Simulation Modelling Practice and Theory 17 (2009) 1143–1159

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Modelling thermal processes in buildings using an object-oriented approach and Modelica

Anton Sodja, Borut Zupančič *

University of Ljubljana, Faculty of Electrical Engineering, Tržaška 25, 1000 Ljubljana, Slovenia

ARTICLE INFO

Article history:
Received 1 August 2008
Received in revised form 26 March 2009
Accepted 2 April 2009
Available online 21 April 2009

Keywords:
Object-oriented modelling
Multi-domain modelling
Modelica
Intelligent building model
Thermal and radiation flows

ABSTRACT

Most of today's modelling and simulation concepts originate from the times and methods of analog computers. Usually, it is assumed that the model must be expressed in an explicit state-space form. Consequently, the topology of the system gets lost and any future extension and reuse of the model is tedious and error-prone. In other words, it is the modeller's task to consider the computational order of the operations during a simulation.

In this paper we discuss the re-implementation of a passive-solar- building simulator in an object-oriented environment; it was originally built in the non-object-oriented simulation environment of *Matlab–Simulink*. The former simulator was designed to resemble a real physical test chamber with regard to the thermal and solar radiation flows. However, due to the lack of object orientation in *Matlab–Simulink* it was very difficult to apply any configuration modifications and extensions.

We start with a brief description of the mathematical modelling which includes thermal dynamics and solar radiation. Then the implementation in *Modelica* is presented. So, a much superior environment in comparison with *Matlab-Simulink* was obtained, giving us the possibility of high-level modular and object-oriented modelling. The model is also extremely efficient in multidisciplinary projects in which control-engineering specialists (our group) cooperate with specialists from civil engineering, because civil engineers can more easily understand graphical and textual models in *Modelica* than schemes in *Simulink*.

We expect that such a model will fulfil and significantly improve several model properties in comparison to the *Matlab–Simulink* implementation, i.e., a better understanding of the influences of thermal and radiation flows on comfortable living conditions, a model-based control-system design, which will enable the harmonization of active and passive energy resources, important energy savings, and a very suitable environment for education in modelling, simulation and control.

1. Introduction

A model that reflects, as much as possible, the behavior of a real system is indispensable if we want to obtain deeper knowledge about the system. In addition, a good model is also very important for successful control design. Usually, complex systems result in complex models; however, it is extremely important to clearly define the modelling aims and to balance all the procedures with regard to a reasonable complexity and accuracy.

The literature from this area is very comprehensive. In particular, there are many studies dealing with traditional modelling. In addition to some papers from our group, which will also be referred to later [14,17], there are some interesting

* Corresponding author. Tel.: +386 14768306.
E-mail addresses: anton.sodja@fe.uni-lj.si (A. Sodja), borut.zupancic@fe.uni-lj.si (B. Zupančič).

1569-190X/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2009.04.003

recent publications that also deal with bond-graph approaches [16] and object-oriented approaches, mainly with the implementation of particular *Modelica* libraries. In [4] the thermal model of a building is described and implemented in *Modelica* and the so-called AT plus library was developed. The convection modelling is rather simplified. As in many other papers the final diagrams are rather ambiguous, especially the parts with solar radiation, which are implemented with many icons. There is also no discussion in this paper about the transfer of direct and diffuse solar radiation through the window and the distribution of direct radiation inside the room. In [12] the authors introduce a commercial simulation environment 'IDA ICE' as one of the first attempts to introduce new approaches with symbolic equations modelling. In addition, there is a comprehensive overview of simulators for buildings. In [6] a group from the Herman–Rietschel institute deals with an interesting ceiling solution, which includes the capillary pipes. In this case the process is modelled with finite elements. However, some parts are perhaps too simplified, with the emphasis being more on the implementation of the model rather than its development. Using a simulation it is shown that with new techniques, based on storing heat energy from rooms during the daytime and releasing it during the night-time, a low-energy-demand chilling can be realised.

Although the majority of the presented papers are focused on the thermal effects in buildings, which are modelled with a theoretical approach, there are also approaches with experimental or combined modelling. In [15,13] genetic algorithms for optimising the model parameters are used. There are also many interesting papers dealing with control [10]. They are particularly interesting for us as our modelling is also mostly intended for control purposes. In [7] the Dymola–Simulink interface, which gives the efficient possibility to use the Matlab environment (and Toolboxes) for control-system optimization, is presented. Such an approach is relatively universal as the Simulink model is used only for the evaluation of the criterion function. However, the Modelica concept enables more efficient optimization solutions, which are, unfortunately, still far from maturity usage (see [1] with a Modelica extension Optimica and appropriate extensions in Modelica compiler).

We also noticed a lack of papers dealing with the harmonization of thermal and illuminance effects [9].

In this paper we are dealing with the modelling of thermal and radiation processes in buildings. The basic aim of the model is to obtain a better understanding of the influences of thermal and radiation flows on comfortable living conditions and to use it for control-system design. This control system should act to harmonize the active and passive energy resources and so important energy savings can also be expected. Furthermore, a very suitable environment for education in modelling, simulation and control was expected.

Our previous activities were based on theoretical modelling of the thermal processes and illumination inside a "real test chamber", which was represented by a small house. An appropriate and well-validated simulator in the *Matlab–Simulink* environment [14] was developed. With the aid of the simulator the efficiency of the energy consumption was improved and comfortable living conditions with regard to the temperature and the illumination were ensured [8,9].

However, *Matlab–Simulink* is a rather conventional modelling tool; a lack of object orientation is its most significant drawback. As a result, the model has to be implemented, more or less, in a state-space form. This is a mathematical description that is very far from basic mass- and energy-balance equations, which are normally understandable by engineers. With such transformations the topology of the basic system is lost, which results in a less-intuitive and transparent model.

Furthermore, traditional modelling tools like *Matlab–Simulink* require the models to be causal, so we have to determine which variables are the inputs (causes) and which are the outputs (consequences) of the model components. This is quite a harsh restriction, often meaning that slight changes to the modelled system or an adaptation of an existing model to a similar system is a very time-consuming and error-prone task, unless the original model was designed with these specific extensions in mind. Our model, which was developed for a test chamber, was so restricted to a rectangular-shaped object, with one room and at most one window in a single wall. Merging two room submodels into a double-room building was impossible without a complete rearrangement of the equations. An additional drawback of *Matlab–Simulink* is its inability to easily incorporate the documentation of a model directly into that model. So the documentation is separated from the model implementation and, usually, the consequence is that the documentation is not up to date with respect to all the recent changes.

In order to obtain the significant support of a tool, also in the modelling phase, so as to achieve better flexibility of the simulator and to make it more generally applicable we decided to re-implement the former simulator in *Matlab–Simulink* in a very sophisticated object-oriented environment, *Dymola* [3], which uses the *Modelica* modelling language standard [11,5]. *Dymola* with *Modelica* is a high-level but general-purpose OO modelling tool, well proven in many complex applications. It is also extremely efficient and helpful in the modelling phase. However, such developed models can easily be used in the *Matlab–Simulink* environment, so one can benefit from all the experimental possibilities and toolboxes, which finally results in a really efficient modelling, simulation and experimentation environment.

Using the object-oriented modelling approach one can build complex model libraries on basic principles, i.e., physical laws and energy- and mass-balance equations, and such model classes can easily be reused in different configurations. So our design was focused on the development of a new library, rather than focusing on a single object (i.e., the test chamber). This approach provides easier future adaptations and extensions in complex intelligent-building modelling.

A very important advantage in the *Dymola–Modelica* environment, is the documentation feature, allowing annotations of formatted descriptions to be added to the model. So, the documentation and the model itself are just two different layers of the same model class, which always gives us the possibility to update both layers simultaneously.

As the variables are also objects, they become real physical quantities, with many other attributes: quantity name, unit, display unit, minimal and maximal values, etc. The environment also uses descriptions given in code at variable and parameter declarations. These descriptions are very helpful in the graphical-editing mode.

The emphasis in this paper is neither the model validation, which is the most important phase of each modelling, nor the use of the model itself. The model validation was extensively performed with the former *Matlab–Simulink* model, and the new *Modelica* library was validated with the aid of the *Simulink* results. The emphasis in this paper is on an illustration of how we can benefit from modern object-oriented approaches in comparison to the conventional ones. Of course it is clear that we have many intentions with the developed library. We expect that such a model will fulfil and significantly improve several model properties in comparison to the *Matlab–Simulink* implementation, e.g., a better understanding of the influences of thermal and radiation flows on comfortable living conditions, a model-based control-system design, which will enable the harmonization of active and passive energy resources and lead to important energy savings, but also as a very suitable environment for education in modelling, simulation and control.

