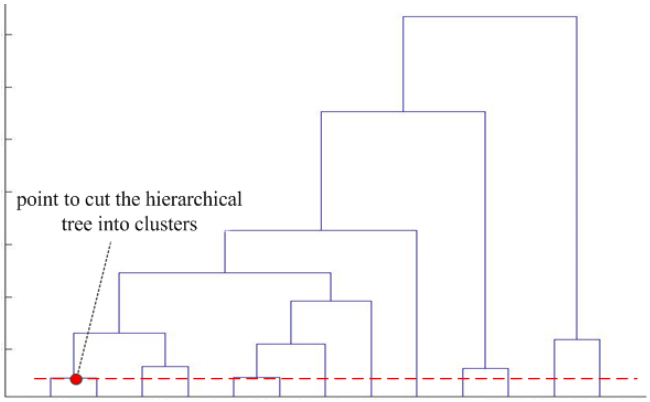

Figure 12: Hierarchical tree – the point to cut the hierarchical tree into clusters is defined as the point where the zones' pair with the minimum distance are linked

Such a definition stems from the following interpretation: When two thermal zones are merged into a new one, the new zone's resulted temperature profile could be conspicuously different from the temperature profiles of the initial zones; hence, the data-set should be updated whenever a two zones' merging is performed. To accomplish this, the agglomerative technique is performed repeatedly in the following manner: (1) Set the desired number of zones; (2) Consider a data-set consisting of time series, derived from the full-scale zonal model simulation; (3) Apply the agglomerative technique, where the point to cut the hierarchical tree into clusters is defined as the point where the zones' pair with the minimum distance are linked; (4) Create a newly formed zonal model according to the results of step 3 and simulate it to receive the updated data-set; (5) Using the updated data-set, repeat steps 3 and 4, until the number of zones is equal to the desired one.

3.2.2 Koopman Mode Analysis

Because the equations describing a building model can be of a high dimension and are often not accessible analytically, methods are required which are measurement, or time-series based, to study such systems. In this context, the Koopman operator can be applied to building models for the visualization and analysis of these systems. By projecting the data-set (time-series of simulated objects) from a building simulation onto eigenfunctions of the operator, spatial features of the system being studied can be extracted.

The procedure of merging thermal zones from a full-scale model using Koopman modes is:

1. Simulate the full-scale thermal simulation model resulting to an objects' data-set of interest (in this work thermal zones' air temperatures.
2. Calculate Koopman eigenvalues and modes by projecting the objects' data-set onto eigenfunctions of the Koopman operator.
3. Merge thermal zones with Koopman modes of similar amplitude and phase at frequencies (modes) of interest.

There are several methods available for calculating Koopman modes [10, 12]. Here, Arnoldi algorithm [12] is selected, since in [11] it is presented as an efficient algorithm to decompose building simulation time series into Koopman modes, able to capture the thermal behavior of a building. Arnoldi algorithm is described below.

Suppose again that we have the data matrix G as introduced above. Then, empirical Ritz values (Koopman eigenvalues) λ_k and empirical Ritz vectors (Koopman modes, eigenfunctions) v_k, $\forall k = 1,2,\ldots,n_{\text{data}}$ are calculated by the following procedure:

1. Find constants c_k, $\forall k = 1,\ldots,n_{\text{data}}-1$ such that $\boldsymbol{r} \perp G$, where:

$$\boldsymbol{r}_i = G_{i,n_{\text{data}}} - \sum_{i=1}^{n_{\text{data}}-1} c_i G_{i,k}. \tag{3.13}$$

   for all $i = 1,\ldots,n_x$. Since $\boldsymbol{r} \perp G$ the following equation holds:

$$G'_{\bullet,k}\boldsymbol{r} = 0 \tag{3.14}$$

   for all $k = 1,\ldots,n_{\text{data}}-1$ where $G_{\bullet,k}$ stands for kth column of matrix G.

   Suppose that $A = \{a_{k,\tilde{k}}\} \in \mathbb{R}^{n_{\text{data}}-1,n_{\text{data}}-1}$, where $a_{k,\tilde{k}} = G'_{\bullet,k}G_{\bullet,\tilde{k}}$ and $b_k = G'_{\bullet,k}G_{\bullet,n_{\text{data}}}$ and $c = \{c_i\} \in \mathbb{R}^{n_{\text{data}}-1}$. Then, constants c_k can be found by solving the following system of linear equations:

$$Ac = b \tag{3.15}$$

2. Define the companion matrix C:

$$C = \begin{bmatrix} 0 & 0 & \cdots & 0 & c_1 \\ 1 & 0 & \cdots & 0 & c_2 \\ 0 & 1 & \cdots & 0 & c_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & c_{n_{\text{data}}-1} \end{bmatrix} \tag{3.16}$$

and find its eigenvalues λ_k, $\forall k \in \{1,2,\ldots,n_{\text{data}}-1\}$.

3. Define the Vandermonde matrix T as follows:

$$T = \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^{n_{\text{data}}-2} \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^{n_{\text{data}}-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_{n_{\text{data}}-1} & \lambda_{n_{\text{data}}-1}^2 & \cdots & \lambda_{n_{\text{data}}-1}^{n_{\text{data}}-2} \end{bmatrix}. \tag{3.17}$$

4. Finally, calculate the matrix $V = GT^{-1}$, columns of which are the Koopman modes. Note that V might contain complex numbers because of inversion of T.

From the calculated Koopman modes of a full-scale building model data, thermal zones are merged if their amplitudes and phases within the Koopman modes considered are within some tolerance. Combining zones using this approach physically corresponds to combining zones which behave similarly due to internal heat generation and environmental heat transfer. The following definition is used for comparing the amplitudes and phases of zones and creating merged zones approximations:

Select $\varepsilon_1, \varepsilon_2 \geq 0$ and consider $i, j \in \{1,2,...,n_x\}$, while the data' objects are the n_x zones' air temperatures. Then, zones i and j can be merged if:

$$|||v_{i,k}|| - ||v_{j,k}|| < \varepsilon_1 \tag{3.18}$$

$$|\angle v_{i,k} - \angle v_{j,k}| < \varepsilon_2 \tag{3.19}$$

for all the k-th Koopman modes of interest where $||\cdot||$ stands for absolute value of a complex number and $\angle$ stands for phase of a complex number. Koopman modes of interest correspond to the largest modes, calculated in Step 4. The main idea of investigating the validity of Equations 3.18 and 3.19 for Koopman modes of interest and not all Koopman modes is based on the fact that only several modes are required to describe important characteristics of the building's thermal response.

3.2.3 Simulation Results

The implementation of Zoning Reduction approaches is performed by a proposed Automatic Process described in Section 5.2, which has been evaluated for a real office building, the CARTIF Building. The main objective of the test case presented here is to verify applicability of the Process and to evaluate the proposed Zoning Reduction Approaches, with respect to the accuracy of the simulation results and their impact (positive or negative) to the computational effort.

A set of experiments were performed for each Zoning Reduction Approach, selecting as data-set the simulated zones' air temperature. A whole year simulation time interval is selected, while assuming all zones to be free floating, the conditions during this period are as follows: shading

devices are completely open; internal gains are equal to zero, since lights and electric equipment are switched off; there is no HVAC system available to control the temperature of the zone; and natural ventilation is not considered.

Evaluating the proposed Automatic Process, the main positive result is that both Zoning Reduction Approaches lead to the same zoning approximations, confirming thereby the effectiveness of the Hierarchical Clustering Approach, since the Koopman Modes Approach has been verified in the context of a recent work [11]. For intuition on how those approaches reduce the number of zone, the 3D geometries of the 25 zones (the initial full-scale model), 24 zones and 23 zones models are depicted in Figure 13.

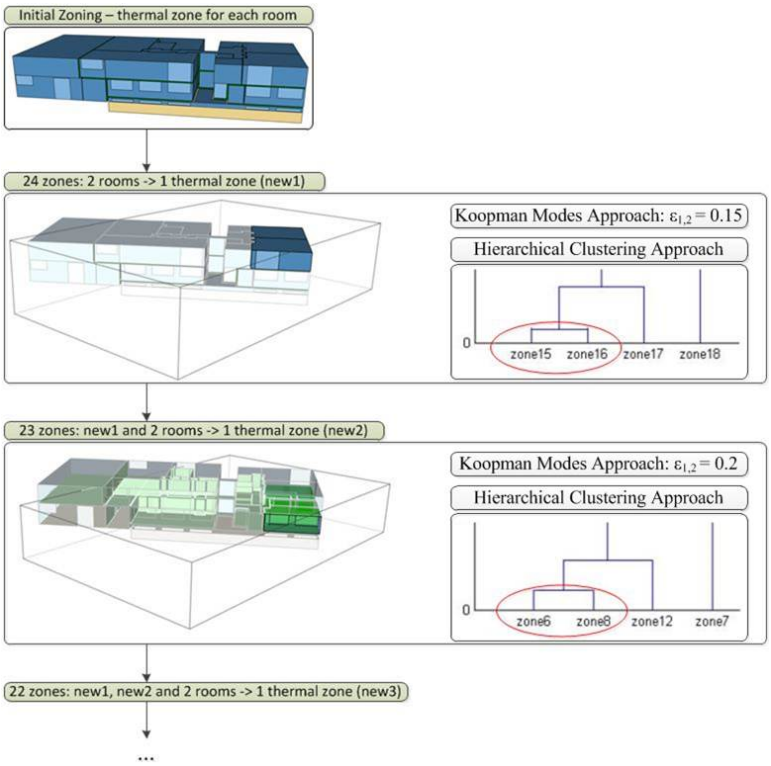

Figure 13: Zoning approximations for the first two steps of Hierarchical Clustering Approach and the respective $\varepsilon_{1,2}$ values of Koopman Modes Approach in CARTIF Building, using as data-set variables the zones' operative temperatures

Although both Zoning Reduction Approaches are equally effective, the ease of use of the Hierarchical Clustering Approach, lies in the fact that the only input to the algorithm is the desired number of zones; in contrast, the difficulty of use of the Koopman Modes Approach, lies in the difficulty of selecting proper values of ε_1, ε_2 and the number of modes of interest, since such selection does not have a physical interpretation, but it is based on experience the modeler has with this Approach.

Towards investigating the impact of the generated zoning approximations on model accuracy, initially a uniform criterion is used, the building's Heating/Cooling (H/C) energy demands. Here, all zones of each evaluated speedup model are simulated for a whole year simulation time interval with heating and cooling. The accuracy of a speedup model is measured by comparing the total energy demands required to maintain the zones' air temperature at the the range $[20, 25]$ °C, while the conditions during this period are as follows: shading devices are completely open; internal gains are equal to zero, since lights and electric equipment are switched off; an ideal HVAC system is available at each zone to control the zone air temperature; and natural ventilation is not considered.

