SEVENTH FRAMEWORK PROGRAMME
THEME – Energy Efficient Buildings

EeB-ICT 2011.6.4. ICT for energy-efficient building and spaces of public use

Self learning Energy Efficient builDing and open Spaces

GA No. 285150

D2.8 – Energy Control Strategy. First Release

Work Package	WP2 – System behaviour models and library generation		
Task	Task 2.6 – Development of energy control strategy		
Revision	0		
Due date	2013/05/31	**Submission date**	2013/07/19
Dissemination level	PU	**Deliverable type**	R

Authors	Ulrich Donath (FhG), Jürgen Haufe (FhG), Rui Esteves (UiS), Tomasz Wlodarczyk (UiS), Shane Montague (USAL), Sunil Vadera (USAL)
Verification	Noemi Jimenez Redondo (CEMOSA)
Approval	Noemi Jimenez Redondo (CEMOSA)

Table of Contents

List of Figures

List of Tables

1 Introduction

The objective of this report is the presentation of the energy control strategy that is used to meet the SEEDS project's goal of minimizing energy consumption by building equipment under ensuring comfort conditions for residents. Basics for the control strategy are the SEEDS BEMS architecture specification with definition of the controller core components: Building Model, Optimizing, Self-learning, Archiving, WISAN communication, and Front Panel. These predefinitions are outlined in the deliverable D2.3 [1] in the scope of WP2 "System behaviour models and library generation" and are summarized in section 2 of this report. Besides, the relations to the process controls are given here. In section 3 the SEEDS's approach of process control is presented. Here, the Building Model which is used to calculate the energy consumption is detailed. It follows the process control with Control Loop, Optimizing, and Self-learning. Data management and interfaces of the components are outlined under two aspects: interactions between the controller internal components and interactions between Controller, Archive, WISAN, and Front Panel.

For the explanations of the process control we use the building example Helicopter Garage (figure 1) which was introduced in deliverable D2.3 [1]. This example is not concerned with the pilots of SEEDS. But, all data are given for it and its size is low enough so that a model based check of the process control can be carried out. The building comprises 10 rooms: helicopter hall, various offices, kitchen, dining room, gym, and garage. The indoor conditions are determined by 1 heat pump, 1 chiller, 1 boiler, and 12 fan coils. The task of process control shall be the energy minimization under unsuring comfort temperatures of the helicopter hall, a selected office room and the dining room. In the control it will be included the occupancy of the rooms, the outside temperature, and the solar intensity.

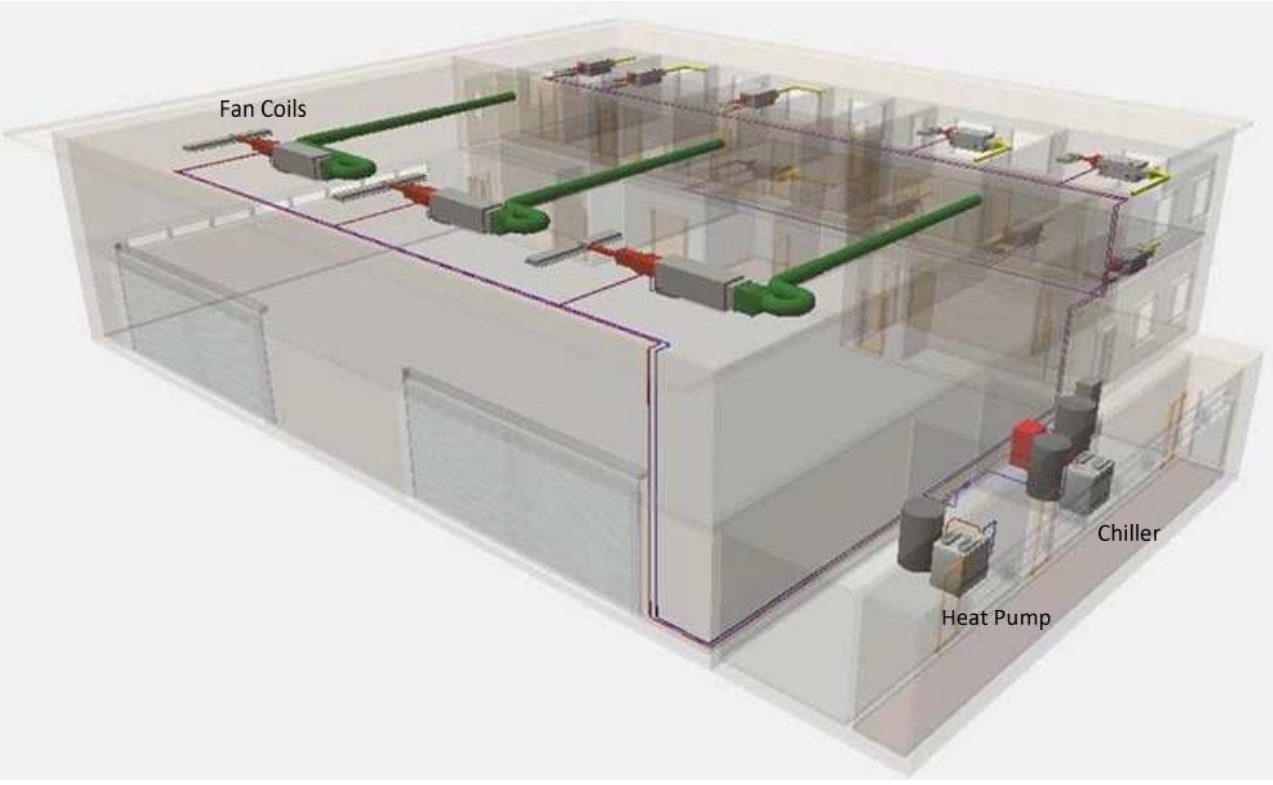

Figure 1: Architecture of Helicopter Garage

A test implementation of the energy control strategie is reported in section 4. This implementation is exemplified for Helicopter Garage. For the check of the controls a physical model of Helicopter Garage is built which runs in interaction with the process control.

The explanations of the process control are carried out by means of ordinary bubble charts and block diagrams and UML diagrams (class diagrams, composite structure diagrams, component diagrams, and activity diagrams) with the semantic of the UML standard [2][3]. The UML diagrams are drawn manually without support of UML software. Therefore no automatic code geneneration is possible.

2 Basics

2.1 Partitioning of the BEMS Architecture

The SEEDS's BEMS was initially structured in the functional groups: field control devices and communication networks, core of the BEMS, and graphical users interface (GUI).

From the viewpoint of the control hierarchy, these groups are considered in the following layers of the control system:

- **Control Layer**
  This Layer includes the core components of the BEMS which receives data from sensors in the building, computes and optimizes new control settings for facilities in the building, and sends these control settings to actuators in the building.
  The core components are:
  - **Building model** of all energy producing and consuming devices or facilities that is evaluated by a building model evaluator.
  - **Self-learning module** which forecasts the values of comfort (air temperatures, humidity, CO2 contents) and the energy consumption in the building.
  - **Optimizing module** that computes new control settings and evaluates these settings by an objective function and restrictions of comfort.

  These components interact closely and are combined in the component **Controller**.
- **Data Management Layer**
  This Layer includes a database which stores historical, present, and future data.
  Historical and present data are:
  - Sensor data, control settings, comfort settings, occupancy, weather conditions, comfort, and energy consumption.

  Future data are:
  - Future comfort settings, occupancy schedules, and weather forecasts.
- **Process Layer**
  This layer includes two components:
  - WISAN input and output, i.e., signals, which come from the sensors in the building and control settings, which are forwarded to the actuators associated with the facilities in the building.
  - Front panel that is a graphical user interface (GUI), which inputs e.g. comfort settings or occupancy schedules and outputs current comfort values and energy consumption. Furthermore dedicated device states are depicted on the front panel and certain set points of the control system are given via the front panel.

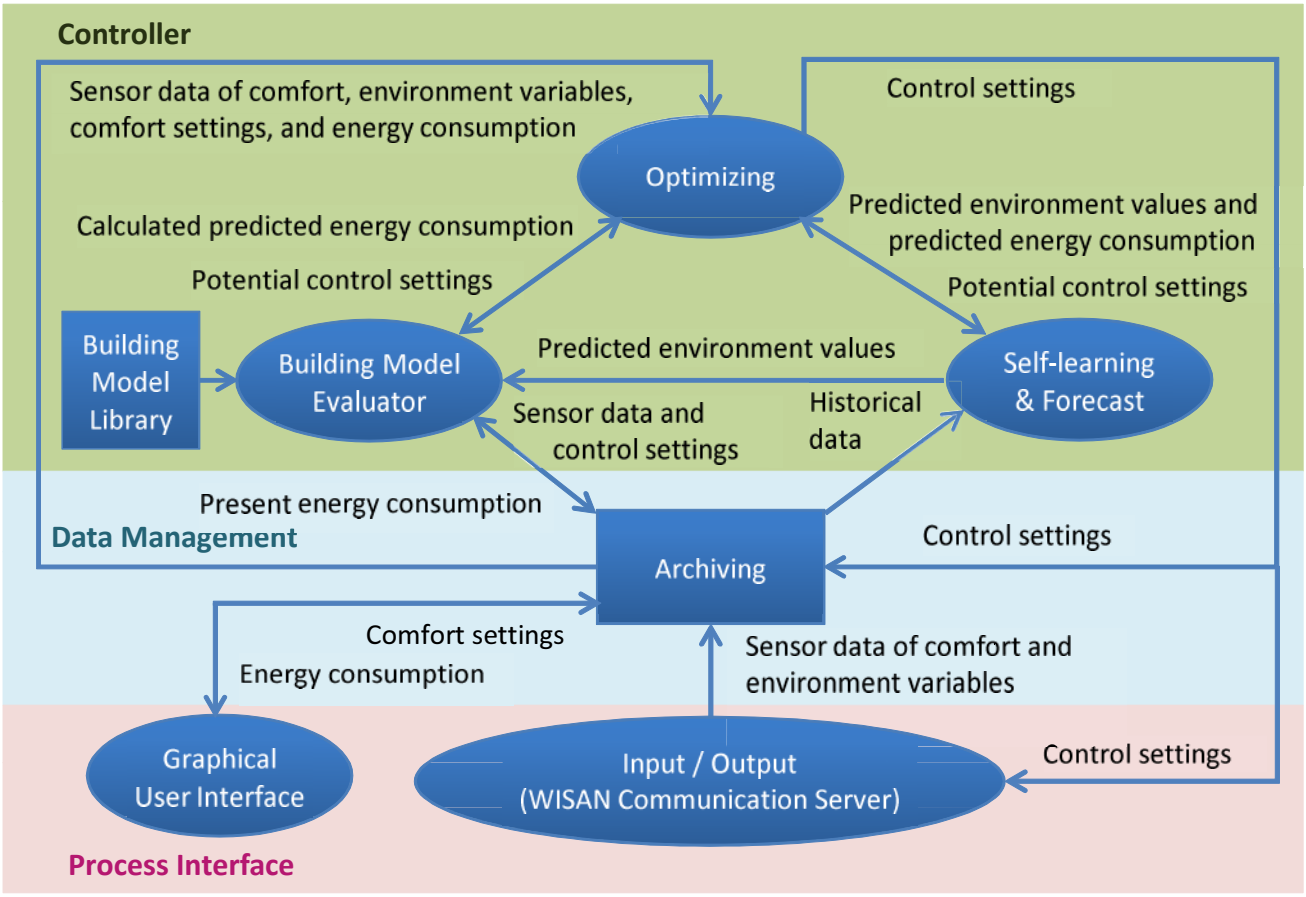

Figure 2: Layer architecture of SEEDS BEMS

2.2 Controller Core Components

The controller core components are the building model inclusive its evaluator, the self-learning module, and the optimizing module.

Theire tasks are:

- Building model

  This module models the energy elements in the building including energy sources, demands and storages. The building model is composed of components of a model library of all devices which are available to build the technical building equipment. It provides the total energy consumption of the building. The model evaluator computes the energy parameters of each device taking into account the characteristic of each device and relationships among them. The building model may be used in a twofold concern:
  - Firstly, calculation of the present energy consumption.
    In doing this, sensor data from the facilities and present control settings are taken into account.
  - Secondly, it is used to calculate the future energy consumption of the building.
    For that, forecasted comfort values predicted by self-learning and potential control settings computed by optimizing are used.

- Self-learning

  This module forecasts the values of those variables which are related to user comfort perception (such as room temperature or humidity). They will be called comfort variables within this document. In order to meet user comfort, such comfort variables should be within an acceptable comfort range.
  This is based on
  - present indoor conditions,
  - future weather conditions (i.e. outdoor temperature and solar intensity),
  - occupancy of the rooms in the given forecast period, and
  - potential control settings.
    The potential control settings are varied by the optimizing module which starts with the present control settings.

  The forecast is extended to internal variables, for instance supply water temperature of the cooling circuit, which determine the run of energy supplier or consumer. The self-learning module calls the building model evaluator which sums up the energy consumption over all included devices.

- Optimizing

  This module creates a set of potential control settings. These are sent to the self-learning module and the building model evaluator which predict the comfort variables and the energy consumption. The optimizer checks whether the comfort variables are within the accepted comfort range and whether the energy consumption is decreased. In the negative case, the potential control settings are discarded. In the positive case, they are registered and are used as reference for the next trial. Rejection of solutions that violate the comfort range is the severest penalty in a penalty technique [16] which is used for solving the constrained optimization problem. The optimizer stops if no further reduction of energy consumption can be found or a fixed number of trials is reached. Then the registered best control settings are outputted to the WISAN which sends the values to the actuators in the building.

2.3 Relations in Terms of Process Control

In the layer architecture of SEEDS BEMS, different variables in familiar terms are used. Table 1 gives the reference of the familiar denotation in figure 2 and the technical terminology as well the common used symbols. Figure 3 shows the relations in connection with a control loop. Figure 4 presents the architecture and data paths of the SEEDS's BEMS in this view.

Familiar Denotation	Technical Terminology	Symbol	Examples
comfort settings, set points of comfort	reference variables	w	feel-good temperature in the room
sensor data of comfort, comfort variables	process variables, controlled variables	x	temperature and humidity in the room
environment variables	disturbance variables, load variables	z	outdoor condition, occupancy
control settings	control variables	u	fan speed level, chiller load level
energy consumption	load variable	e	chiller's input of electrical energy

Table 1: Variables in SEEDS BEMS

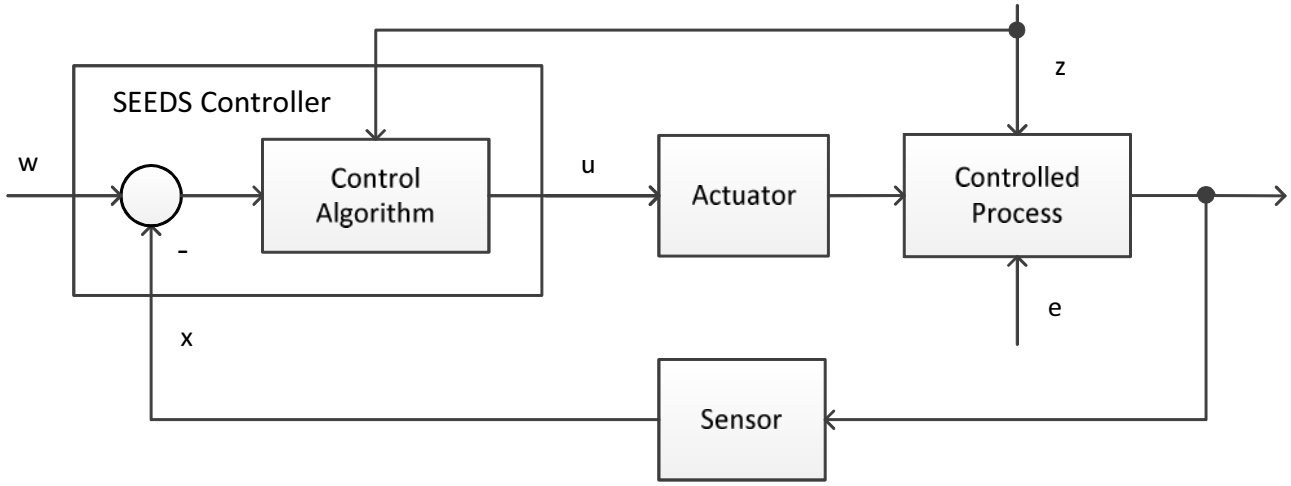

Figure 3: SEEDS BEMS in terms of process control

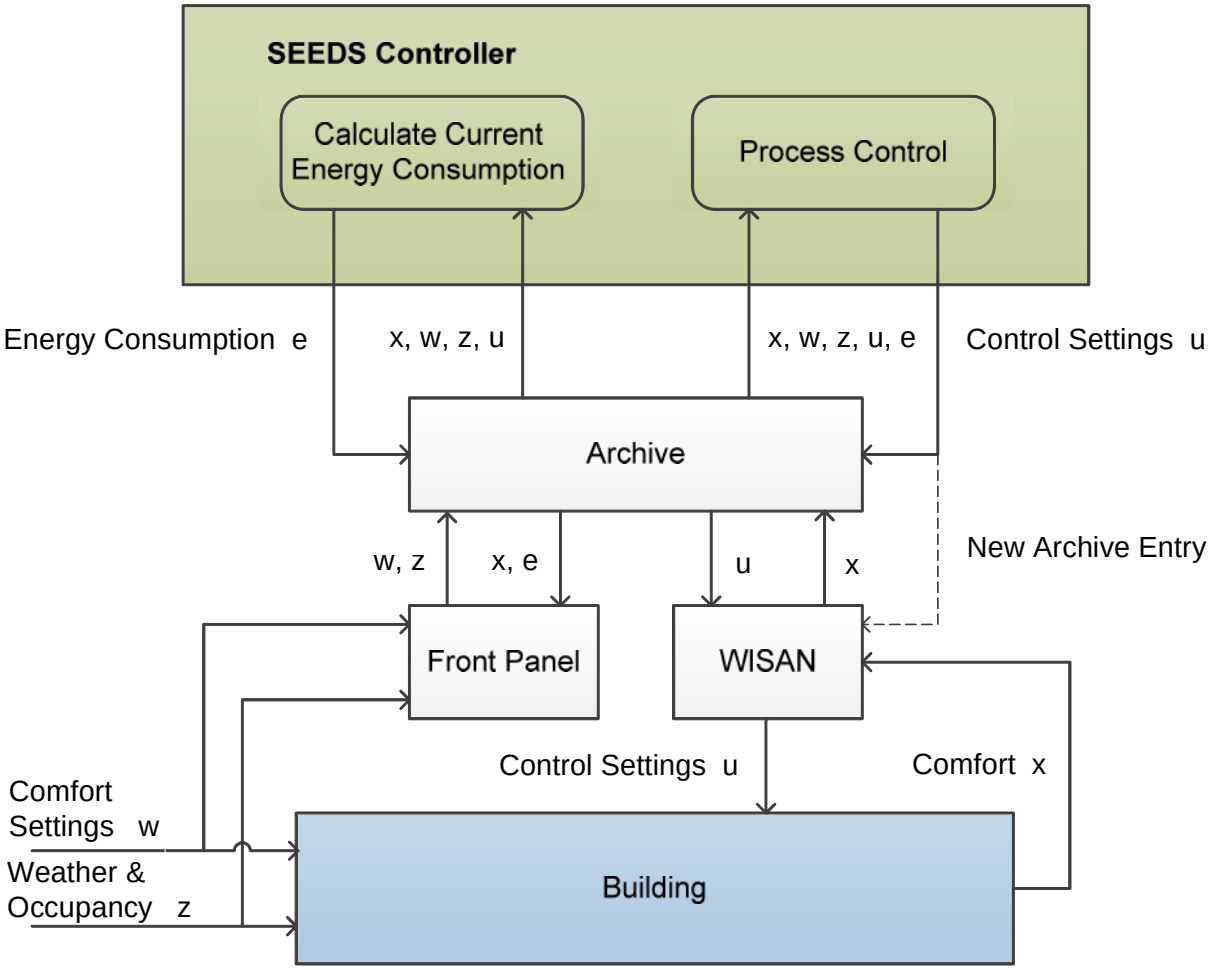

Figure 4: Architecture of SEEDS BEMS in logical view

3 Process Control and related Tasks

3.1 Continuous Calculation of Energy Consumption

3.1.1 Building Models

Each building model is represented by an instance of the *building model* class (figure 5). The *building model* class implements arrays and methods which allow the modeller to describe the structure of the technical equipment in the building [4]. That means all technical components for indoor conditioning and their connections are aggregated in the *building model.* The descriptions of these components are implemented in instances of the *component* class, which comprise the names and identifiers of the components, theire locations, the states, and the input and output ports. For deeper modelling, specialisations are made in *control component* and *flow component* classes.

The *control component* class is detailed in the subclasses

- *sensor*
- *controller*
- *actuator*.

The *flow component* class is detailed in the subclasses

- *energy conversion device* and further in *boiler, chiller, heat pump, …*
- *flow controller* and further in *valve, damper*
- *flow moving device* and further in *fan, pump*
- *flow storage device* and further in *tank*
- *composite element* and further in *fan coil, AHU.*

Control components and flow components can be directly linked. In addition, all components can be connected via ports which are associated with the components and used as intput or output. Furtermore, sensors can be associated with the ports to indicate temperature values or mass flow values of the media passing the ports.

For the calculation of the energy consumption there are two methods implemented in the *flow component* classes:

- *calc thermal_power( )*
- *calc_electrical_power( ).*

These calculations are carried out by means of measurements or data sheets and may be omitted if one of them is missing, e.g. electrical power of a tank. The first ones are expression statements, the latter ones are table-lookup algorithms. As an example the power calculation of a chiller is detailed in Table 2.

Figure 6 shows the declaration of the *flow component* class and its subclasses *chiller* and *air to water chiller*. Besides further specialized flow classes, *air to water chiller* is instantiated in the building model of Helicopter Garage, for instance. For the building model of Helicopter Garage, figure 7 depicts all class instances with their ports for the input and output of the media *air* and *water* and the required sensors for performing the calculation of their power consumption. The instances of *air to water chiller, air to water heat pump, boiler,* and *fan coil* are linked with instances of the *controller* class. These controllers realise the controls of their partners as On/Off Control with hysteresis or Proportional Control.