2. Mathematical modelling

2.1. Thermal dynamics

In order to obtain a relatively simple model that can be used for simulator implementation, the thermal processes in buildings were modelled with lump parameters. The model consists of the building's interior (air mass and furniture), the building's envelope (walls, ceiling, floor and windows) and the surroundings. For additional simplification, the following assumptions were made:

- The wall layers and window panes were considered homogenous and small enough to neglect all the temperature gradients along the surface. The conduction of heat was therefore reduced to a 1-dimensional problem in the direction perpendicular to the wall layer.
- In the case of the "real test chamber", where the dimensions are relatively small, the temperature of the air mass inside can be sufficiently well represented by a single average temperature point. This assumption is, of course, questionable in the case of bigger rooms with more significant temperature gradients.
- It was supposed that most of the heat exchange with the furniture is made with the air mass inside the room, and the contributions of the walls and the floor are small enough to be neglected.

2.1.1. Walls

Each wall layer is modelled as a single temperature point representing an average temperature $\bar{T}_i$ in the layer (located in the middle of the layer profile). All the layer's mass is aggregated at this point, having heat capacity C_i, and it exchanges heat with the inner and outer layer's boundaries (with the average surface temperatures designated as T'_i and T''_i in Fig. 1) with thermal conduction, shown by Eq. 1, where the average thermal conductivity between the layer's average temperature point and the boundary is denoted by $1/R_i$.

$$C_i \frac{d\bar{T}_i}{dt} = \frac{1}{R_i}(T'_i - \bar{T}_i) + \frac{1}{R_i}(T''_i - \bar{T}_i) \tag{1}$$

A layer can be in the middle of the wall, having boundary temperatures with respect to adjacent layers, or it can be a layer at the side, with direct contact to the inner and outer air. In the latter case the interaction of the inner wall layer with the surroundings includes convection with the inner air, long-wave radiation between the walls and short-wave solar-radiation flow penetrating through the windows [14]. At the outer side of the walls only convection with the surrounding air is taken

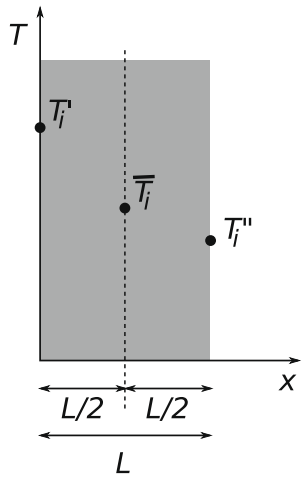

Fig. 1. Temperatures in the wall-layer profile.

into consideration because it is assumed that the walls are white and so the contribution of the solar radiation is small and can be compensated with a higher outside air temperature. The heat flow caused by convection depends on the difference between the air and the wall-surface temperatures and is empirically determined with a convection factor. The thermal radiation exchange between the walls is calculated with Stephan's law. The radiation heat flow $q_{i,j}$ coming from the ith wall or window with a surface temperature T_i and flowing to the jth wall or window with surface temperature T_j is given by

$$q_{i,j} = \epsilon_i \cdot \epsilon_j \cdot A_i \cdot F_{i,j} \cdot \sigma \cdot \left(T_i^4 - T_j^4\right). \tag{2}$$

In Eq. 2 ϵ_i and ϵ_j denote the emissivity of the surfaces, A_i is the area of the ith surface and $F_{i,j}$ is the radiation shape factor, which denotes the amount of radiation that strikes the surface j, caused by the radiation of surface i. In the case of the special mutual positions of parallel or perpendicular surfaces, the geometrical coefficients can be evaluated analytically [14]. σ denotes the Stephan–Boltzman constant.

The dynamics of the surface temperature of the wall's inner side T'_j is determined with the equation

$$0 = \frac{1}{R} \cdot \left(\bar{T}_j - T'_j\right) + \sum_i q_{i,j} + \alpha_{in,j} \cdot \left(T_{air} - T'_j\right) + q_{sol\_j} \tag{3}$$

where $\bar{T}_j$ is the average temperature of the jth inner wall layer. The first term on the right-hand side of Eq. 3 is the conduction heat flow into the wall, the second term denotes the thermal radiation flows from other parts of the envelope, the third term is the convection through the air inside (T_{air} denotes the air temperature and $\alpha_{in,j}$ is an empirically determined convection factor [14]) and the final term q_{sol_j} is the solar radiation flow, which is transmitted through transparent parts of the envelope (as well as reflected from other walls).

The dynamics of the outer-wall surface temperature T''_j is simpler, since only the interaction with the surroundings through convection with the outer air is considered:

$$0 = \frac{1}{R} \cdot \left(\bar{T}_j - T''_j\right) + \alpha_{out,j} \cdot \left(T_{out} - T''_j\right) \tag{4}$$

where $\alpha_{out,j}$ is an empirically determined convection factor [14].

2.1.2. Windows

The windows are modelled in a similar way to the walls. They are all double glazed, having two panes, with each pane being represented by the heat capacity aggregated at a single mass point, which has the average temperature of the pane. However, due to the thickness of the panes, their thermal resistance is negligible in comparison to the thermal resistance of the air in the gap between both panes. The temperatures and heat flows in the window profile are shown in Fig. 2, where the temperatures of the outer and inner panes are denoted with T_{op} and T_{ip}, respectively.

The dynamics of both panes' temperatures T_{op} and T_{ip} for the outer and inner panes are described with the equations

$$C_{op} \cdot \frac{dT_{op}}{dt} = -q_{cond} - q_{rad} + q_{conv\_op} + q_{sol\_op} \tag{5a}$$

$$C_{ip} \cdot \frac{dT_{ip}}{dt} = q_{cond} + q_{rad} - q_{rad\_ip} - q_{conv\_ip} + q_{sol\_ip} \tag{5b}$$

$$q_{cond} = \frac{1}{R} \cdot (T_{ip} - T_{op}) \tag{5c}$$

$$q_{rad} = A \cdot \sigma \cdot \left(T_{ip}^4 - T_{op}^4\right) \tag{5d}$$

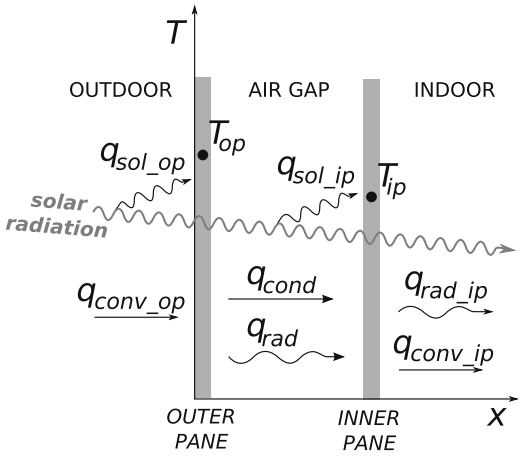

Fig. 2. Temperatures and heat flows in the window profile.

The thermal capacitances of the outer and inner panes are denoted by C_{op} and C_{ip}, respectively. The heat exchange between the panes consists of the heat transfer through the air inside the gap q_{cond} (considered as conduction due to the thinness of the gap) and the thermal radiation flow q_{rad}. The heat flow q_{cond} is determined by Eq. 5c, where R denotes the thermal resistance of the air layer inside the gap. The heat flow q_{rad} is determined with Stephan's law (Eq. 5d), where A is the area of the window and σ is Stephan's constant. A simplification was made, and the panes are considered to be perfect black bodies. Therefore, the ϵ constants (albedos of both panes) are set to 1 and are thus left out of Eq. 5d. The outer pane's temperature is further affected by heat exchange with the air outside through the convection flow q_{conv_op} and by the absorption of the incident solar radiation q_{sol_op}, while the thermal radiation of the building's surroundings is neglected and can be compensated with a slightly higher outside temperature, like in the case of the walls. The other terms in Eq. 5b are the following heat flows: the thermal radiation of the building's envelope and the reflected solar radiation inside the room q_{rad_ip}, the inner air convection q_{conv_ip} and the absorbed solar radiation transmitted through the outer pane q_{sol_ip}.

2.1.3. Night isolation with the roller blind

The roller blind affects heat loss through the window. It is modelled as an additional thermal resistor between the outer pane and outdoor air. The thermal conductance of the pane is inversely proportional to the portion of the window that is shaded, and the resulting heat flow is described by the equation

$$q = \frac{G}{p} \cdot (T_{op} - T_{out}) \tag{6}$$

where G is the heat conductance of the shade, p is the portion of the window that is shaded, and T_{op} and T_{out} are the temperatures of the outer pane and the air outside.