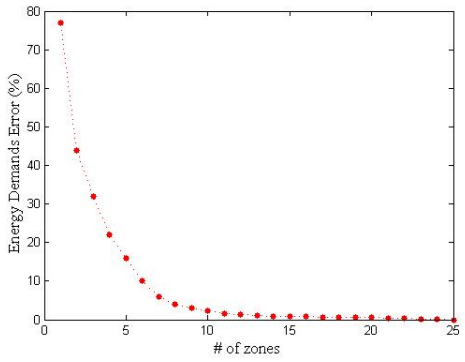

Figure 14: H/C energy demands' error in prediction as the number of zones increases

Table 11: Results of Zoning Reduction Approaches — Different Koopman modes tolerances $\varepsilon_{1,2}$ result to different H/C energy demands' error and simulation runtimes

♯ of zones	$\varepsilon_{1,2}$	Demands' error %	Runtime (*sec*)
25	0	0	382
20	0.3	0.5	296
15	0.68	0.8	223
10	0.7	2.3	160
5	0.83	16	115
1	1	77	92

Moreover, the speedup models are designed for the purpose of control design. Control design requires repeated simulations for the prediction horizon in connection with a stochastic optimization algorithm; a prediction horizon of one day along with six days of the warming up phase, reveals a time interval of one week to be set for each simulation, and as such the simulation runtime of each zoning approximation is compared to that of the full-scale model, considering a one-week time interval.

As Table 11 shows, for the Koopman Modes Approach, setting the number of Koopman modes of interest to 10 constantly, as increased values of ε_1 and ε_2 are used to check if Equations 3.18 and 3.19 hold, more zones are determined to be sufficiently similar to each other, reducing the total number of zones. As the number of zones reduces, the calculated H/C energy demands' error increases, while the simulation runtime decreases. In words of the Hierarchical Clustering Approach, as the desired number of zones decreases, the model's accuracy increases, while the simulation is less computationally expensive.

According to Figure 14, the H/C energy demands' error in prediction is inversely proportional to the number of zones.

The effectiveness of each zoning approximation to predict the zone's operative temperature, in terms of the root mean square error (*rmse*) and the maximum deviation indicators ($\delta_{\max}$), is also investigated, where both the full-scale model and a group of reduced models — specifically, 25 zones, 20 zones, 15 zones, 10 zones, 5 zones and 1 zone models — are simulated for a whole year time interval (run period).

Let h be a function that assigns to an index of zone in the original settings the index of the an aggregated zone. Let $T_{\text{full-scale},k,i}$ be the zone i of the full-scale model operative temperature at time step k and $T_{\text{reduced},k,h(i)}$ be the zone $h(i)$ of the zoning reduced model, part of which is the zone i of the full-scale model, operative temperature. The $rmse_i$ between the full-scale model consisting of N zones and a reduced model consisting of $M < N$ simulation results is given by:

$$rmse_i = \sqrt{\sum_{k=1}^{n_{\text{data}}} \left(T_{\text{full-scale},k,i} - T_{\text{reduced},k,h(i)}\right)^2 / n_{\text{data}}} \tag{3.20}$$

where n_{data} is the total number of timesteps (e.g. $n_{\text{data}} = 52560$, for a whole year simulation with 10min timestep), while the $\delta_{\max,i}$ is calculated as follows:

$$\delta_{\max,i} = \max_{k} |T_{\text{full-scale},k,i} - T_{\text{reduced},k,h(i)}|. \tag{3.21}$$

Table 12 summarize the change in model predictive capability of operative temperature as the number of zones is reduced. The *rmse* and the $\delta_{\max}$ results, clearly confirm what was discussed above. The accuracy of the model decreases exponentially as the number of zones is reduced. Thus, a critical point in the number of zones exists, beyond which accuracy does not substantiate the use of zonal models as surrogates of real buildings. Any number of zones greater than this critical point is acceptable, but the following trade-off is obvious: for more accurate results much more number of zones must be provided and calculations are more time demanding, therefore it should carefully be considered which is the computational complexity for a specific task and what is its required level of accuracy: one-size-fits-all does not apply here.

Table 12: Error in model predictive capability of zones' operative temperature as the number of zones is reduced, in terms of the *rmse* and the $\delta_{\max}$

Initial IDF	root mean square error					maximum deviation				
zone name	20 zones	15 zones	10 zones	5 zones	1 zone	20 zones	15 zones	10 zones	5 zones	1 zone
zone 1	0.0235	0.0311	0.0452	0.3419	1.2311	0.0448	0.0647	0.1067	0.7687	2.2385
zone 2	0.0213	0.0319	0.3793	0.3945	0.8915	0.0454	0.0725	1.0913	1.0269	1.8302
zone 3	0.0064	0.0096	0.1487	0.5426	0.5339	0.0129	0.0192	0.4061	1.2614	1.7699
zone 4	0.1090	0.1692	0.1935	0.3842	1.5404	0.3179	0.4484	0.4627	0.8759	2.8924
zone 5	0.1338	0.1448	0.1172	0.2381	1.6115	0.3926	0.5088	0.4401	0.6902	2.6272
zone 6	0.1286	0.1306	0.2054	0.6142	2.5790	0.3137	0.3179	0.4579	1.2131	3.7374
zone 7	0.0739	0.1711	0.6941	0.5365	2.2538	0.1081	0.5436	1.3722	1.0363	3.8359
zone 8	0.0896	0.0981	0.1629	0.6103	2.5425	0.1795	0.2342	0.3973	1.2083	3.7845
zone 9	0.0776	0.2338	0.5129	0.4205	2.1229	0.1369	0.6879	1.2964	1.1483	3.0774
zone 10	0.0714	0.0844	0.4137	0.3204	2.5445	0.1229	0.2349	0.7189	0.8309	3.8221
zone 11	0.0685	0.3629	0.3684	0.3764	1.9367	0.1138	1.0128	1.0954	1.0858	3.8671
zone 12	0.1673	0.1843	0.2106	0.5963	2.5178	0.5787	0.6077	0.6199	1.2752	3.9272
zone 13	0.0411	0.1646	0.2002	0.2836	1.7620	0.0612	0.5328	0.6045	0.7692	3.3716
zone 14	0.0314	0.0523	0.6994	0.9008	1.1261	0.0822	0.1614	1.3544	1.8000	2.5878
zone 15	0.1681	0.2129	0.2276	0.3976	4.2139	0.3373	0.4042	0.5463	0.7732	5.3185
zone 16	0.1910	0.2376	0.2389	0.4282	4.1810	0.4409	0.5197	0.5949	0.9343	5.2351
zone 17	0.0889	0.2910	0.3275	0.4242	4.2731	0.2387	1.0085	1.0045	1.3040	5.3474
zone 18	0.0507	0.0689	0.0957	0.4654	4.7275	0.0720	0.1986	0.2732	1.2421	5.6199
zone 19	0.0533	0.2971	0.3355	0.9223	3.4709	0.0892	0.7344	0.8387	2.7311	5.4492
zone 20	0.1423	0.1908	0.2139	0.2363	2.9260	0.3835	0.4894	0.5486	0.8117	3.3700
zone 21	0.1327	0.1420	0.1839	0.3211	3.0695	0.3777	0.4468	0.5088	0.7459	3.6459
zone 22	0.0745	0.1264	0.3980	0.7728	2.6029	0.2533	0.4184	1.2235	2.5238	3.8112
zone 23	0.0344	0.0574	0.3464	0.8766	2.1281	0.0708	0.1519	0.7460	2.1886	2.5584
zone 24	0.0706	0.1387	0.1555	0.6730	3.1931	0.1320	0.3947	0.4241	2.3259	4.7794
zone 25	0.0141	0.0210	0.1499	0.2301	2.1884	0.0286	0.0426	0.2984	0.8617	3.2731

4 Inductive Approach

This chapter describes a way how to infer the simplified model from available data, either simulated or real.

4.1 General Methodology

The principle of the proposed methodology is that we will explore the hierarchy of simplified models from simple to more complex ones. Namely, let the setting of the model be represented by an integer vector $r \in \mathbb{N}^{n_r}$. The search for the best model starts from the simplest model with $r = (1,1,\ldots,1)$. Consequently, the algorithm examines more complex models by trying to increase individual components of c. For instance, if $n_r = 3$, the models with settings $(1,1,2)$, $(1,2,1)$ and $(2,1,1)$ are tried. The setting of the tried model with the lowest prediction error is considered as starting point for next iteration of trials. E.g. the setting $(1,2,1)$ had the lowest prediction error and the next iteration will examine settings $(2,2,1)$, $(1,3,1)$, and $(1,2,2)$. This process is iterated till the termination conditions are satisfied which could be (i) achieving sufficient prediction error, (ii) exceeding the maximal calculation time or maximal number of iterations, (iii) the precision error does not decrease with more complex models. This approach corresponds to the (1+1) evolutionary algorithm which has been successfully used for acceleration of combinatorial optimization [13, 14], including the case of model selection [15].

The algorithm is summarized in Pseudocode 1. Note that a part of data, namely D_{test} is not used during the model selection. This part of data is used for the realistic quantification of the prediction error. We will assume that D_{test} is the last block of data D.

Pseudocode 1: General Methodology

input : data $D = (x_t, a_t, d_t)_{t=1}^{n_{\text{data}}}$
output : optimal model ${}^{O}\mathscr{M}$, error on testing set ${}^{O}e$
divite D into data D_{trv} for training and validation and D_{test} for testing
$r \leftarrow (\underbrace{1,1,\ldots,1}_{n_r})$
$[{}^{O}\mathscr{M}, e_{\min}] \leftarrow \textsc{TrainAndValidate}(D_{\text{trv}}, r)$
while *termination conditions not satisfied* **do**
  $e_{\text{tmp}} \leftarrow \infty$
  forall the $j = 1,\ldots,n_r$ **do**
    $r' \leftarrow r$
    $r'_j \leftarrow r'_j + 1$
    $[\mathscr{M}, e] \leftarrow \textsc{TrainAndValidate}(D_{\text{trv}}, r')$
    if $e < e_{\text{tmp}}$ **then**
      $e_{\text{tmp}} \leftarrow e$
      $r_{\text{tmp}} \leftarrow r'$
      if $e < e_{\min}$ **then**
        $e_{\min} \leftarrow e$
        ${}^{O}\mathscr{M} \leftarrow \mathscr{M}$
  $r \leftarrow r_{\text{tmp}}$
${}^{O}e \leftarrow \textsc{CalculateError}({}^{O}\mathscr{M}, D_{\text{test}})$
return ${}^{O}\mathscr{M}$

4.2 Training and Validation

4.2.1 Block-Based Cross Validation

In this section, we will specify the developed procedure for training and validation of autoregressive models. This procedure is based on the blocked cross validation which is known to provide robust models of autoregressive time series [16].

The principle is that the available data are divided in several blocks. For each block (considered as validation set) $D_{[k]}$, a model is learned from the other blocks (considered as training set)

$$D_{[1]}, D_{[2]}, \ldots D_{[k-1]}, D_{[k+1]}, \ldots D_{[n_{\text{blc}}]}$$

and the error is calculated for $D_{[k]}$.

