Integrated Project
NMP 026771-2
Start: 2006-10-01
Duration: 48 months

I3CON - Industrialised, Integrated, Intelligent Construction

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

Chapter title:

Building Performance Simulation within I3CON

Table of contents

1 INTRODUCTION

In this chapter, an overview of IDA ICE is given. IDA ICE is the main deliverable of I3CON, Task 6.2. The first section is intended for a reader with a limited background in building simulation. After this, we dive deeper down into the issues involved in designing a modern tool for the field.

1.1 Introduction to IDA ICE simulator application

The main predictive capability of IDA ICE concerns the core purpose of a building – its ability to provide a controlled climate more pleasant than that outdoors.

The most immediate climatic factor is obviously temperature, and consequently being able to foresee how hot or cold a room will be under given conditions is the most important facility of IDA ICE.

Other aspects of indoor climate are also computed such as relative humidity and general air quality as indicated by the level of carbon dioxide in the air. However, local air velocity, i.e. how air moves within a room, is not computed. For this type of study, one needs 3D CFD models, where the air volume of a room is divided into many small cells.

The prediction is based on detailed physical models of the main processes in a building. Heat loads from the sun, from people and from electrical equipment must be described. Natural and mechanical ventilation airflows and heat storage in building structures are some of the processes that are modeled.

Active measures for influencing the indoor climate are obviously also a concern. How the heat is supplied or removed, for example the temperature of circulated air or water becomes important. Heating by a low temperature is potentially more economical (and ecological) since high quality energy, such as that from electricity or very hot water, can be better utilized (for example by using a heat pump).

The devices that provide fresh air and heat must be governed by active control systems. The most attractive measures for improving indoor conditions or energy use are often related to changes in the control system. IDA ICE has unique abilities in this area. Control systems may be described in great detail and when needed with very fine time resolution. In alternative tools, fixed and often long time steps are used, making crass simplifications of control behavior a necessity.

Another characteristic of IDA ICE is that pressure of air and water is modeled throughout. This means for example that an advanced user may study valve authority and balancing issues. Hybrid ventilation schemes, where natural and mechanical forces are combined, are also straightforward to model without any drastic simplifications. For basic usage, the pressure is seldom a concern, since idealized models with all pressure drop in the terminal device are used by default.

A common misconception is that a detailed dynamical model such as that of IDA ICE is inherently difficult to use, requiring massive amounts of input data at great detail. Granted, one is able to build very complex models of buildings using IDA ICE. However, simple models are also possible to build and these models do not require any more details than the input of many so called “simplified tools.” Any reasonable estimate of building thermal performance will require things like heat loads, envelope and window areas to be specified.

And with only these very basic numbers available, an ICE model can be constructed. This model will produce more accurate predictions than any simplified method will. The only drawback is execution time. Since the "simplified methods" are often originally designed for hand calculation, they execute in a millisecond or so on a modern computer. For a single zone ICE model, one may have to be patient for a few seconds for a yearly simulation.

Results are presented both in terms of general consumption statistics, such as yearly energy and electricity use, but also as diagrams, where conditions may be monitored at any part of the building for any segment of time. The finest time resolution is normally on the order of minutes, but if required much finer resolution can be obtained.

Contrary to other tools, IDA provides the advanced user with insight into each detail of the mathematical model. The model equations are available for inspection. If the development environment (IDA SE) has been installed, any user may also change the built-in models or add his/her own. (IDA SE is further discussed in Section 4.) In terms of results, the time history of any variable of the model may be recorded. There are about a thousand variables available for inspection for each zone. This allows the critical user to monitor exactly what is going on during the simulation and to understand the background of a suspicious result.

Models are described in terms of differential and algebraic equations (DAE). The total system of equations is solved using a general-purpose solver, which adapts time-step and integration order according to what is going on in the solution. At night, when little is happening, time-steps may exceed an hour, while for example a quickly switching on-off controller, may force time-steps to become extremely short in order to capture the precise development.

There are several advantages with using a modern general-purpose solver rather than the hand-coded component subroutines of all other available whole-building simulators. First, it removes handcrafting of solution algorithms and this not only saves massive amounts of development time, it also ensures quality. By instructing the solver to use a fine solution tolerance, the numerical solution error can effectively be made negligible and one can be sure that any remaining error stems from the actual equation model, i.e. equations do not reflect reality sufficiently well.

1.2 DAE Based Building Simulation

Any designer of a building simulator must first of all find or construct some framework in which to express the physical models to be implemented. A great majority of existing tools and also some recent projects such as EnergyPlus, rely on a domain dependent infrastructure which is formulated in the early stages of the project to accommodate the models envisioned at this stage. As for any design project, early decisions will have tremendous impact on the rest of the work throughout the lifecycle. The designers of TRNSYS, for example, in the early seventies, had a great vision and constructed a simple, yet expressive "world" which has served the building simulation community well for soon thirty years.

When deciding on a basic framework, one should ask what the truly domain specific features of the target area are so these can be fully exploited in the tool. Voltage and current are for example important for an analog electronics simulator and early building simulation efforts were often rooted in clever methods to predict heat propagation through walls. We think the TRNSYS designers in the seventies realized something fundamental about buildings and their service systems; it is virtually impossible to formulate anything reasonably strict and truly domain specific about a building simulation. An awesome mixture of various models are needed for state-of-the-art building simulation. Fortunately, the situation is similarly

disordered in many other domains such as chemical process plant, whole-vehicle, mechatronic and aeronautical simulation. True multi-domain approaches are needed.

