A TOOL FOR DATA MAPPING FROM DESIGN PRODUCT MODELS TO OBJECT-ORIENTED SIMULATION MODELS

Per Sahlin

Bris Data AB
Västerlånggatan 27, 111 29 Stockholm, SWEDEN
per.sahlin@brisdata.se
www.brisdata.se

KEYWORDS

CAD, product model, continuous simulation, hybrid simulation, data mapping

ABSTRACT

The advent of equation-based modelling languages such as Modelica opens a new world of improved productivity to advanced simulationists. However, they also form an excellent basis for end-user applications intended for the less sophisticated user, who typically uses simulation as one of several aids for design decision support. In this context, the issue of automated data mapping from CAD and other more specialized design representations becomes important. This paper discusses the various design representations that are required for simulation and the mappings of data between them. A concrete example from automated generation of a large-scale equation model for building indoor climate and energy simulation is presented.

INTRODUCTION

Object-oriented equation-based modelling tools and associated languages such as Dymola (Elmqvist et al. 1996), gPROMS (Barton and Pantelides 1994, Oh and Pantelides 1996), IDA/NMF (Sahlin 1996), and lately Modelica (Elmqvist et al. 1999) and VHDL-AMS (IEEE, 1997) allow management of large-scale simulation models with unprecedented efficiency. However, for a design engineer, with a need to rapidly and repeatedly evaluate a range of performance measures, these tools are rarely useful in their basic form. Several different simulation evaluations must be carried out to optimize the design, each one typically involving hundreds of parameters, driving functions and initial guesses. Task specific tools are therefore needed to set up the problem and large amounts of data have to be transferred from a CAD environment.

3D CAD tools are today fully implemented in many (but not all) engineering fields. The evolution of Integrated Design Environments (IDE) such as IDEAS is a natural consequence, providing the user with advanced model analysis capabilities. In an IDE, not just geometry but other properties, such as material, color and texture of the design object are handled. Such an extended CAD representation which accommodates a large class of relevant data of a product is sometimes referred to as a product model.

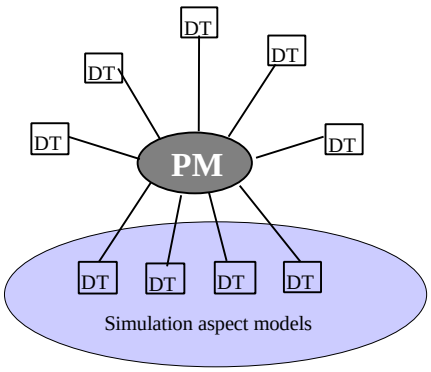

Figure 1. Several Design Tasks interact with a central Product Model. Some design tasks utilize simulation decision support. Each type of simulation requires a separate aspect model of the design.

Product Data Models (PDM) are of course immensely more useful if they are standardized, so that product data can be communicated between different tools and actors. STEP (STandard for the Exchange of Product model data, ISO TC 184, 1993) is since the mid eighties an international effort to standardize product descriptions for many domains. EXPRESS (Schenck and Wilson 1994) is the data modelling language in which STEP models are described. However, the idea of a single, standardized product model that holds all relevant data of a complex design is today regarded to be quite utopic for most real-scale applications with multiple actors. Every aspect of and actor on the design has special information needs that in most cases would be impossible to fully integrate within a finite amount of work. And even if an agreement about the content and structure of such a grand model indeed was reached, it would be utterly impractical to use and soon be dated.

The solution is to limit the scope of the central product model to the key data that is of interest to several different actors and to introduce instead a number of *aspect models* that are tailored to certain design tasks. An aspect model may contain additional information and there may even be a semantic mismatch between a given aspect model and the central product model. The existence of multiple aspect models, the integrity of which must be kept intact through the design evolution, necessitates efficient methods for data mapping between different design representations.

The objective of this paper is to discuss some of the aspect models and mapping methods that are needed for simulation design tasks. We will especially focus on data mapping to the aspect models that are formed by modern equation-based simulation models.

In the next section, an overview of needed representations and mapping methods is given. This is followed by a presentation of a tool for two of the final stages in the required chain of representations, IDA Modeller. Finally a concrete application example of data mapping in IDA Modeller is presented and discussed.