Note that the input-output pairs that are provided from each block depend on the considered order of the autoregressive models. For instance, if a block contains 1000 records and the order is 1, then there are 999 pairs $[(x_{t+1}), (x_t, a_t, d_t)]$. If the order is 2, then we have 998 pairs

$$[(x_{t+1}), (x_t, a_t, d_t, x_{t-1}, a_{t-1}, d_{t-1})].$$

Pseudocode 2: Train and Validate

input : data $D = (x_t, a_t, d_t)_{t=1}^{n_{\text{data}}}$ setting c
output : learned model $\mathscr{M}$, cross validation error e
divide the data D into blocks $D_{[1]} \ldots D_{[m]}$
forall the $k = 1, \ldots, m$ **do**
  $\mathscr{M} \leftarrow \textsc{Train}([D_{[1]}, D_{[2]}, \ldots D_{[k-1]}, D_{[k+1]}, \ldots D_{[m]}], c)$
  $e_k \leftarrow \textsc{CalculateError}(\mathscr{M}, D_{[k]})$
$e \leftarrow \frac{1}{m} \sum_{k=1}^{n_{\text{blc}}} e_k$
$\mathscr{M} \leftarrow \textsc{Train}(D, c)$

4.2.2 Calculating the Prediction Error

Given a model and a block of data, we need to quantify the quality of the model. In our algorithms, we need this at two places: first for the cross validation, second for the final estimate of the model's precision using the test set.

A very standard way to calculate the prediction error is the mean square error. In our case we have to treat two more aspects of the prediction error calculation.

1. We consider the forecasting for a longer time horizon, e.g. 24 hours. Therefore, the values at 11 a.m. are predicted at 10 a.m., 9 a.m., 8 a.m. etc.
2. We assume that we deal with multiple state variables, i.e. x_t is multidimensional and the variables might be of different physical units and be directly incomparable.

With respect to these points, we consider the calculation of the aggregated mean squared error as follows:

$$e = \sum_{t=1}^{(n_{\text{data}} - n_{\text{hor}})} \sum_{j=1}^{n_{\text{hor}}} \sum_{i=1}^{n_x} w_i \left(\frac{x_{t+j,i} - \hat{x}_{t,j,i}}{\sigma_i} \right)^2 \tag{4.1}$$

where

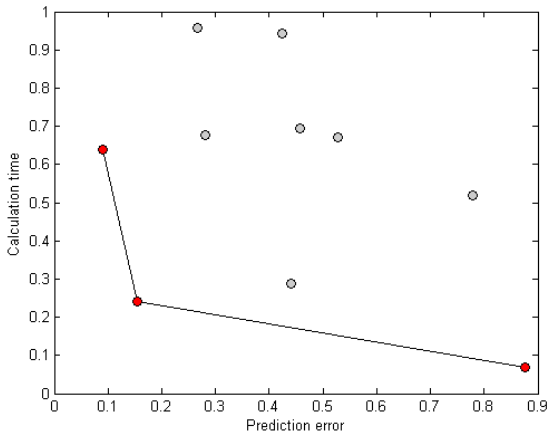

Figure 15: Illustration of Pareto frontier.

- w_i - relative weight of variable i after normalization. To be provided by user.
- $x_{t+j,i}$ - measured value of ith variable at time $t+j$
- $\hat{x}_{t,j,i}$ forecast of measured value of ith variable at time $t+j$ calculated at time t
- σ_i - deviation considered errors, can be calculated semi-automatically.

4.2.3 Multi-criteria interpretation of the results

As already discussed in Section 2, we consider two quantitative criteria: (i) prediction error, (ii) computation time. In order to provide the user with all information that is needed to cope with this trade-off, we propose to log for each training and validation both criteria. This can be shown in a scatter plot with visualization of the Pareto frontier.

4.3 Implementing the Methodology with Autoregressive Gaussian Mixtures

We have decided to implement the methodology using autoregressive Gaussian mixtures. Gaussian mixtures approximate probability density functions by weighted combination of multivariate normal distributions [17].

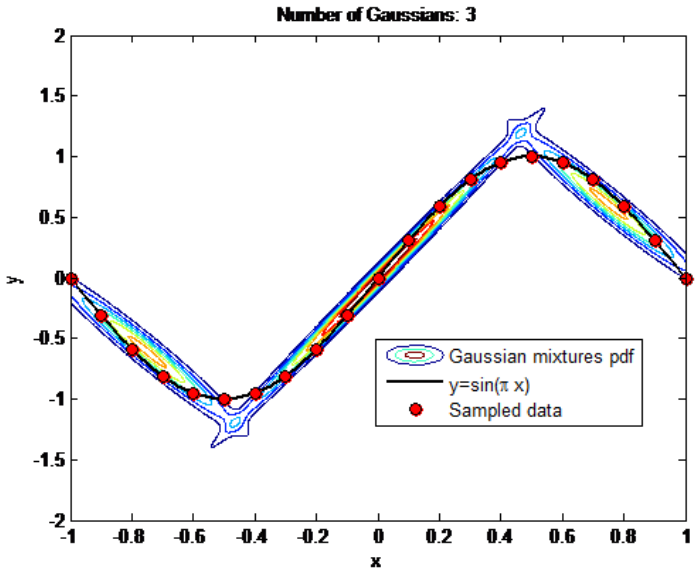

Figure 16: Illustration of Gaussian mixture regression

We will use the Gaussian mixtures to model the states x_τ where $\tau = t+1, t+2, \ldots t+n_{\text{hor}}$ using the probability density function

$$f(x_\tau, x_{\tau-1}, a_{\tau-1}, d_{\tau-1}, \ldots x_{\tau-n_{\text{ord}}}, a_{\tau-n_{\text{ord}}}, d_{\tau-n_{\text{ord}}})$$

where $n_{\text{ord}} \geq 1$ is the order of the autoregressive model. From this model, it is possible to infer easily the conditional pdf

$$f(x_\tau | x_{\tau-1}, a_{\tau-1}, d_{\tau-1}, \ldots x_{\tau-n_{\text{ord}}}, a_{\tau-n_{\text{ord}}}, d_{\tau-n_{\text{ord}}})$$

and consequently the deterministic version as mean

$$\hat{x}_\tau = E[x_\tau] = M(x_{\tau-1}, a_{\tau-1}, d_{\tau-1}, \ldots x_{\tau-n_{\text{ord}}}, a_{\tau-n_{\text{ord}}}, d_{\tau-n_{\text{ord}}})$$

where $E[\cdot]$ stands for expected value. Gaussian mixtures have several suitable properties. With the increased number of components, the precision of the model increases as well as its complexity, see Figure 17.

Gaussian mixtures are able to deal with missing values and use statistically sound Bayesian learning and its modifications. In contrast to just-in-time local regression, Gaussian mixtures are very fast in the evaluation. An informal comparison of Gaussian mixtures with other approaches (Neural networks, Gaussian processes, polynomial regression, local regression, ensemble local models) is summarized in Figure 18.

Now, let us specify how the general methodology is implemented using these models. We consider the complexity settings r to be two-dimensional r_1 corresponds to the number of components (Gaussians) in the mixture while r_2 is the parameter expressing the order of the autoregression. For this case, the general methodology as described in Pseudocode 1 is illustrated in Figure 19.

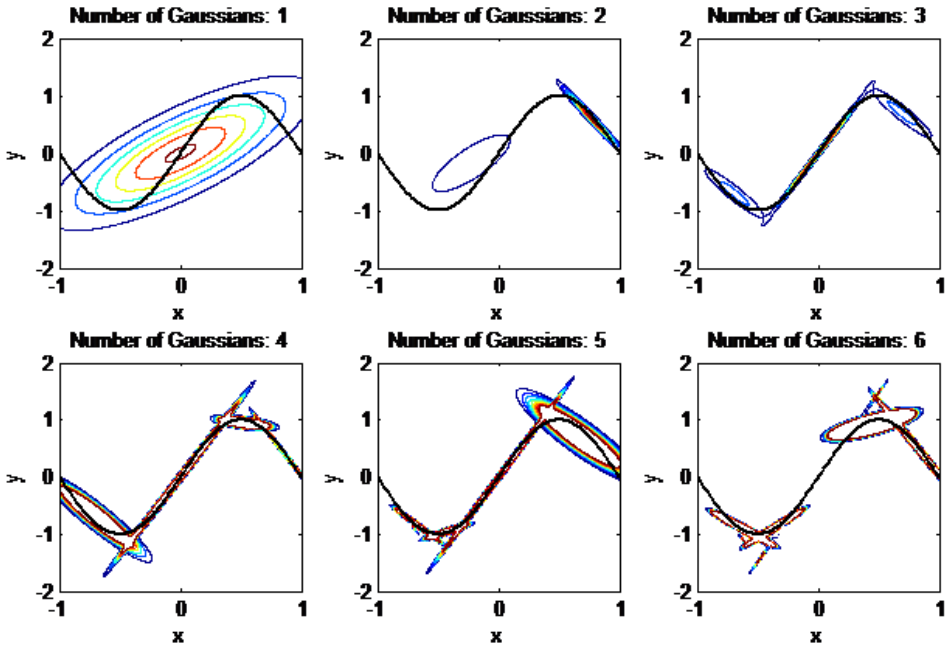

Figure 17: Growing number of components in Gaussian mixture regression.

	NN	GP	PR	LR	ELM	GM
Fast to evaluate	✓		✓		✓	✓
Universal approximation	✓	✓	✓	✓	✓	✓
Avoids over-fitting	✓	✓		✓	✓	✓
Reliable learning		✓	✓	✓	✓	✓
Explicit treatment of uncertainty		✓				✓
Small representation	✓		✓		✓	✓
Works with missing values				✓		✓

Figure 18: Why Gaussian mixtures - qualitative comparison to alternative approaches: Neural Networks (NN), Gaussian Processes (GP), Polynomial Regression (PR), Local Regression (LR), Ensamble Linear Models (ELM), Gaussian Mixtures (GM)

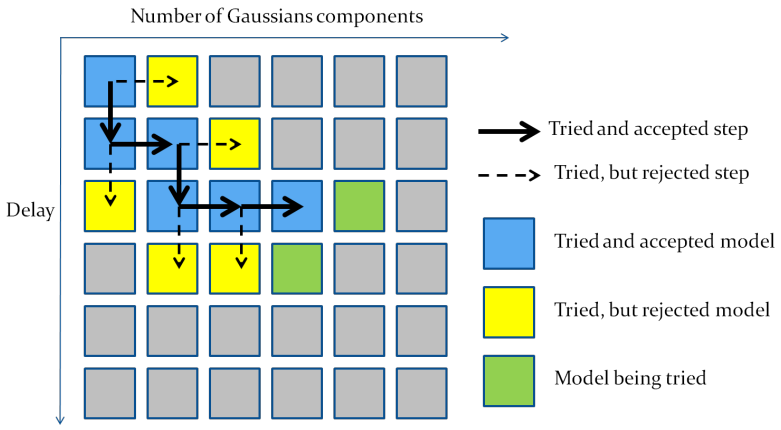

Figure 19: Evolution of the methodology for Gaussian mixtures - from simple to more complex models.