2.1.4. Interior of the room

The air mass inside the room is modelled as a single, lumped heat capacity (mass point) that exchanges heat with the envelope of the room through convection. The furniture is modelled as a rectangular block with a heat capacity and a surface area corresponding to all the furniture pieces in the room and it only exchanges heat with the surrounding air through convection. The temperatures of the inner air and the furniture can be described by the equations

$$C_{air} \cdot \frac{dT_{air}}{dt} = \sum_j [\alpha_{in,j} \cdot A_j \cdot (T_j - T_{air})] - \alpha_{in,f} \cdot A_f \cdot (T_{air} - T_f) + Q_h + Q_v \tag{7a}$$

$$C_f \cdot \frac{dT_f}{dt} = \alpha_{in,f} \cdot A_f \cdot (T_{air} - T_f) \tag{7b}$$

The quantities describing the properties of the air mass inside and the furniture are designated by the indices *air* and *f*, respectively. In Eq. 7a there is a sum over all the surfaces' convection contributions in the envelope. There are additional two terms: Q_v denotes the heat flow due to air exchange with the outside air (ventilation) and Q_h is the flow of heating or cooling. Eq. 7b has only one term on the right-hand side due to the heat exchange only being via convection.

2.2. Solar radiation

Solar radiation is, of course, an important generator of the thermal dynamics effects, and the influence of solar radiation on the temperature inside the building is calculated in three steps.

In the first step the global position of the sun must be determined. From this information the inclination angles for all the windows of the building can be calculated. The position of the sun in the sky is determined by two angles, as illustrated in Fig. 3. α is the solar-incidence angle and Φ is the solar-azimuth. Both can be calculated using equations of spherical geometry

$$\alpha = [\sin\beta \cdot \sin\delta + \cos\beta \cdot \cos\delta \cdot \cos\psi] \tag{8}$$

$$\Phi = \arcsin\left[\frac{\sin\psi \cdot \cos\delta}{\cos\alpha}\right] \tag{9}$$

where β is the geographical latitude, δ is the zenith angle and ψ is the hour angle. The angles ψ and δ are determined by the equations

$$\psi = 2\pi \cdot \left(\frac{1}{2} - \frac{t[s]}{86400}\right) \tag{10a}$$

$$\delta = 23.45^\circ \cdot \sin\left(2\pi \cdot \frac{284 + \text{day in year}}{365}\right) \tag{10b}$$

where t is the current time of day in seconds.

Finally, the direction of the solar rays is expressed as a normalized vector $\vec{r}$ in a coordinate system having its x-axis directed from south to north and the z-axis from the terrain upwards

$$\vec{r} = (\cos\alpha \cdot \cos\Phi, \cos\alpha \cdot \sin\Phi, -\sin\alpha)^T \tag{11}$$

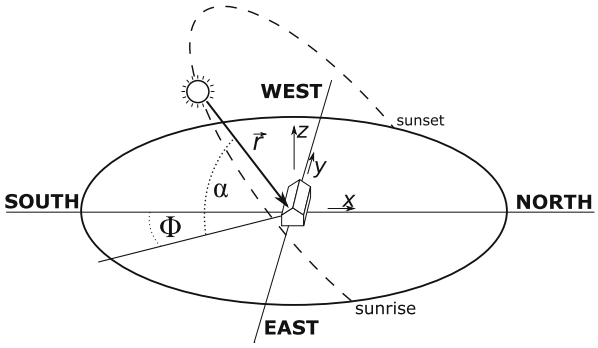

Fig. 3. Position of the sun in the sky is determined by angles α and Φ.

In the second step the portion of the solar radiation that is transmitted through the window and the portions that both panes of the window absorb must be determined. The solar radiation is compounded from the direct and diffuse component. The transmitted, *direct radiation* q_{dir_tr} depends on the incidence angle ϑ (and angles ϑ_i and ϑ_o – see Eqs. (15a)).

$$q_{dir\_tr} = q_{dir} \cdot \left[TR \cdot e^{\frac{\ln\left(PP_i \cdot \frac{n_i^2+1}{2 \cdot n_i}\right)}{\cos \vartheta_i}} \cdot e^{\frac{\ln\left(PP_o \frac{n_o^2+1}{2 \cdot n_o}\right)}{\cos \vartheta_o}} \right] \tag{12}$$

Also, the absorbed, direct radiation depends on the incidence angle and is given for the inner and outer panes ($q_{dir\_ab\_i}$ and $q_{dir\_ab\_o}$)

$$q_{dir\_ab\_i} = q_{dir} \cdot \left[TR \cdot e^{\frac{\ln\left(PP_o \cdot \frac{n_o^2+1}{2 \cdot n_o}\right)}{\cos \vartheta_o}} \cdot \left(1 - e^{\frac{\ln\left(PP_i \cdot \frac{n_i^2+1}{2 \cdot n_i}\right)}{\cos \vartheta_i}} \right) \right] \tag{13}$$

$$q_{dir\_ab\_o} = q_{dir} \cdot \left[TR_o \cdot \left(1 - e^{\frac{\ln\left(PP_o \cdot \frac{n_o^2+1}{2 \cdot n_o}\right)}{\cos \vartheta_o}} \right) \right] \tag{14}$$

In the upper equations q_{dir} is the direct solar radiation flow, n_i and n_o are the refraction indices, PP_i and PP_o are the transmission coefficients when the rays are perpendicular to the pane, TR is the portion of the radiation that passes through the surface of both panes (i.e., all the incident radiation minus the reflected radiation) and TR_o is the portion of radiation that passes through the surface of the outer pane. Both coefficients, TR and TR_o, are calculated from the equations:

$$\vartheta_i = \arcsin\left(\frac{\sin \vartheta}{n_i}\right) \tag{15a}$$

$$\vartheta_o = \arcsin\left(\frac{\sin \vartheta}{n_o}\right) \tag{15b}$$

$$A_1 = \frac{1}{2} \cdot \left[\frac{\sin^2(\vartheta - \vartheta_i)}{\sin^2(\vartheta + \vartheta_i)} + \frac{\tan^2(\vartheta - \vartheta_i)}{\tan^2(\vartheta + \vartheta_i)} \right] \tag{15c}$$

$$A_2 = \frac{1}{2} \cdot \left[\frac{\sin^2(\vartheta - \vartheta_o)}{\sin^2(\vartheta + \vartheta_o)} + \frac{\tan^2(\vartheta - \vartheta_o)}{\tan^2(\vartheta + \vartheta_o)} \right] \tag{15d}$$

$$TR = \frac{1 - A_1}{1 + A_1} \cdot \frac{2 \cdot A_2 \cdot (1 - A_1)}{(1 + A_1) \cdot (1 + A_2) - 4 \cdot A_1 \cdot A_2} \tag{15e}$$

$$TR_o = \frac{(1 - A_1) \cdot (1 - A_2)}{(1 + A_1) \cdot (1 + A_2) - 4 \cdot A_1 \cdot A_2} \tag{15f}$$

We suppose that the *diffuse radiation* component is incident on the window from all directions with equal intensity and, based on this assumption, the same expressions for transmission and absorption as in the case of the direct component are used. However, averaging for all the possible incidence angle ϑ values from the interval $[0, \pi/2]$ is used.

The heat flows due to the absorbed radiation q_{sol_i} (inner pane) and q_{sol_o} (outer pane) in Eqs. (5b) and (5a) are the sum of both components being absorbed:

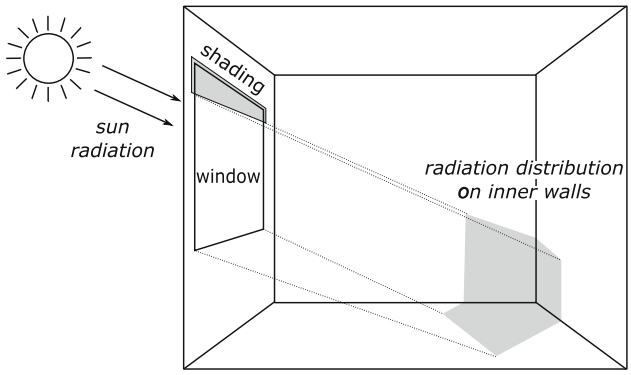

Fig. 4. Distribution of transmitted direct solar radiation on the inside walls.