SIMULATION DATA FROM PRODUCT MODELS

In a future oriented design scenario, multiple formal representations of the design must coexist. A number of special purpose models (aspect models) derive their data from a central product model (Figure 1). For a certain object-oriented simulation exercise, an appropriate chain of representations might be:

1. A general product model. This is the repository of all common project data. From this representation, all interesting views are derived, e.g., drawings, bills of materials, costs, and input data for various simulation tools. All members of a design team interact with the product model. The product model must evolve in structure and complexity in parallel with the progress of the design.

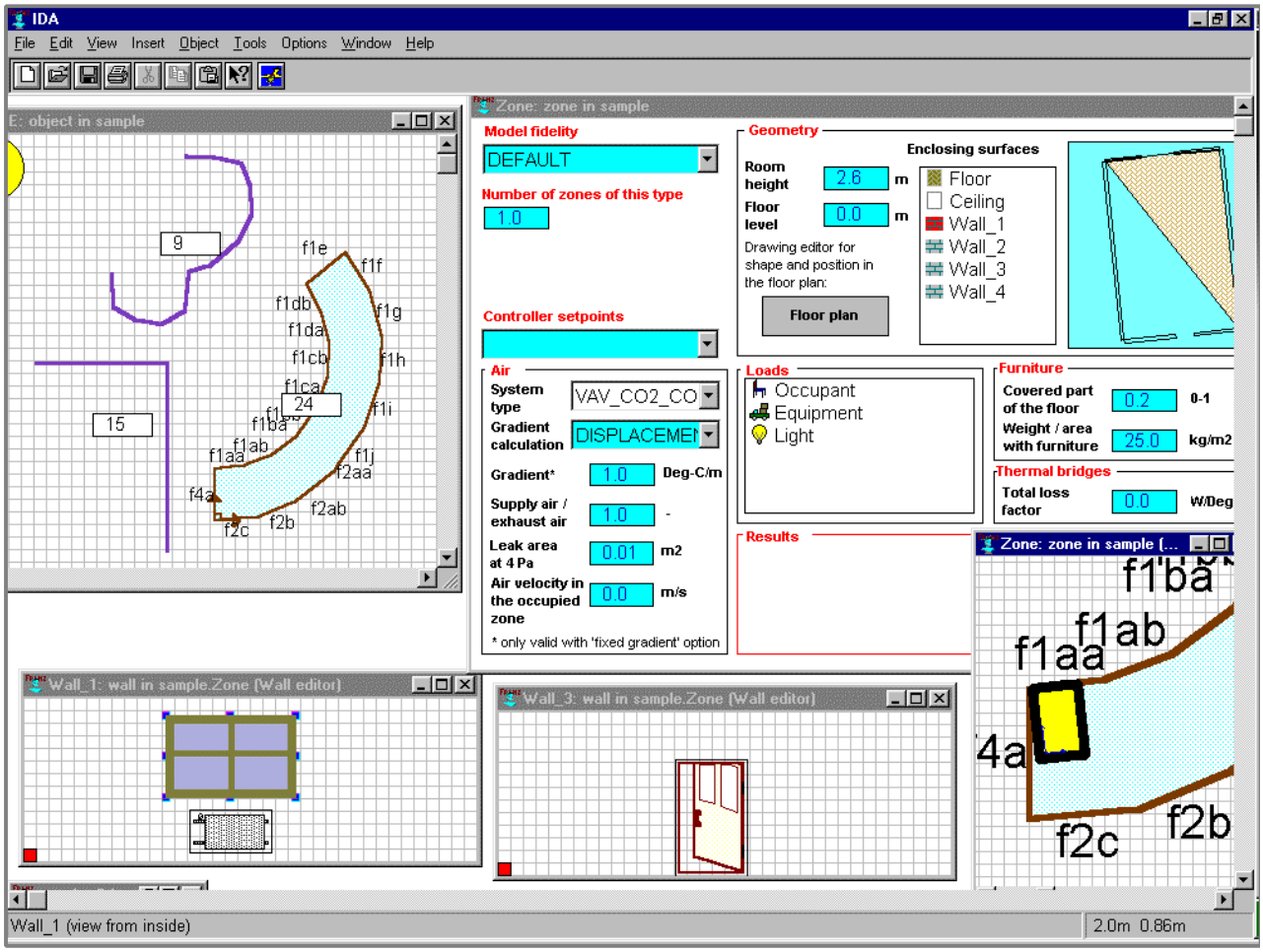

Figure 2. The (level 2) physical view of a building for thermal simulation of indoor climate. Geometrical and material data may be imported from a CAD representation. The user adds data as necessary for the simulation experiment. For a one-room model, as depicted, the user has access to a few hundred parameters in the physical view.