Chiller	Thermal Power	
	Measurement-based	Datasheet-based
Heating Mode	$P_{th} = \dot{Q}_w \times c_{pw} \times (T_{w\,out} - T_{w\,in})$	$P_{th} = f(T_{c\,in}, T_{w\,out}, EER)$
Cooling Mode	$P_{th} = \dot{Q}_c \times c_{pc} \times (T_{c\,out} - T_{c\,in})$	$P_{th} = f(T_{c\,in}, T_{w\,out}, COP)$
	Electrical Power	
	Measurement-based	Datasheet-based
Heating Mode		$P_{el} = f(P_{th}, T_{c\,in}, T_{w\,out}, EER)$
Heating Mode		$P_{el} = f(P_{th}, T_{c\,in}, T_{w\,out}, COP)$

Table 2: Power calculation methods for chiller

Q_w – Water mass flow
Q_c – Condenser mass flow
c_{pw} – Specific heat capacity of water at const. pressure
c_{pc} – Specific heat capacity of condenser at const. pressure
$T_{w\,out}$ – Temperature water out (water supply)
$T_{w\,in}$ – Temperature water in (water return)
$T_{c\,out}$ – Temperature condenser out
$T_{c\,in}$ – Temperature condenser in
EER – Energy Efficiency Ratio
COP – Coefficient of Performance

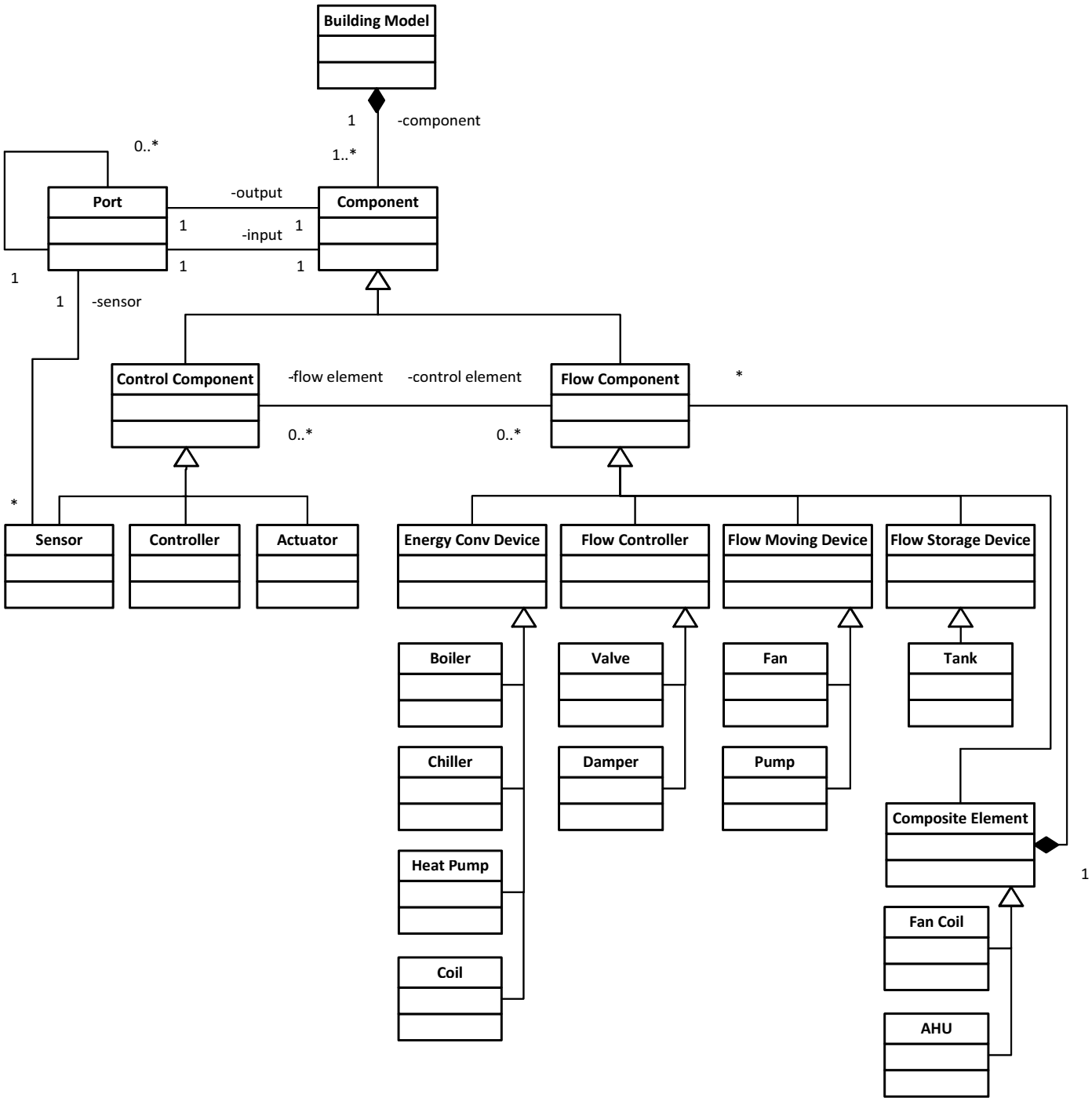

Figure 5: Class diagram of the technical building equipment

Figure 6: Class declarations of flow component, chiller and air to water chiller

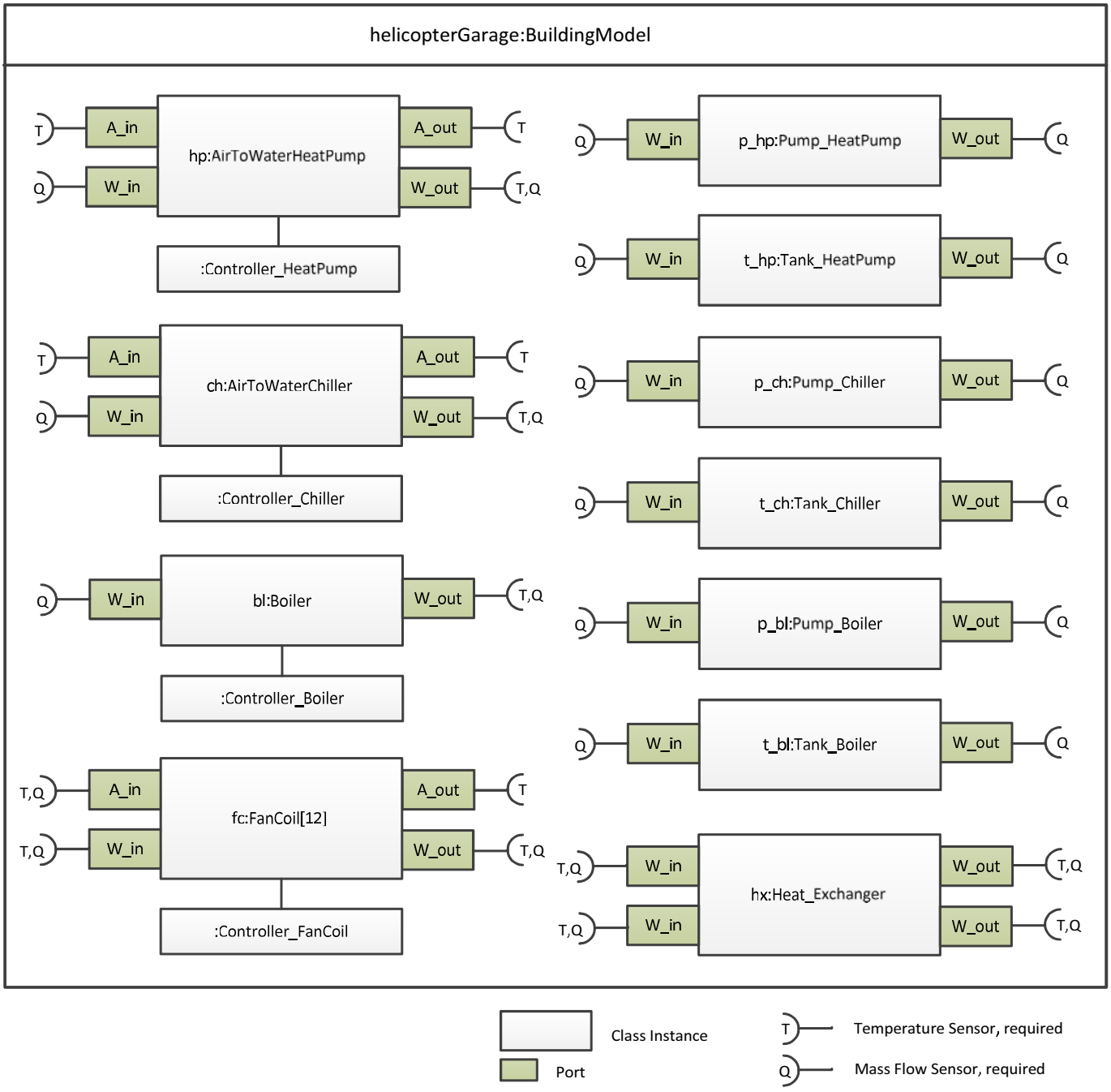

Figure 7: Composite structure diagram of the technical building equipment of Helicopter Garage

3.1.2 Building Model Evaluator

The Building Model Evaluator has the task to calculate the energy consumption of a building by using the building model instance, e.g. Helicopter Garage model according figure 7, and the present values of the process variables, environment variables, and control variables. This task is carried out periodically within the sample period Δt.

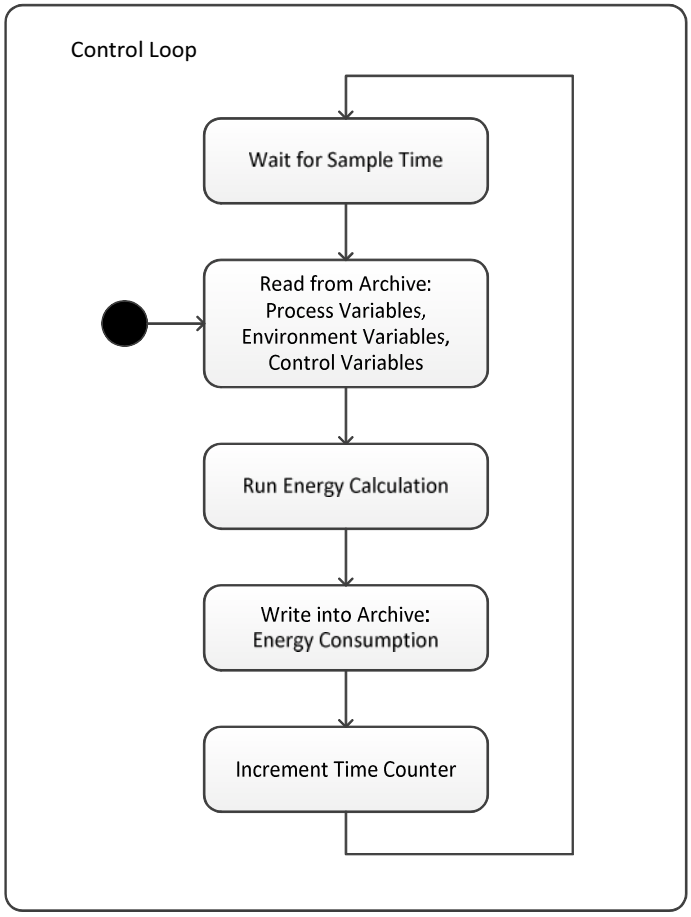

Figure 8: Control loop for calculation of current energy consumption

In the control loop (figure 8) the evaluator calls the power calculation methods (table 2) of all involved devices and computes the present thermal and electrical powers. These values are multiplied by Δt to obtain the consumed energy (figure 9).

In mathematical terms, for each device are valid:

$$E_{th}^{t_j} = P_{th}^{t_j} \times \Delta t$$

$$E_{el}^{t_j} = P_{el}^{t_j} \times \Delta t.$$

Summation over all devices results the total energy consumption of the building within Δt.

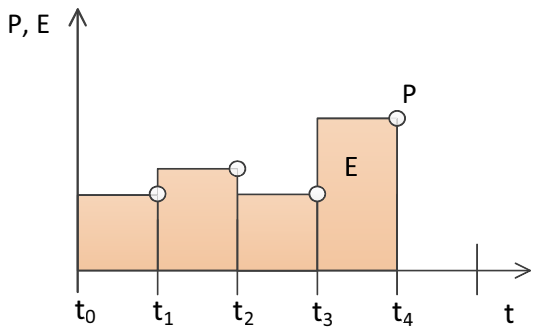

Figure 9: Power and energy calculations at sample times

For the entire observation period it results:

$$E_{th_{total}} = \sum_{j}\sum_{n} E_{th_n}^{t_j}$$

$$E_{el_{total}} = \sum_{j}\sum_{n} E_{el_n}^{t_j}$$

with n – device index and j – sample index.

The observation period can be a day, a year, or a forecast period, which are given as upper limit of the sample index.

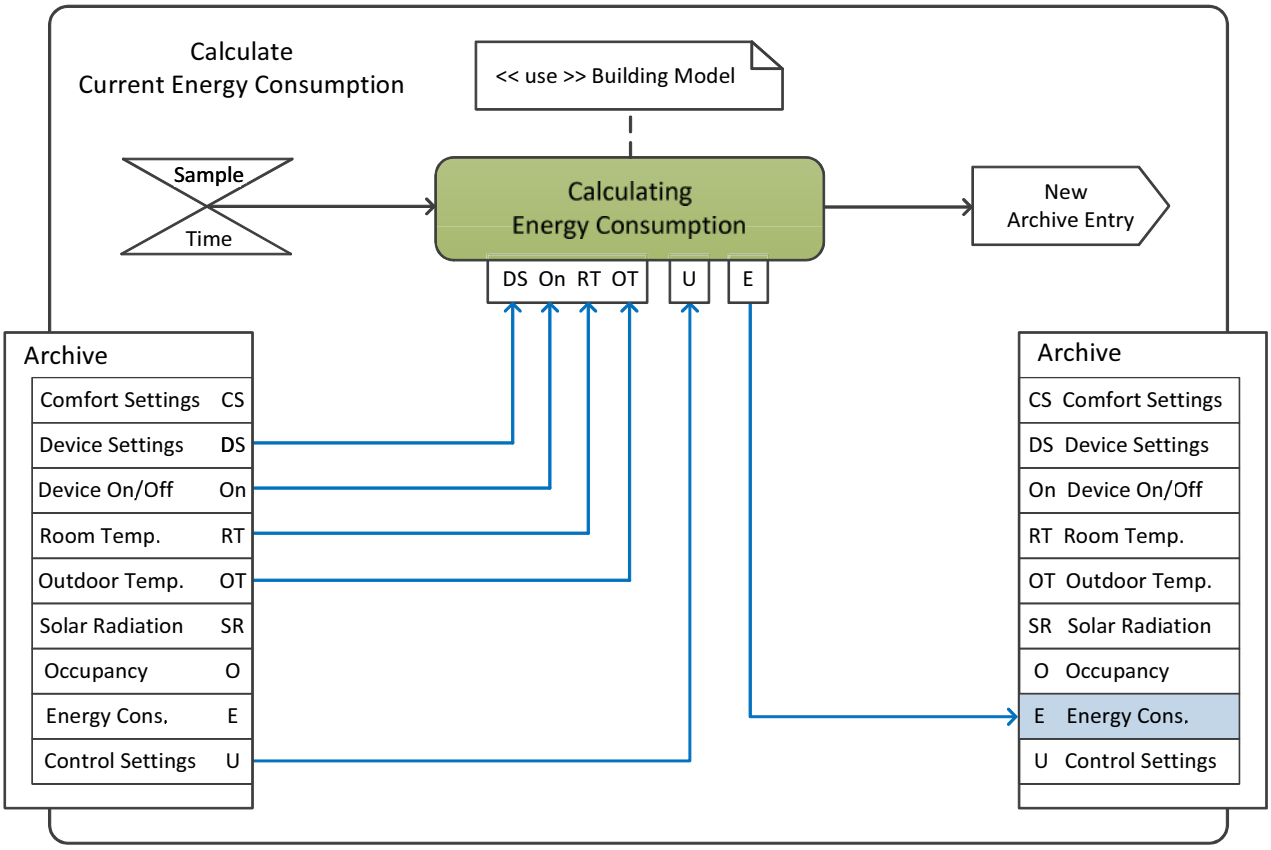

Figure 10: Activity diagram of calculation of current energy consumption

Figure 10 shows the relations of energy calculation to the parameters of Helicopter Garage given as archive entries.

3.1.3 Data Input, Output and Archiving

The data input and output for energy calculation is performed entirely via the rchive. The controller queries the archive periodically in the period Δt and then writes the calculated energy values in the archive (figure 11).

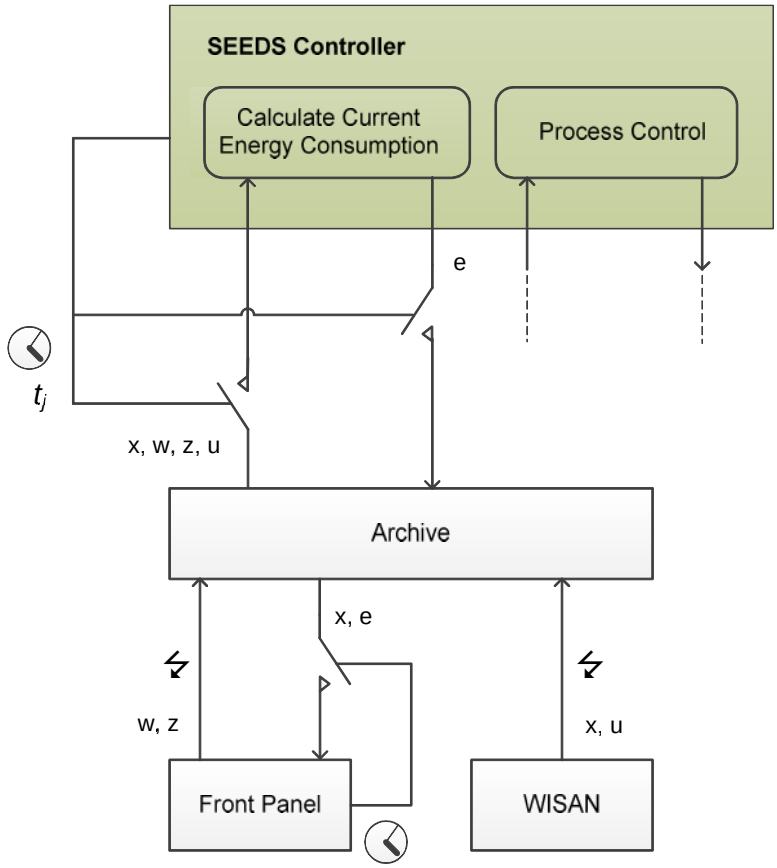

Figure 11: Data IO for energy calculation

Front Panel and WISAN execute the archive entries event-driven. This concerns the reference variables, disturbance variables, and process variables. The output to the users about the temperatures and humidities in the rooms and the current energy consumption as well its trend is performed by Front Panel time-controlled.

The data input and output via Front Panel and Archive is detailled in report [5] *D6.3 - Specification of Hardware and Software Platform.*

3.2 Process Control

3.2.1 Control Loop

3.2.1.1 Components, Interfaces, and Data Structures

Beside the continuous calculation of the present energy consumption, the SEEDS Controller fulfills the process control to optimize the energy consumption in the near future step-by-step. In Controller the components Building Model, Self-learning, and Optimizing are instantiated (figure 12) and their interactions are managed. In addition, a port is defined in Controller, through which the whole data traffic is handled with Archive. As Archive a relational database is used, which allows to store historical, present and future data. Historical and present data relate to process variables, control variables and energy consumption. Future data are weather forecasts or occupancy schedules, for example, which are inputed in Archive via Front Panel. The values of the present process variables are given by WISAN. To transfer the present values of control variables to to the building equipment a special connection is established between Controller and WISAN.

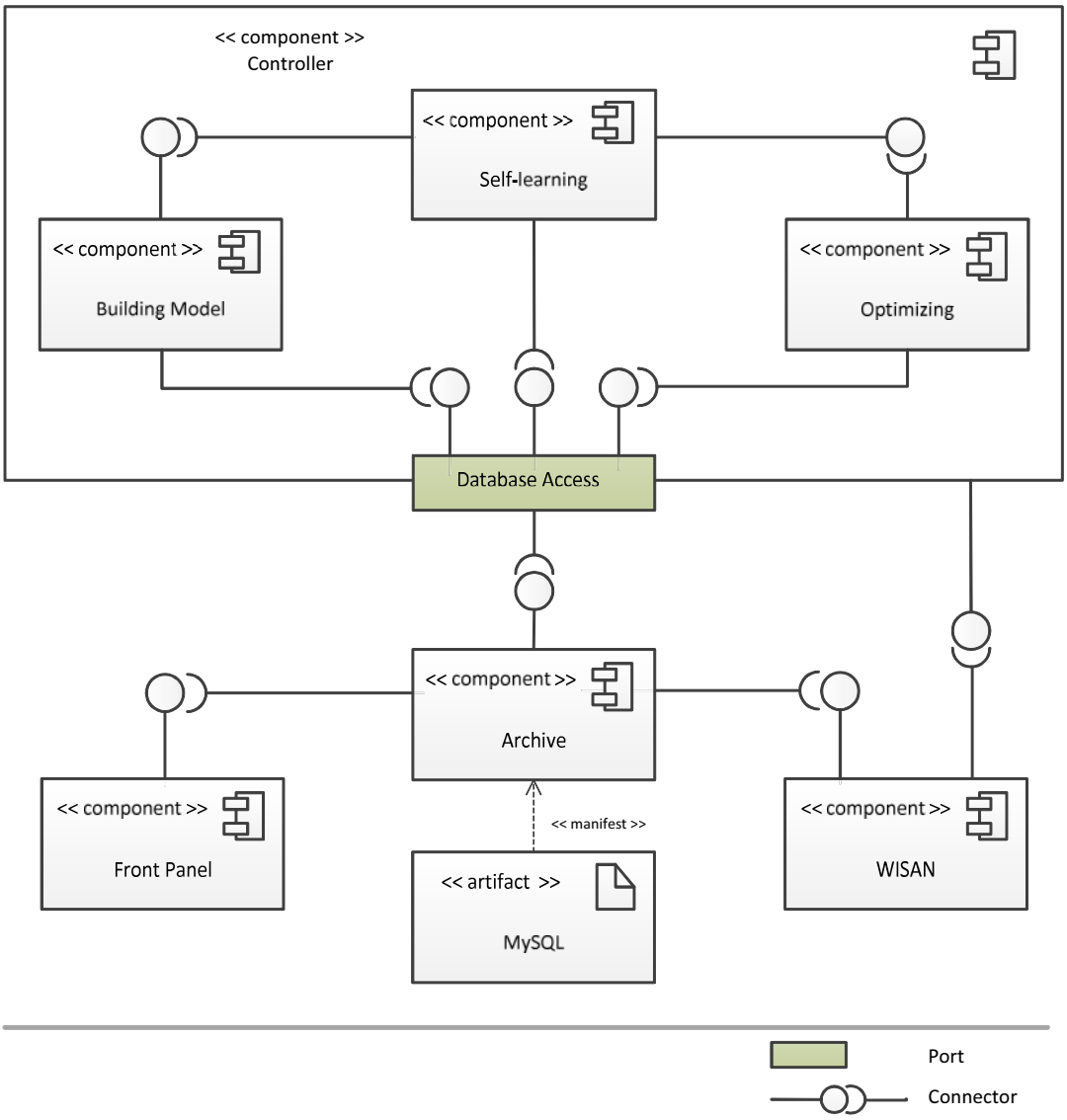

Figure 12: Component diagram of SEEDS BEMS software

The controller and its components uses time related variables or data classes, respectively. They are declared as TimePoint comprising the tuple [time, value]. Summaries of several time points under an identifier build TimePoints which are used to compose, store and transfer time series (figure 13).

The controller samples the database in a fixed time period. In the same cycle the present values of control variables and energy consumption are written into the database. As shown in figure 12, each core component also has access to Archive. To unify the transfer, an interface is defined that includes all access methods to the database (figure 14). In addition, the controller defines an interface that regulates the call of the core components and the data transfer between them (figure 15). Detailed information about the interface implementation is given in the report [12] *D5.4 - Specification and implementation of interfaces that have been integrated and tested*.