4.4 Case Studies

4.4.1 Simulated FIBP Data

In this case study, we used data simulated from BEPS model of the FIBP building. The weather was taken from the Meteonorm for Frankfurt am Main [18] namely the data from January and February and we have focused on the so called tower model only. The available data points are summarized in Table 13.

We have focused on the temperatures, thus we set $w_1 = w_2 = w_3 = 1/3$ and $\sigma_1 = \sigma_2 = \sigma_3 = 1$ and worked with the horizon of 24, i.e. 6 hours with 15 minutes sampling.

The results show that of the BEPS simulation form the tower model can be approximated with a very good precision. In Figure 20, we can see the trade-off between the cross-validation error and the time that is needed for evaluation of 521 simulations (this number corresponds to the size of the block in the block-based cross validation). The most accurate model has $n_{\text{gau}} = 6$ Gaussian components and autoregressive order $n_{\text{ord}} = 5$. Increasing the complexity by one more Gaussian component leads to loss of the precision due to over-fitting, see point r =[7 5]. The fastest one is naturally $n_{\text{gau}} = 1$ and $n_{\text{ord}} = 1$.

The very high accuracy is not very surprising in this case: the original tower model is defined in terms of noise-free nonlinear dependencies. Since Gaussian mixtures are known to approximate rich class of distributions since the Gaussian radial basis functions are able to approximate any rich class of continuous functions [19]. The discussed results only confirm this theoretical property in the practice.

Regarding the complexity, Figure 20 indicates that the transition from linear, i.e. r =[1 1] to simple nonlinear r =[3 1] model improves the precision significantly without serious increase of computational complexity. To decrease the cross-validation error further is related to significant increase of complexity. The six-hour ahead prediction of the most precise model [6 5] is shown in Figure 21.

To make the results comparable with the inductive approach, we know that the time is related to 521 simulations and 6 hours. In order to estimate the simulation time for a single simulation of 72

Table 13: Data points available for the simulation study

Data point	**Comment**
Energy007	State variable x_1
Energy107	State variable x_2
Energy207	State variable x_3
Temp007	State variable x_4
Temp107	State variable x_5
Temp207	State variable x_6
RadiationSouth	Predictable disturbance d_1
AmbientTemp	Predictable disturbance d_2
Occupancy	Predictable disturbance d_3
ClothingFactor	Predictable disturbance d_4
Blinds007	Controlled input a_1
Blinds107	Controlled input a_2
Blinds207	Controlled input a_3
HotWaterTemp	Controlled input a_4
Massflow007	Controlled input (ignored since constant)
Massflow107	Controlled input (ignored since constant)
Massflow207	Controlled input (ignored since constant)

hours, as in the deductive approach multiply the time by

$$\frac{72}{6 \cdot 521}$$

In this way, the simulation time ranges from 0.3 to 5.5 seconds. The accuracy is very high as shown in Table 14. Comparing to Table 7, the considered criteria are smaller by one digit place.

Table 14: Precision of Inductive Models

Zone	$\overline{\Delta T}$	s
007	0.0032	0.0848
107	0.0082	0.0538
207	0.0105	0.0729

The inductive approach has demonstrated good prediction capabilities and it seems that it could be considered as an alternative to the deductive one that has more assumptions.

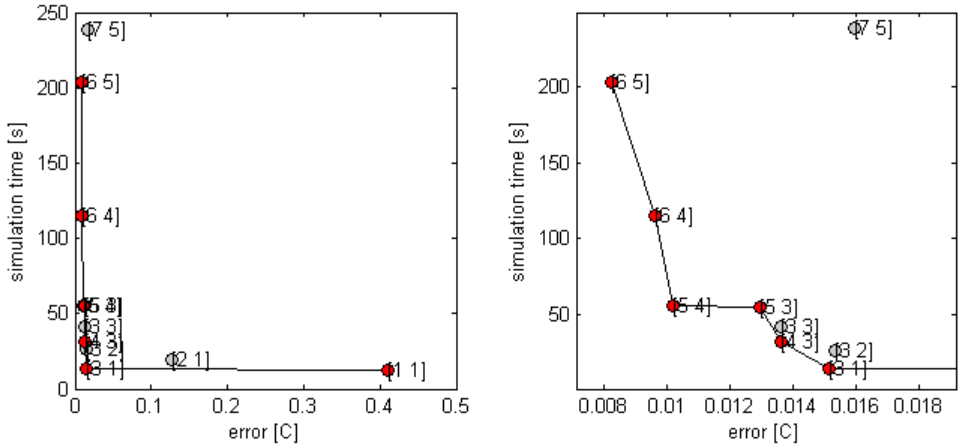

Figure 20: Results of the inductive approach. Each point correspond to a model with given number of components and autoregressive order $[n_{\text{gau}}\ n_{\text{ord}}]$. The right chart is detail of the left one.

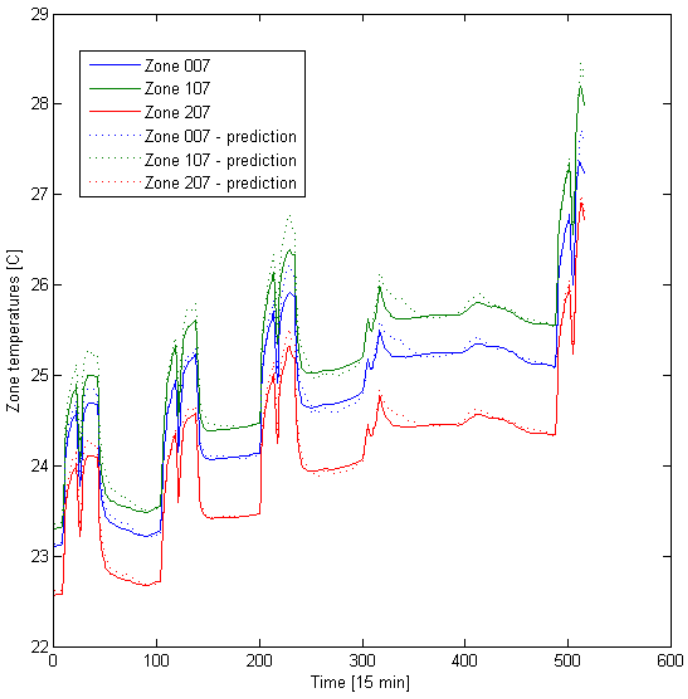

Figure 21: Zone air temperatures and their 6 hours old prediction by model with order $n_{\text{ord}} = 5$ and $n_{\text{gau}} = 6$ Gaussian components

5 Implementation and Integration in the BaaS Platform

After discussing the algorithms and approaches, this section informs about the practical use of them, namely it discusses how they are implemented and integrated in the BaaS platform.

First, Subsection 5.1 provides a general overview of the considered procedures, specifying their inputs, outputs, and triggers. In Subsections 5.2 and 5.3, more information on the implementation of particular approaches is provided, while in Subsection 5.4 the integration of derived simplified model with the Control Design process is configured.

5.1 General Overview of Considered Procedures

In terms of the integration within the platform, we can distinguish the following patterns:

- Generation of the inductive model
  - Inputs: Data, distinguishing states and disturbances.
  - Outputs: Dynamic model of the system.
  - Triggered: Once, when necessary to be initiated.
- Generation of the deductive model based on Zoning Reduction approaches
  - Hierarchical Clustering Approach Inputs: (1) full-scale zonal model; (2) weather file; (3) data-set; (4) desired number of zones.
  - Koopman Modes Approach Inputs: (1) full-scale zonal model; (2) weather file; (3) data-set; (4) desired error of merging; (5) Koopman modes of interest number.
  - Outputs: Simplified zonal model.
  - Triggered: Once, when necessary to be initiated.
- Adaptation of the inductive model (not implemented yet)
  - Inputs: Model, data.
  - Outputs: Updated model.
  - Triggered: Whenever new data collected.
- Use of the model for simulation
  - Inputs: Model, initial states, predicted disturbances, future inputs.
  - Outputs: predicted states.
  - Triggered: Whenever needed by some other services (control, diagnosis).

Note that there is no automatic procedure for the Geometry Reduction approaches since they are based on manual configuration. The ways how to automate this process can be considered, e.g. using ontological description of some templates representing repeating patterns [20], but they are not subject of this deliverable.

5.2 Implementation Notes - Deductive Approach

Summarizing, given a full-scale zonal model and a representative data-set consisting of a zone's level output variable (for example, air temperature for all zones) on which zoning reduction relies: (1) in Hierarchical clustering approach, substantial recreations of intermediate zonal models are required, before the recreation of the final speedup model that consists of the desired number of zones; (2) in Koopman modes approach, a zoning approximation, where Equations 3.18 and 3.19 hold for a predefined Koopman modes of interest number, is used to recreate the final speedup model.

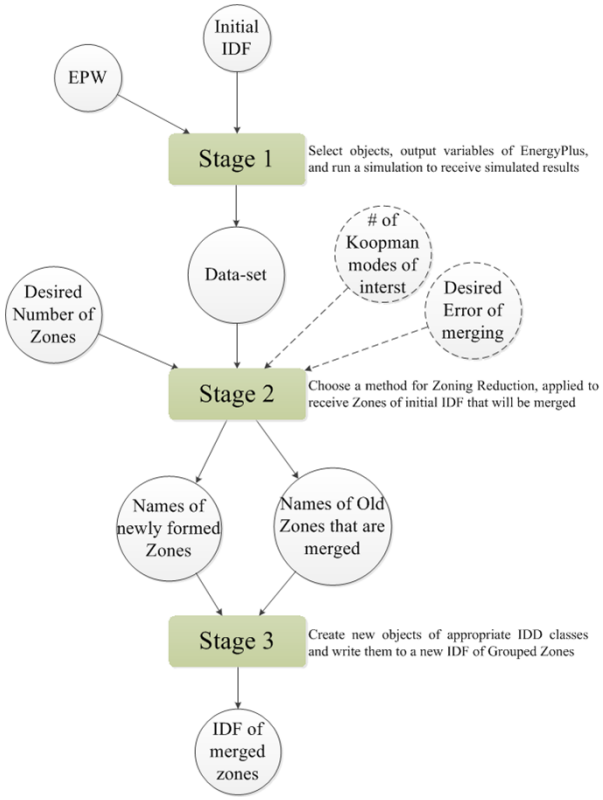

Figure 22: An overview of the Automatic Process for Generating Speed-up Model based on Zoning Reduction Approaches

In both approaches, recreation of a thermal simulation model is required, a tedious, slow and error-pruning process, commonly performed by a thermal simulation modeler manually. The benefit of an automatic process for generating speedup models based on Zoning Reduction Approaches would be twofold: (1) it would be orders of magnitude faster than manually recreating building geometry; and (2) it would be less susceptible to human error. Towards this direction, the three stages process presented in Figure 22 is proposed, utilizing EnergyPlus as the simulation engine to develop zonal models. In Stage 1, the objects of the data-set, output variables of EnergyPlus, are defined and a full-scale zonal model (intial model) simulation is performed to receive simulated results. A detailed representation of steps embedded in Stage 1 is shown in Figure 23.