2. A physically oriented aspect model for a certain type of simulation. This is a physical description of the design that contains all the data that are relevant for a certain class of simulation experiments. Designers (not simulation experts) interact with this representation to formulate simulation experiments in physical terms. The aspect model may be locally edited to optimize the design without need to involve the general product model in each design iteration.
3. The mathematical representation. This is the realm of object-oriented simulation languages such as Modelica. At this level, the design is represented by a large hybrid differential-algebraic system of equations. Manipulation of the model generally requires mathematical modelling skills far beyond those of a design engineer.

Data for each successive stage is derived (mapped) from the previous. The user must also at each stage be able to view, manipulate, and add to the automatically derived data. The mapping of data between each stage must be transparent, so that a critical user can resolve the origin (in the previous stage) and processing background of each datum. Enabling quality assurance of the data mapping process is equally important as the validation of the general, unparameterized, simulation model itself.

Traditional task specific industrial simulation tools are typically limited to level 2 in the depicted scenario. The mathematical representation of the simulated system (level 3) is frequently hidden inside the simulation tool, without possibility of user inspection or manipulation. Usually both the mathematical representation itself and the mapping of data to this representation are informal. The user is unable to inspect the mapping code and resulting equations. This lack of transparency and formality creates a situation where the user is left to trust that the simulation tool does a sufficiently good job, without having a real possibility to check this in detail for a given case. If, on the other hand, a formal language such as Modelica is used for the simulation model, sufficient transparency can be provided to enable quality control of the generated model.

Development and standardization of general product models (level 1) is still to a large extent a research topic. The issue of data mapping from a STEP based general product model to various aspect models for simulation has been studied in several projects, e.g., the European COMBINE projects (Augenbroe 1995). For several applications, where the complexity and/or standardization level of the underlying product model is limited, commercial integrated design environments with simulation support has existed for some time. However, most (all?) of these environments make use of traditional, monolithic, simulation tools and neither the mapping process, nor the generated mathematical model are open for quality assurance.

In the remainder of this paper, we will concentrate on a tool for design representation at levels 2 and 3, IDA Modeller, and associated transparent data mapping techniques. Usage of IDA Modeller is illustrated by an IDA-based building simulation application, IDA