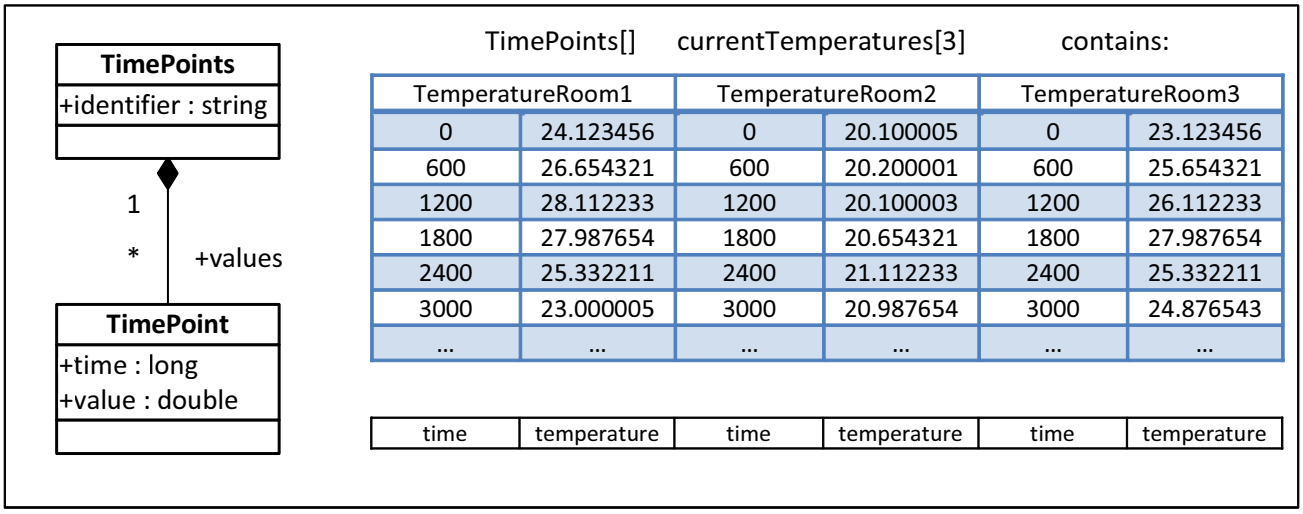

Figure 13: Classes of time related variables

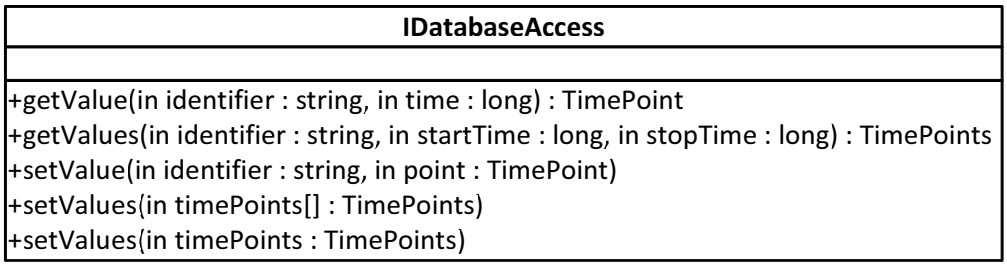

Figure 14: Interface definition for database acces

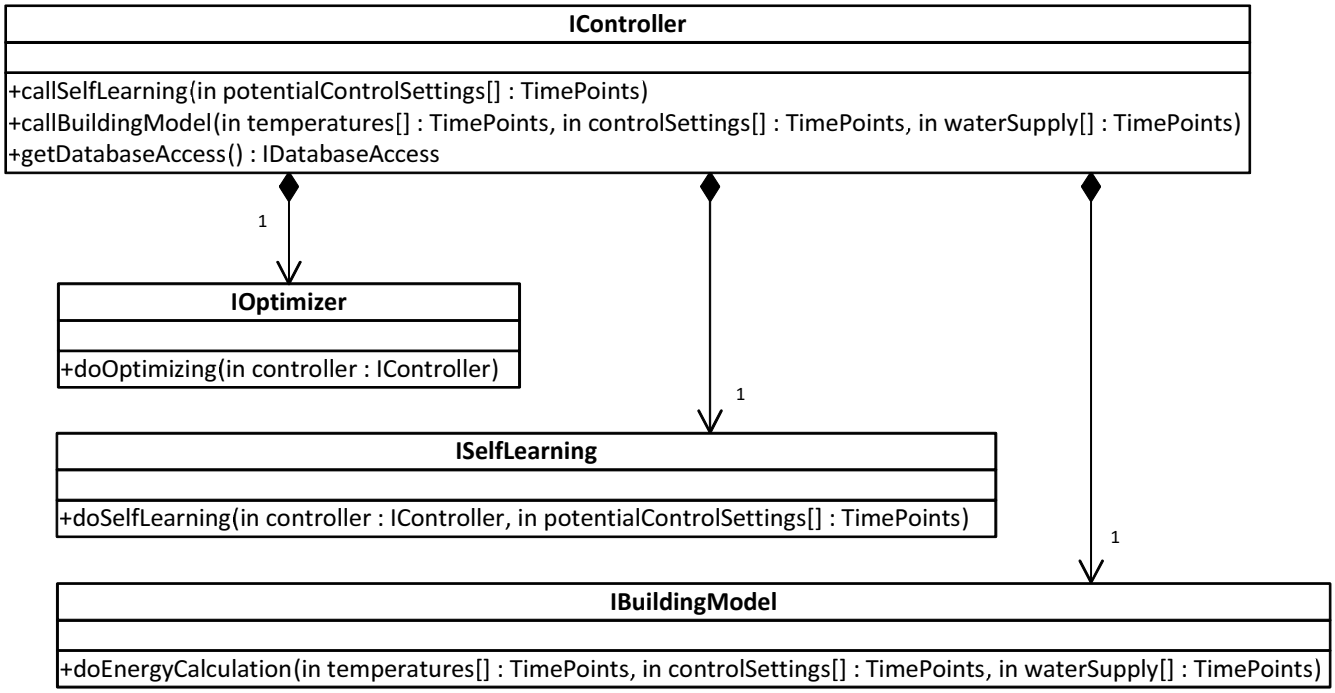

Figure 15: Interface definition for core component calls

3.2.1.2 Control and Data Flow

The loop of process control is a simple extension of the loop for energy calculation, in the first instance. A forecast period is defined as a multiple of the sampling period. If a forecast time point is reached, a time event is triggered that starts the optimization (figure 16). The optimization runs in a parallel process. The optimization performs the entire data transfer independently of the further activities in the control loop.

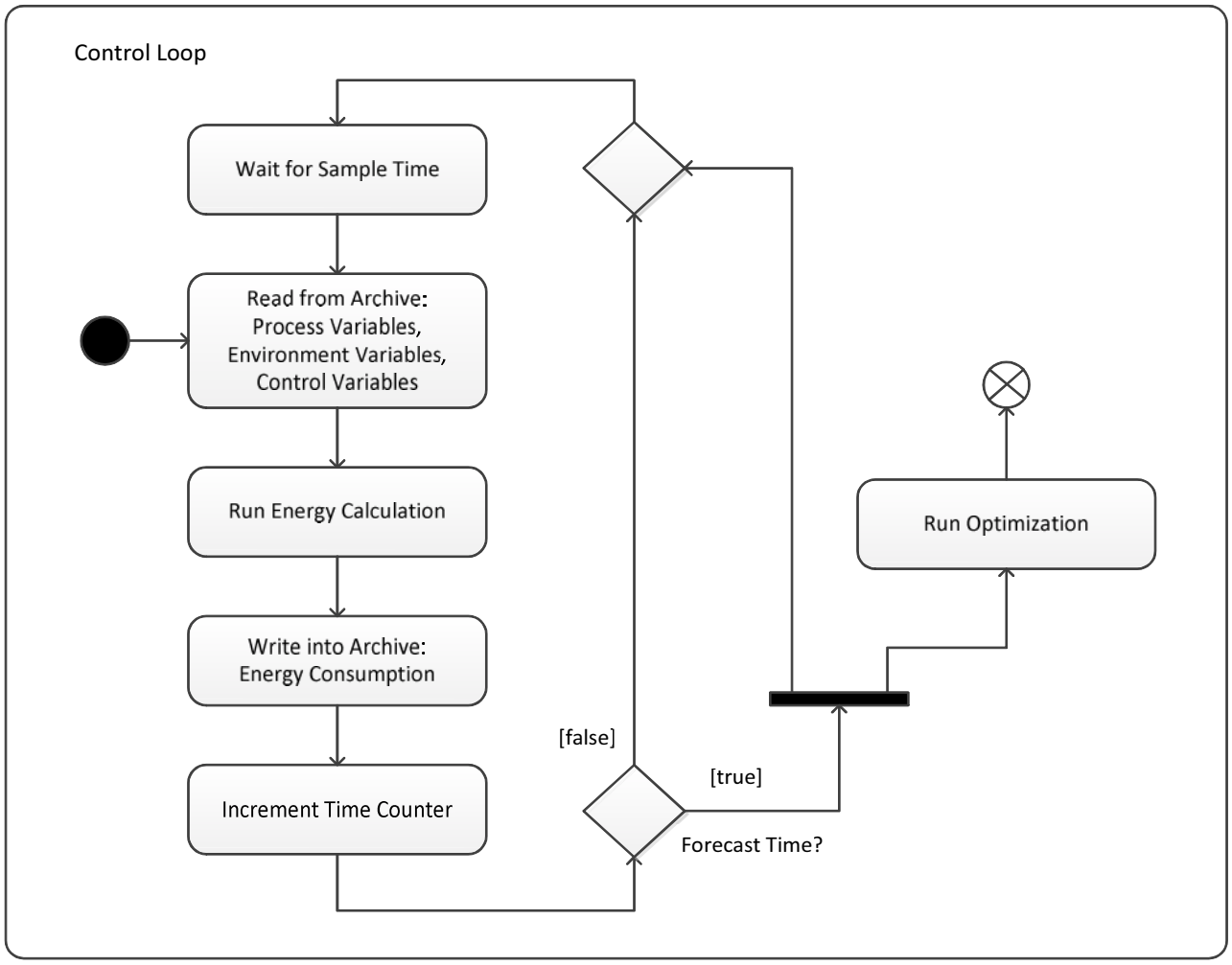

Figure 16: Control loop for starting optimization

The optimization comprises two activities: optimizing itself and the forecast of comfort variables and energy consumption by self-learning (figure 17). Optimizing varies the control variables according a given strategy, e.g. Paricle Swarm Optimization, checks the limitations of the comfort variables and evaluates the objective function. If no further reduction of energy consumption can be found or a preset number of trials is reached, then the optimization is terminated. The control variable vector related to the best trail is written into Archive.

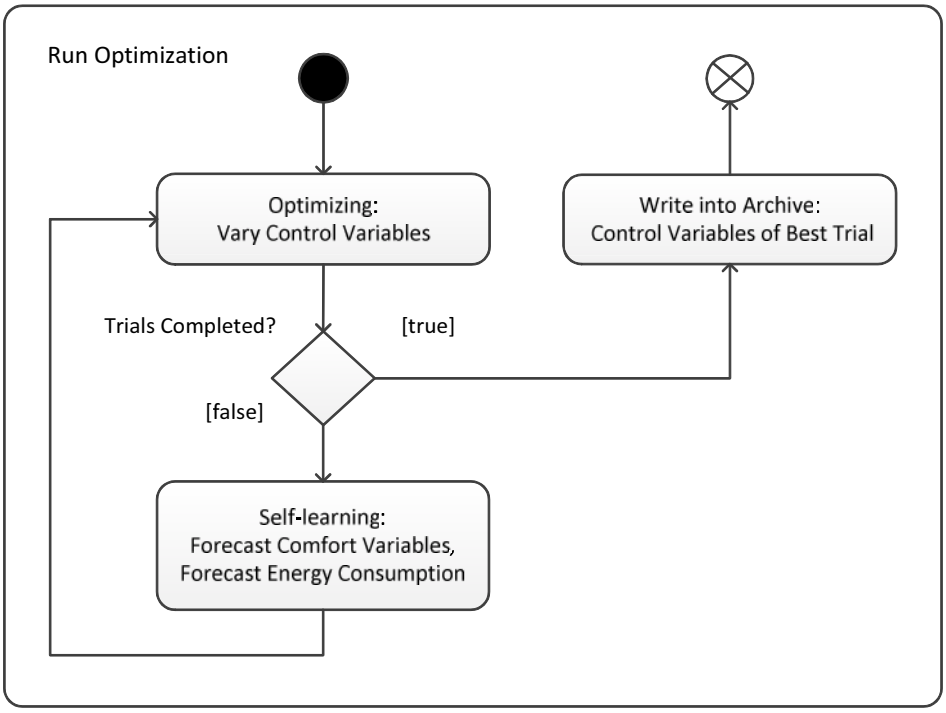

Figure 17: Optimization with forecasting by self-learning

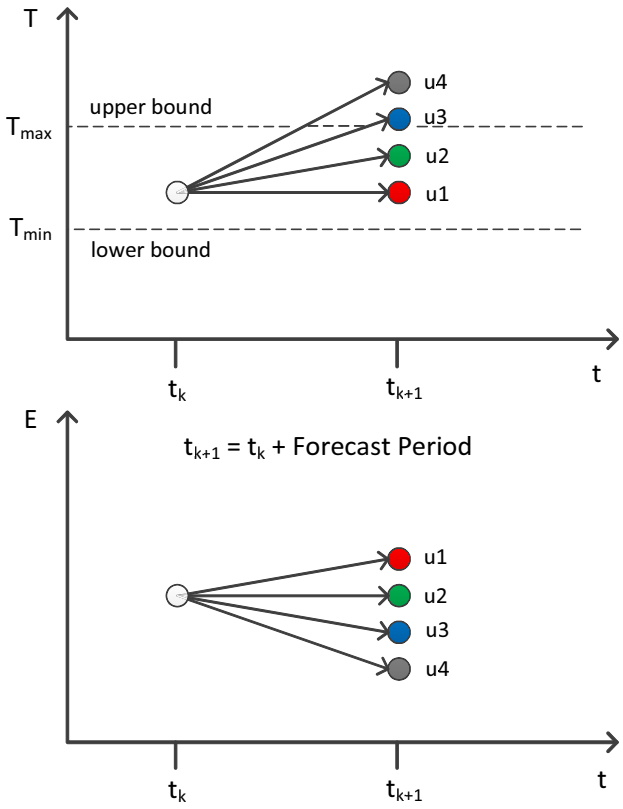

Figure 18: Task for optimization: finding the best compromise u = u2

The optimization task is illustrated by figure 18. Time t_k is the time point of optimizing. The optimization algorithm varies the control variable u, i.e., determines successively the potential control settings u_i, i= 1, 2, 3… and calls forecasting. Forecasting predicts the room temperature T

and the increase of energy consumption ΔE for time t_{k+1}, which is defined by the preset forecast period. The control settings u_3 and u_4 with a limit violation of T are discarded by the optimization algorithm. The optimization will continue that u_2 leads to lower energy consumption than u_1 and selects consequently u_2 as best compromise. Until the next optimization u_2 is output as control setting. Following an advanced optimization technique [17], the objective function or cost function is composed of the energy consumption and a penalty term. The latter delivers penalty values when the comfort limits are violated. Thereby optimizing is directed to the permitted range. Details are explained in section 3.2.2.

Figure 19 gives an overview of the relation of control and data flow by means of Helicopter Garage.

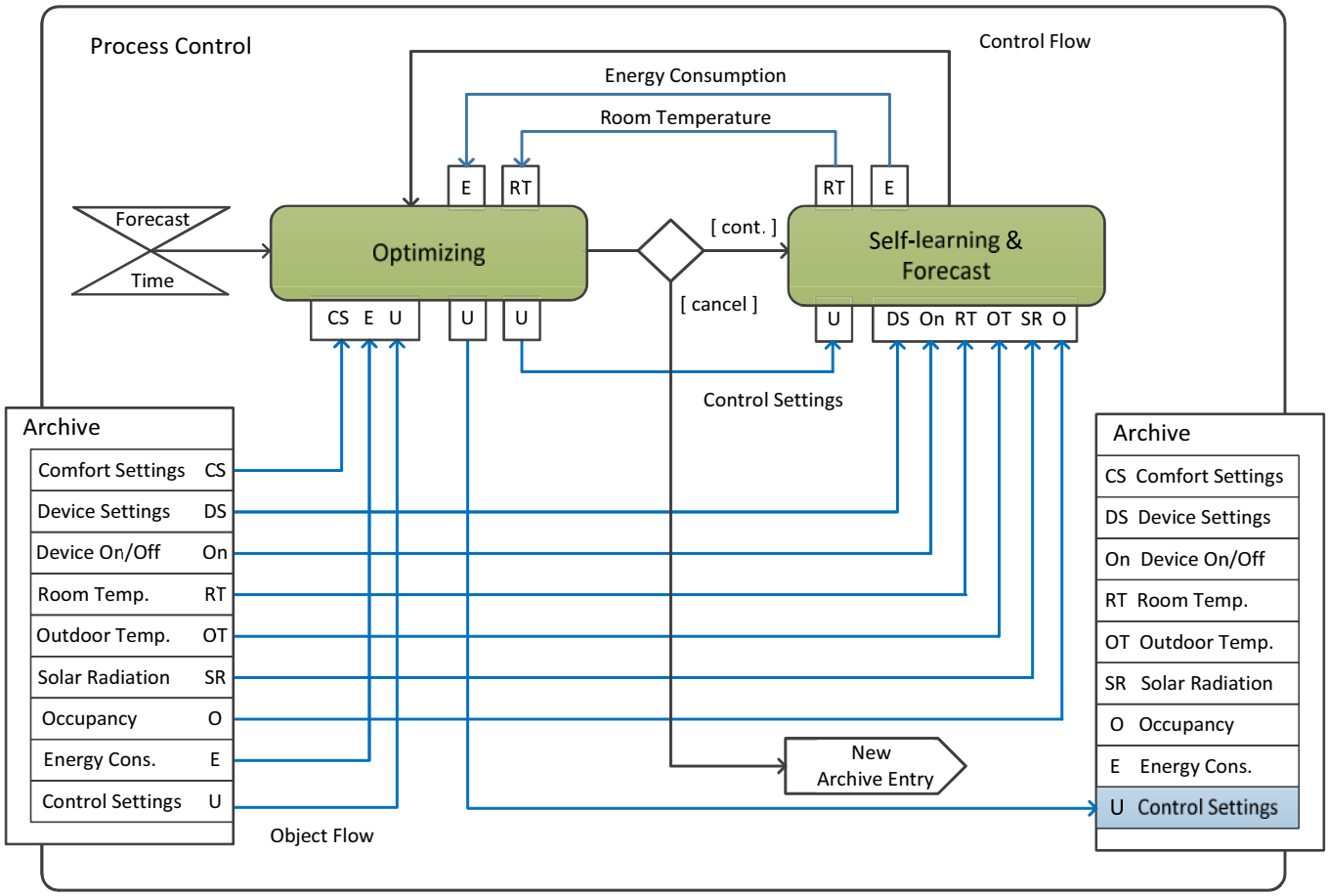

Figure 19: Activity diagram of optimization of control settings

Optimizing and Self-learning read data from Archive and writes data into Archive:

- **CS – comfort settings** (reference variables)
  CT_i – Comfort temperature in °C of $Room_i$ in a time table, i=1…10
  $\Delta Tmax$ – Maximum delta of the comfort temperatures in °K
- **DS – device settings** (reference variables)
  AT_j – Set point of the air return temperature in °C of the $Fan\text{-}Coil_j$, j=1… 12
  WT_{CH} – Set point of the water supply temperature in °C of the Chiller
  WT_{HP} – Set point of the water supply temperature in °C of the Heat Pump

- **On – device states** (process variables)
  $OnFC_j$ – On/Off of the Fan-$Coil_j$, j=1… 12
  OnCH – On/Off of the Chiller
  OnHP – On/Off of the Heat Pump

- **RT – comfort variables** (process variables)
  RT_i – Current temperature in °C of $Room_i$, i=1… 10
- **OT – environment variable** (disturbance variable)
  OT – Outdoor Temperature in °C in a time table
- **SR – environment variable** (disturbance variable)
  SR_i – Solar Radiation Intensity of $Room_i$ in the range 0 to 1, i=1… 10
- **O – environment variable** (disturbance variable)
  O_i – Occupancy of $Room_i$ in Number of Persons in a time table, i=1… 10
- **E – energy consumption** (load variable)
  E – Total energy consumption of the Building in kWh
- **U – control settings** (control variables)
  FC_Speed_j – Speed level [1, 2, 3] of the Fan-$Coil_j$, j=1… 12
  CH_Load – Load level [0.25, 0.50, 0.75, 1.00] of the Chiller
  HP_Load – Load level [0.25, 0.50, 0.75, 1.00] of the Heat Pump

The control flow starts with a Forecast Time event and circulates subsequently between Optimizing and Self-learning until Optimizing finishs it. Optimizing activates Self-learning with giving the potential control settings as arguments. Self-learning returns the forecasts of room temperatures and energy consumption.

3.2.1.3 Inclusion of Occupancy in Controls

In the forecast of comfort variables, e.g. room temperatures, the influence of occupancy is integrated. The occupancy may be given by the facility manager as a time table which is stored in Archive. Table 3 shows an occupancy schedule of certain rooms of Helicopter Garage; figure 20 depicts this in a time line.

Time	Hall No of People	Office1 No of People	Dining Room No of People
00:00	2	0	0
07:00	8	2	4
09:00	8	2	0
11:00	8	2	4
13:00	8	2	0
15:00	5	2	0
16:00	8	2	4
17:00	5	2	0
22:00	2	0	0

Table 3: Occupancy schedule of Helicopter Garage

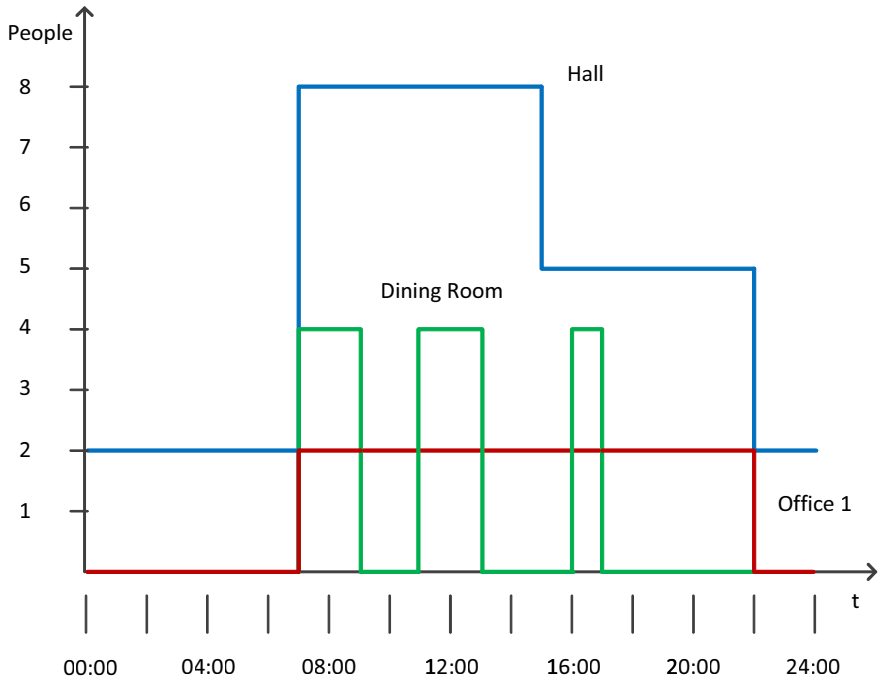

Figure 20: Time line of occupancy

As outlined above, optimization and forecast run at given discrete points in time t_k and refer to a fixed period, called Forecast Period. The occupancy schedule and the timed calls of optimization and forecast must be brought into a close relation, in the sens, that all events in the Forecast Period are transferred to the interval limit t_k (figure 21). Self-learning performs this by reading Archive with access routines of the database interface. For instance, self-learning calls *getvalue (Room1.Occupancy, current_time)* with *current_time=t_k* and gets back the tupel *(event_time, number_of_people)*.

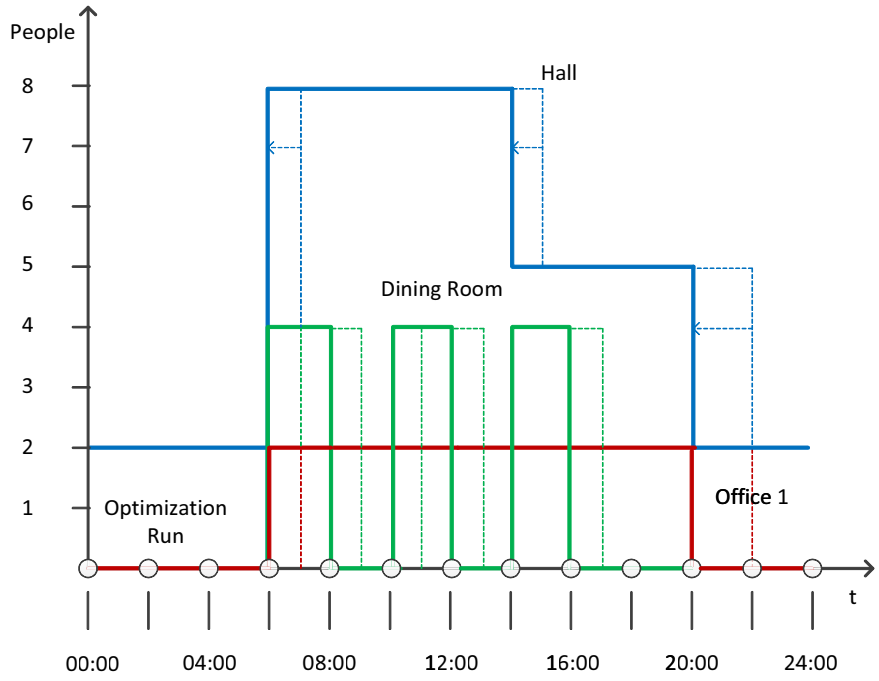

Figure 21: Shifted time line for optimization runs at forecast points

3.2.1.4 Inclusion of Environmental Conditions in Controls

As example of environment conditions, the outside temperature is considered here. Its integration is similar to the integration of occupancy schedules. Temperature curves are given by a weather forecast agency. They are transformed in a time table which is stored in Archive. Self-learning reads this with access routines of the database interface (figure 14). Return value is the outdoor temperature at the latest time point in the Forecast Period (figure 22).

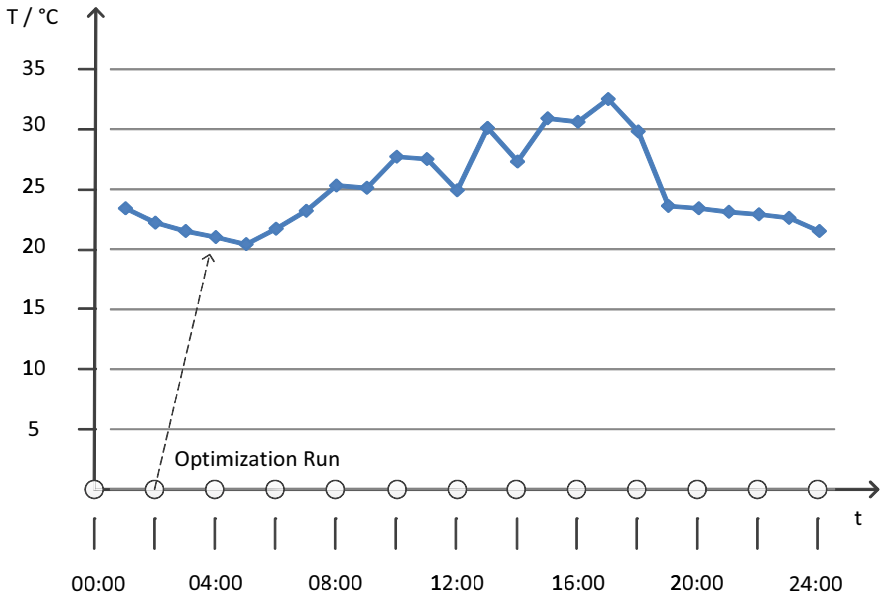

Figure 22: Inclusion of outside temperatures in optimization runs

3.2.1.5 Time and Event Triggered Start of Optimisation

In the previous discourses, the optimization was cyclically executed with a fixed time period, called Forecast Period (figure 16). Because at certain times an optimization produces no effect (e.g. in phases at rest in the night hours), the start of optimization is bound on an event as well its stop. Such an event may be the change of a system variable, e.g. the sudden magnification of the occupancy of the helicopter hall (figure 23).