In EnergyPlus, model input data are supplied by two ASCII (text) files: the Input Data Dictionary (IDD) and the Input Data File (IDF).

All possible EnergyPlus classes, and a specification of the properties each class has, are defined in the IDD file. IDD file consists of a comma separated text by a semi column, where an entry defines a class of input objects and specifies all the data needed to model it. The IDD structure defines an

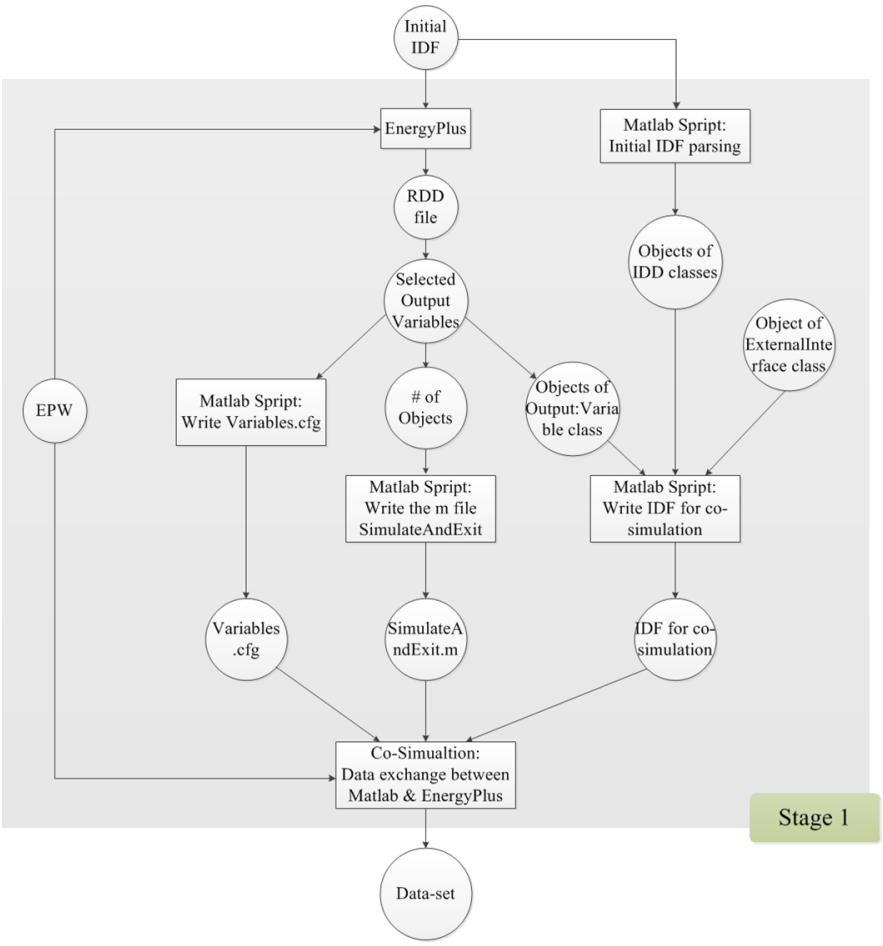

Figure 23: Stage 1 – Select objects — output variables of EnergyPlus — and run a simulation to receive simulated results

input object as a keyword. Each version of Energyplus has a different IDD file. Refer to [21] for a full description of the IDD. IDF file [22] consists of all the necessary IDD classes' objects to properly define a thermal simulation model of a certain building. Each thermal simulation model has a different IDF file.

In order to utilize all this information, two Matlab scripts have been developed: the first script identifies the version of the IDF file, parses the appropriate IDD file, and creates a library ($MatlabIDD_{xx}$, where xx is the EnergyPlus version) of Matlab classes, corresponding to EnergyPlus classes; the second script identifies the version of a certain IDF file and parses this file conducing to $MatlabIDD_{xx}$ objects definition.

Beyond a wide variety of EnergyPlus output variables, particular variables can be reported depending on the actual simulation problem described in the IDF. The Report Data Dictionary (RDD) is a text file listing those variables available for reporting during the simulation of a certain IDF, including possible objects of the data-set required for a Zoning Reduction Approach execution. For instance, Fanger's predicted mean vote could not be reported, if People class objects for all zones have not been defined. Selecting a zone's level output variable from that list, an object of the *Output:Variable* class is defined and imported in the initial IDF. A plethora of zone's level output variables could exist in the RDD file, commonly including zone air temperature, zone operative temperature, zone relative humidity and zone Fanger Predicted Percentage of Dissatisfied people (Fangeer PPD) to name but a few. However from this point forward, we assume that the selected

output variable is the zone air temperature.

After an initial IDF — enriched with the selected *Output:Variable* — simulation run, the resulted data-set of the selected variable is printed in a comma separated text by a semi column, where each column corresponds to a unique zone's variable time-series (zone air temperature with reporting frequency equal to the simulation timestep). The order of columns is alphabetical, according to the names of the zones, and as such not allowing the user to set a preferable order. To achieve this, EnergyPlus could be used in conjunction with the Building Controls Virtual Test Bed.

To configure the data exchange between EnergyPlus and Matlab through BCVTB, the following four steps (described further in Section 5.4) are required:

- Enrich the intial IDF with the selected *Output:Variable* object and an object of the *External-Interface* class.
- Develop an xml file, named variables.cfg, that defines the mapping between EnergyPlus and BCVTB variables.
- Create an m file, named simulateAndExit.m, to determine the data exchange between MATLAB and the BCVTB variables.
- Create a Ptolemy model.

In order to automatically create — when required — the enriched IDF, the variables.cfg and the simulateAndExit.m files, three respective Matlab scripts have been developed.

The data-set consisting of the simulated results (outputs of Stage 1) along with the desired number of zones, or the desired errors of zones' merging ε_1, ε_2 and the Koopman modes of interest number, are forwarded to Stage 2, where a Zoning Reduction Approach is chosen and is applied to receive groups of zones that will be merged (see Figure 24).

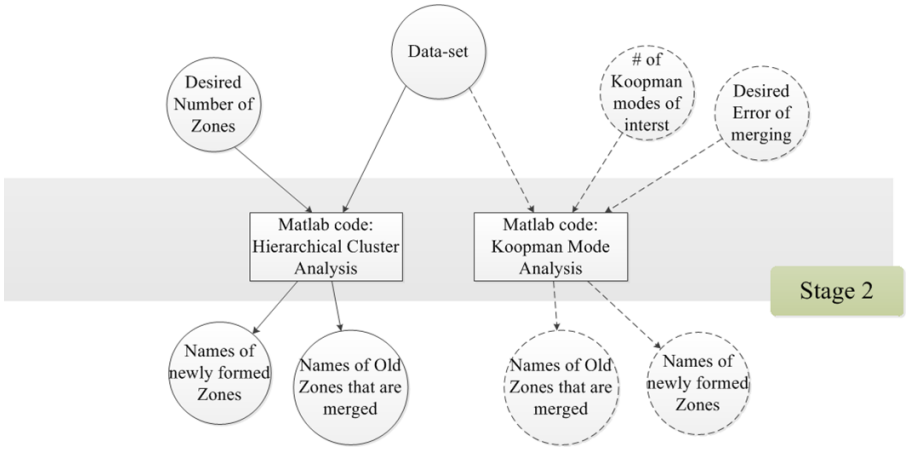

Figure 24: Stage 2 – Choose a method for Zoning Reduction to receive Zones of initial IDF that are merged

Names of newly formed zones and names of initial model's zones that belong to each newly formed zone, are finally used to automatically recreate the final speedup model in Stage 3 (see Figure 25), in the following manner:

- For each newly formed zone, a new object of the *Zone* class is determined.
- When zones are merged forming a new zone, shared walls and openings (doors, windows) are no more *BuildingSurface:Detailed* and *FenestrationSurface:Detailed* class objects, re-

spectively, but *InternalMass* class objects acting as thermal mass for the new zone. Moreover, internal loads from electric-equipment, lighting, and occupancy, objects of *ElectricEquipment*, *Lights* and *People* classes, respectively, are combined for the newly formed zone. With the initial IDF parsed objects of *BuildingSurface:Detailed*, *FenestrationSurface:Detailed*, *IntrnalMass*, *ElectricEquipment*, *Lights* and *People* classes as inputs, a Matlab script has been developed and is applied to properly determine new objects of these classes.

- For each new object of *Zone*, *BuildingSurface:Detailed*, *FenestrationSurface:Detailed*, *IntrnalMass*, *ElectricEquipment*, *Lights* and *People* classes the respective class "write" method is called, implementing the writing operations performed on objects of the corresponding class to create the new IDF of merged zones.

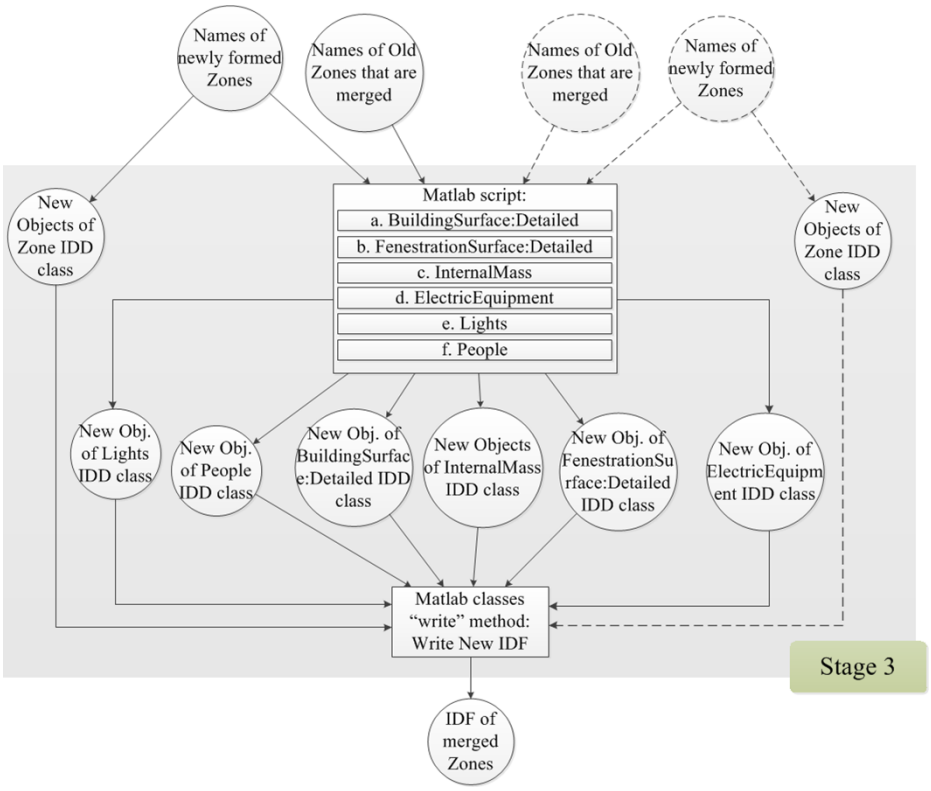

Figure 25: Stage 3 – Create new objects of appropriate IDD classes and write them to a new IDF of Grouped Zones

5.3 Implementation Notes - Inductive Approach

The inductive approach is implemented in Matlab and the model based on autoregressive Gaussian mixtures is represented as a Matlab object. Figure 26 shows the properties and methods of the object. The properties are very specific for the given architectures: `Ncomponents` stands for the number of Gaussian components, `order` for the order of the autoregression, `Mu` are centers of particular components, and `Sigmas` corresponding covariance matrices. Finally, `Priors` are priors of the components. More details on these concepts are provided e.g. in [17].