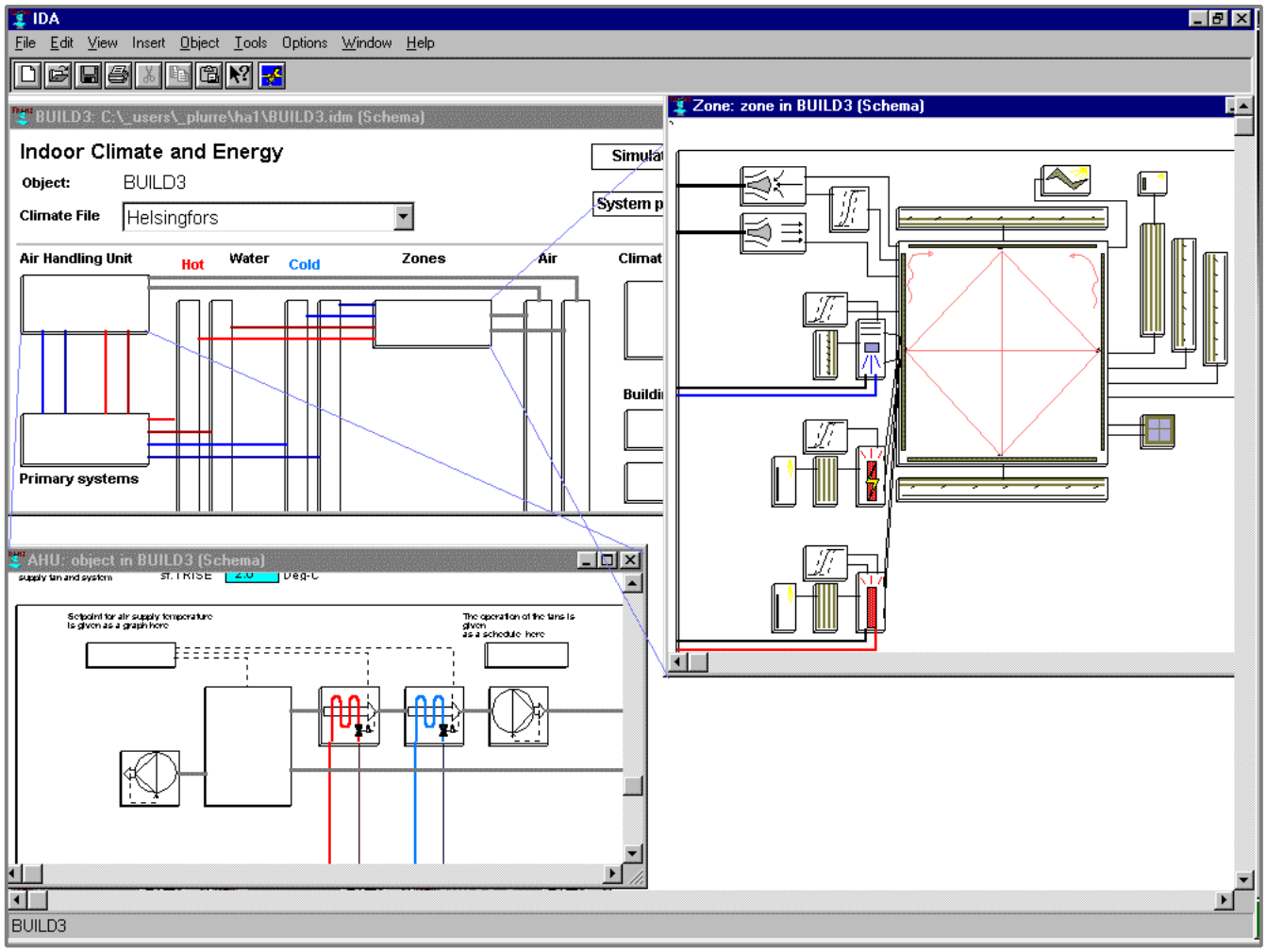

Figure 3. The (level 3) mathematical view of the simulated system. In IDA Modeller, this view is described with NMF. The depicted model of a single room has approximately 600 equations and a similar number of parameters. It may be interactively edited by the user. Typical simulation experiments take a few minutes to simulate and encompass a few thousand timesteps. Large models may involve some ten thousand equations. Simulation time increases about linearly with problem size.

Indoor Climate and Energy (ICE), that gives users full access to levels 2 and 3, and to the data mapping between these levels. This application is installed in about two hundred copies in Scandinavian design offices.

IDA MODELLER

IDA Modeller is the front-end of the IDA Simulation Environment. Models in IDA are described by hybrid differential-algebraic systems of equations. Presently the Neutral Model Format (NMF) (Sahlin et al. 1996) is used to express models. Work is underway to also allow Modelica models. IDA Solver is the numerical solver of the IDA Simulation Environment. IDA Solver relies on pre-compiled (and non-causal) component models (packaged as Windows DLL's.) Due to this, users of IDA applications may freely interconnect component models, without need for a compiler. This allows shipment of very flexible end-user applications at a low cost.

In the most common ICE simulation project, the user builds a model of his system in the physical view, performs a simulation experiment and inspects the results in the same view. The level 3 (math) representation of the system is generated "on the fly" and is never presented to the user. However, in many cases, there is a need to inspect or adapt the math model directly. In such cases, a separate instantiated data model is kept for each level. A third alternative is available for advanced users. They may directly build a math model of the simulated system using the precompiled component models. Conceptually, this corresponds to the work of building a SIMULINK model with ready-made components. However, acausal (input-output free) port-level connections, variable units and descriptions, make the work a great deal more productive.

In other IDA end-user applications, the user interface is less sophisticated, and the level 3 model is the only representation of the simulated system. Figure 4 shows an example of such an application for clean-room design. Additional (physical level) parameters are then tagged on to the instantiated components and subsystems to provide the user with modelling support. The application also typically contains assertion algorithms to support the user in the correct formulation of solvable problems and to distribute "global" parameters into the various components.