$$q_{sol\_i} = q_{dir\_ab\_i} + q_{dif\_ab\_i} \tag{16}$$

$$q_{sol\_o} = q_{dir\_ab\_o} + q_{dif\_ab\_o} \tag{17}$$

In the third step we calculate the distribution of the transmitted radiation through the window on the surfaces inside the room. Diffuse radiation spreads in all directions equally, the same as long-wave radiation, and the portion of radiation transmitted through a window illuminating a certain surface inside is equal to the radiation shape factor $F_{i,j}$ between the *i*th window and *j*th irradiated surface. The total heat flow caused by the diffuse radiation component to the *j*th surface is the sum of the contributions from all the windows in a room:

$$q_{dif\_j} = \sum_i F_{i,j} \cdot q_{dif\_tr\_i} \tag{18}$$

The calculation of the distribution of the direct radiation is more complex. Fig. 4 shows an example of the illuminated interior surfaces. The distribution over the internal surfaces depends on the shape of the room, the direction of the rays and the shape of the window aperture. Also possible window shades were taken into account. Finally, the reflections are considered. It is assumed that the reflected direct solar radiation can also be considered as diffuse. All the solar-radiation heat flow coming directly from the windows and hitting the indoor surfaces of the building's envelope is then the sum of both components in Eq. 19a. The contributions of successive reflections are calculated recursively by Eq. 19b until the desired accuracy is achieved (a negligible part of the radiation energy is "lost"). ϵ_i designates the albedo of the surface.

$${}^0q_j = q_{dir\_j} + q_{dif\_j} \tag{19a}$$

$${}^{k+1}q_j = \sum_i (1 - \epsilon_i) \cdot F_{i,j} \cdot {}^kq_i, \text{where } i \neq j. \tag{19b}$$

All the heat flow q_{sol_j} absorbed by the surface *j* is the sum of the contributions of the *N* considered reflections, reduced by the factor ϵ_j (the albedo of the surface):

$$q_{sol\_j} = \sum_{k=0}^{N} \epsilon_j \cdot {}^kq_j \tag{20}$$

3. Implementation of the model in Modelica

As mentioned in the introduction, the described model was, with some simplifications, originally built in a non-object-oriented simulation environment, *Matlab–Simulink* [14,17]. The previous simulator was designed to resemble a real physical test chamber. However, due to the lack of object orientation in *Matlab–Simulink*, it was very difficult to apply configuration modifications and extensions. Because of several other disadvantages we started with the re-implementation of the simulator in the object-oriented environment *Dymola* with the *Modelica* language.

3.1. Modelica – an object-oriented modelling language

Modelica [11,5] is a modelling language that supports both high-level modelling using pre-prepared complex model components in various libraries and detailed modelling by equations. The graphical diagram layer and the textual layer can be efficiently combined. The basic construct in Modelica is class. There are several types of classes with several restrictions: class, model, block, function, package, type, etc.

The concepts in OO modelling follow in many aspects to OO programming (e.g., inheritance). From a modeller point of view, OO means that one can build a model similar to a real system by taking the components and connecting them into a model. It is very important that OO environments enable the so-called acausal modelling approach, so that modellers

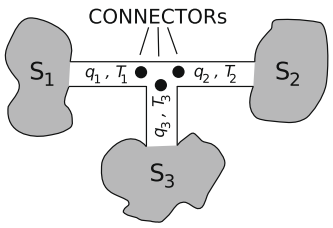

Fig. 5. Connection of three therodynamical classes.

can concentrate on physical laws and then translation process transforms all the model equations into a set of vertically and horizontally sorted equations [2]. More precisely, the translation process transforms an OO model into a flat set of equations, constants, variables and function definitions. After the flattening, all of the equations are topologically sorted according to the data-flow dependencies between the equations. In the case of general differential algebraic equations (DAEs), this is not just sorting, but also the manipulation of the equations to convert the coefficient matrix into block-lower-triangular form, a so-called BLT transformation. Then an optimizer module containing algebraic simplification algorithms, symbolic index reduction methods, etc., eliminates most equations, keeping only a minimal set that eventually will be solved numerically. Next, independent equations in explicit form are converted to assignment statements [5].

The most important difference with regard to the traditional block-oriented simulation tools is in the different way of connecting components. So, a special-purpose class *connector* as an interface defines the variables of the model shared with other (sub)models, without prejudicing any kind of computational order. In this way the connections can be, besides inheritance concepts, thought of as one of the key features of OO modelling, enabling effective model reuse. Fig. 5 illustrates the connection of three thermodynamical classes with the appropriate connectors.

The classes are interacted with two relevant types of variables: temperatures and heat flows. All the temperatures in each connector must be equalized

$$T_1 = T_2 = T_3 \tag{21}$$

while the sum of all the heat flows should be zero

$$q_1 + q_2 + q_3 = 0 \tag{22}$$

The variables from Eq. 21 are often referred to as *across* variables (in Modelica nomenclature *effort*), while the variables from Eq. 22 are *through* (in Modelica nomenclature *flow*) variables. During the model translation the Eqs. (21) and (22), originating from the connector definitions, are automatically generated and added to the other equations of the model.

For the model implementation in Modelica an appropriate modelling environment is needed. We chose Dymola [3], which has a long tradition and good maintenance.

The basic idea of implementation in Modelica is to decompose the described system into components that are as simple as possible and then to start from the bottom up, connecting basic components (classes) into more complicated classes, until the top-level model is achieved.

3.2. Implementation of thermal flows

In our modelling we start with the interactions between two bodies or an accumulation of the heat inside the body. Model classes describing these basic interactions can be found in libraries, and by connecting them, appropriate equations are implicitly generated from connector definitions and added to the other equations.

An extensive library of the most common elementary models, called the *Modelica Standard Library*, is provided in *Dymola*. It also contains a *HeatTransfer* library with the components for 1-dimensional heat transfer modelling with lumped parameters, e.g., "*ThermalConductor*", "*BodyRadiation*", "*Convection*", etc. All these components contain a single connector (port), defined as follows:

```
connector HeatPort "Thermal port for
   1-dim. heat transfer"
SI.Temperature T "Port temperature";
flow SI.HeatFlowRate Q_flow
   "Heat Flow Rate (positive if flowing
      from outside into the component)";
end HeatPort;
```

The connector `HeatPort` contains two variable declarations: the temperature `T` and the heat flow rate `Q_flow`. The former is without any qualifier, making it, by default, the *across* variable, and it is connected according to Eq. 21. The latter has an additional qualifier `flow`, which specifies it as the *through* variable, and is connected according to Eq. 22. The variables in

Modelica are more complex than the simple variables in programming languages. They are actually physical quantities, which also contain units according to the International System of Units – SI prefix, boundary limitations, etc.

The components provided by the *Modelica Standard Library* are sufficient to start with the Modelica implementation of the described mathematical model in a graphical way – by connecting the appropriate components (icons).

3.2.1. Implementation of the wall

Eq. 1 describes the temperature dynamics inside a wall's layer and contains three terms: the term on the left-hand side describes the accumulation of heat inside the layer and the terms on the right-hand side describe the heat conduction through the layer to its boundaries. The layer's thermal dynamics in *Modelica* can be implemented by connecting the components "`HeatCapacitor`" and two "`ThermalConductor`"s. The latter two components of the type "`ThermalConductor`" have two connectors, and the difference in their temperatures is proportional to the heat flow through the element.

The resulting scheme for the presented example, as seen in *Dymola*'s GUI, is depicted in Fig. 6.

The block called "*LayerCapacity*" is a model of a heat capacitor, while the blocks "*InnerSide*" and "*OuterSide*" are models of the thermal conduction through the layer, and are connected on one side with "*LayerCapacity*" and on the other side with the stand-alone connectors "*inside*" and "*outside*". The described structure is defined as a layer model class. We notice three connecting points, with three different temperatures: in the middle the average temperature of the layer (where all three components are connected) and two boundary-layer temperatures on both sides.

The model of the wall is obtained by simply connecting serval layer submodels in series. The structure of the wall layer is further connected to the other connectors according to Eqs. (3) or (4), if the layer's boundary is also the wall's boundary.

3.2.2. Implementation of the window

The same procedure was used to implement the model class of a window. It is described mathematically with Eqs. (5a) and (5b), which describe the dynamics of the temperatures of both panes. The scheme obtained in *Dymola*'s GUI is shown in Fig. 7.

The heat capacities of the outer and inner panes are modelled with two "`HeatCapacitor`" model classes, "*OuterPane*" and "*InnerPane*", the connectors of which also contain the panes' average temperatures. Both panes interact with each other via thermal radiation and thermal conduction through the air in the gap between the panes, as is evident from the first two terms on the right-hand side of Eqs. (5a) and (5b). Therefore, "*OuterPane*" and "*InnerPane*" are connected with the model classes "*AirInside*" and "*PaneRadiation*". There are also model classes named "*OPAbsorbedLight*" and "*IPAbsorbedLight*" in Fig. 7. These are conversion blocks and transform absorbed solar-radiation flows (in Eqs. (5a) and (5b) designated with q_{sol_o} and q_{sol_i}, respectively) into connections of the panes' heat-capacity blocks. They are needed to convert the absorbed radiation flows, calculated as a real variable, into the `HeatPort` connector type. All the other blocks, which model other thermal flows comming from the window's surroundings, are connected to the stand-alone connectors "*Outside*" and "*Inside*", respectively.