Method `train` estimates the parameters of the model using the expectation-maximization algorithm. Method `meanNext` calculate the expected value of the next state and method `simulateNext` samples from the distribution of the next state. The headers of the methods are very general and other architectures (e.g. multi-layer perceptrons)can be implemented easily.

```
classdef GMDynamicModel
    %GMDYNAMICMODEL Summary of this class goes here
    %   Gaussian Mixture Model for Dynamic Systems
    %   Copyright Honeywell spol. s r.o., Czech Republic
    %   Version 1.0

    properties
        Ncomponents;
        order;
        Mu;
        Sigma;
        Priors;
    end

    methods
        function obj = GMDynamicModel()
            addpathgm
        end
        function obj = train(obj,xnext,xpart,apart) ...
        function next = meanNext(obj,xpart,apart) ...
        function next = simulateNext(obj,xpart,apart) ...
    end
end
```

Figure 26: Matlab code - structure of the autoregressive Gaussian mixture.

The models are used by the `generalMethodology` which is a Matlab function that implements Pseudocode 1. The construction is intended to be integrated in the Application layer where it shall form a basis of context-free FDD.

```
function [optimalModel,...
    errorOnTesting]...
    = generalMethodology(...
                        xdata,...
                        addata)
```

Figure 27: Header of the general methodology function for the inductive simplification.

5.4 Integration of the Simplified Models within the Control Design Process

According to deliverable 5.3, the Control Design methodology towards designing efficient controllers relies on the availability of accurate enough models, able to predict the thermal state of the building. Additionally, the proposed optimization iterative algorithms have to to converge to a “good” controller after a finite number of iterations. Convergence-time heavily depends on the size of the problem at hand: for small problems convergence can be achieved after few iterations, while for more complex settings the number of iterations required increases. An increase in the number of iterations, allows algorithms to converge to better solutions, thus, simulation-time has to be minimum. To that direction, the reduced building thermal simulation models, derived from approaches described in the previous sections, are used.

Figure 28 presents the integration of the Control Design process within BaaS system and highlights the role of simplified models within this process.

To facilitate the integration of different controllers within the proposed Control Design approach, the controllers are defined using an actor-based library called Buildings Control Virtual Test Bed

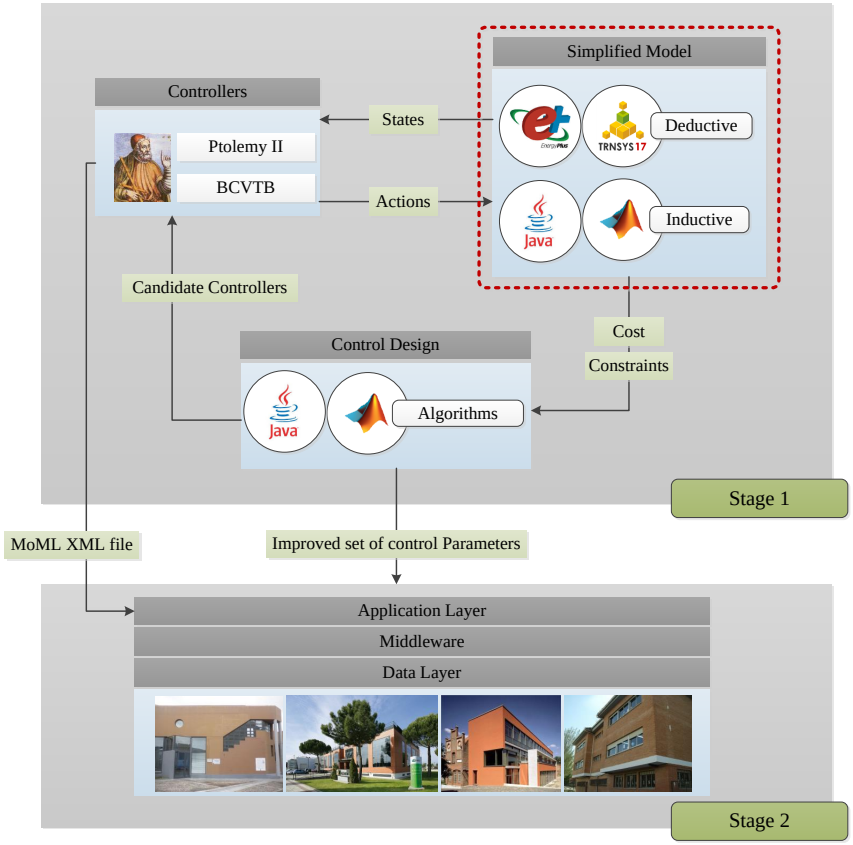

Figure 28: Integration of the Control Design and the Simplified Simulation Models within BaaS system

(BCVTB) [8], which is based on Ptolemy II [23] heterogeneous design and actor-oriented modeling open source environment.

Next, the defined controllers need to be coupled with the simplified simulation model.

In the final step, the Control Design process is setup, deployed and generates improved control parameters that are communicated to the building. Here, the properties of the surrogate-based stochastic optimization algorithm developed in Matlab and can be exported as jar files utilizing the Matlab Builder JA for Java export) are defined, such as the termination criterion and the bounds defining allowed values for the control parameters. Then Control Design process is initiated either in scheduled intervals either as a response to an event and communicates the optimized controller parameters to the building.

In the building side, the configuration of the control logic is performed semi-automatically, using the controller actors designed in BCVTB. This is possible since a Ptolemy model contains all necessary information required to execute the model (inputs, outputs, interconnection, execution intervals, etc.) in the MoML format, which is also the format used for configuring the control actors residing in the kernel. In practice, this means that an XML file in MoML format containing the description of the controllers designed in Ptolemy can be imported by the APO kernel during the configuration phase. This can ensure the consistency of the controller structure in both “simulation” and “real” worlds, as well as enable a laborious-free and less error-prone configuration of the controller functions and their data requirements on the building-side.

Facilitating the controllers/simulation models coupling, the simplified simulation zonal (developed in EnergyPlus or TRNSYS following deductive approaches) or data driven (developed in Matlab following inductive approaches or exported as jar files utilizing the Matlab Builder JA for Java export) models can be included as actors in BCVTB, as shown in Figure 29. Here, a Ptolemy model which is a Ptolemy II flow chart diagram is defined. Regarding the Simulator actor, configuring the flow chart diagram consists of three elements, enumerated in Figure 29:

1. *SDF Director*: In Ptolemy II, different models of computations can be used to define how the different actors interact with each other. The model of computation is defined by a director that needs to be included in the Ptolemy II flow chart diagram. For the BCVTB, the Synchronous Data Flow (SDF) Director is used.
2. *beginTime, timeStep, endTime*: These three parameters have units of seconds and needs to be equal to the start time, time step and final time that are used in the simulation program. These parameters are used to configure the SDF Director.
3. *Simulator actor*: The Simulator actor conducts the data exchange with the simulation program. Here, three types of data exchange need to be configured: (1) EnergyPlus - BCVTB; (2) TRNSYS - BCVTB; and (3) Matlab - BCVTB data exchange. The requirements' definition of such configurations is the main topic of this Section and described below.

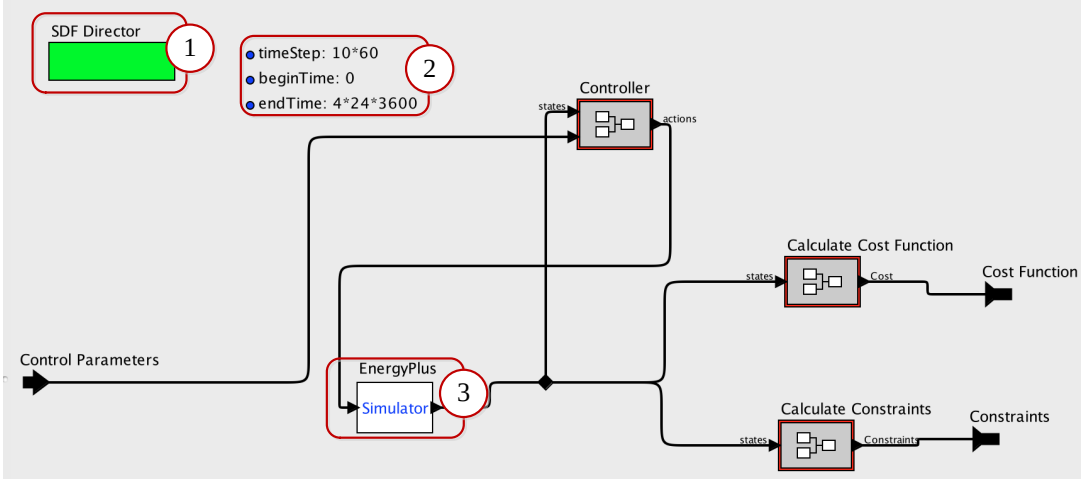

Figure 29: Integration of the Control Design and the Simplified Simulation Models within BaaS system

5.4.1 EnergyPlus – BCVTB data exchange

To configure the data exchange between EnergyPlus and BCVTB, the following three steps are required:

- Enrich the simplified IDF with the essential objects of the variables we would like to exchange.
- Develop an xml file, named variables.cfg, that defines the mapping between EnergyPlus and BCVTB variables.
- Define a Simulator actor to the Ptolemy II model.

IDF file

With the simplified IDF at hand, modification of it must be performed in order to achieve the EnergyPlus — BCVTB data exchange. The first object needs to be filled is an *ExternalInterface* class object, in which the name property is set to PtolemyServer so as to activate the BCVTB. Objects of the *Output:Variable* class can be used to send data from EnergyPlus to the BCVTB, while objects of the following IDD classes can be used to receive data from the BCVTB, at each zone time step:

1. ExternalInterface:Schedule;
2. ExternalInterface:Actuator;
3. ExternalInterface:Variable.

XML file

The data mapping between EnergyPlus thermal model and the BCVTB is defined in an xml file called variables.cfg. This file needs to be in the same directory as the EnergyPlus idf file.