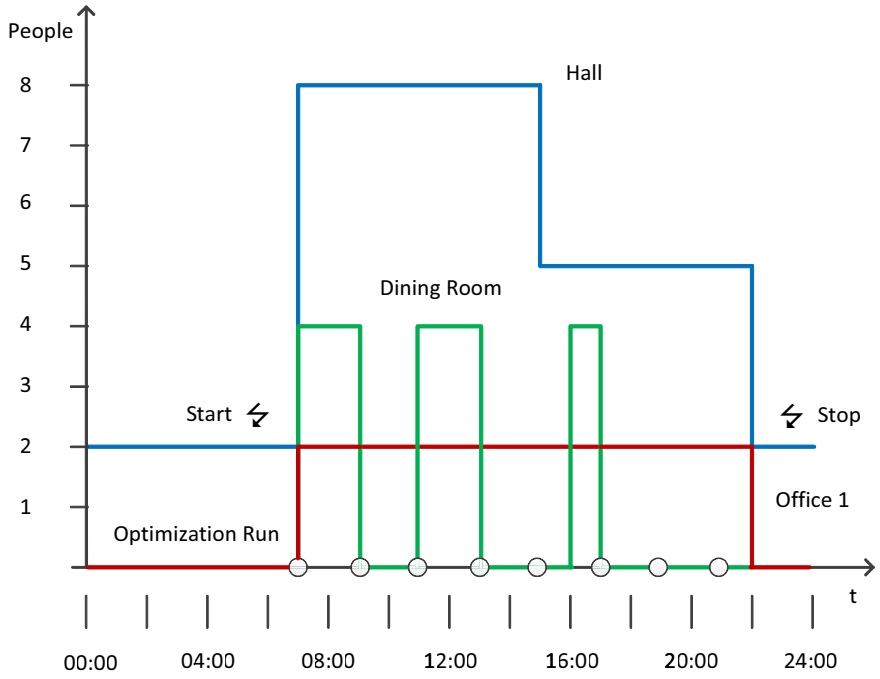

Figure 23: Occupation increase/decrease as start/stop event for optimization

The event can be generated by a presence detection sensor or a software module who evaluates the occupany schedule in Archive. The latter has the advantage that a foresight can be performed. After detection of a Start Event, a Forecast Timer is set and Optimization is run (figure 24). After this, the optimization will be performed periodically until the Stop Event is detected. Forecast Timer is set on the Forecast Period. The optimisation runs in parallel to the control loop.

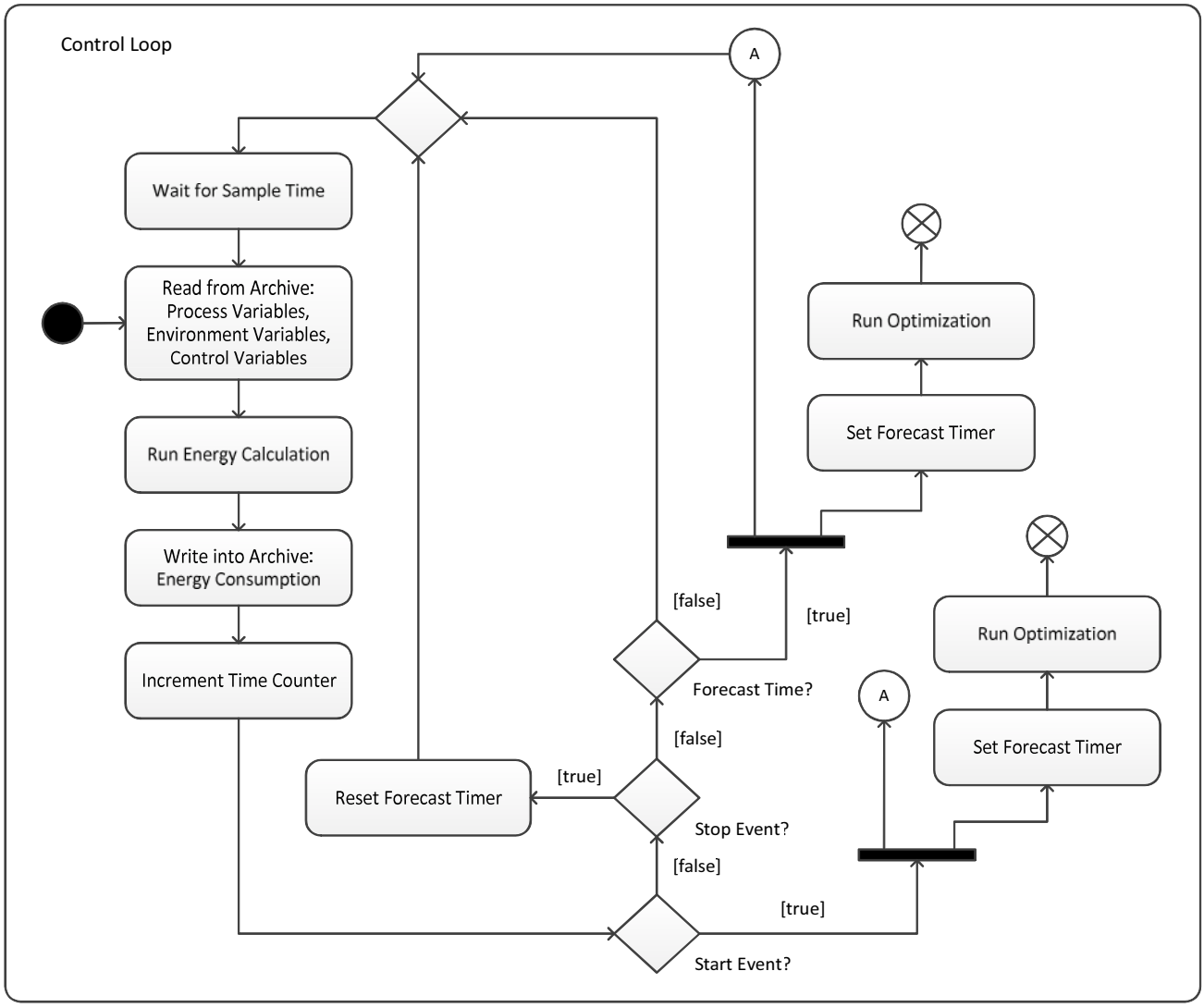

Figure 24: Control loop for event triggered start/stop of optimization

3.2.1.6 Transmission of the Optimised Control Settings

Each optimization cycle ends with providing of new control settings. These values are transmitted to Archive and WISAN. The former is carried out by the *setValues()* method of the controller (section 3.2.1.1). The latter is performed by the *SetActuatorValue ByNodeNameAndActuatorName()* method of the WISAN Web Service immediately. As the method identifier suggests, the names of the node and the actuator must be specified by the caller to address the target. Detailed information about the WISAN communication infrastructure is given in the report [6] *D4.3 – Plug & Play conformance requirements: API, Integration Webservices and libraries.*

An alternative solution is the transmission of the optimal control settings only to Archive and the notification of WISAN about it. WISAN then queries Archive and transmits the control settings to the actuators. Figure 4 illustrates these data paths.

3.2.2 Optimizing

The SEEDS optimization can be overviewed in the sequence diagram presented in figure 25. The optimization module gets the energy predictions through the self-learning module transparently from the SEEDS model chosen for energy prediction (i.e. Self-learning or Building Model). The modul “Fitness” relates to the algorithm used for optimization.

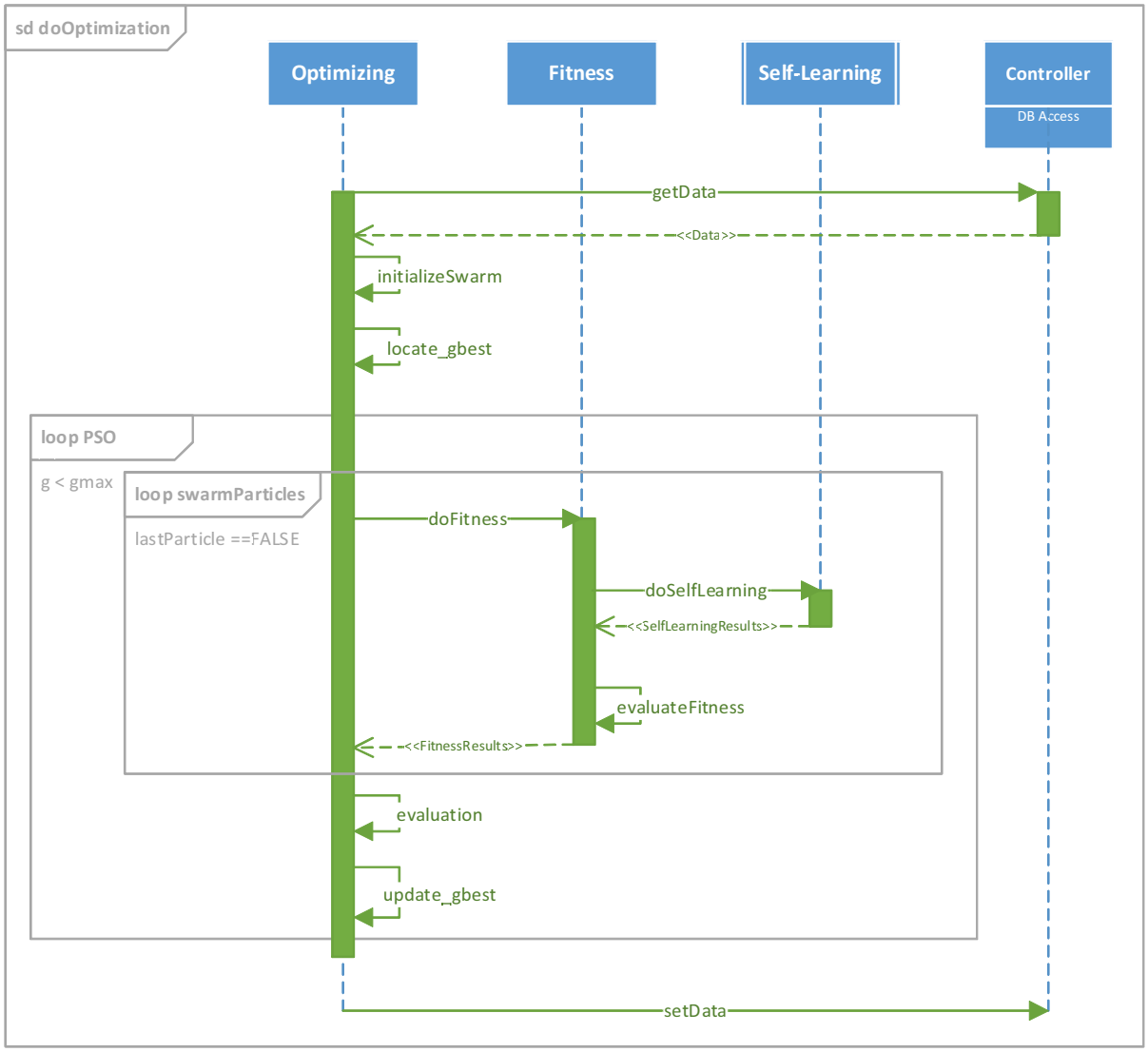

Figure 25: Optimization sequence diagram for Helicopter Garage

3.2.2.1 Selected Algorithm – Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO shares many similarities with evolutionary computation techniques such as Evolutionary Algorithms (EA). In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. As so, the algorithm works by updating the position of

particles from a swarm to find the optimal solutions for a cost function. Each particle adjusts its flying trajectory by incessantly updating its position and velocity.

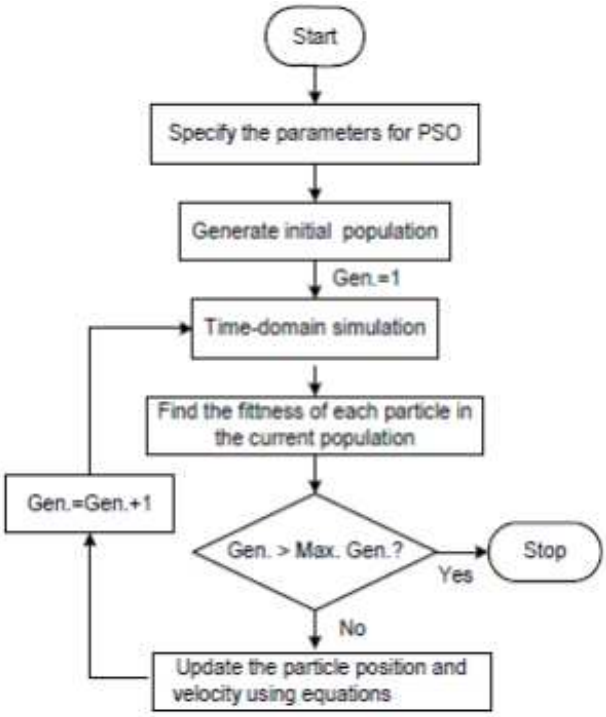

Figure 26: Particle Swarm Optimization flowchart (Panda and Padhy 2008)

First, the swarm population is initialized including the initialization of both position and velocity for each particle in the population (figure 26). The personal best *pbest* is initialized for each particle and the corresponding global best *gbest* is chosen to be the leader of the swarm. Then, for a maximum number of iterations, each particle flies over the search space by updating its velocity and position, using:

$$vi = w*vi + c1*rand()*(pbesti- xi) + c2*rand()*(gbest-xi)$$

$$xi= xi + vi,$$

where vi is the velocity of particle; w is the inertia weight; c1 and c2 are learning factors; xi is the position of particle. The *pbest* for each particle and the leader for the whole population are also updated within the iterations.

The pseudo code of how PSO works is presented in table 4.

```
Begin
    Initialize swarm
    Locate gbest
    generation=0
    While generation<gmax
    For each particle
        Update velocity and position
        Evaluation
        Update pbest
    End For
    Update gbest
    g++
    End While
End
```

Table 4: Pseudo code of PSO

The integration of the PSO with the SEEDS optimization is shown in the sequence diagram previously presented in figure 26. The SEEDS PSO starts by initializing a swarm and *gbest* using data read from Archive. The PSO proceeds to a loop. In each iteration of the loop, a swarm of particles is evaluated. Each particle of the swarm has a different set of coordinates. The coordinates of each particle correspond to a possible set of control settings. For each particle, the PSO calls Self-Learning to get the predicted values of the comfort parameters and the predicted values of the energy consumption. The predicted values are used in the cost function to calculate the fitness of each particle. The fitness of all particles of the swarm is evaluated and if a particle is found with better fitness the *gbest* is updated. The information of the *gbest* is used for calculating the coordinates of the particles in the next iteration. The PSO loop finishes when a certain number of iterations (or convergence criteria) are met. The final value of the *gbest* is sent to Archive.

3.2.2.2 Optimization Variables with Bounds and Discretization

In the original PSO algorithm, the optimization variables domain is continuous. However, in SEEDS the optimization variables are discretized and bounded by lower and upper bounds.

E.g.: Fan speed level $\in \{1, 2, 3, 4\}$

The optimization assigns the discret and finite optimization variables to equivalent variables that are normalized and continuous

E.g.: Fan speed level $\leftrightarrow x \in [0, 1]$,
where
Fan speed level = 1 if $x \in [0, 0.25]$
Fan speed level = 2 if $x \in [0.25, 0.5]$
…

Thus, the variable *x* can be used by the PSO.

3.2.2.3 Call of Forecasting of Comfort Variable Values and Energy Consumption

The cost function that SEEDS PSO uses to calculate the fitness is expressed for the forecast period Δft as:

$$cf(\hat{e}, w)_{\Delta ft} = \hat{e}_{\Delta ft} + \alpha_{\Delta ft}$$

where

ê is the predicted energy consumption
α is the comfort penalty.

The general idea of the cost function is to score the solution candidates according to the predicted energy consumption penalized by how the candidates diverge from the desired comfort level. For a certain candidate, the cost function needs the predicted comfort variables given by Self-learning and the predicted energy consumption given by Self-learning or Building Model.

3.2.2.4 Evaluation of Restrictions and Objectives

To deal with the comfort levels, a comfort penalty is calculated by the Comfort Level function. The Comfort Level function filters out the candidates that fall out of the boundaries. A candidate that falls within the boundaries is scored directly by the predicted energy consumption. When the predicted comfort of a candidate does not fall within the boundaries, its score is calculated by adding a penalty factor to the predicted energy. The penalty factor is proportional to the distance from the desired comfort level:

$$\alpha_{\Delta ft}(\hat{e}, \omega) = \begin{cases} 0, & wl \leq \omega \leq wh \\ f(\hat{e}, d), & \omega < wl \vee \omega > wh \end{cases}$$

where

ê is the predicted energy consumption
ω is the predicted comfort level
wl is the lower comfort boundary
wh is the higher comfort boundary
d is the discomfort.

The usage of a penalty factor allows Optimizing to find a solution in a situation where there are no candidates that fulfil the desired comfort level.

3.2.3 Self-learning & Forecast

3.2.3.1 Features of the Self-learning Component

The Self-learning component builds a model using training data. A model represents a pattern in the training data that can be used to forecast (predict) future values of a variable.

The Self-learning component is defined with respect to the following time points and time periods:

- sample time period for Self-learning;
- t_k: time point of optimization, that determines the time of forecast;
- $t_{k+\Delta ft}$: time point about which a forecast is made, where Δft [1] represents the duration of the forecast period.

For Helicopter Garage, the Self-learning sample time period is 10 minutes and the forecast period Δft is 30 minutes.

3.2.3.2 Structure of the Selected Neural Networks

We have adopted neural networks as the Self-learning method for Helicopter Garage example. A neural network is a collection of connected input and output units, also known as “neurons”. A weight is associated with each connection. The output of a neural network normally is for classification or prediction.

Neural networks for BEMS were described in deliverable [7] D5.1 *Report on applicability of different learning and optimization method* and empirical results on some sample scenarios were presented in deliverable [8] *D5.2 - Self-learning algorithms that have been verified with sample scenarios and test data*. For completeness, it is worth to summarize their structure which includes:

- Input Layer: The inputs are fed simultaneously into the units in the input layer.
- Hidden Layers: The inputs from the input layer are then weighted and fed simultaneously to a hidden layer of neuron like units. A neural network may include multiple hidden layers. Figure 27 shows a neural network with two hidden layers. However in practice, normally only one hidden layer is used.
- Output Layer: This layer includes the units to which the weighted outputs of the last hidden layer are input.

[1] We use the notation - Δft in an index is considered a multiple of the sample time e.g. 3. Otherwise, Δft is considered a length of time e.g. 30 minutes.

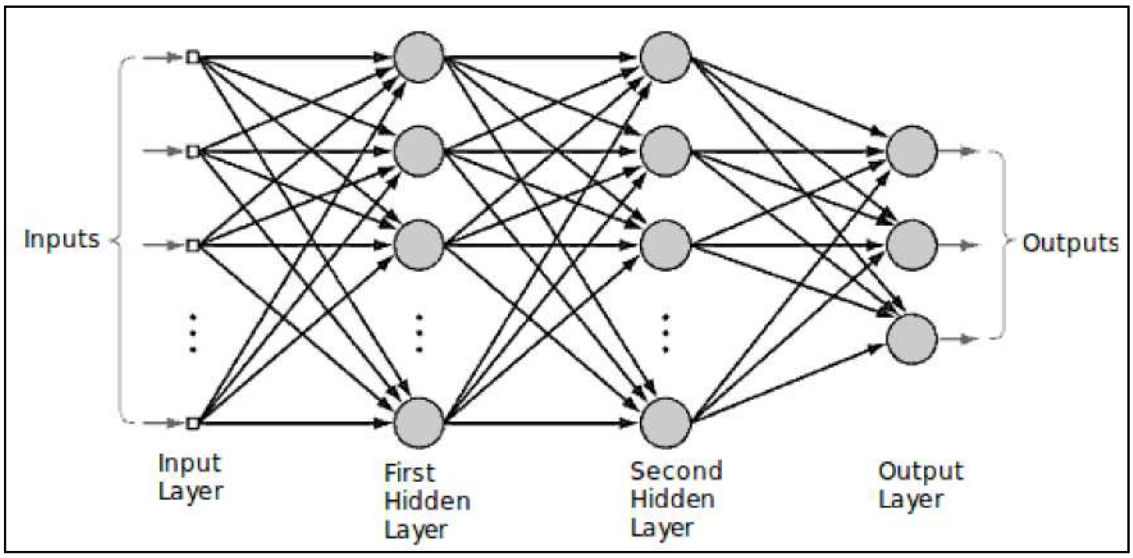

Figure 27: Neural network structure with two hidden layers

3.2.3.3 List of Features of Selected Neural Networks

This section explains the list of features that are chosen in the neural networks for Helicopter Garage dataset. We can classify the features into three categories: provided features, derived features, predicted features.

Provided Features

The values of provided features are provided either by Optimization or Archive. A provided feature is used on the input layer of a selected neural network.

The Archive provides the following:
- **CS – comfort settings**
- **DS – device settings**
- **On – device states**
- **RT – comfort variables**
- **OT – environment variable**
- **SR – environment variable**
- **O – environment variable**
- **E – energy consumption**

The Optimizer provides the following:
- **U – control settings** (control variables)

The list of provided features chosen for Helicopter Garage includes:
- **temp_outside (OT$_{t_m}$)**: Outdoor air temperature at time point t_m in °C. This value is from the Archive.
- **occupancy_room (O$_{i_{t_m}}$)**: Occupancy of Room i at time point t_m. This value is from the Archive.

- **heatpump (HP_Load $_{t_m}$)**: Heatpump load level at time point t_m. This value is from the Optimization component.
- **chiller (CH_Load $_{t_m}$)**: Chiller load level at time point t_m. This value is from the Optimization component.
- **fanspeed (FC_Speed $_{j_{t_m}}$)**: Fan-coil *j* Speed at time point t_m. This value is from the Optimization component.
- **temp_room (RT $_{i_{t_m}}$):** Room *i* temperature at the current time point t_m. This value is from the Archive.

where $i = 1\ldots10$, and $j = 1\ldots12$.

Derived Features

The value of a provided feature can be used to calculate the value of a derived feature. The derived features are used on the input layer of a selected neural network.

The list of derived features chosen for Helicopter Garage includes:

- **timeOfday**: defined according to the timepoint t_m where:
  night time (NT) is 6p.m.-7a.m.;
  morning time (MT) is 7a.m.-12a.m.;
  afternoon time (AT) is 12a.m.-6p.m..
- **temp_room_diff (ΔRT $_{i_{t_m}}$ = RT $_{i_{t_m}}$ - RT $_{i_{t_{m-1}}}$)**: Room *i* temperature difference between current time point t_m and previous time point t_{m-1}. This value is calculated based on the temperature values from the Archive**.**
- **energy_consumed (ΔE $_{t_m}$ = E $_{t_m}$ - E $_{t_{m-1}}$):** Energy consumption at timepoint t_m . This value is calculated based on the energy consumption values from the Archive.
- **energy_consumed_lag_1 (ΔE $_{t_{m-1}}$ = E $_{t_{m-1}}$ - E $_{t_{m-2}}$):** Energy consumption between timepoint t_{m-1} and t_{m-2} . This value is calculated based on the energy consumption values from the Archive.
- **energy_consumed_lag_2 (ΔE $_{t_{m-2}}$ =E $_{t_{m-2}}$ - E $_{t_{m-3}}$):** Energy consumption between timepoints t_{m-2} and t_{m-3}. This value is calculated based on the energy consumption values from the Archive.
- **energy_consumed_lag_3 (ΔE $_{t_{m-3}}$ = E $_{t_{m-3}}$ - E $_{t_{m-4}}$):** Energy consumption between timepoints t_{m-3} and t_{m-4} . This value is calculated based on the energy consumption values from the Archive.
- **energy_consumed_diff (ΔE $_{t_m}$ - ΔE $_{t_{m-1}}$):** Difference in lagged energy consumption ΔE_{t_m} and $\Delta E_{t_{m-1}}$.
- **energy_consumed_diff_1 (ΔE $_{t_{m-1}}$ - ΔE $_{t_{m-2}}$):** Difference in lagged energy consumption $\Delta E_{t_{m-1}}$ and $\Delta E_{t_{m-2}}$.
- **energy_consumed_diff_2 (ΔE $_{t_{m-2}}$ - ΔE $_{t_{m-3}}$):** Difference in lagged energy consumption $\Delta E_{t_{m-2}}$ and $\Delta E_{t_{m-3}}$.

where $i = 1\ldots10$, and $j = 1\ldots12$.