DATA MAPPING

The code for generation of the math model from the physical (level 2) system description may become quite complex for real-scale applications. The readability and structure of this code is essential to reach the objective of a maintainable and quality assurable application. Several dedicated data mapping languages exist in the STEP world, e.g. EXPRESS-M, C, V, and X (Verhoef et al. 1995). A 3D CAD mapping tool for ICE (Nordqvist and Noack 1997) relies for example on EXPRESS-C. However, a detailed discussion of mapping language properties is beyond our scope here.

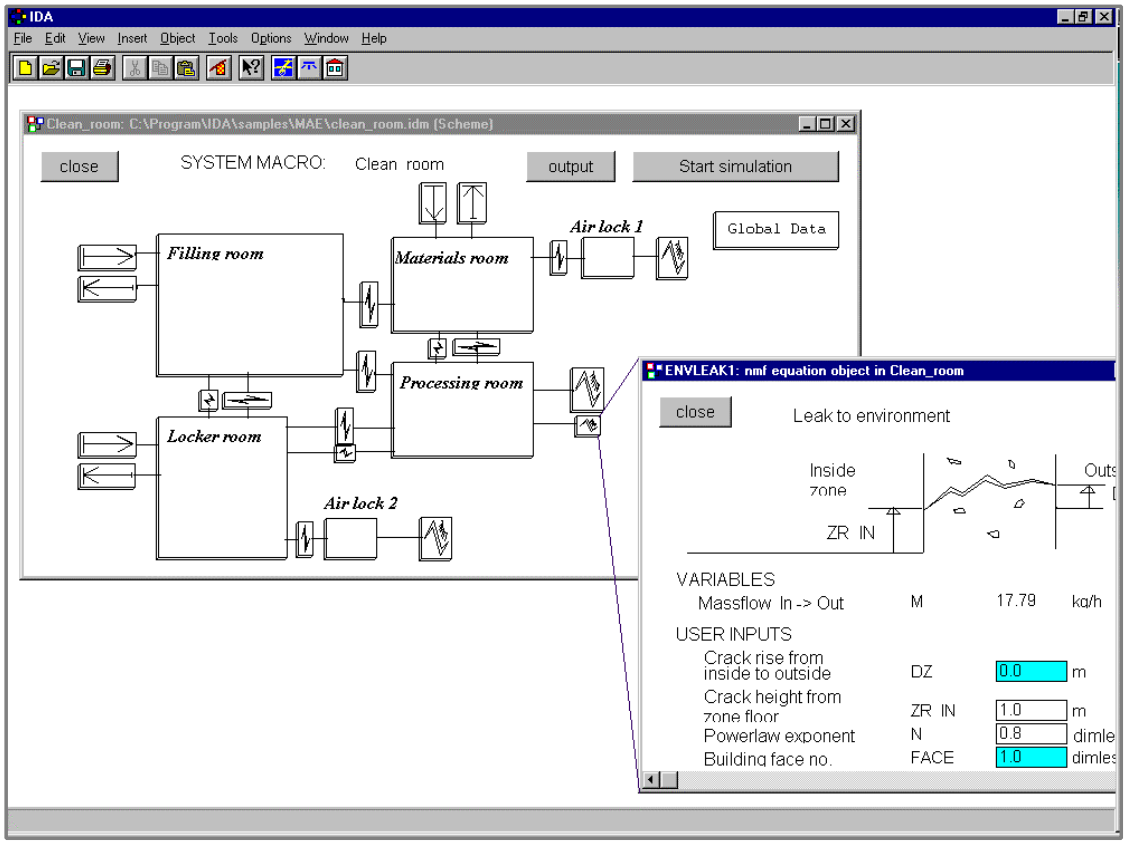

Figure 4. A sample project from a clean room design application.

IDA Modeller relies on a tailored mapping language for the instantiation, parameterization and connection of math models. The ICE application contains approximately two thousand lines of such mapping code. Below is an example of the operations and code that are involved in a typical generation segment.