The file begins with the following line:

```
<?xml version="1.0" encoding="ISO-8859-1"? ><!DOCTYPE BCVTB-variables SYSTEM "variables.dtd">
```

<BCVTB-variables> and < /BCVTB-variables> define the start and the end of the xml file, respectively. In between them, we need to specify how the exchanged data is mapped to EnergyPlus objects. They contain all the elements which define the variable mapping. The order of the elements which are defined between them matters and it needs to be the same as the order of the elements in the input and output signal vector of the BCVTB actor that calls EnergyPlus.

The exchanged variables are declared in elements that are called "variable" and have a corresponding source. The BCVTB can send data to objects of three classes, *ExternalInterface:Schedule*, *ExternalInterface:Actuator* and *ExternalInterface:Variable*. For these objects, the source needs to be set to Ptolemy, because they are computed in Ptolemy. The xml elements for these objects look as follows:

For the ExternalInterface:Schedule objects, NAME needs to be the same as the EnergyPlus schedule name.

```
<variable source="Ptolemy"><EnergyPlus schedule="NAME"/ >< /variable>
```

For the ExternalInterface:Actuator objects, NAME needs to be the same as theEnergyPlus actuator name.

```
<variable source="Ptolemy"><EnergyPlus actuator="NAME"/ >< /variable>
```

For the ExternalInterface:Variable objects, NAME needs to be the same as the EnergyPlus Energy Runtime Language variable name.

```
<variable source="Ptolemy"><EnergyPlus variable="NAME"/ >< /variable>
```

As we discussed above, the BCVTB can also read data from any Output:Variable object. For each object, the source attribute is set to EnergyPlus, because it is computed by EnergyPlus. The xml elements for each object look as follows:

```
<variable source="EnergyPlus"><EnergyPlus name="NAME" type="TYPE"/ >< /variable>
```

Output:Variable: NAME needs to be the same as the EnergyPlus “Variable Name” and TYPE needs to be the same as the corresponding EnergyPlus “Key Value”.

Simulator actor – EnergyPlus

A Simulator actor is used to conduct the data exchange with EnergyPlus. The parameters of the Simulator actor are as shown in Figure 30 and a short description is provided below:

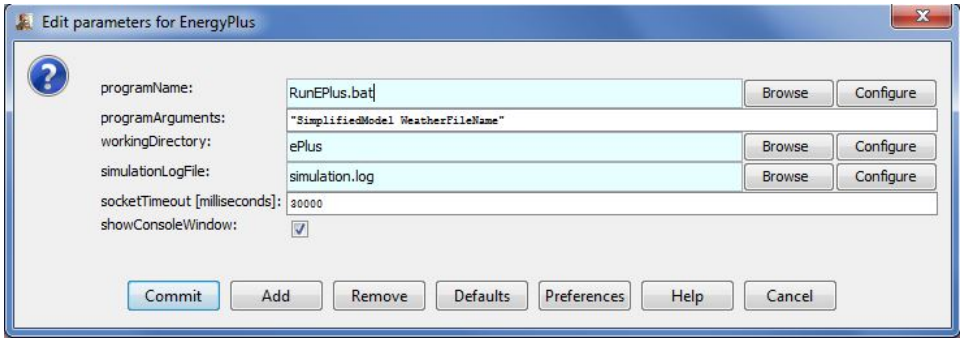

Figure 30: Simulator actor – EnergyPlus and its parameters

- programName: The name of the executable that starts the simulation is RunEPus.bat.
- programArguments: Arguments needed by EnergyPlus. In this field we set the idf file name (SimplifiedModel) and the weather file name (WeatherFileName).
- workingDirectory: Working directory of EnergyPlus is ePlus folder.
- simulationLogFile: Name of the file to which the BCVTB will write the console output and error stream that it receives from EnergyPlus. This file typically shows what may have caused an error.
- socketTimeout: Time out in milliseconds for the initial socket connection. At the start of the simulation, the BCVTB waits for EnergyPlus to connect through a socket connection to the BCVTB. If EnergyPlus does not connect within the here specified time, the BCVTB will stop with an error.

5.4.2 Matlab – BCVTB data exchange

To configure the data exchange between Matlab and BCVTB, the following three steps are required:

- Create an m file, named simulateAndExit.m preferably, to determine the data exchange between the MATLAB and the BCVTB variables.
- Define a Simulator actor to the Ptolemy II model.

M file

The simulateAndExit.m file has the following structure:

1. Initialize variables
2. Add path to BCVTB Matlab libraries:

```
addpath( strcat(getenv('BCVTB_HOME'),'/lib/matlab'));
```

3. Establish the socket connection:
```
sockfd = establishClientSocket('socket.cfg');
```
4. Exchange data (called at each timestep):
```
[retVal, flaRea, simTimRea, dblValRea] = ...
... exchangeDoublesWithSocket(sockfd, flaWri, nDblRea, simTimWri, dblValWri);
```
The input arguments are:
   - sockfd: Socket file descriptor.
   - flaWri: Communication flag to write to the socket stream. It is set to zero for normal operation, or to a negative value to stop the exchange.
   - nDblRea: Number of double values which will be read.
   - simTimWri: Current simulation time in seconds to write to BCVTB.
   - dblValWri: Vector of double values to write to BCVTB.

   The return values are:
   - retVal: Has to be a non negative value if the data exchange was successful or a negative value if an error occurs.
   - flaWri: communication flag read from the socket stream. A negative value indicates that the BCVTB will stop due to an error and not send any more data. A value equal to zero indicates normal operation and a value equal to one means that the final simulation time has been reached and no more data will be exchanged.
   - simTimRea: current simulation time in seconds read from the socket.
   - dblValRea: vector of double values read from the socket.
5. Close socket at the end of the simulation:
   closeIPC(sockfd);
6. Exit Matlab:
   exit

Simulator actor – Matlab

A Simulator actor is used to conduct the data exchange with Matlab. The parameters of the Simulator actor are shown in Figure 31 and a short description is provided below:

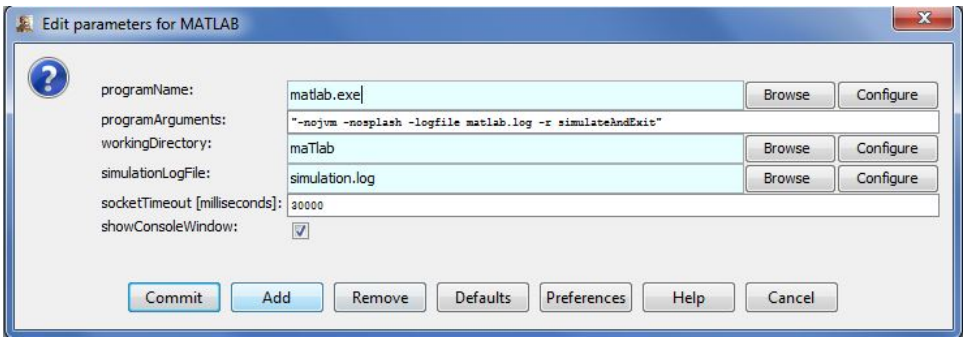

Figure 31: Simulator actor – Matlab and its parameters

- programName: The name of the executable that starts the simulation is matlab.
- programArguments: Arguments needed by the simulation. Text arguments need to be enclosed in apostrophes.
- workingDirectory: Working directory of Matlab is maTlab folder.
- simulationLogFile: Name of the file to which the BCVTB will write the console output and error stream that it receives from Matlab. This file typically shows what may have caused an error.
- socketTimeout: Time out in milliseconds for the initial socket connection. At the start of the simulation, the BCVTB waits for Matlab to connect through a socket connection to the BCVTB. If EnergyPlus does not connect within the here specified time, the BCVTB will stop with an error.

5.4.3 TRNSYS - BCVTB data exchange

Any TRNSYS simulation model is developed in the Simulation Studio (user interface of TRNSYS). The Simulation Studio creates not only the trnsys project file (tpf), but also the basic input file, named deck file (dck), a text file that contains all the information on the simulation but no graphical information. To configure the data exchange between TRNSYS and BCVTB, the following three steps are required:

- Enrich the simplified tpf with the BCVTB component (Type 6666).
- Create the dck file
- Define a Simulator actor to the Ptolemy II model.

BCVTB compontent and dck file

The BCVTB component (Type 6666) controls how the variables are communicated between TRNSYS and the BCVTB. There are 3 parameters need to be defined: the number of variables passed to the BCVTB; the number of variables received from the BCVTB; and the number of TRNSYS timesteps per data exchange with the BCVTB. For the data mapping between TRNSYS thermal model and the BCVTB, the usual TRNSYS linking process is used.

Once the BCVTB component is added to the tpf file, the dck file for the project must be created. The BCVTB uses the dck file directly to run the TRNSYS simulation and not the Studio project file (tpf file). The dck file is written by the pen icon on the left side of the Simulation Studio window.

Simulator actor – TRNSYS

A Simulator actor is used to conduct the data exchange with TRNSYS. The parameters of the Simulator actor are shown in Figure 32 and a short description is provided below:

- programName: The name of the executable that starts the simulation is runTRNSYS.bat.
- programArguments: Arguments needed by TRNSYS. In this field we set the dck file name (SimplifiedModel).
- workingDirectory: Working directory of TRNSYS is TrnSyS folder.

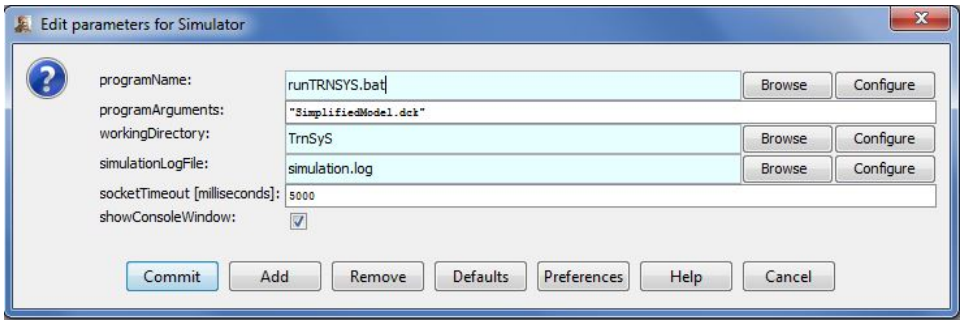

Figure 32: Simulator actor – TRNSYS and its parameters

- simulationLogFile: Name of the file to which the BCVTB will write the console output and error stream that it receives from TRNSYS. This file typically shows what may have caused an error.
- socketTimeout: Time out in milliseconds for the initial socket connection. At the start of the simulation, the BCVTB waits for TRNSYS to connect through a socket connection to the BCVTB. If TRNSYS does not connect within the here specified time, the BCVTB will stop with an error.