As shown above, the calculation of a derived feature involves the previous values of certain provided features. Note, the lagged values for features before the start of the sample are unknown and are denoted '?'.

Sample data from the Helicopter Garage example are presented in table 5.

Timepoint	energy_consumed_lag_1 ($\Delta E_{t_{m-1}} = E_{t_{m-1}} - E_{t_{m-2}}$)	energy_consumed ($\Delta E_{t_m} = E_{t_m} - E_{t_{m-1}}$)	energy_consumed_diff ($\Delta E_{t_m} - \Delta E_{t_{m-1}}$)
0	?	583	?
600	583	583	0
1200	583	583	0

Table 5: Sample data from Helicopter Garage example

Predicted Features

The value of a predicted feature is forecasted (predicted) by the selected neural network. A predicted feature is used on the output layer of the neural networks.

The list of prrdicted features chosen for Helicopter Garage includes:

- **temp_room_predict ($RT_{i_{t_{m+\Delta ft}}}$):** Predicted future Room *i* temperature at the time point $t_{m+\Delta ft}$. This feature is on the output layer of the neural network of room temperature.
- **energy_consumed_predict ($\Delta E_{t_{m+\Delta it}}$):** Predicted future energy consumption between timepoint t_m and $t_{m+\Delta ft}$. This feature is on the output layer of the neural network of energy consumption.

3.2.3.4 Sample NNs of Comfort Variables (Air Temperature, Humidity, CO2)

Table 6 shows the features that are used in the neural network for room temperature. The definition of features is shown on section 3.2.3.3.

Feature Name	Variable *	Data Type	Value Range
timeOfday	-	Nominal	{NT,MT,AT}
temp_outside	OT_{t_m}	Numeric	25 – 35 [°C]
Occupancy_room	$O_{i_{t_m}}$	Numeric	0 – 8 [Persons]
Heatpump	$HP\_Load_{t_m}$	Nominal	{25,50,75,100}
Chiller	$CH\_Load_{t_m}$	Nominal	{25,50,75,100}
Fanspeed	$FC\_Speed_{j_{t_m}}$	Nominal	{1, 2, 3, 4}
temp_room_diff	$\Delta RT_{i_{t_m}} = RT_{i_{t_m}} - RT_{i_{t_{m-1}}}$	Numeric	-3.67 – 1.197 [°C]
temp_room	$RT_{i_{t_m}}$	Numeric	22.639 – 27.007 [°C]
temp_room_predict	$RT_{i_{t_{m+\Delta ft}}}$	Numeric	22.639 – 27.007 [°C]

Table 6: Features of NN for room temperature

The parameters that were adopted for the training of the neural network are Learning Rate = 0.2 and Momentum = 0.25. Figure 28 shows the produced neural network for room temperature by using

the Helicopter Garage dataset. It has 19 neurons for the input layer, 10 neurons for the hidden layer, and 1 neuron for the output layer.

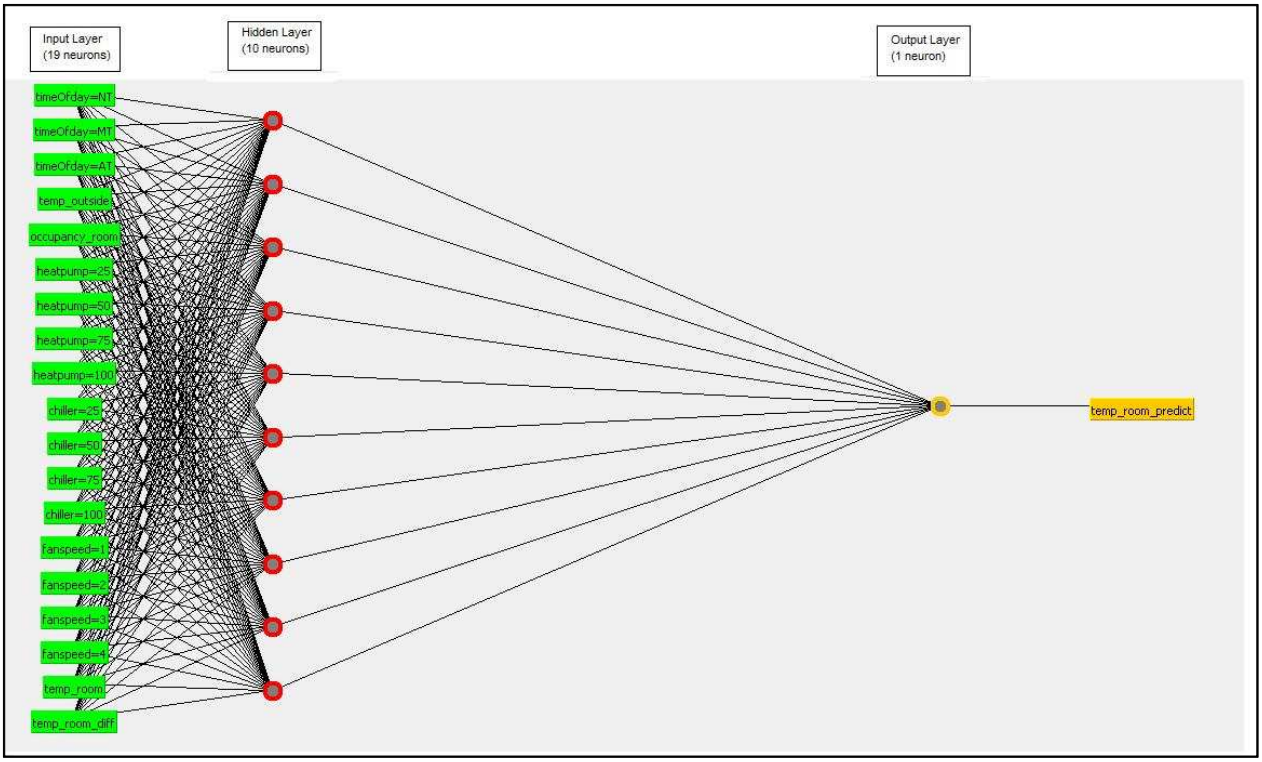

Figure 28: Neural Network for Room Temperature

3.2.3.5 Sample NNs of Energy Consumption

Table 7 shows the features that are used in the neural network for energy consumption. The definition of features is shown on section 3.2.3.3.

The parameters that were adopted for the training of the neural network are Learning Rate = 0.2 and Momentum = 0.25. Figure 29 shows the produced neural network for energy consumption by using the Helicopter Garage dataset. It has 23 neurons for the input layer, 12 neurons for the hidden layer, and 1 neuron for the output layer.

Feature Name	Variable *	Data Type	Value Range
timeOfday	-	Nominal	{NT,MT,AT}
temp_outside	OT_{t_m}	Numeric	25.0 - 35.0 [°C]
Heatpump	$HP\_Load_{t_m}$	Nominal	{25,50,75,100}
Chiller	$CH\_Load_{t_m}$	Nominal	{25,50,75,100}
Fanspeed	$FC\_Speed_{j\,t_m}$	Nominal	{1, 2, 3, 4}
energy_consumed	$\Delta E_{t_m} - E_{t_m} - E_{t_{m-1}}$	Numeric	583 – 7825 [kWh]
energy_consumed_lag_1	$\Delta E_{t_{m-1}} - E_{t_{m-1}} - E_{t_{m-2}}$	Numeric	583 – 7825 [kWh]
energy_consumed_lag_2	$\Delta E_{t_{m-2}} - E_{t_{m-2}} - E_{t_{m-3}}$	Numeric	583 – 7825 [kWh]
energy_consumed_lag_3	$\Delta E_{t_{m-3}} - E_{t_{m-3}} - E_{t_{m-4}}$	Numeric	583 – 7825 [kWh]
energy_consumed_diff	$\Delta E_{t_{m-1}} - \Delta E_{t_{m-2}}$	Numeric	-6820 – 7107 [kWh]
energy_consumed_diff_1	$\Delta E_{t_{m-1}} - \Delta E_{t_{m-2}}$	Numeric	-6820 – 7107 [kWh]
energy_consumed_diff_2	$\Delta E_{t_{m-2}} - \Delta E_{t_{m-3}}$	Numeric	-6820 – 7107 [kWh]
energy_consumed_predict	$\Delta E_{t_{m+\Delta t}}$	Numeric	1750 – 15568 [kWh]

Table 7: Features for NN of Energy Consumption

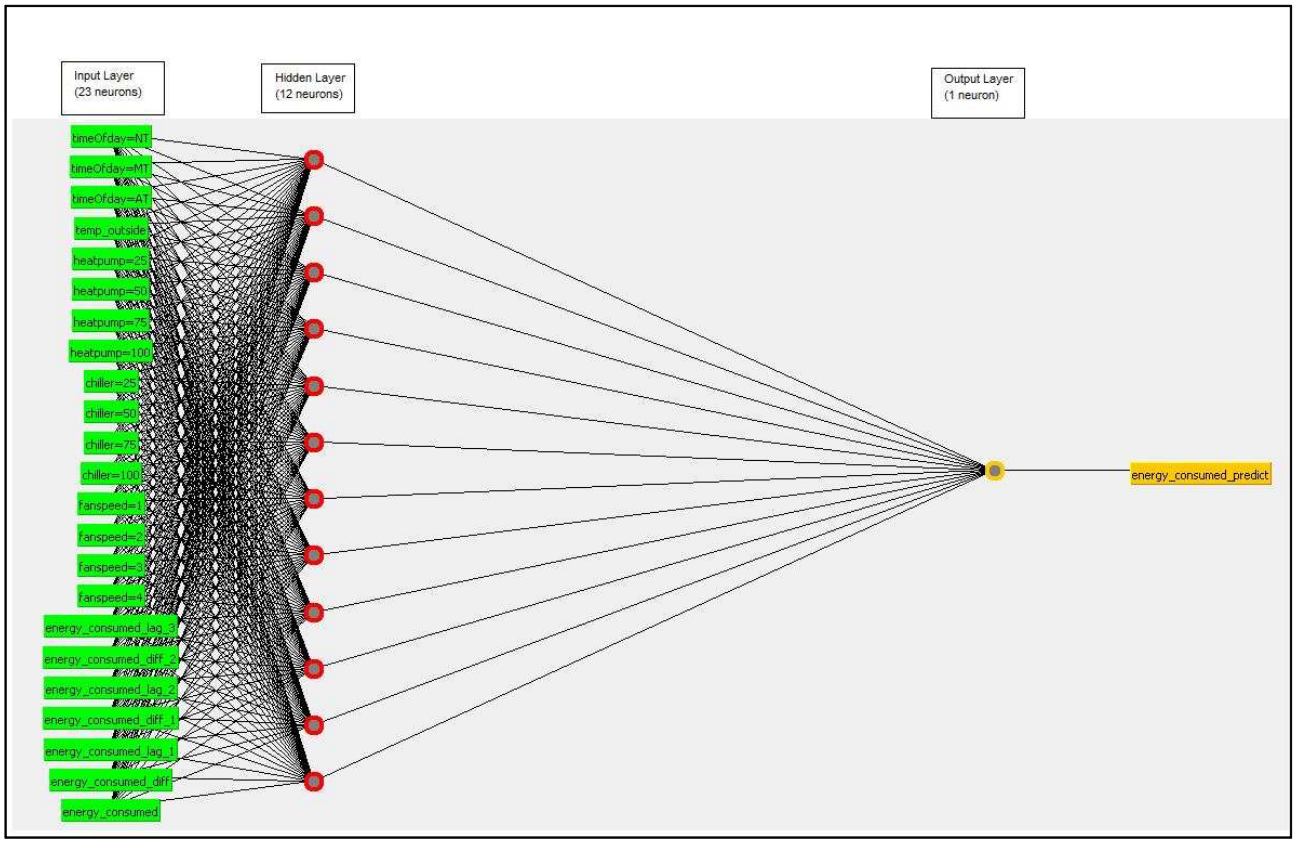

Figure 29: Neural Network for energy consumption

3.2.4 Energy Calculation using Building Model

Beside the optimization of energy consumption with a pure forecast of comfort variables and energy consumption, there is the possibility to calculate the energy consumption in the preset forecast period by means of the building model. To do this, the calculation of the future building behavior is performed in two steps: forecast of process variables by Self-learning and calculation of energy consumption by Building Model Evaluator (figure 30). In this case, Self-learning not only has to predict the values of the comfort variables, but also the values of system variables that are needed for energy calculation. In Helicopter Garage example, these are the supply water temperatures of the heat pump and the chiller. In comparison to the continuous energy calculation the difference consists only in the use of predicted values rather than measured values. The building model evaluator, including the building model, is instantiated for energy calculation in the control loop for a second time.

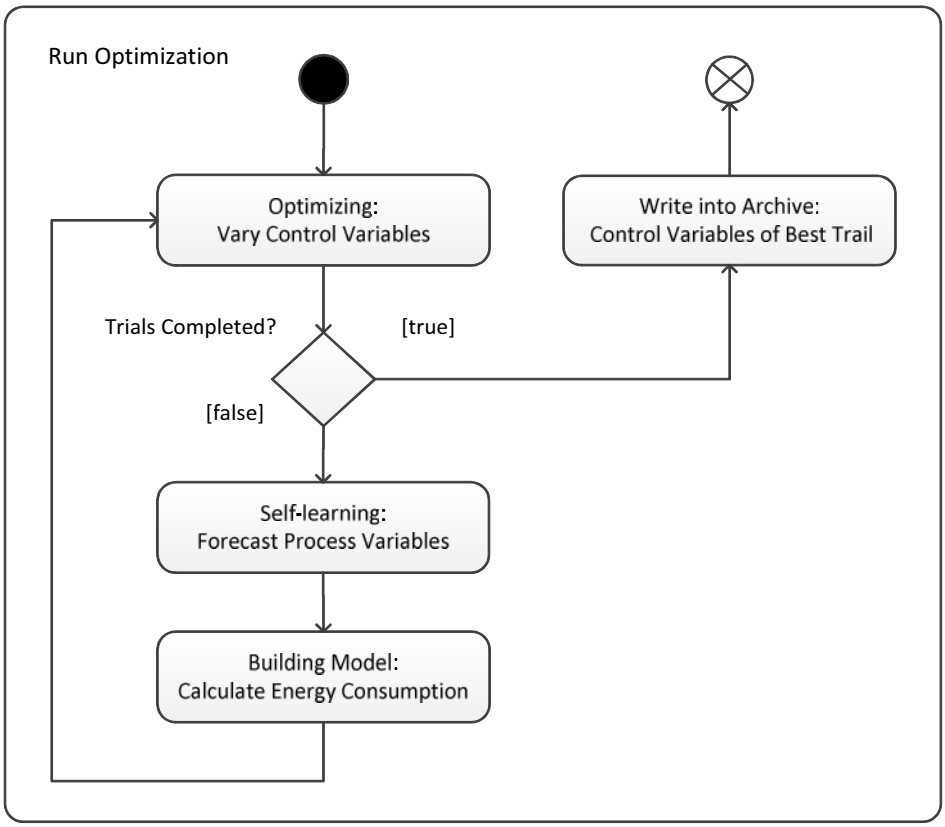

Figure 30: Optimization with self-learning and building model evaluation

For instance, in Helicopter Garage building model the operations of heat pump and chiller are extrapolated from the predicted supply water temperatures in the cooling circuits. Then, according to the device states the energy consumptions are calculated (figure 31).

This approach requires of the forecast period:

$$T_{forecast} \leq T_{min}/2 \,,$$

assuming supply water temperature curves are approximated by Sinus functions with periods T.

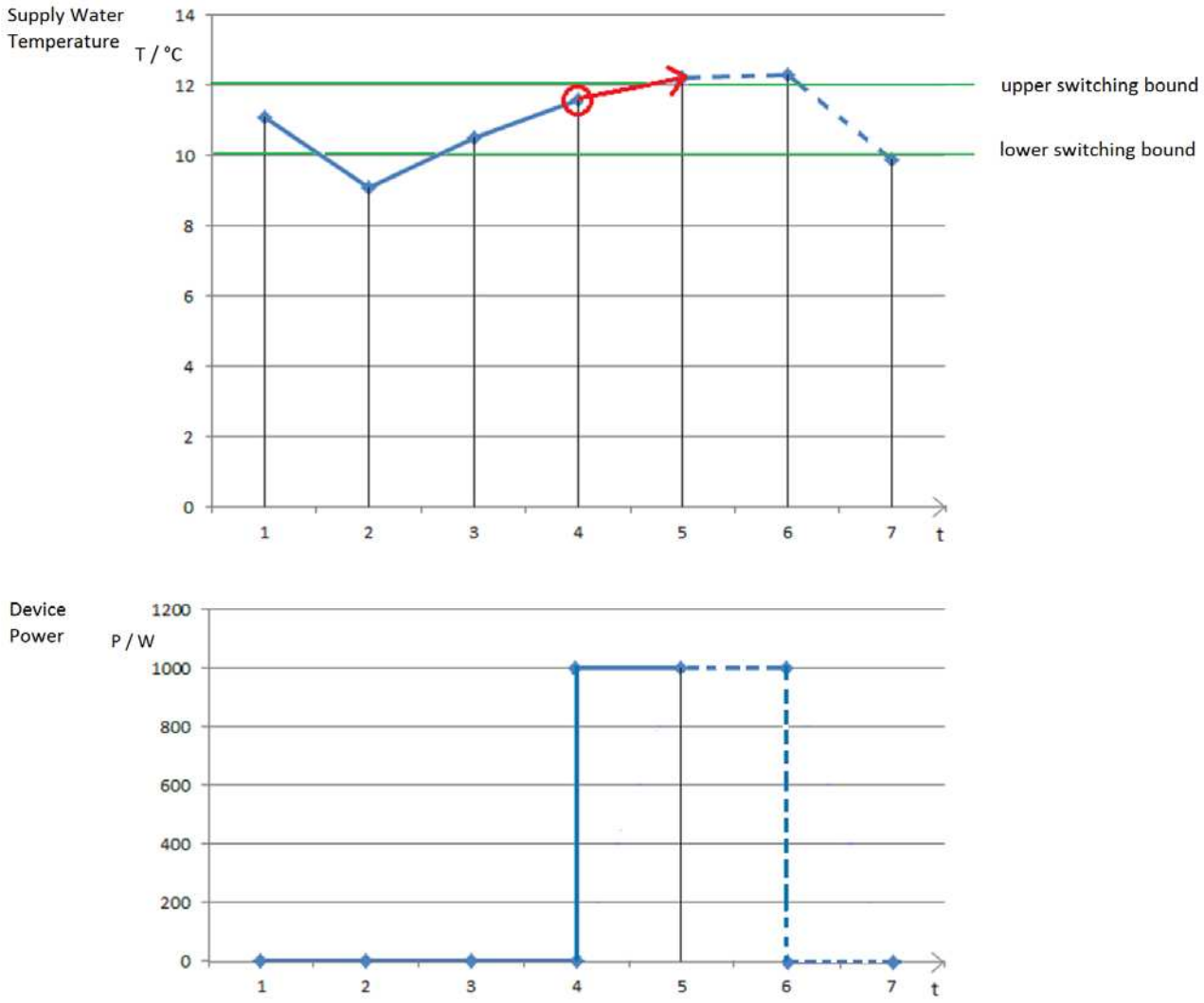

Figure 31: Principle of prediction-based energy calculation

3.3 Data Management

3.3.1 Temporary Data for Control Operations

Figure 19 provides an overview which data are used for optimizing Helicopter Garage's control settings. These data are the present values of certain process, reference, and disturbance variables. Moreover, the NNs need also past values for the prediction of comfort variables and the energy consumption. The past ranges from 1 to 4 sample time steps backward (table 6 and table 7). Of occupancy schedules and weather forecasts, ranging several time points in the future, the currently valid values are to read. In addition to these time-referenced relations, the relations of the variables must be described to the device instances and its locations.

The aforementioned requirements are best met by a **relational database.**
A relational database contains a collection of normalized tables with the following properties [9]:

- number of columns (⁻ attributs) of a table is fixed
- number of rows (= tuples) of a table is variable
- every table row has an unique identifier
- table rows are put in any order
- table entries are atomic, i.e. only one value of an attibute is content of a table element
- all tables are in a certain relationship with each other
- all tables are jointly managed.

For SEEDS the relational database management system **MySQL** was evaluated and used in the test implementation described in section 4. MySQL is available as open source software as well as a commercial enterprise version for different operating systems and is used worldwide.

During evaluation, the data volume which must be stored was estimated for the Stavanger Pilot. This data volume comprises about 3 GByte a year for approximately 1900 signals and can be stored in a laptop-based MySQL system. The query all data points from the last hour of all 1900 signals takes altogether 13 seconds. Due to this features, MySQL was chosen to store the temporary data during the control operations.

3.3.2 Historical Data for Training Self-learning Components

For training the self-learning components, precisely the NNs for forecasting, a large number of time series is required. These time series can be gained by recording the values of the control and process variables in their chronological sequence. With the values of energy consumption it is equally proceeded. The energy consumption is measured in the building or computed by the SEEDS controller.

For the storage of very large data volumes, **openTSDB** is a suitable data base. OpenTSDB is an open source, distributed and **scalable time series database** [10]. It uses HBase for storage. OpenTSDB's GUI provides all basic functionality including some management functionality, basic analytics and plotting. Collected data can be accessed either through OpenTSDB API for easy data extraction or directly through HBase interfaces. HBase level access allows performing additional analysis in a scalable way.

OpenTSDB was originally design for fine grained, real-time monitoring of data traffic in computer networks and data centers. Since then it was adapted for sensor networks data. In case of data that do not obviously conform to time series format e.g. occupancy schedules or weather forecasts

applicability of OpenTSDB is based on proper data representation. In SEEDS project we were successful with storing multi-layer weather data in OpenTSDB, but occupancy schedules have not been evaluated.

Therefore, application of OpenTSDB in SEEDS is focused on data archiving for: training or retraining of the NNs for forecasting, quick visualization of energy consumptions and comfort conditions over long times and ultimate evaluation of project results. The last functionality is enabled by OpenTSDB integration is statistical processing frameworks (e.g. R).

3.3.3 Structural Data for Configuration of the SEEDS BEMS

For each SEEDS application, the interfaces of Controller, Front Panel, and WISAN and the layout of Archive have to be adapted. That should be done automatically with special configuration software. Basis is the information about the structure of both the building and the automation system. This information is stored by means of **Microsoft Access**, which is a database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools [11]. The database engine is less powerful than MySQL because it can only store static data, but its GUI is extraordinarily user-friendly. For instance, all devices, which the composite structure diagram of Helicopter Garage (figure 7) shows, are registered in the Access database. Data are added to the sensors and actuators as well as for site description.

3.4 Interfaces

3.4.1 Controller to Archive

The archive fulfills the tasks:

- Data coupling between the controller and the process components
- Synchronizing of the concurrent operation of the controller and the process components.

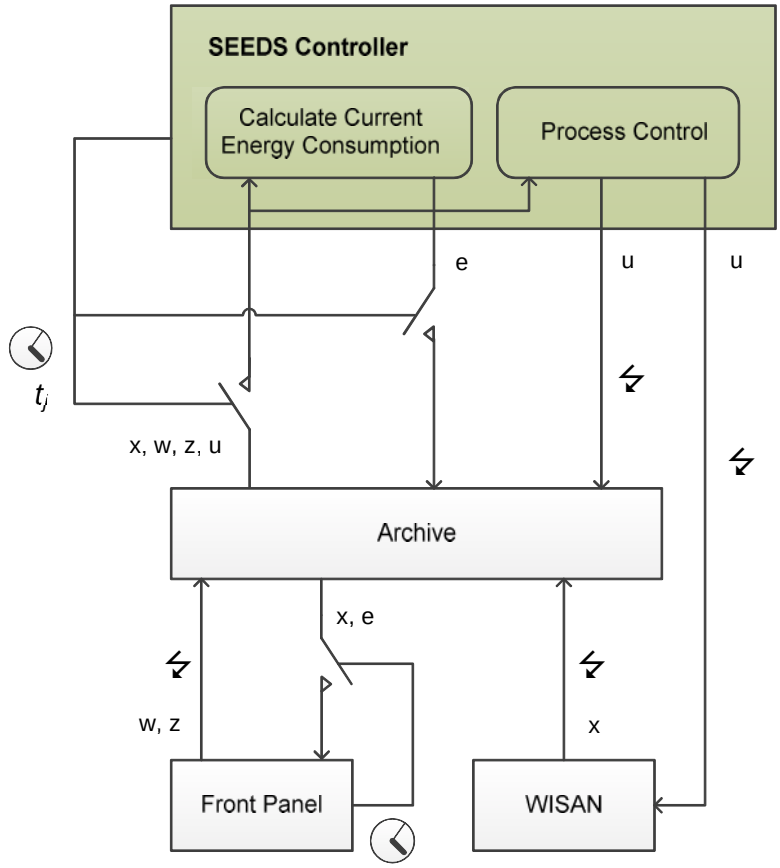

Figure 32: Data IO for energy calculation and process control

The data coupling is realized by storage of temporary data and its reading and writing by specialized access methods (figure 32). The basis is formed by time related data classes of process, control, and load variables. An overview is given in section 3.2.1.1. The report [12] *D5.4 – Specification and implementation of interfaces that have been integrated and tested* informs in detail about the interface implementations.