Example: spawning a radiator

In the lower left corner of Figure 2, a water-based radiator has been added to the level 2 model. The size of the physical box represents in this case the physical dimensions of the radiator. Opening the radiator will reveal an additional four parameters that are required to characterize the device in level 2. (These may also be accessed from a data base). However, the single object (the radiator) that has been added to the level 2 description leads to a number of required level 3 objects and generation operations. The following steps are carried out:

1) Generate the following NMF objects in the room subsystem (c.f. Figure 3, right window, lower left corner)
   a) A proportional controller for the mass flow through the radiator
   b) A piece of wall (1D transient conduction model) behind the radiator
   c) A model for calculation of solar radiation and other ambient conditions behind the radiator
   d) The radiator itself
2) Extend the vector of Hot-and-cold-subsurfaces in the radiation model of the room; provide the physical coordinates of the new surface
3) On the parent level above the room, instantiate a supply and return water system, if it does not already exist.

The automated instantiation of a level 3 component, e.g. the radiator, involves the following principal steps:

i. Compute any local variables
ii. Compute the graphical position of the component
iii. Compute and transfer all non-default parameters
iv. Make logical and graphical connections to neighboring components
v. Make logical and graphical connections to the boundary of the current subsystem

Below is a sample fragment of the generation code for the instantiation of the radiator objects of a room (step 1d above). The lisp oriented syntax has been chosen for ease of implementation rather than beauty. The mapping language will in the future most likely be made a part of the general script language of IDA, but this is still an area of active experimentation and development.

```
(:group WatRad ;;generate a group of objects with names starting with "WatRad"

 :source
     ;generate an ordered list of source objects
     (:call list-water_radiator :zone)

 :type CEWatHet ; NMF type to instantiate

 :set ((nHCSurf (:call get-HCSurf-number :zone :source)) ; compute object's number
       )

     ;; instantiate at this graphical position
 :at (180 (+ 70 (i* 120 nHCSurf)) 205 (+ 125 (i* 120 nHCSurf)))

 :parameters ;; parameter mapping from source to target object
    ;;<to par>  <from source>
     ((k          k)
      (n          n)
      (strip_h    strip_width)
      (length     (/ (* dx dy) strip_width))
      (cp_liq     [:building syspars cpliq])
      (hback      -1)
      (dP0        [:building syspars dp0_water])
      (mmax       design_massflow)
      (mmin       [:building syspars water_mmin])

      ;initial values
      (DpOK       1)
      (Mok        1)
```

```
    )

:set ((yincr (i* 120 nhcsurf)))
:connections ;; make logical and graphical port connections

        ((Front    (nmfzone TqHCFront nhcsurf)  :at  ((204 (+ 83 yincr))  (219 205)))
         (BackConv (nmfzone TqHCBack  nhcsurf)  :at ((204 (+ 112 yincr)) (219 227)))
         (BackWall ((:format "HCBackwall_~D" nhcsurf) Term_a)
            :at ((181 (+ 92 yincr)) (159 (+ 92 yincr))))
         (Control        ((:format "WatRadCtrl_~D" :number) OutSignalLink 1)
            :at ((198 (+ 71 yincr)) (198 (+ 53 yincr)) (169 (+ 53 yincr))))
         (AirTemp        ((:format "WatRadCtrl_~D" :number) MeasureLink)
            :at  ((188 (+ 71 yincr)) (188 (+ 62 yincr)) (169 (+ 62 yincr))))
        )
:boundary-connections ;; connect with subsystem boundary (for further connection by parent)

        ((Inlet      (:format "Sup_hot_~D" :number)
           :at ((197 (+ 124 yincr)) (197 (+ 143 yincr)) (10 (+ 143 yincr)))
           :role sup_hot
           :line-color #.red
           :line-style 2)
         (Outlet     (:format "Rtn_hot_~D" :number)
           :at ((187 (+ 124 yincr)) (187 (+ 135 yincr)) (10 (+ 135 yincr)))
           :role rtn_hot
           :line-style 2)
        )

)
```

The generation clause creates a group of level 3 objects. It is driven by a corresponding list of level 2 source objects. Each water based radiator belongs to the array of source objects of class HCSurf (Hot and Cold Surfaces.) Other members in this array may be, e.g., electric radiators and cooling panels. nHCSurf refers to the radiator's identity in the array of HCSurfaces. This number is used to identify the proper connection partners and ports in the neighboring components.