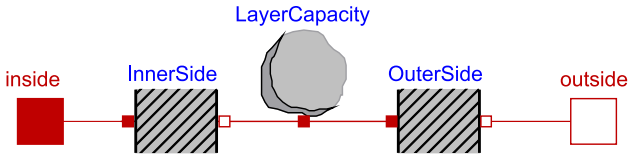

Fig. 6. Scheme of a wall layer in Modelica.

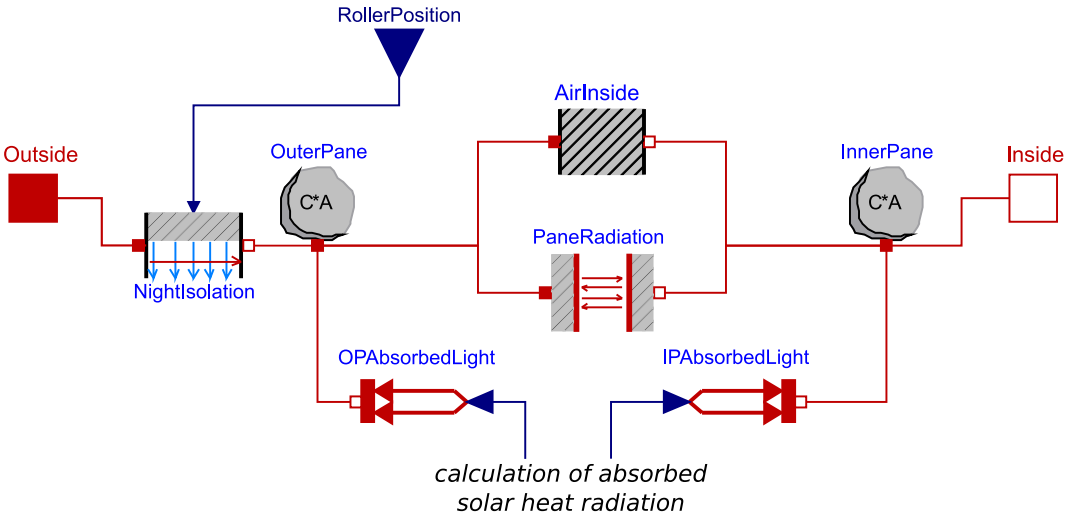

Fig. 7. Scheme of the model describing thermal processes in a window in Modelica.

It is clear that the connector "*Outside*" is not connected directly to the heat-capacity model class "*OuterPane*" of the pane in Fig. 7, but through the "*NightIsolation*" model class that models the influence of a (partially) shaded window, which influences the thermal conductance to the air outside (see Eq. 6). For the development of the model class "*NightIsolation*", the base class "`Element1D`" of all the components with two connectors of the *HeatTransfer* sublibrary is extended. The control input is added (in Fig. 7 it is denoted with "*RollerPosition*") and the variables of both connectors are linked together by adding Eq. 6 to the equation section of the extended model:

```
equation
  Position * Q_flow = G * dT;
```

`Position` is the component's internal name for the control input (portion of window shaded), `Q_flow` and `dT` are the variables provided from the base class, the former is the heat flow through the "*NightIsolation*" and the latter is the difference between the connectors' temperatures.

Of course we could also build the class "*NightIsolation*" using primitive components from the *Standard Modelica Library*. However, such an implementation would be less transparent. The same problem appears in the calculation of convection flows that are based on experimentally obtained convection coefficients. So all these blocks are also implemented using the base class "`Element1D`" with the appropriate extensions – adding empirical formulae.

The specification of parameters and model documentation possibilities are also very important modelling features. For example, the thermal properties of a wall layer are usually described by quantities like the density ρ, the thermal conductivity λ, the specific heat capacity c, the layer's area A and the thickness L. But these are not the parameters used in Eq. 1, where the layer heat capacity C and the heat resistance R are used. Those two parameters must be calculated from the former parameters by using the equations

$$C = \rho \cdot c \cdot L \cdot A \tag{23a}$$

$$\frac{1}{R} = 2 \cdot \lambda \cdot \frac{A}{L} \tag{23b}$$

Below, a *Modelica* model class for the layer is a container class holding the parameters and the structure from Fig. 6.

```
model WallLayer
  // parameters declaration
  parameter SI.ThermalConductivity lambda
    "Thermal Conductivity";
  parameter SI.Density rho
    "Density of material";
  parameter SI.SpecificHeatCapacity c
    "Specific heat capacity";
  parameter SI.Thickness L "Layer thickness";
  parameter SI.Area A "Surface of layer";
  // subcomponents declaration
  HeatCapacitor LayerCapacity(C=
    rho*c*L*A);
  ThermalConductor InnerSide(G=
    2*lambda*A/L);
  ThermalConductor OuterSide(G=
    2*lambda*A/L);
  // connectors declaration
  Interfaces.HeatPort_a inside;
  Interfaces.HeatPort_b outside;
  ...
end WallLayer;
```

The parameter expressions are calculated at the initialization phase before the simulation run. To each parameter a documentation string can be appended and used for the GUI for the submodels' parameter specifications, together with the type specification (providing quantity and unit name), as depicted in Fig. 8. All other documentation concerning the model can be appended as annotations. Fig. 8 depicts the dialog for the wall-layer parameter specifications. Similarly, a new model class for holding the windows' parameters was made (window scheme – see Fig. 7).

3.3. Implementation of radiation flows

The calculation of solar-radiation flows is very complicated. The irradiation of surfaces inside the room depends on the position of the sun in the sky and the geometry of the room and windows. As the most significant absorption in walls

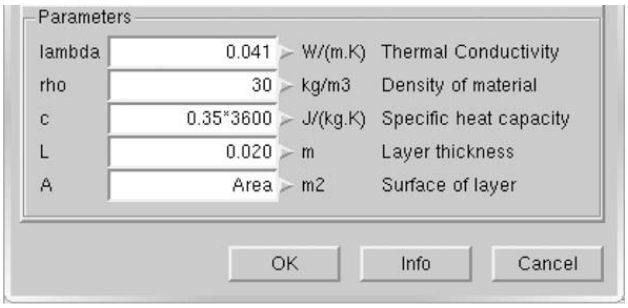

Fig. 8. Dialog for specifying the values of a submodel's parameters.

and windows was taken into account and some other influences were neglected (the most notable probably being the absorption of solar radiation in the internal air and furniture).

The position of the sun in the sky is calculated with two angles: the zenith and the solar-azimuth angles (Eqs. (8) and (9), respectively). For calculation purposes a direction vector of the solar rays is determined from these two angles (Eq. 11) and distributed to window models together with the intensity of the two components comprising the solar-heat radiation: the direct and diffuse components (the reflected radiation of the building's surroundings is neglected). The position of the sun in the model is calculated inside the "*Sun*" block and the relevant data is distributed to each block of the window model by connections with the pair of custom connector types:

```
connector SunLightOutput
  output SI.HeatFlux q_dir
    "Direct solar radiation";
  output SI.HeatFlux q_dif
    "Diffuse solar radiation";
  output Real dir[3] "Direction of sun's rays";
end SunLightOutput;

connector SunLightInput
  input SI.HeatFlux q_dir
    "Direct solar radiation";
  input SI.HeatFlux q_dif
    "Diffuse solar radiation";
  input Real dir[3] "Direction of sun's rays";
end SunLightInput;
```

where `dir[3]` is the information according to Eq. 11. The connection is causal and the only advantage over the global variable is that in the former case it is easier to manipulate with the data without touching the receiving models.

The portion of radiation absorbed and transmitted through the glass of the window is determined by factors that depend on the inclination angle of the sun's rays ϑ coming to the window's surface.

The transmitted and absorbed portions of the solar radiation are calculated separately for the direct and diffuse components (Eqs. (12)–(14)), implemented as an algorithm inside a block with causal connectors.

The calculation of the solar-radiation distribution on the walls inside is split between the calculation of the direct and diffuse components. The portion of the diffuse component irradiated on each wall or the other window directly from the source window is determined simply by the shape factors F_{ij}, as is evident from Eq. 18. The distribution of direct solar radiation is calculated by a projection of the window surface to each wall in the direction of the sun's rays. This surface is used to calculate the portion of heat radiation of the direct component transmitted through the window and received by that wall.