The synchronization between the controller and the process components is achieved by tagging all entries in Archive with a timestamp. The controller queries Archive periodically by means of its access methods under specification of a time point or time period. The time points or time periods can refer to presence, past, and future.

3.4.2 Controller to WISAN

If the controller has computed new optimised control settings, then these values have to be transmitted to the corresponding actuators immediately. This transmission is carried out event-

driven by a spcialized method of the WISAN Web Service. Beside the numerical value, the names of the node and the actuator must be specified by the controller.

Detailed information about the WISAN communication infrastructure is given in the report [6] *D4.3 – Plug & Play conformance requirements: API, Integration Webservices and libraries.*

3.4.3 Front Panel to Archive

The Front Panel is the graphical user interface (GUI) between the user and the SEEDS BEMS. Via the GUI the user inputs comfort settings, certain device settings, and load variables like occupancy schedule or weather conditions into Archive. Reversely, the GUI reads the values of comfort, the energy consumption, and certain device states from Archive and shows these values on the screen.

The GUI uses ADO.Net which is a set of software components within Microsoft .NET Framework to access and modify data stored in relational database systems. For the data input the GUI initiates event-driven INSERT queries. The variables that are displayed on the screen are periodically read from Archive. This is realized by periodically sending a SELECT query to Archive. Detailed information about the GUI implementation is given in the report [5] *D6.3 – Specification of Hardware and Software Platform.*

3.4.4 WISAN to Archive

The WISAN stores the last value of each sensor/actuator in Archive. The WISAN Server uses the ADO.NET Entity framework to update the values of the sensors/actuators in Archive. Detailed information about the WISAN Server implementation is given in the report [13] *D4.4 – Communication infrastructure and tool support implementation.*

4 Test Implementation

4.1 Helicopter Garage

4.1.1 Building Structure

Helicopter Garage is located in Spain, Barajas Airport Madrid. The main use of this building is parking and reparing of helicopters. The building has two floors. The main room of the building is the helicopter hall. This room has a volume of about 3200m^3 and extends over two floors. On the ground floor (figure 33) there are four additional rooms: kitchen, dining room, gym room and garage. On the first floor (figure 34) there are other five office rooms. Due to its size, the helicopter hall has three fan coils for the heating and cooling, the rest of the rooms have only one fan coil. The energy producing plants are outside the building.

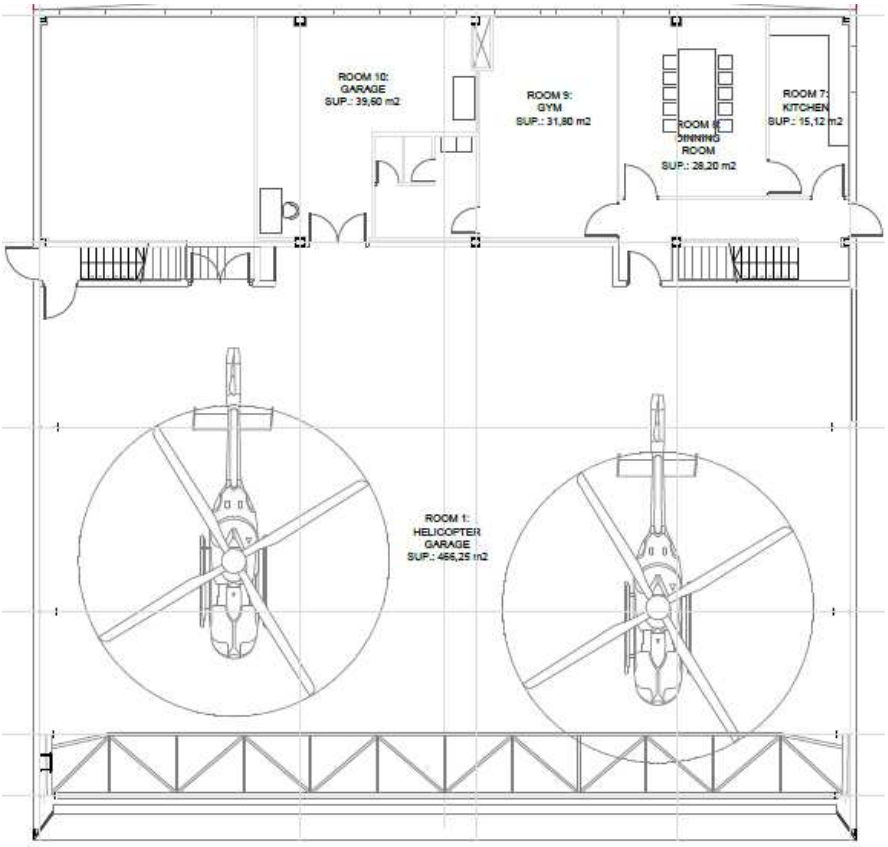

Figure 33: Ground floor of Helicopter Garage

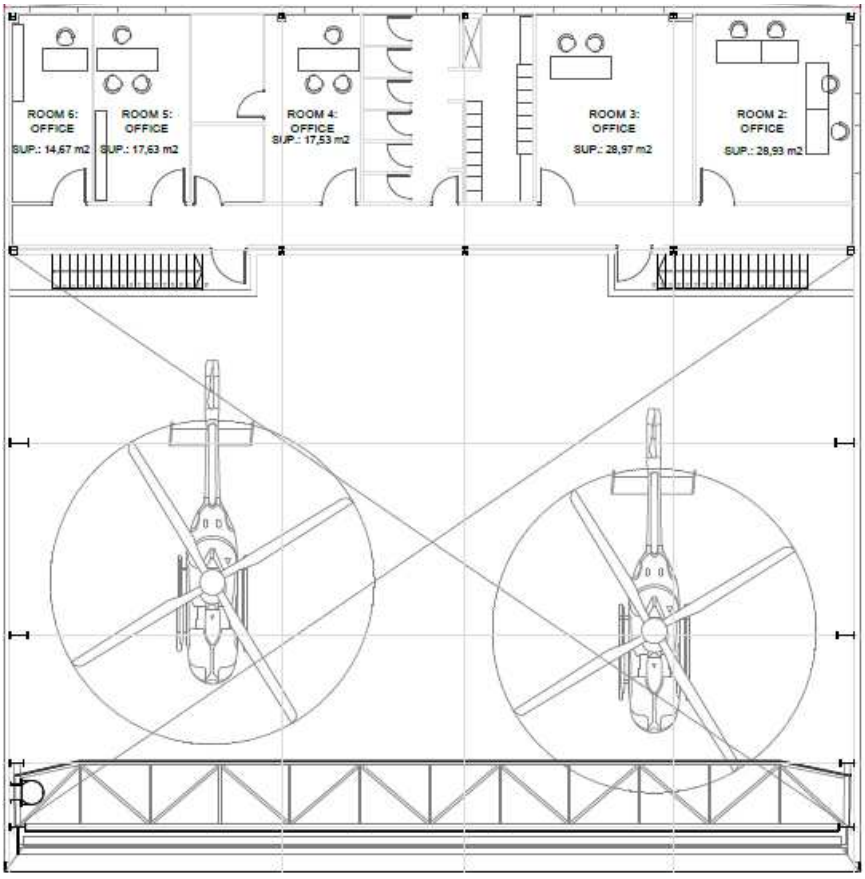

Figure 34: First floor of Helicopter Garage

Detailed information about Helicopter Garage is given in the report [1] *D2.3 - Modelling Methodology. ANNEX B - Application of SEEDS Modelling Methodology in an example (Helicopter Garage, HG).*

4.1.2 Energetic System for Air Conditioning

The energetic system for air conditioning (figure 35) can be divided into three subsystems:

- Thermal energy distribution system
- Cooling plant
- Heating plant.

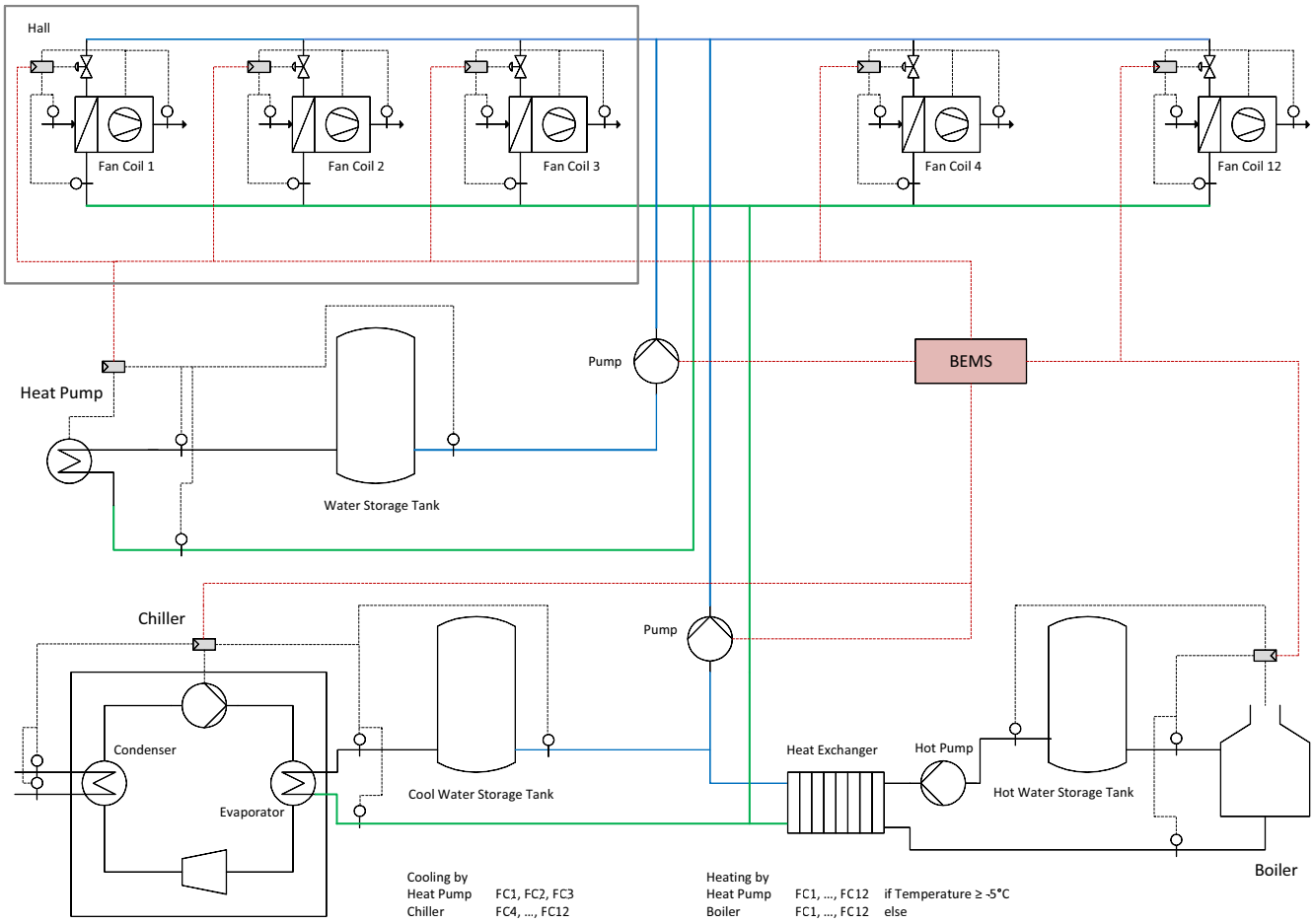

Figure 35: PI Diagram of Helicopter Garage

The thermal energy distribution system is based on fan coils. There are 3 fan coils in the hall and 1 fan coil in each one of the other rooms. The cooling plant (used in summer time) is made up of the heat pump and the chiller, where the heat pump supplies the fan coils in the hall and the chiller supplies all other fan coils. The heating plant (used in winter time) includes the heat pump and the boiler. If outdoor temperature is over -5°C, the heat pump heats all rooms; but if it falls below -5°C then the boiler heats all rooms. Each water cycle has a tank to store thermal energy for an efficient performance of heat pump, chiller, and boiler.

4.1.3 Building Model

In the test implementation, described in section 4.2 and 4.3, the behavior of the building during the **summer period** is considered. Therefore, the building model is reduced to the components of the energy distribution system and the cooling plant (figure 36). These components are instantiated and parameterized and their energy consumption will be calculated.

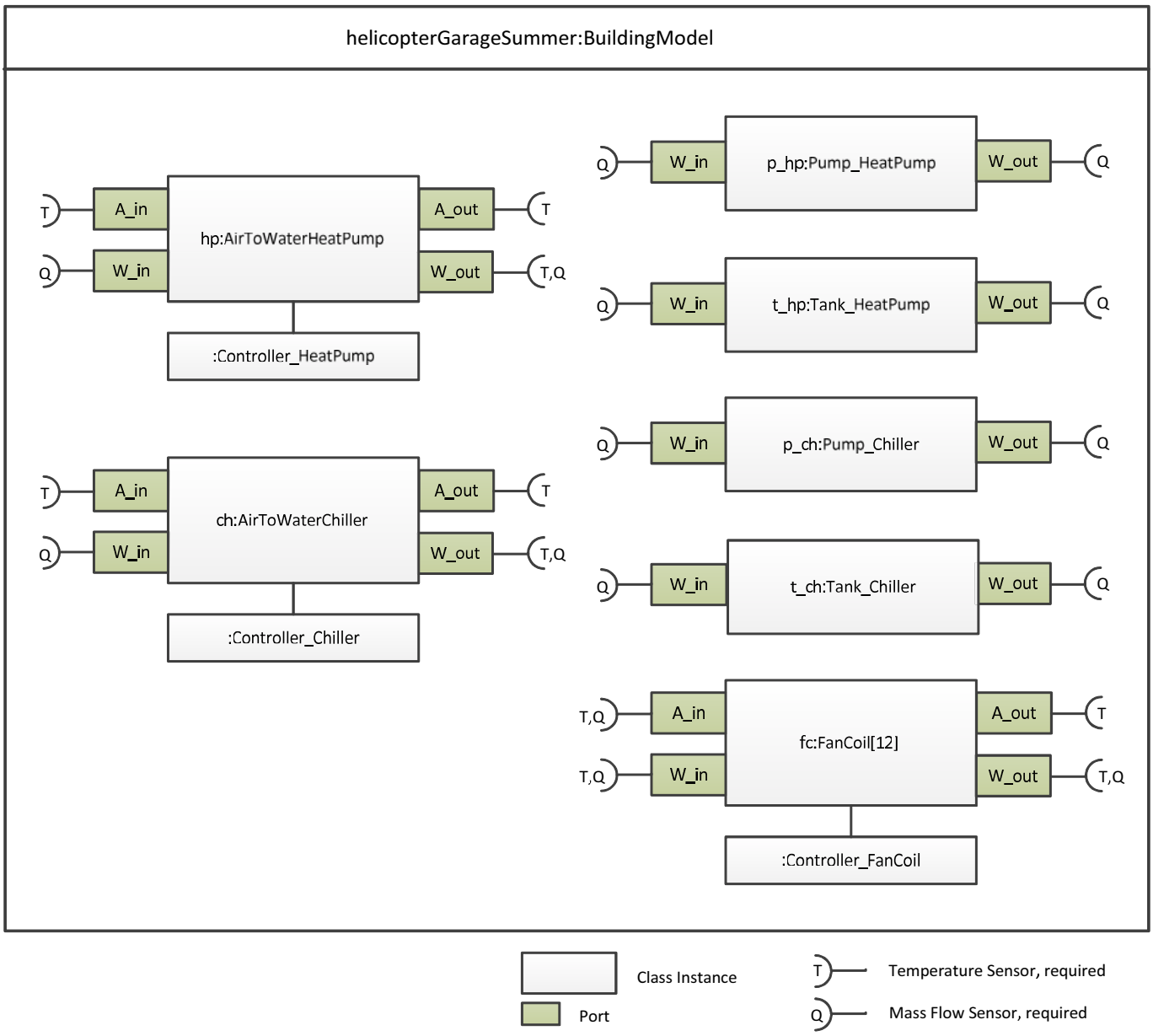

Figure 36: Instantiated building model of Helicopter Garage

4.1.4 Self-learning Components

According to the comfort variables which are to be controlled in Helicopter Garage example, the self-learning component is instantiated multiple times. Table 6 defines the input and output of these components. In addition, the self-learning component is instantiated for the energy demand prediction. Table 7 defines the input and output of this component.

4.2 Test Bed

To test the continous calculation of energy consumption and the process control which are performed by the SEEDS Controller, a physical building model of Helicopter Garage was developed. It calculates the process variables, especially the values of comfort, as reponse on the control settings, the comfort settings, weather and occupancy. In SEEDS Controller the software implementations of the SEEDS core components are instantiated and set up for controlling Helicopter Garage. Furthermore, Archive and Front Panel are included in the test bed (figure 37) in the target implementation. Only the WISAN component is omitted; its communication is replaced by inputs and outputs of the physical building simulator.

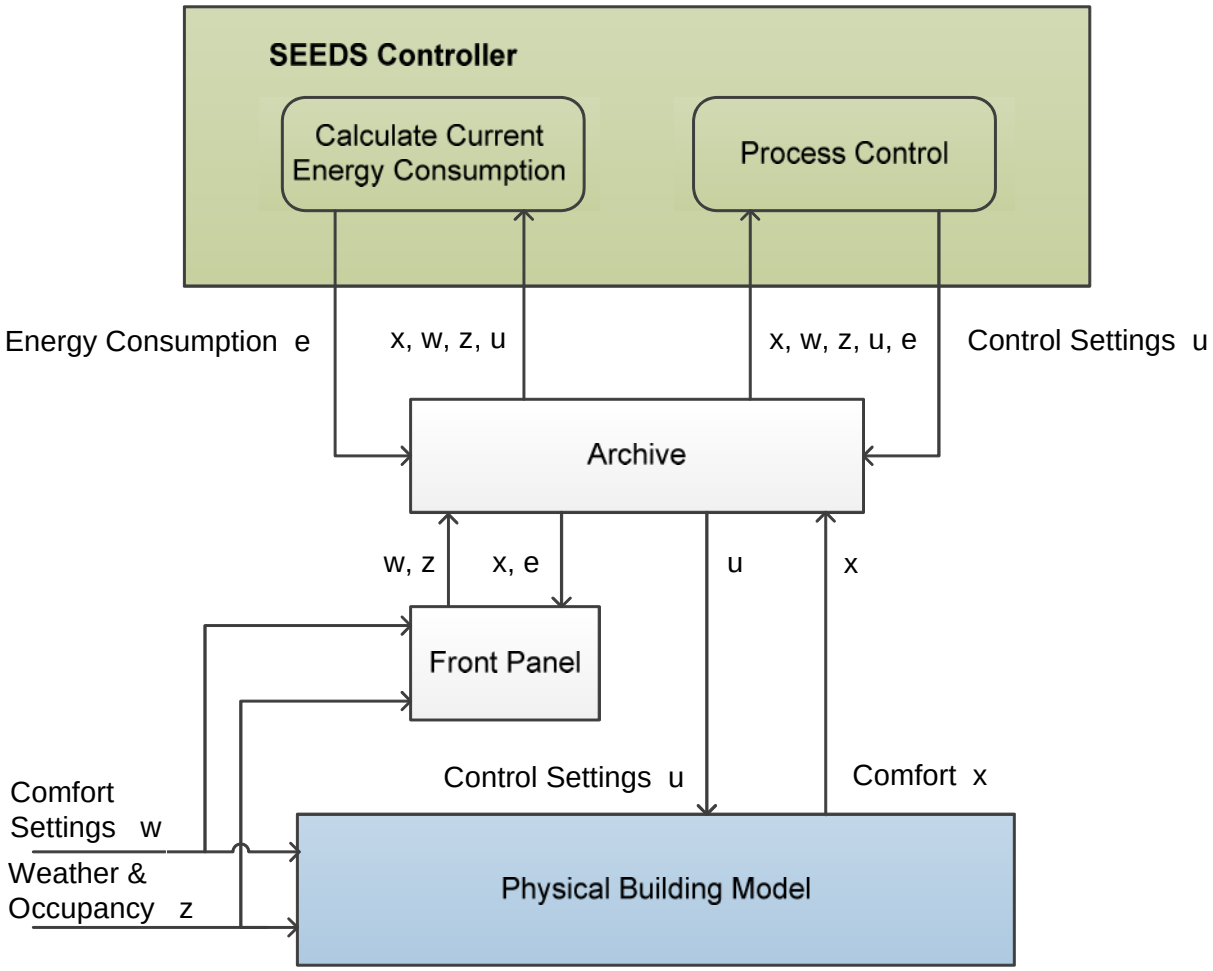

Figure 37: Architecture of SEEDS Test Bed

The **physical building model** is formed from components or blocks, which are described in the Modelica language. Figure 38 shows a detail of Helicopter Garage model [14]. In the room models, it is included both temperature balancing and thermal energy entries given by the fan coils, the outdoor condition (outside temperature, sun radiation), and occupation. In the model of the cooling circuit, the device models of heat pump, chiller, tanks, and pumps are included. Curves of outside temperature, solar gain and occupancy are added. The control variables (control settings) are the load levels of the heat pump and the chiller, which vary in the range [0.25, 0.5, 0.75, 1.0] and the speed levels of the fan coils, which vary in the range [1, 2, 3].

The full model is outlined in report [15] *D2.5 - Optimized Models for the energy system and for sub-systems verified and optimized for the demonstrator.*

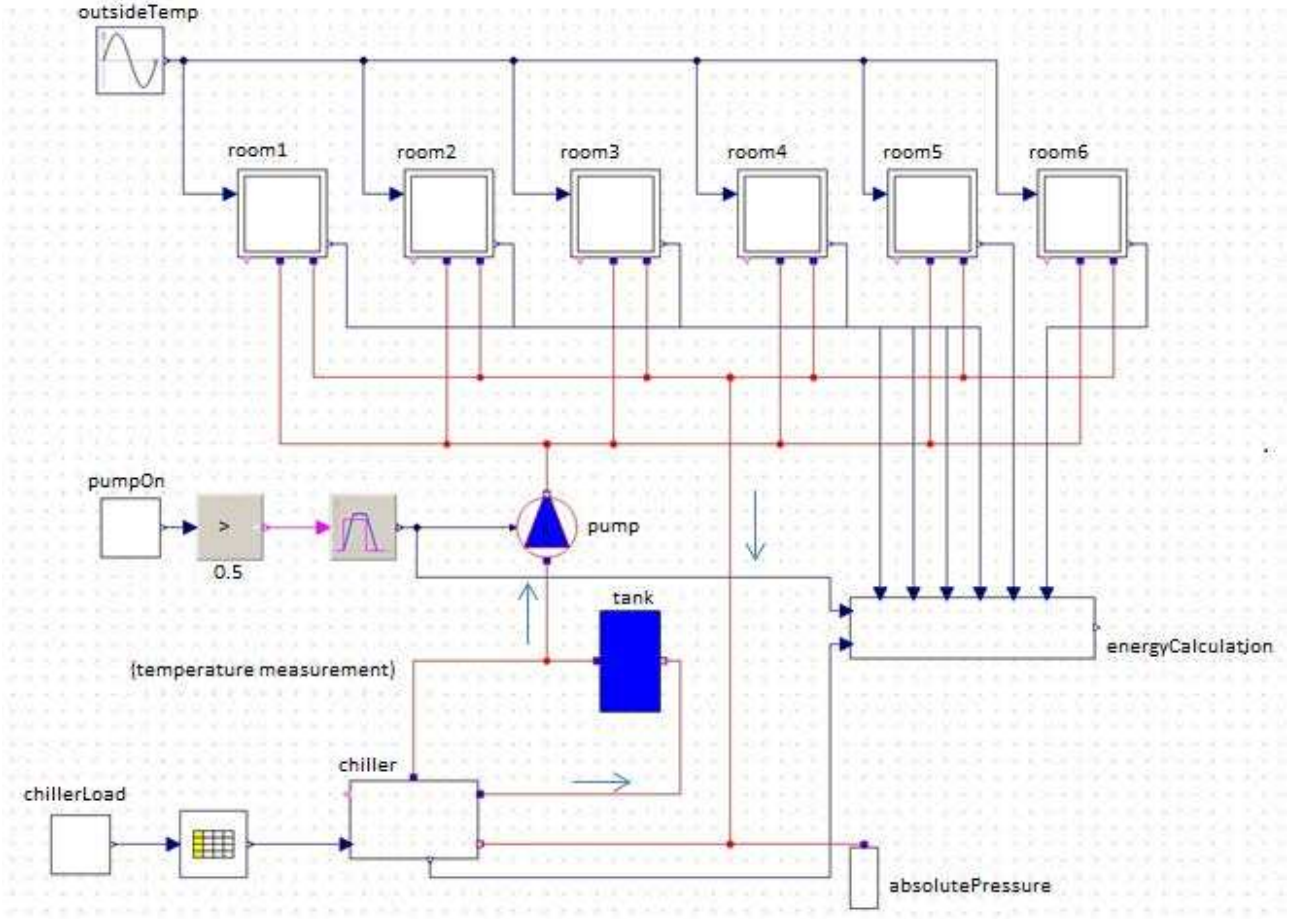

Figure 38: Detail of the Modelica model of Helicopter Garage

The following simulation results are provided:

- Air temperature of the helicopter hall (room 1)
- Air temperature of a selected office (room 2)
- Air temperature of the dining room (room 8)
- Supply water temperature of the chiller
- Supply water temperature of the heat pump
- Total energy consumption of Helicopter Garage.