EXPERIENCES WITH THE CURRENT IMPLEMENTATION

The current level 2 to 3 mapping method has proven to be easy to implement and maintain. It is believed to be a great deal more practical than a corresponding hard coding (as normal methods of source and target objects) would have been. Nevertheless, it is wanting in several ways. From the point of view of the end-user, two major problems exist:

1. Each update of level 2 and corresponding level 3 generation will overwrite the previous level 3 model. There is currently no way to use the powerful generation mechanisms repeatedly while working at level 3. A simple way to partly alleviate this problem would be to allow recording of level 3 user operations and then provide the possibility of replay after a new level 3 model has been created. Such a solution is underway but is likely to have the common problems of any macro recording, i.e. unpredictable results when applied in a different context. Another way to solve the overwrite problem would be to let the user explicitly control what part of the model that is regenerated. This, on the other hand, may be difficult to use correctly, since most users are only vaguely aware of the structure of the source and target models. A third way might be to allow incremental generation of the level 3 model triggered by changes in the level 2 description.
2. The current generation algorithm will name spawned objects according to its own needs. It is difficult for the user to trace the origin of a level 3 object without resorting to identification by parameter values. This problem is rather straightforward to solve at the expense of some algorithm and/or name structure complexity.

In spite of these problems, the current implementation works well for a great majority of ICE users. There is ample opportunity to study user behavior in order to create practical solutions to the problems discussed.

CONCLUSIONS

The new object-oriented and equation-based simulation methods are structured and robust enough to enable usage of automated model generation mechanisms. This in turn enables us to build end-user simulation applications with unprecedented structure, transparency and maintenance properties. However, new areas of needed research and development are also uncovered in the area of mapping between different types of data models.

At this point, we would argue that equation-based target models are sufficiently special to motivate tailored mapping methods, some examples of which have been illustrated in this paper. However, it would be an interesting exercise to use some of the general mapping languages (Verhoef et al. 1995) for the same purpose, to investigate their relative performance.

If the tailored mapping methods indeed prove superior, it is a natural consequence to standardize their form in languages such as Modelica.

REFERENCES

Augenbroe, G.L.M. (ed.). 1995. Combine 2 Final report, Contract JOU2-CT92-0196, Delft Univ. of Technology, 1995

Barton P.I., and C.C. Pantelides. 1994. "Modeling of combined discrete/continuous processes". AIChE J., 40, pp. 966--979, 1994

Bris Data AB. 1999. IDA Simulation Environment - User's Manual, Bris Data AB, Stockholm, Sweden, September 1999 (see also http://www.brisdata.se)

Elmqvist H., D. Brück, and M. Otter. 1996. Dymola --- User's Manual. Dynasim AB, Research Park Ideon, Lund, Sweden, 1996

H. Elmqvist, B. Bachmann, F. Boudaud, J. Broenink, D. Brück, T. Ernst, R. Franke, P. Fritzson, A. Jeandel, P. Grozman, K. Juslin, D. Kågedahl, M. Klose, N. Loubere, S. E. Mattsson, P. Mostermann, H. Nilsson, M. Otter, P. Sahlin, A. Schneider, H. Tummescheit, H.Vangheluwe. 1999. "Modelica 1.2 - A Unified Object-Oriented Language for Physical Systems Modeling. TUTORIAL and RATIONALE". June 15, 1999 (available at http://www.modelica.org)

IEEE. 1997. "Standard VHDL Analog and Mixed-Signal Extensions". Technical Report IEEE 1076.1, IEEE, March 1997

ISO TC 184. 1993. *The STEP Standard*, draft international standard DIS 10303, continuously since 1992 published in several different parts

Oh M., and C.C. Pantelides 1996. "A modelling and simulation language for combined lumped and distributed parameter systems". Computers and Chemical Engineering, 20, pp. 611--633, 1996

Nordqvist, W. and R. Noack. 1997. "A Mapping Study", Dept. of Construction Management and Economics, KTH, Stockholm, 1997

Sahlin. P., A. Bring, E.F.Sowell. 1996. "The Neutral Model Format for Building Simulation, Version 3.02", Dept. of Building Sciences, KTH, Stockholm, June 1996 (available at http://www.brisdata.se/nmf)

Schenck, D.A., and P.R. Wilson. 1994. Information Modeling: The EXPRESS Way, ISBN 0-19-508714-3, Oxford Univ. Press, 1994

Verhoef, M., T. Liebich and R. Amor. 1995. A multi-paradigm mapping method survey, Fischer, Law and Luiten (eds), "Modelling Of Buildings Through Their Life-Cycle", IB/W78-TG10 publication 180, p 233-247, Stanford University, August 1995