Finally, n successive reflections are added to the total radiation heat irradiated to each wall or window. It is assumed that all the reflected heat radiation is diffuse and the sum of all the contributions of successive reflections is calculated with Eq. 19b.

The calculation of the distribution for the direct component radiation is the most complex part of the distribution calculations. It also requires parameters that describe the geometry of the room in more detail (besides the already mentioned shape factors), so it was feasible to separate it from the other code and put it in its own block.

The algorithm for the calculation of the radiation distribution depends closely on its source – the window – and was therefore included in the window model. The corresponding part of the window model, which has to be joined with the part described in Fig. 7, is depicted in Fig. 9. In the block *Glazing* the amount of absorbed and transmitted heat radiation is determined. The transmitted radiation goes further to the block *Reflector*, where the distribution over the walls inside the room is determined. This block also needs the portions of the direct radiation components received by each wall, which is calculated

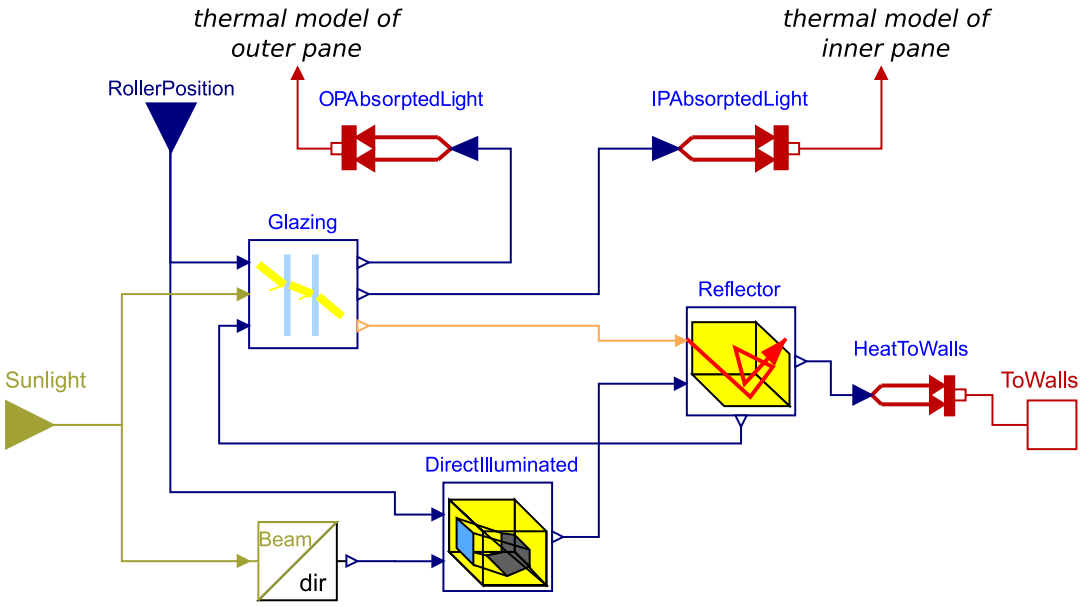

Fig. 9. Model for calculation of the transmitted radiation distribution on the inner walls.

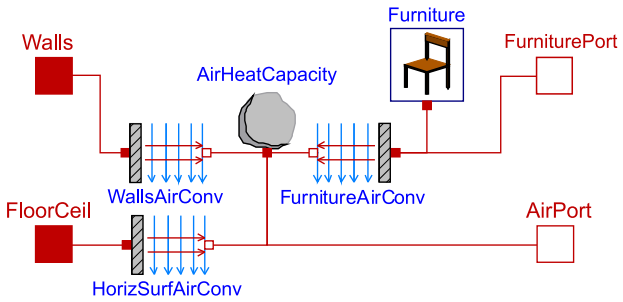

Fig. 10. Model of the interior in Modelica.

in the block *DirectIlluminated*. Note that all the connections between the blocks have arrows indicating the causal interactions and the result must be converted to the *HeatPort* connector type with the blocks *HeatToWalls*, *OPAbsorptedLight* and *IPAbsorptedLight*.

3.4. Higher level composition of the model

3.4.1. Modelling of the room

The described low-level classes are sufficient to compose higher levels of the model. In that way we avoid a behemoth scheme.

The model of the wall is realized with several cascade-connected wall layers. So the wall class and the window class are used to build up the building's envelope. All the surfaces (walls and windows) exchange the heat with thermal radiation. A common room consists of four walls, a floor, a ceiling and at least one window. This means 21 radiation interactions. In order to develop a more transparent model we join all the radiation-exchange calculations (blocks) into one block with seven connectors.

Similar problems and solutions appear when modelling the inside air and furniture, where the majority of the calculations are used to model convection flows. To further rationalize the OO scheme of the room model, a model class for the interior – "*InteriorModel*" – was developed using Eqs. (7a) and (7b). Fig. 10 shows the appropriate model scheme.

The air mass ("*AirHeatCapacity*" class) and furniture ("*Furniture*" class) interact with each other by convection, while the air mass also exchanges heat with the surfaces that embrace it (connectors "*Walls*" and "*FloorCeil*)" also through convection. As convection is treated differently for the vertical and horizontal envelope elements, the appropriate connections are grouped into two arrays (for vertical and for horizontal surfaces). Also, the classes for convection ("*WallsAirConv*" and "*HorizSurfAirConv*") must be placed in appropriate loops and some manual editing must be performed to connect them to the class "*AirHeatCapacity*". The *Modelica* definition is the following:

```
for i in 1:NV loop
   connect(WallsAirConv[i].Air,
      AirHeatCapacity.port);
end for;
for i in 1:NH loop
   connect(HorizSurfAirConv[i].Air,
      AirHeatCapacity.port);
end for;
```

NH is the number of horizontal surfaces and *NV* is the number of vertical surfaces. Each element of the array must be explicitly connected with the connector of the air's heat-capacity model ("*AirHeatCapacity.port*"). The parameters are also grouped into arrays. Below is the declaration part from the model class "*InteriorModel*".

```
parameter SI.Area VArea[NV];
parameter SI.Height VHeights[NV];
ConvectionWallAir WallsAirConv[NV](area=
   VArea, height=VHeights);
```

With such a design the model structure can be efficiently changed by the appropriate change of a parameter.

Finally, a model of a room can be built from the prepared model classes. The overall scheme consists of classes that model the room's envelope and those from the interior model class. The appropriate model scheme is shown in Fig. 11.

The class "*Interior*" in the middle is surrounded with the classes of the room envelope. According to Eq. 3, the inner surfaces of the envelope (represented by the connector facing towards "*Interior*") are connected to the "*RadiationBox*" class, which models the thermal radiation exchange between the surfaces, "*Interior*" class (model of air mass and furniture inside) and to the lower-right connector of the "*Window*" class, which is an array of solar- radiation heat flows received by each surface (in Fig. 9 the connector called "*ToWalls*"). The external surfaces of the envelope are connected to connectors that are visible from the outside of the model of the room and used in the top-level experimental model according to Eq. 4, except for the "*Floor*" connector, which is connected to a constant ground temperature.

3.4.2. Top-level experimenting model

Fig. 12 depicts the top-level model for a single-room building that can be used for experimental purposes.

The "*Room*" model class (see Fig. 11) is linked with the building's surroundings, which are represented by all the other classes in the scheme. The measured variables of the building's environment are the outdoor temperature and both components of the solar radiation (direct and diffuse). This data is read from a file in the block "*Measurements*". The value of the