The corresponding time series are written in Excel sheets and are used to adjust the neuronal networks for forecasting.

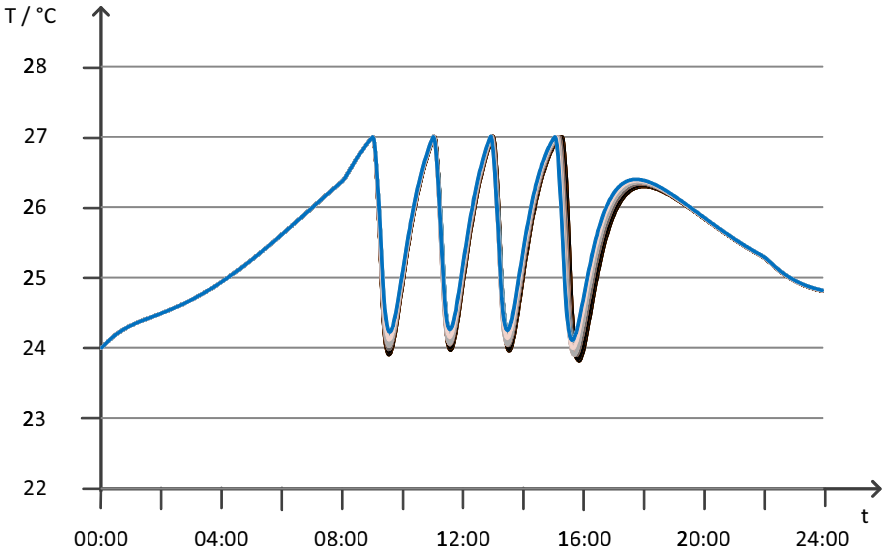

Figure 39: Air temperatures of helicopter hall (room 1)

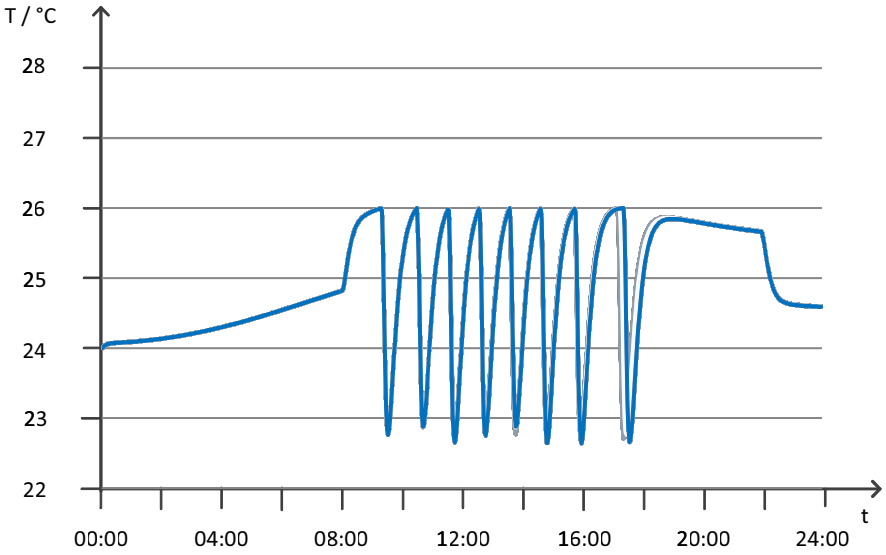

Figure 40: Air temperatures of office (room 2)

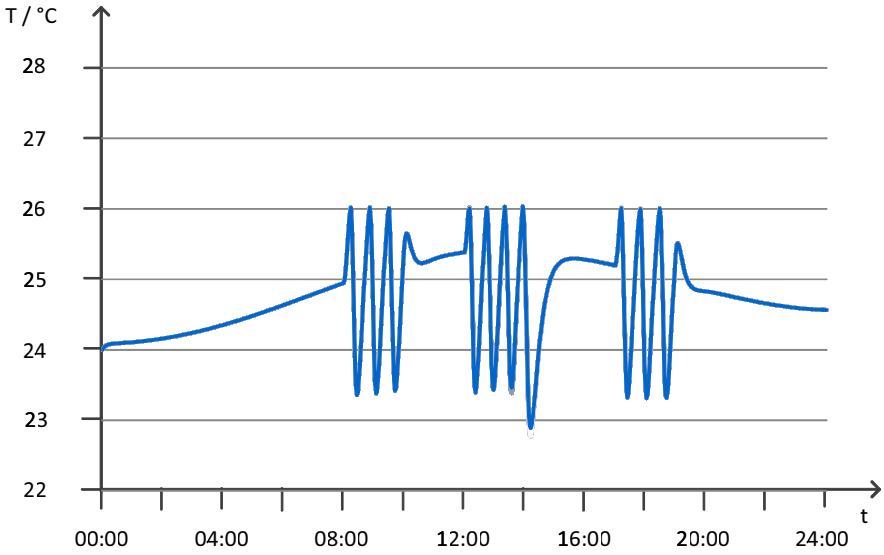

Figure 41: Air temperatures of dining room (room 8)

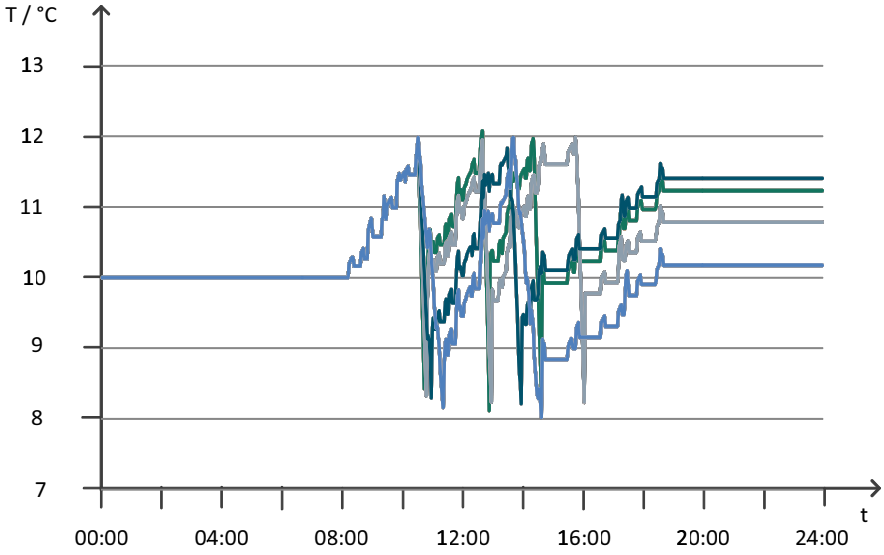

Figure 42: Water supply temperatures of chiller

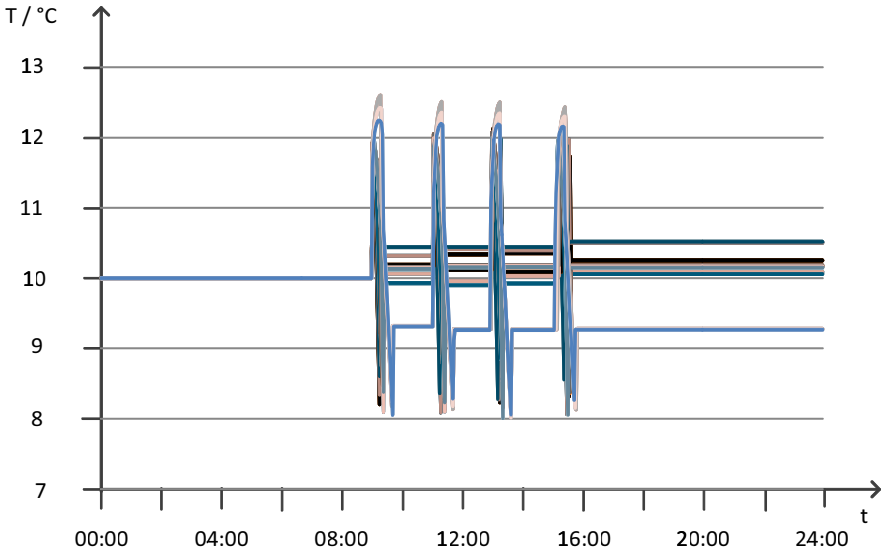

Figure 43: Water supply temperatures of heat pump

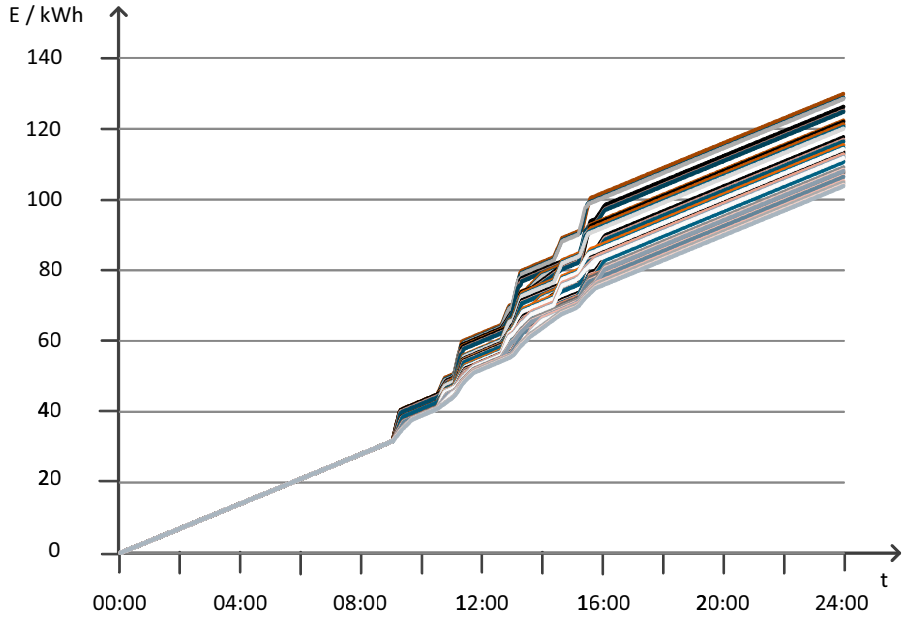

Figure 44: Total energy consumptions of Helicopter Garage

4.3 Evaluation of Energy Calculation

The PI Diagram of Helicopter Garage (figure 45) shows that each device of the equipment has its own local controller. All these controllers are On/Off-controllers with hysteresis. They form the control loops for air conditioning in the rooms and the control loops for cooling and heating water production (figure 45).

There are only two additional possibilities of the BEMS for energy minimization:

- Variation of the load profiles of the devices
- Variation of the set point temperatures of the local On/Off-controllers.

In the first case, the load levels of the heat pump and the chiller and the speed levels of the fan coils are the control variables. In the second case, the positions of the hysteresis and thus their upper and lower switching bounds are manipulated.

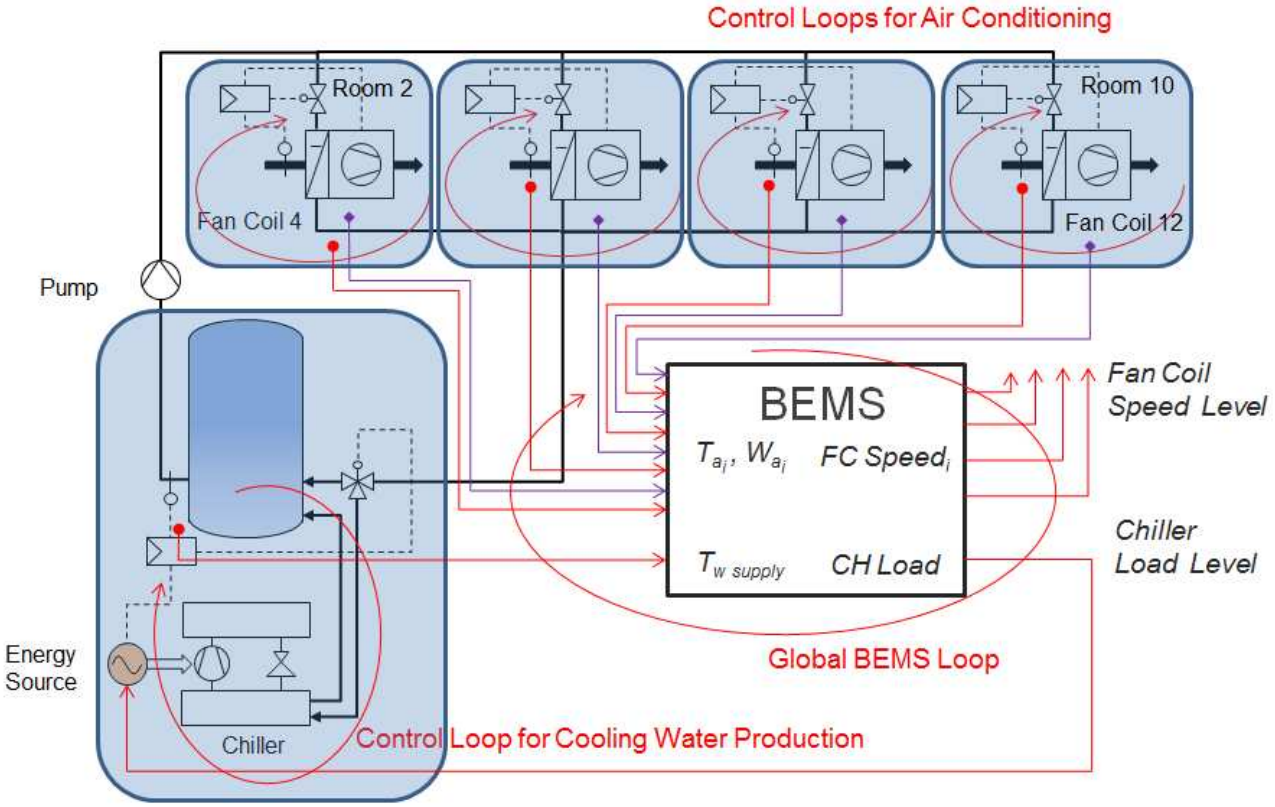

Figure 45: Hierarchy of the SEEDS Controls

To check the SEEDS's energy calculation the first case was chosen. The load levels of the heat pump and the chiller vary in the range [0.25, 0.5, 0.75, 1.0] and the fan coils' speed levels diversify in the range [1, 2, 3].

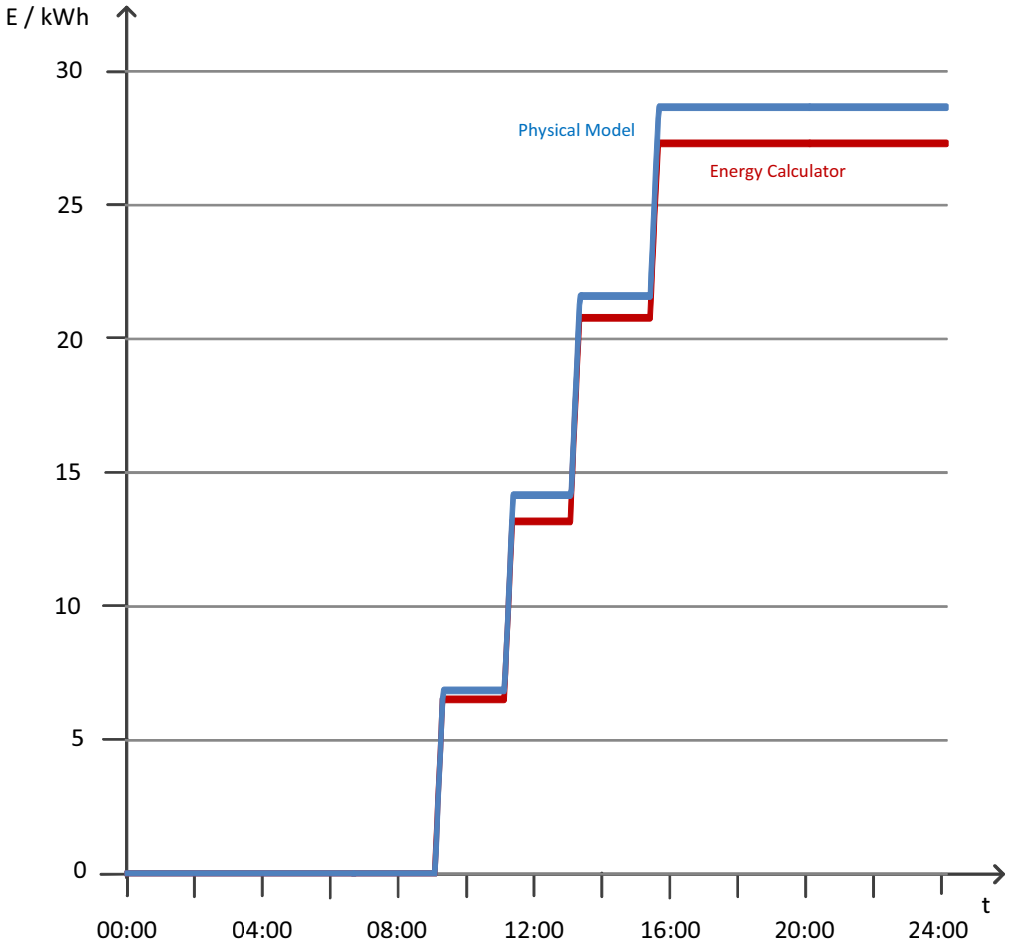

Figure 46: Energy consumption of heat pump

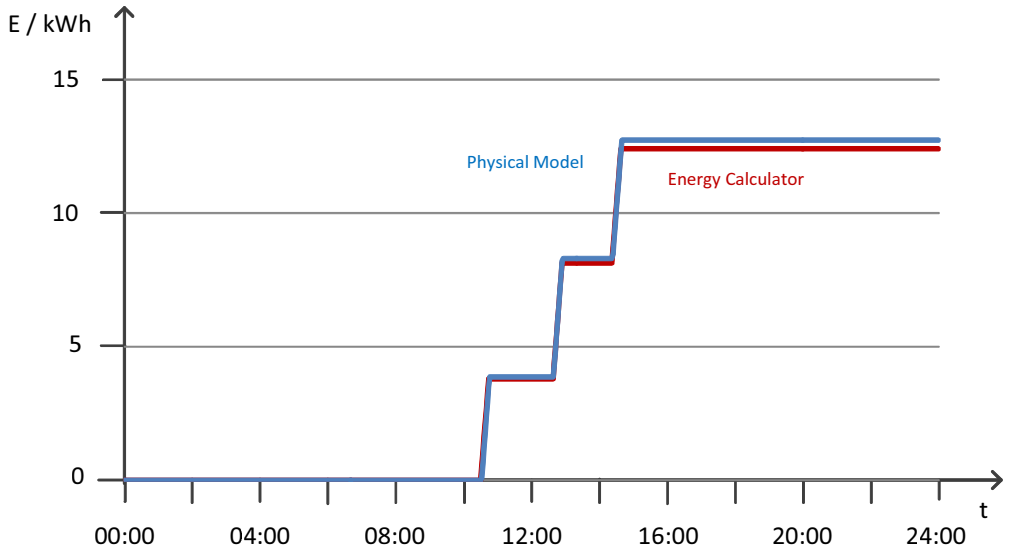

Figure 47: Energy consumption of chiller

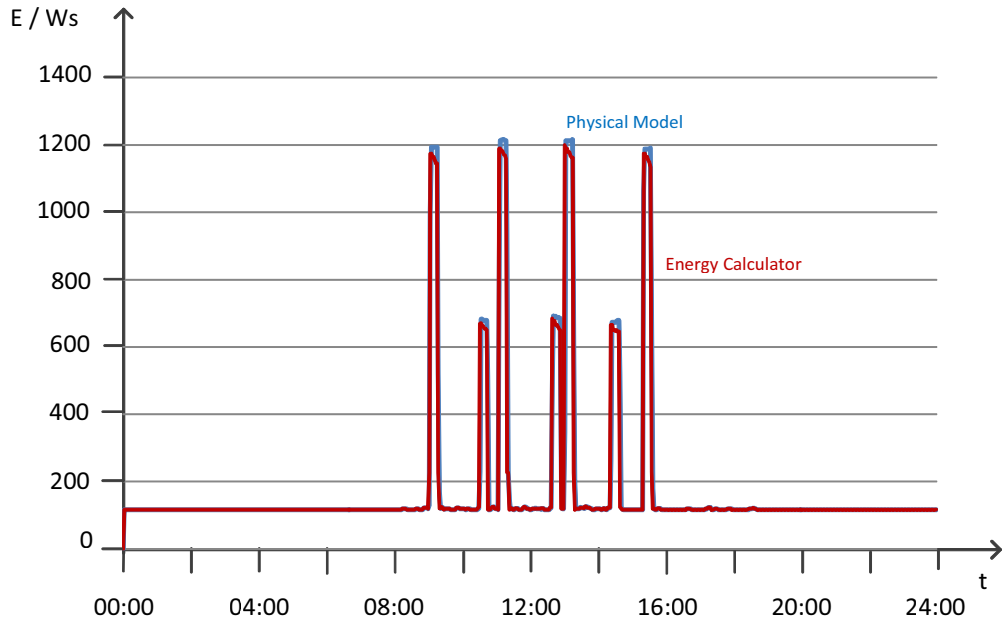

Figure 48: Energy entry into Helicopter Garage within sample periods

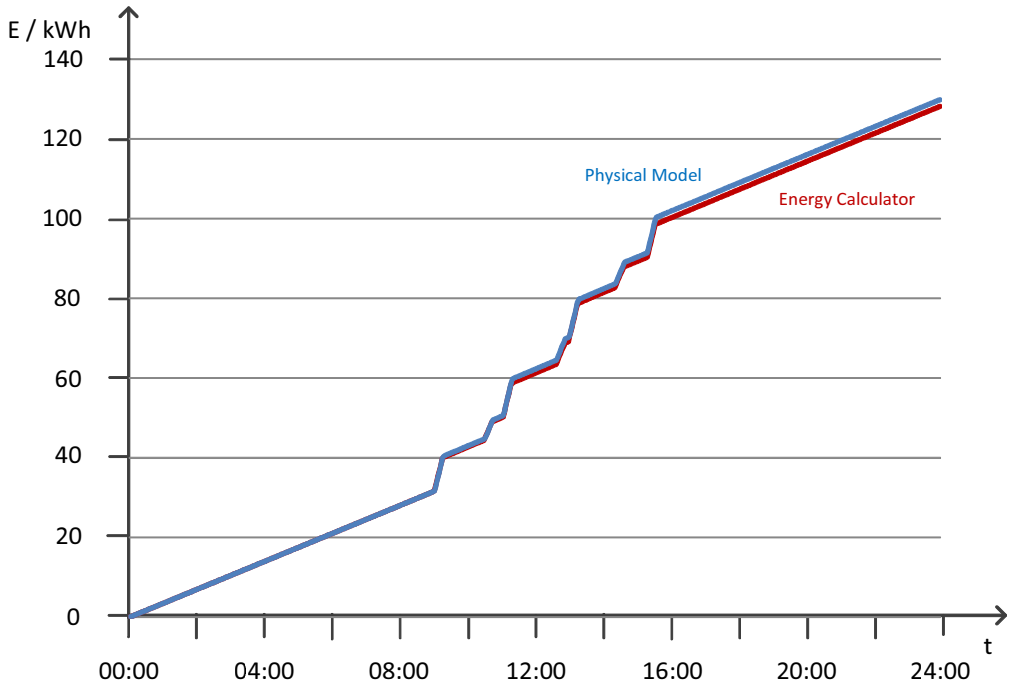

Figure 49: Total energy consumption of Helicopter Garage

Figures 46 to 49 contrast the **outcomes** of the physical model simulation and the SEEDS's energy calculation for the operation with full load of heat pump and chiller (HP_LL=CH_LL=1.0) and highest fan coil speed levels (FC_SL=3). There are only minor differences. That was achieved **by reading the device states**, on or off, by the energy calculator from the simulated devices in a sample period of 2 min. The alternative, to calculate the device states from the sampled cooling water temperatures, shows differences in the energy calculation of the chiller. The causes lie in the small time constants for the energy input in the water pipeline network and in the loss of data by sampling. The small time constants are a lack of the physical model, in reality, they are

considerably larger. This issue is discussed in more detail in [15] *D2.5 - Optimized Models for the energy system and for sub-systems verified and optimized for the demonstrator. 4.3 Evaluation of the Energy Demand Calculation.*

4.4 Load Scenarios for Optimization

To validate the implementation of the controller by means of limited resources, a minimal control task has been selected. This task is the temperature control of the helicopter hall (room 1), a selected office room (room 2), and the dining room (room 8). That interaction between Controller as whole, Archive, and Front Panel has been tested successfully. In addition, the data links among the internal core components Controller, Optimizing, and Self-learning are established.