6 Conclusions

6.1 Achievements

This deliverable has provided in Sections 3 and 4 alternative ways how to simplify complex BEMS models in order to provide optimal control and advanced diagnostic services.

The deductive approaches work with rich context information and simplify the models in terms of spacial hierarchy using representative geometric patterns or aggregation of zones with similar behavior. The inductive approach offers a way how to learn models from data. Both approaches has demonstrated on several data-sets to accelerate the execution of the models while the accuracy has been assured. The current status of deployment of the considered models is summarized in the following table:

Table 15: Current status of the deployment. Yes - discussed in this document. Upcoming - not yet discussed.

Pilot	geometry	zoning	GM simulated	GM real
FIBP	yes	yes	yes	upcoming
TUC	yes	yes		
CARTIF	yes	yes		
SE	yes*			

* Covered in D4.2 already.

6.2 Applicability of the developed methods - summary

The deliverable provides several methods how to simplify a BEMS model and Section 5 deeply discusses how they are implemented and integrated in the BaaS platform. To summarize this discussion, Table 16 shows basic properties of these methods

Table 16: Primary applicability of deductive and inductive approaches

Approach	Deductive	Inductive
Primary use	Model-based control	Monitoring
Key functionality	Fast long term simulations	Filtration of the current state
Integrated within	Ptolemy	Application layer

In order to depict their applicability in more detail as well as in wider context and future possibilities, we provide Figure 33. The blue boxes correspond to the case where the prior information is either unavailable or expensive to be integrated. After the data collection is established, it is possible to use the data to train a simplified model based on Gaussian mixture, as described in Section 4. Consequently, these models are used for the context-free FDD based on monitoring of abnormal values. The context-free FDD is specified in deliverable D5.2. If there is no anomaly detected, then advanced control methods, as described in D5.3 can be used. Note that if the trained model is sufficiently precise, it is possible to use algorithms for model-based control. If some anomalies are detected, the application of advanced control mechanism has to be postponed. The anomalies are reported and after some maintenance actions are carried out, the FDD can be launched again.

When the prior information is available and the building owner is willing to invest in the configuration of a more complex solution, then more options are available. This situation is depicted by the orange boxes in Figure 33. The very first use of the prior knowledge is the application of the rule-based FDD that can detect some extra anomalies and classify the building as irrelevant for advanced control methods. The prior information serves also as an important input to the BEMS modeling as well as to the deductive model reduction methods. The geometric reduction requires a building with repeating structural patterns, the time series clustering is more general, but also more complex in terms of computational time. Note that the deductive reduction of the model can be useful also for the inductive training since it can significantly reduce the dimension of the problem. Another option is to use the data from BEPS simulation as an input to the training of inductive models, as depicted in Figure 33 by the dashed line.

From this discussion, it is obvious that both deductive and inductive approaches have strong and clear relationships to the rest of the BaaS platform. Moreover, the developed simplification methods shall are rather complementary than competing. This is markable from the fact that there is the option to combine them. However, their primary application is separate. As already discussed in Section 5, the deductive approaches are fully connected to the Ptolemy platform while the inductive learning is considered as a part of the Application layer.

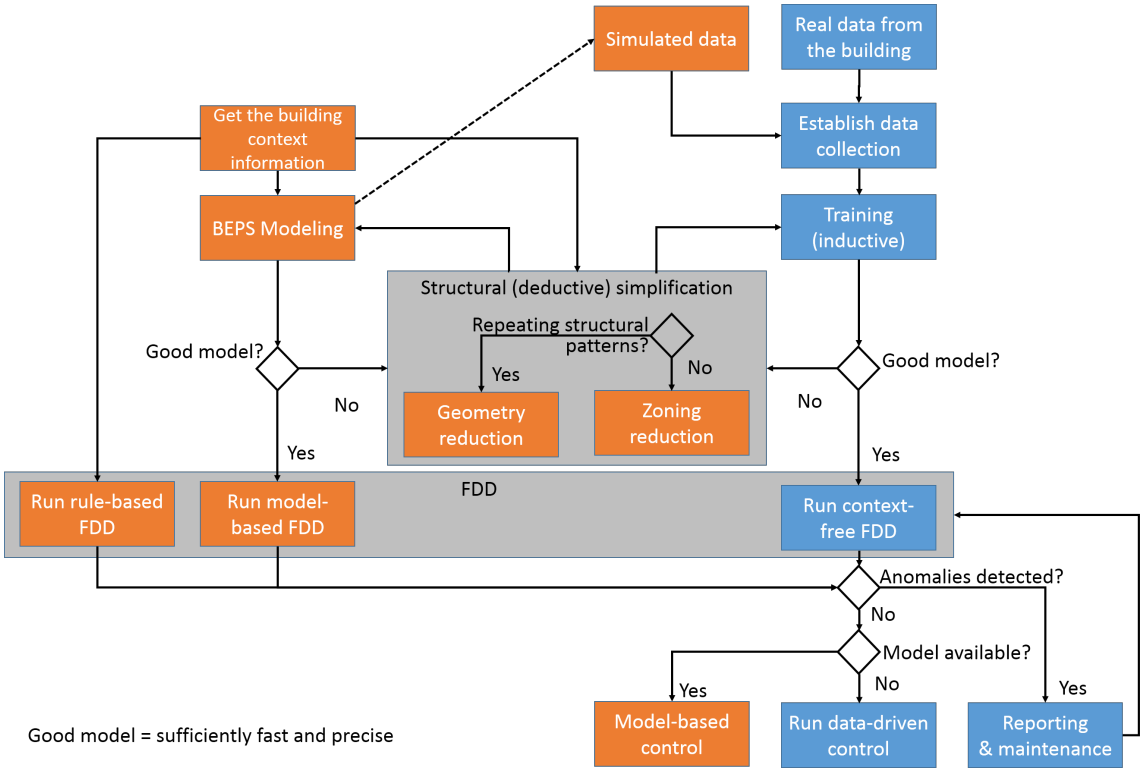

Figure 33: Relations of the simplification methods.

6.3 Intended Further Development

The deliverable provides recent information about the status of the development and the team intends to elaborate on some topics in the short term, namely:

- To test the prediction quality of the models that are obtained in the inductive way from real data.

- To integrate the simplified models obtained in the inductive way as an essential part of the context-free FDD as discussed in D5.2.

6.4 Other Possible Development

There are also some other ideas on the further development, but they are not intended to be addressed:

- To examine combination of deductive and inductive approaches. For instance, to use the hierarchical model reduction for dimension reduction and consequently the modeling using Gaussian mixtures.
- To investigate deeply and quantitatively other considered architectures for the inductive learning than Gaussian mixtures, e.g. ensemble neural networks.
- To check whether the time-series clustering leads to similar results as the geometric simplification.
- To use the inductive way to train the decision rules by optimal weighted learning as described in [24].

References

[1] ASHRAE Handbook, "Fundamentals," *American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., Atlanta GA*, 2009.

[2] A. Droscher, H. Schranzhofer, A. C. Juan Rodriguez Santiago, R. Streblow, D. Muller, N. Exizidou, G. Giannakis, and D. Rovas, "Integrated Thermal Simulation Models for the Three Buildings." PEBBLE Deliverable 2.1, 2010.

[3] K. Hottges, J. Santiago, M. Garcia, G. Giannakis, G. Kontes, D. Rovas, C. Valmaseda, O. Hidalgo, and J. Martin, "Development of simulation models for the pilot buildings." BaaS Deliverable 4.2, 2014.

[4] G. Giannakis, G. Kontes, E. Kosmatopoulos, and D. Rovas, "A model-assisted adaptive controller fine-tunning methodology for efficient energy use in buildings," in *Mediterranean Conference on Control & Automation*, June 2011, pp. 49 –54.

[5] G. Kontes, G. Giannakis, E. Kosmatopoulos, and D. Rovas, "Adaptive Fine-Tuning of Building Energy Management Systems Using Co-Simulation," in *IEEE Multi-Conference on Systems and Control*, 2012, pp. 1664–1669.

[6] T. Hong, F. Buhl, and P. Haves, "Energyplus run time analysis." Lawrence Berkeley National Laboratory, 2009.

[7] DOE, "Energyplus: Engineering reference." United States Department of Energy, California, 2014.

[8] M. Wetter, "Co-simulation of building energy and control systems with the building controls virtual test bed," *Journal of Building Performance Simulation*, vol. 4, no. 3, pp. 185–203, 2011.

[9] O. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*. Springer, 2005, vol. 2.

[10] I. Mezić, "Spectral properties of dynamical systems, model reduction and decompositions," *Nonlinear Dynamics*, vol. 41, no. 1-3, pp. 309–325, 2005.

[11] M. Georgescu and I. Mezić, "Building energy modeling: A systematic approach to zoning and model reduction using koopman mode analysis," *Energy and Buildings*, vol. 86, pp. 794–802, 2015.

[12] Y. Susuki and I. Mezic, "Nonlinear koopman modes of coupled swing dynamics and coherency identification," in *Power and Energy Society General Meeting, 2010 IEEE*. IEEE, 2010, pp. 1–8.

[13] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+ 1) evolutionary algorithm," *Theor. Comput. Sci.*, vol. 276, no. 1-2, pp. 51–81, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1016/S0304-3975(01)00182-7

[14] C. Igel and B. Sendhoff, "Genesis of organic computing systems: Coupling evolution and learning," in *Organic Computing*. Springer, 2008, pp. 141–166.

[15] W. Lu and I. Traore, "A novel evolutionary clustering algorithm based on gaussian mixture model," in *Proceedings of the 10th WSEAS international conference on Computers*. World Scientific and Engineering Academy and Society (WSEAS), 2006, pp. 686–691.

[16] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," *Information Sciences*, vol. 191, pp. 192–213, 2012.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[18] J. Remund, S. Kunz, and R. Lang, "Meteonorm-global meteorological database for solar energy and applied climatology," *Solar Engineering Handbook, version*, vol. 4, 1999.

[19] M. A. Figueiredo, "On gaussian radial basis function approximations: Interpretation, extensions, and learning strategies," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 2. IEEE, 2000, pp. 618–621.

[20] H. Dibowski, J. Ploennigs, and K. Kabitzsch, "Automated design of building automation systems," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3606–3613, 2010.

[21] DOE, "Energyplus: Guide for interface developers." United States Department of Energy, California, 2013.

[22] ——, "Energyplus: Input-output refernce." United States Department of Energy, California, 2014.

[23] C. Brooks, E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng, "Heterogeneous Concurrent Modeling and Design in Java (Volume 1: Introduction to Ptolemy II)," EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M05/21, Jul 2005. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/9571.html

[24] K. Macek, D. Rovas, M. Schmidt, and C. Valmaseda, "Holistic optimization of hvac systems via distributed data-driven control," in *Proceedings of ISA Conference*, Lisbon, Portugal, July 2014.