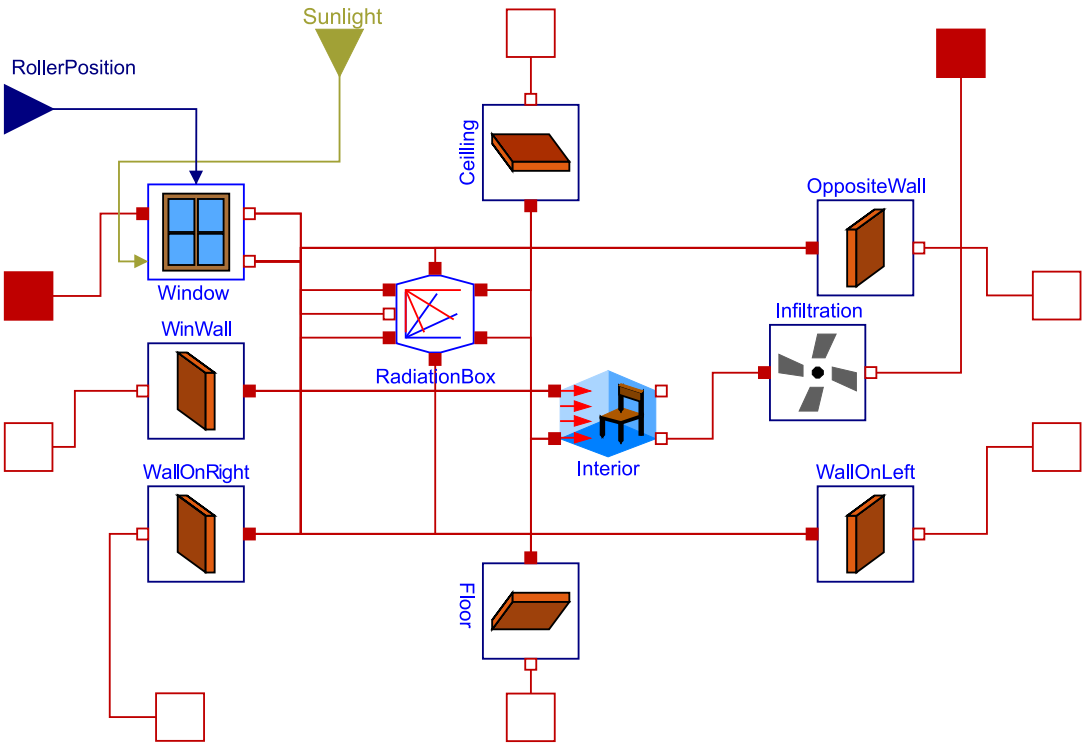

Fig. 11. Modelica model of the room.

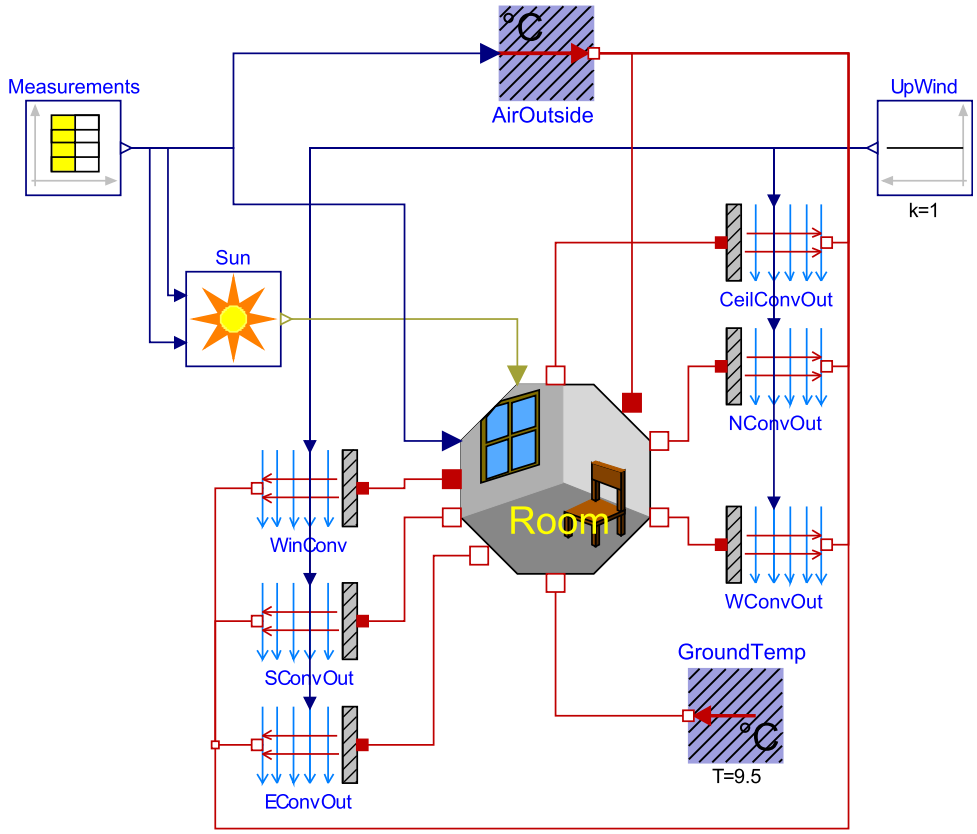

Fig. 12. Scheme of the top-level experimenting model.

outdoor temperature is then routed to the block "*AirOutside*", which converts the *RealInput* connector to the *HeatPort* connector, which enables a further connection to the blocks that model the convection between the outdoor air and the walls of the building (ceiling, north, south, east and west wals). The intensity of the solar radiation is routed to the class named "*Sun*", where the direction vector of the solar rays is also calculated from a specified start date and simulation time and packed together with the solar-radiation component intensity into one connector. Some variables of the environment were not measured, but only estimated, and are considered as constant during the simulation. These variables are the temperature of the ground (block "*GroundTemp*") and the velocity of the wind (block "*UpWind*"), which is used in the outdoor convection-coefficient calculation. The block "*Measurements*" also generates the signal for the roller blind positioning.

3.4.3. *Modelling of a building with several rooms*

The complexity of this simple room model is evident. The room model consists of ten blocks, which are further compounded and some of them again are very complex. A rectangular-shaped room with one window is very general, but buildings can be, of course, much more complex. With our former simulator implementation in the traditional block-oriented approach – in the *Matlab–Simulink* environment – we did not have any chance to use a one-room model for possible extensions. Actually, this was an important reason to start with the OO simulator design in *Dymola–Modelica*.

Let us consider the following example: we want to join two room models into a two-room building model. The joining wall is split into two halves, belonging to each room model. The joining wall is a simple one-layer-wall, while the outer walls have additional insulation, and so three layers were used. If the rooms of the two-rooms building are similar (e.g., have same components connected in the same way), it can be built up of two instances of the single-room building's model. The inner wall is split into two halves, each modeled by its own wall-model component belonging to each room model. In this case, the component of the room's wall, which is now the inner wall of the two-rooms building, needs to be replaced since it consists of only one layer. *Modelica* supports a redeclaration of the subcomponents of the model, but they need to be marked as `replaceable` in the original model's definition, and a replaceable component can be changed only with its subtype. In *Modelica* the subtype of a model is defined as follows [11]:

"*For any classes S and C, S is a supertype of C and C is a subtype of S if they are equivalent or if:*

- *every public declaration element of S also exists in C (according to their names)*
- *those element types in S are supertypes of the corresponding element types in C.*"

However, while single-layer-wall model has a different structure then the three-layers-wall model, a simpler model must be provided as a constraint for the subsequent redeclaration of the replaceable wall-model. In our case of the wall model we

introduce a partial wall model `PartialWall` consisting of only the appropriately defined connectors. Instances of the tree-layer-wall model are then, in the single-room building model, defined as replaceable and constrained with the `PartialWall`, for example, `WallOnRight` is declared as:

```
replaceable BuildingParts.Wall3 WallOnRight(
   /*...*/) constrainedby PartialWall;
```

So declared wall-models' components of the room model can now be replaced with a different wall-model as long as the former contains the same type and same named connectors as the `PartialWall` class, most conveniently, it is extended from `PartialWall`.

In the implementation of a double-room building, in the declaration of the rooms' models, the component of the inner half-wall also has to be redeclared:

```
model DoubleRoomBuilding"Model of the
   building with two rooms"
BuildingParts.OneWindowRoom NorthRoom(
   redeclare BuildingParts.Walll
      OppositeWall);
end DoubleRoomBuilding;
```

This approach of easily reusing the existing components only works until the structure of the component does not need to be changed, otherwise a new component has to be created (by extending the existing one or creating it from scratch).

3.4.4. Dealing with parameters

A large number of parameters has to be set for a room model instance, and of course all the submodels have to be supplied with all the parameters. As the goal is to enter each parameter only once, we used appropriate references. For example, the model of the convection between the wall and indoor air requires the area of the contact surface (e.g., wall) as a parameter, but this information is already provided in the wall model (in order to calculate the layers' thermal resistance), so the parameter of the thermal-convection model is referenced to that of the wall model. This is perfectly acceptable for unrelated parameters like the wall-absorption factor, which is totally independent of the absorption factor of the floor. But there are many tightly correlated parameters, i.e., the geometrical data of the room (e.g., the areas of the walls). For these parameters we used a *record* with all the geometrical information about the room, and from the basic parameters (e.g., height, width, orientation, etc.) many other parameters are calculated (e.g., areas, radiation shape factors etc.). The corresponding parameters in the submodels of the room model are then referenced to those of the record.

4. Simulation results

As the paper is not focused on model validation, only some simulation results will be compared with the appropriate measurements performed on a test room. The measurements were taken for 5 days in the autumn, and the measured quantities were the solar radiation (direct and diffuse components separately), the temperature of the outside air and the temperature inside the chamber. Except for the inside temperature, these measured quantities were also the inputs to the simulator. Fig. 13 shows the simulated indoor temperature, the measured indoor temperature and the measured outdoor temperature. The intensity of the solar radiation (direct and diffuse) is shown in Fig. 14 (upper part), while the lower part presents the position of the roller blind during the experiment.