In the following two different load scenarios for optimization are investigated.

In the **first scenario**, the control variables are:

- Load level of the heat pump, HP_LL ∈ [0.25, 0.5, 0.75, 1.0]
- Load level of the chiller, CH_LL ∈ [0.25, 0.5, 0.75, 1.0]
- Speed level of the fan coil in the dining room, FC_SL ∈ [1, 2, 3].

The optimization starts 8:00 and ends 19:00, coupled with the usage of the dining room. The optimization period is 30 minutes. At each of these time points of the optimization, the energy consumption is predicted for the duration of 30 minutes. For example, figure 50 show all possible values of energy consumption at 11:00 in the next 30 minutes corresponding to all combinations of the values (4x4x3) of the control variables. That is, each column of the chart is based on a vector (HP_LL, CH_LL, FC_SL). In figure 50, the green column highlights the prediction of minimum energy consumption with the vector (HP_LL=0.25, CH_LL=0.25, FC_SL=2). The selection from several minima is performed by an additional weighting of the energy consumers. The predictions are computed by means of the physical model of Helicopter Garage.

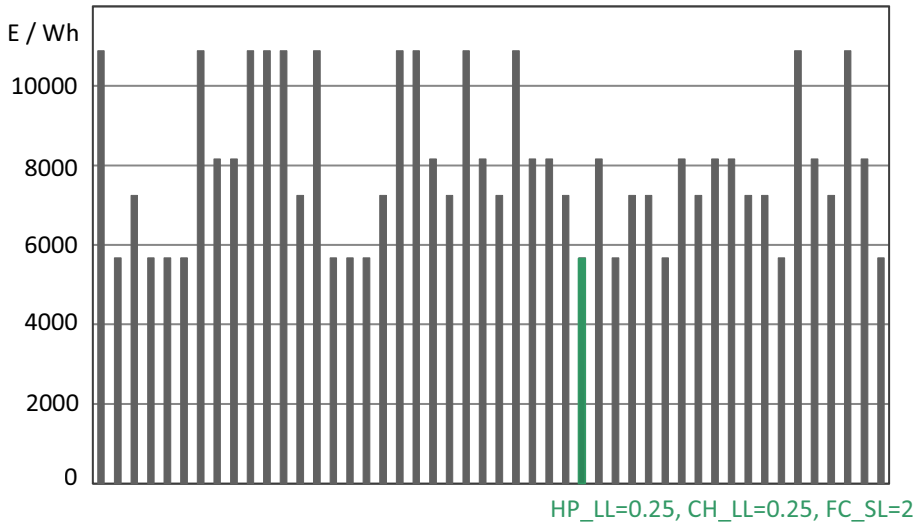

Figure 50: Variations of energy consumption prediction at 11:00 for 30 minutes

If the column with the minimum energy value at all time points of optimization is chosen and the minimum energy values are accumulated, we obtain the curve of minimum energy consumption (figure 51). If the same is done with columns with the maximum energy values, we get the curve of maximum energy consumption (figure 51). These two curves bound the search space of optimization.

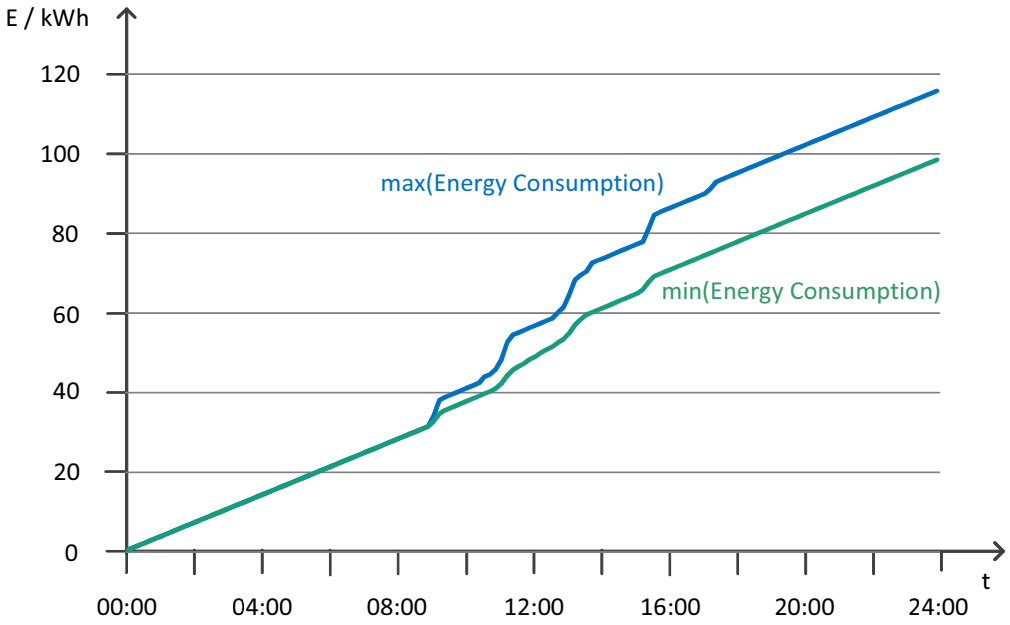

Figure 51: Minimum and maximum energy consumption in the optimization intervals

Starting 8:00 and finishing 19:00, the optimizer selects the vector of the control variables with the minimum value of energy consumtion for all 30 minutes. Due to the small number of control variables, the optimizer uses the grid search. Both the prediction of energy consumption and the response of the building are calculated by means of the physical building model of Helicopter Garage. Figures 52 to 56 show the results. The accumulated energy consumption is depicted by the green curve in figure 51.

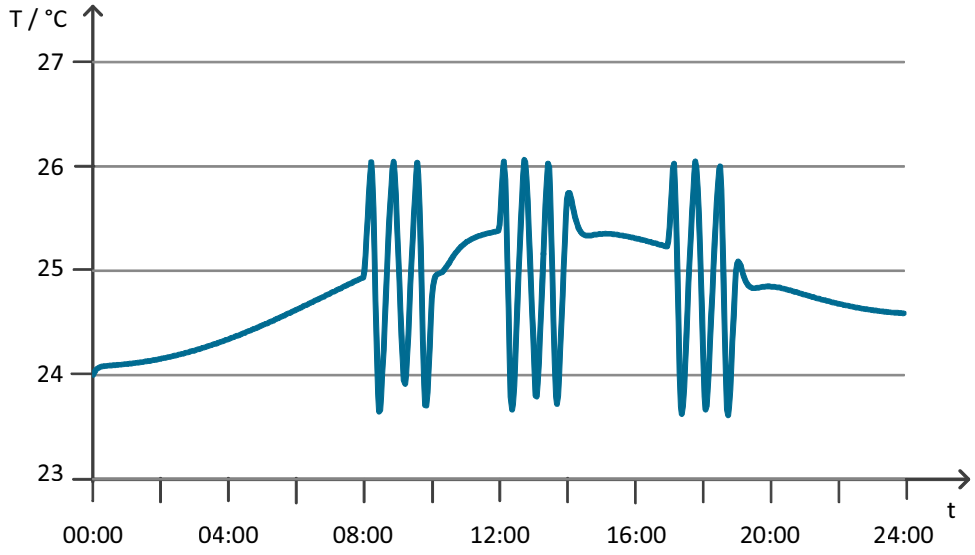

Figure 52: Temperature curve of the dining room

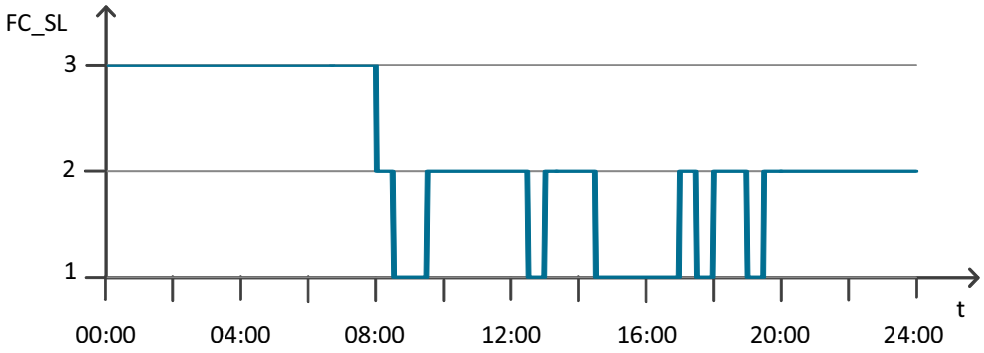

Figure 53: Adjusted fan speed levels in the dining room

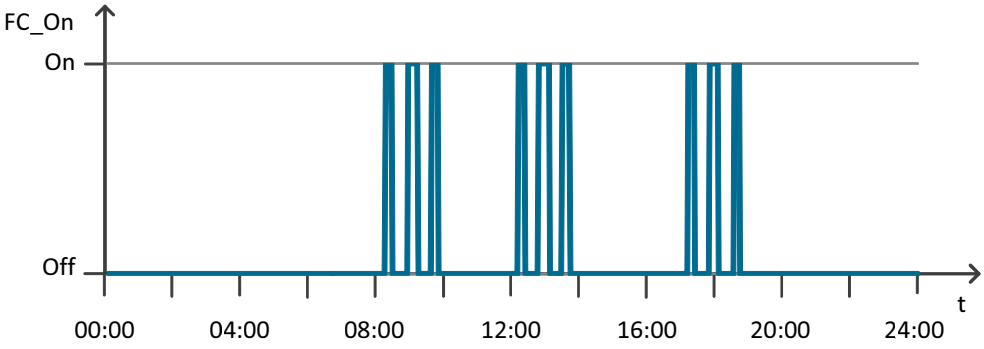

Figure 54: Operation of fan coil in the dining room

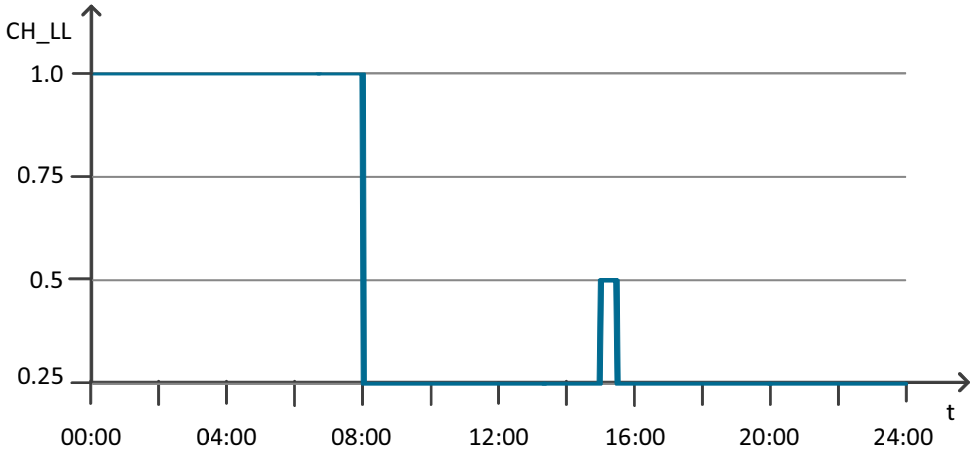

Figure 55: Adjusted chiller load levels

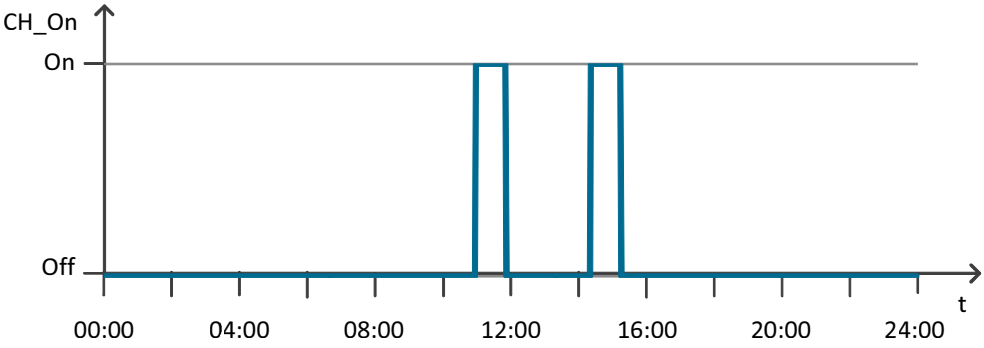

Figure 56: Operation of chiller

In the **second scenario**, the control variables are:

- Load level of the heat pump, HP_LL ∈ [0.25, 0.5, 0.75, 1.0]
- Load level of the chiller, CH_LL ∈ [0.25, 0.5, 0.75, 1.0]
- Set point temperature of the heat pump for cooling, HP_TC = [10°C…18°C], in steps of 2°C and switching differential for cooling HP_SDC = +/- 2°C.
- Set point temperature of the chiller for cooling, CH_TC = [10°C…18°C], in steps of 2°C and switching differential for cooling CH_SDC = +/- 2°C.

The speed levels of all fan coils are fixed at FC_SL = 3.

The goal is to expand the search space for optimization and to increase the sensitivity of the energetic system towards changes of the control variables. Figures 57 and 58 show the simulation results for 2 days. The background of figure 57 is the run of the energetic system without SEEDS control. Heat pump and chiller load level are fixed at HP_LL = CH_LL = 1.0; their set point temperatures are set on HP_TC = CH_TC = 10°C. At the end of the second day the energy consumption amounts about 274 kWh.

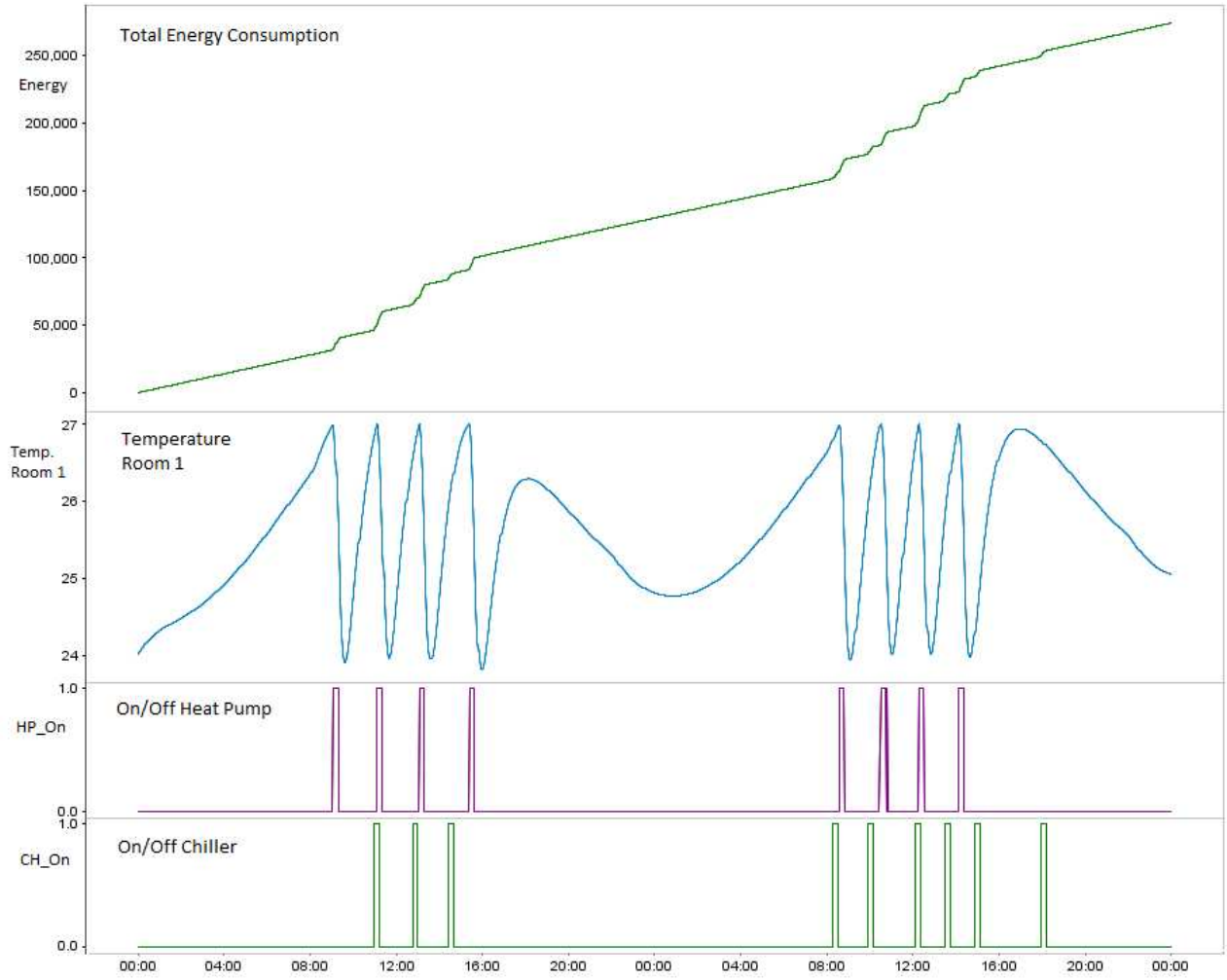

Figure 57: Energy consumption of the Helicopter Garage for HP_TC = CH_TC = 10°C

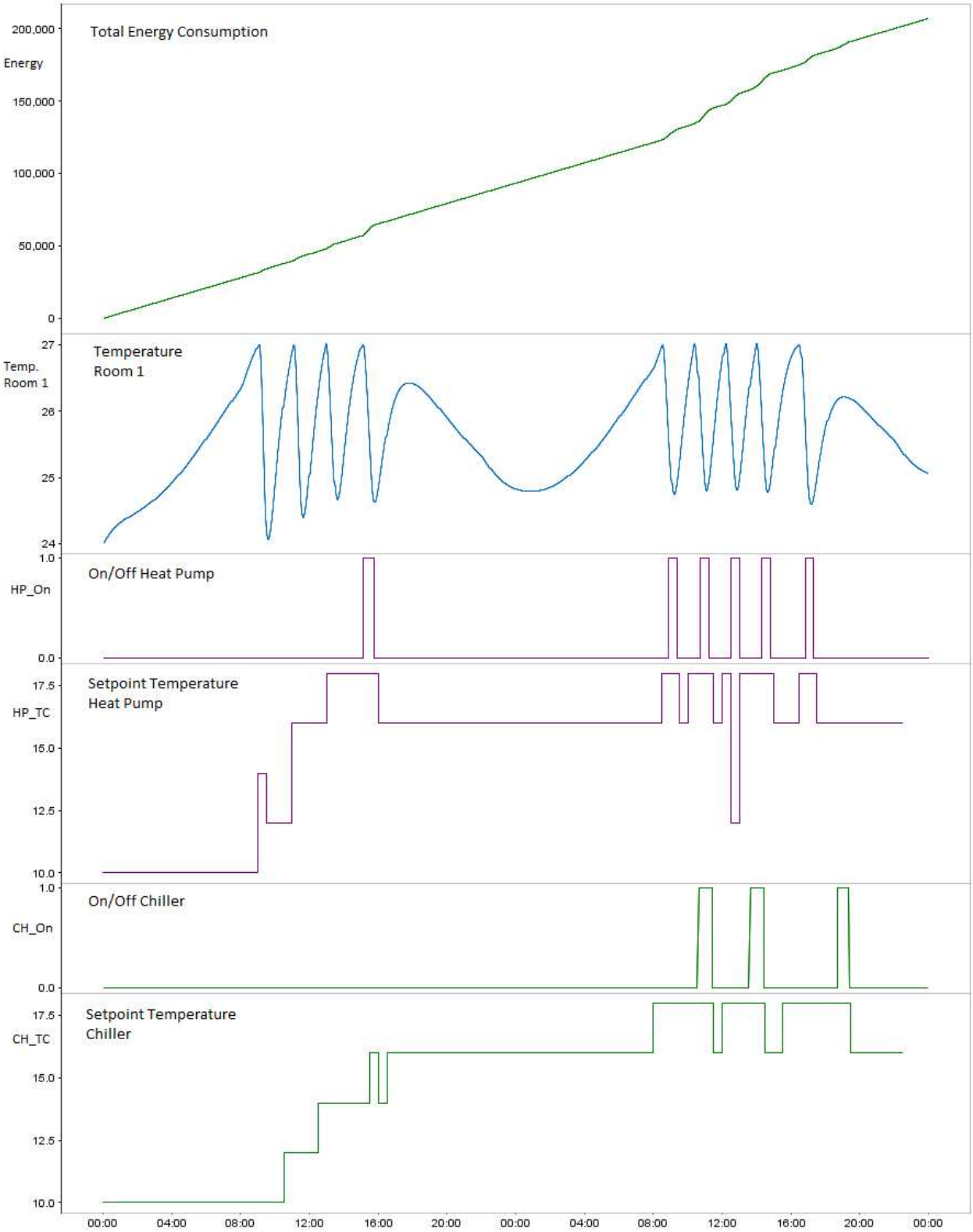

Figure 58: Energy consumption of Helicopter Garage for optimized HP_TC and CH_TC

The background of figure 58 is the run of the energetic system with SEEDS control under variation of the control variables as described above. The optimization is carried out from 7:00 to 22:00 each day. The room occupancy and weather conditions are the same on both days. At the end of the second day the energy consumption now amounts about 207 kWh which corresponds to an energy

saving of about 24%. Compared with figure 1, figure 2 shows a substantial reduction of the heat pump and chiller operations (HP_On and CH_On) while maintaining the room temperature.

At this point it must be emphasized that the exclusive utilization of the physical model for large objects such as the demonstrators in Stavangar and Madrid cannot be used due to the high computation time. The goal here is to provide a comparison for the self-learning based optimization in SEEDS.

5 Bibliography

[1] Márquez, F.; Jiménez, N.; Barragán, A.: D2.3 Modelling Methodology. 2013

[2] Oestereich, B.: Analyse und Design mit UML2. Oldenbourg Verlag München Wien. 2005

[3] Jeckle, M. et al.: UML2 glasklar. Hanser Verlag München Wien. 2004

[4] Stenzel, P.: Validierung und Anwendung des IFC-Schemas für eine raumlufttechnische Anlage zur Generierung einer Energiebedarfsberechnung im Rahmen der Open-BIM-Initiative. Diplomarbeit, TU Dresden, Januar, 2013

[5] Peneva, R.: D6.3 Specification of Hardware and Software Platform. 2013

[6] España, J.; Díaz, F.: D4.3 Plug & Play conformance requirements: API, Integration Webservices and libraries. 2012

[7] Vadera, S. et al: D5.1 Report on applicability of different learning and optimization methods. 2012

[8] Vadera, S.; Wu J., Montague, S., Meziane, F.: D5.2 Self-learning algorithms that have been verified with sample scenarios and test data. 2013

[9] Rausch, C.: Relationale Datenbanken (DB): Das Relationenmodel. 23.05.2013 http://ab.inf.uni-tuebingen.de/teaching/ss03/asa/db_intro.pdf

[10] Wlodarczyk, T. W.: Overview of Time Series Storage and Processing in a Cloud Environment. IEEE 4th International Conference on Cloud Computing Technolgy and Science. 2012

[11] Wikipedia: Microsoft Access. 23.05.2013 http://en.wikipedia.org/wiki/Microsoft_Access

[12] Meyer, R.; Haufe, J.: D5.4 Specification and implementation of interfaces that have been integrated and tested. 2013

[13] Diaz, F. et al: D4.4 Communication infrastructure and tool support implementation. 2013

[14] Tuch, S.: Modellbasierte Minimierung des Energieverbrauchs einer raumlufttechnischen Anlage. Diplomarbeit (FH), HTW Dresden, September, 2012

[15] Wurm, J.; Donath, U.; Haufe, J.: D2.5 Optimized Models for the energy system and for sub-systems verified and optimized for the demonstrator. 2013

[16] Gen, M.; Cheng, R.: A survey of penalty techniques in genetic algorithms. Proceedings of IEEE International Conference on Evolutionary Computation, pp. 804-809, 1996

[17] Yeniay, Ö.: Penalty Function Methods for Constrained Optimization with Genetic Algorithms. Mathematical and Computational Applications, Vol. 10, No. 1, pp. 45-56, 2005.