Also, many other experiments and simulations were performed for different periods. The results confirm the suitability of the Modelica model. Of course, some deviations of several degrees Kelvin can be noticed, especially at the daily temperature peaks. One of the major reasons for these deviations is the unconsidered compensation for the absorption of the solar and

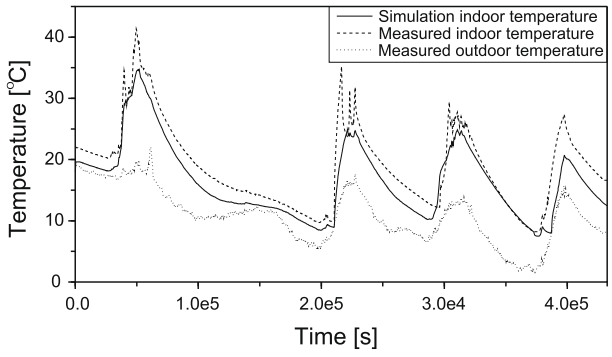

Fig. 13. Comparison of simulation result with measured data from real-world test chamber.

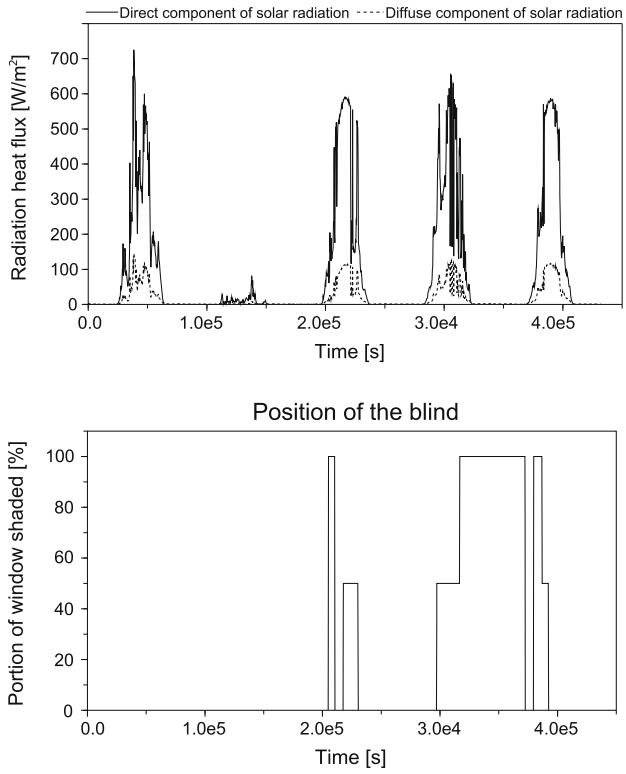

Fig. 14. Upper part: measurements of both components of solar radiation; bottom: position of the roller blind.

surrounding radiation in the walls with the higher temperature of the air outside. Of course, there are many other simplifications and possibly inaccurately estimated parameters.

We also compared the simulation results from *Dymola–Modelica* and *Matlab–Simulink* (the previous simulator implementation). We noticed the better fitting of the *Modelica* model to the measurements, because the solar-heat radiation modelling was improved by a better consideration of the window shading and the use of the nonlinear Stephan's law.

5. Conclusion

A one-room model was originally built in a non-object-oriented simulation environment *Matlab–Simulink*. Due to the lack of object orientation, it was very difficult to apply configuration modifications and extensions.

So we decided to re-implement and to improve the model using the modelling tool *Dymola–Modelica*, which strongly supports an *object-oriented approach*. The advantage of this approach is the ability to build a model without the need for an analytical transformation of the equations and thus the topology of the physical system is preserved. Also, all the connections among the submodels are acausal, so the models or submodels can easily be reused.

The model is also extremely efficient in multidisciplinary projects in which control-engineering specialists (our group) cooperate with specialists from civil engineering, because civil engineers can more easily understand the graphical and textual models in *Modelica* than the schemes in *Simulink*.

We hope that our models are in some parts more realistic and more transparent than some other implementations found in the literature [4,12,6], where the final simulation schemes are rather ambiguous – especially the part with solar-radiation calculations, which is implemented with many icons and is therefore rather confusing. Unfortunately, we were not able to compare our approach to modelling the transfer of direct and diffuse solar radiation through the window and the distribution of the direct radiation inside the room – there is no deeper discussion about it in the cited papers. The convection in [4] is also rather simplified, as the constant convection coefficients are used rather than the more complex experimental expressions we used and which, in our opinion, significantly contribute to the results.

We believe that accurate modelling of solar thermal radiation is one of the most important results in our work and a lot of attention has been devoted to its implementation in *Modelica* in order to develop efficient and reusable models for the future work.

Currently, we are working on the validation, and therefore a new building was equipped with the required measurements.

We expect that such a model will significantly improve several model properties in comparison with the *Matlab–Simulink* implementation; this will include better understanding of the influences of thermal and radiation flows on comfortable living conditions, model-based control-system design, which will enable the harmonization of active and passive energy

resources, and so important energy savings will be made, but also it will be a very suitable environment for education in modelling, simulation and control.

References

[1] Åkesson, Johan. Tools and Languages for Optimization of Large-Scale System. Ph.D. Thesis, ISRN LUTFD2/TFRT–1081–SE, Department of Automatic Control, Lund University, Sweden, 2007.
[2] F.E. Cellier, Contineous System Modeling, Springer-Verlag, New York, 1991.
[3] Dymola, Multi-engineering modeling and simulation. Users manual, ver 7.0. Dessault System, Dynasim AB, Sweden, Lund, 2008.
[4] F. Felgner, L. Merz, L. Litz, Modular modelling of thermal building behaviour using Modelica, Mathematical and Computer Modelling of Dynamical Systems 12 (1) (2006) 35–49.
[5] P. Fritzson, Principles of Object Oriented Modeling and Simulation with Modelica 2.1, IEEE Press, John Wiley&Sons Inc., Publication, USA, 2004.
[6] T. Haase, A. Hoh, P. Matthes, D. Mller. Integrated Simulation of Building Structure and Building Services Installations with Modelica, in: Proceedings of Roomvent 2007; Gummerus Printing, Finnland 2007.
[7] C. Hoffmann, H. Puta, Dynamic Optimization of Energy Supply Systems with Modelica Models, in: Proceedings of the Modelica 2006 Conference, Vienna, Austria, 2006, pp. 599–605.
[8] M.T. Lah, B. Zupančič, A. Krainer, Fuzzy control for the illumination and temperature comfort in a test chamber, Building and Environment 40 (12) (2005) 1629–1637.
[9] M.T. Lah, B. Zupančič, J. Peternelj, A. Krainer, Daylight illuminance control with fuzzy logic, Solar Energy 80 (2006) 307–321.
[10] K. Lee, J.E. Braun, Model-based demand-limiting control of building thermal mass, Building and Environment 43 (2008) 1633–1646.
[11] Modelica – A Unified Object-Oriented Language for Physical Systems Modeling. Language Specification, Version 3.0. Modelica Association, September 2007, <http://www.modelica.org/documents/ModelicaSpec30.pdf>.
[12] P. Sahlin, L. Eriksson, P. Grozman, H. Johnsson, A. Shapovalov, M. Vuolle, Whole-building simulation with symbolic DAE equations and general purpose solvers, Building and Environment 29 (8) (2004) 949–958.
[13] I. Sartori, A.G. Hestnes, Energy use in the life cycle of conventional and low-energy buildings: a review article, Energy and Buildings 39 (3) (2007) 249–257.
[14] I. Škrjanc, B. Zupančič, B. Furlan, A. Krainer, Theoretical and experimental fuzzy modelling of building thermal dynamic response, Building and Environment 36 (9) (2001) 1023–1038.
[15] X. Xu, S. Wang, A simplified dynamic model for existing buildings using CTF and thermal network models, International Journal of Thermal Sciences 47 (2008) 1249–1262.
[16] M. Weiner, F.E. Cellier, Modeling and simulation of a solar energy system by use of bond graphs, in: Proceedings First SCS International Conference on Bond Graph Modeling, San Diego, CA, 1993, pp. 301–306.
[17] B. Zupančič, I. Škrjanc, A. Krainer, M.T. Lah, Harmonization of thermal and daylight flows with modelling, simulation and control system design in buildings, in: ITI 2004: Proceedings of the 26th International Conference on Information Technology Interfaces, Dubrovnik, Croatia, 2004, pp. 579–584.