Also for a general class of problems such as lumped parameter models of piecewise continuous modular dynamical systems, there is a range of expression paradigms to choose between such as Bond Graphs (20-Sim, MS1) block diagrams (Simulink, VisSim), subroutines or classes with specified interfaces (TRNSYS, HVACSIM+) or symbolic differential-algebraic equations (DAEs) (as in IDA, Dymola, SPARK, SMILE, ALLAN.Simulation and CLIM 2000-ESACAP). With the possible exception of Bond Graphs, serious efforts at modeling buildings have been made with all of the mentioned tools, Felgner et al. [1], Musy et al. [2], Nytsch-Geusen and Bartsch [3]; Jeandel et al. [4]; Murphy and Deque [5]. In recent years, MATLAB-Simulink has grown to almost a de-facto standard in non-CFD scientific computation. However, closer inspection of building simulation models made with the listed tools will reveal that Simulink models are orders of magnitude smaller (or slower) than those made with more powerful tools. The current version of Simulink is not likely to be able to contend with the state-of-the-art efforts.

The new and ambitious modeling language Modelica (www.modelica.org), has shown potential to bring order to the fragmented world of DAE based simulation. It draws on the collective experience of a large number of first-generation languages and since the first tool, Dynasim [5], appeared in 1999, several large industries such as Toyota, Ford, United Technologies, Caterpillar, ABB, Alstom, TetraPak etc. have adopted it. Impressive Modelica models have been reported at three well-attended international Modelica conferences. In the automotive industry, multi-domain, whole-car simulators of previously unprecedented size and complexity have been developed, Tiller et al. [6]; Bowles [7].

The IDA team has been active in DAE language design since the late eighties when the Neutral Model Format (NMF) was first proposed to ASHRAE and the building simulation community, Sahlin and Sowell [8]. In addition to several prototypes for various target environments, quality translators have been developed for IDA, TRNSYS and HVACSIM+, Sahlin [9]. The first version of the public domain model library of IDA ICE was developed in NMF within IEA Task 22, Bring et al. [10].

NMF was designed to bring the power of DAE based modeling to the building simulation community and yet be compatible with major building simulators such as TRNSYS, IDA and SPARK. From a technology point of view, this effort has been a success but the language has never caught the sustained interest of independent building simulation developers.

The IDA team has been part of the Modelica development effort since the first design meeting in 1996. During this time several prototypes have been developed for the successively evolving Modelica specification. The I3CON version of IDA ICE is planned to provide Modelica support and it is envisioned that most future major library development efforts around IDA ICE will be Modelica based. However, NMF will also be supported for the foreseeable future, since the straightforward language structure and lower cost of tools is better suited to many less frequent modelers. An automatic translator from NMF to Modelica has been developed.

1.3 IDA ICE Model Library Overview

The basic design principle behind the IDA ICE library has been to provide the best possible resolution of key phenomena while enabling whole-building, full-year simulation within commercially acceptable execution times. The first version of the library was developed within IEA Task 22. Admittedly, in the first version of the simulator in 1998, whole-building

simulation was reserved for those with sufficient patience. However, with current IDA numerical methods on today's hardware, the library is more appropriately placed. We will briefly outline the phenomena modeled (and not yet modeled) with special emphasis on features that are unusual in a whole-building simulator context.

Flow networks. Pressure is modeled throughout air and water flow networks. For air this means that stack and wind driven flows are consistently modeled as in air flow network programs such as COMIS. For mechanical circuits, ideal flow controllers are often used to maintain a given flow in order to avoid the need to input accurate pressure drop data. However, the advanced user may study systems with realistically distributed pressure drops. CO_2 and moisture are modeled for all air streams. High on the development agenda are models for large horizontal openings.

Control principles and dynamics. Most control loops are explicitly modeled using separated P or PI controllers. Variables such as massflows, temperatures and pressures are never regarded as given, but are always computed, often via more or less idealized control loops. To compute, e.g., a cooling load, some large but finite capacity room unit and associated sensor and controller are always used. Fast local loop dynamics are often not modeled in the basic library in the interest of calculation time. However, to study fast timescales in some part of the system, sensor, actuator and relevant process dynamics may be modeled for the local loop. An example of such a study is presented in the next section. This possibility of modeling local circuits to an arbitrary level of time resolution in the context of a whole-building model we believe to be unique.

Long-wave radiation. Full non-linear Stefan-Boltzman long-wave radiation is modeled for shoebox zones only. An MRT approach is presently used for zones with more complex geometry. View factors for arbitrary obstructed surfaces will be included.

Short-wave radiation. ASHRAE (default), Kondratjev and Perez sky models are available. External shading is calculated for direct and diffuse light for arbitrary four cornered flat opaque surfaces. Shade presence may be controlled. Semi-transparent surfaces will be implemented. Inside the zone, incoming short-wave is distributed diffusely according to view factors. Internal windows are currently not implemented but are high on the user's wish list. A special object for five-surface skylights enable modeling of complex roof shapes. For these, multiple beam reflections is modeled.

Convection (Film coefficients). An empirical non-linear model for internal film coefficients depending on surface slope and temperature difference is used. Another fit is used for wind dependence of external film coefficients.

Solid heat transfer and thermal storage. The default wall is a finite difference model with automatically suggested, non-uniform grid spacing. It uses a special integration technique to fully utilize matrix structure. A variant of the same model without native integration is also available. In addition, an automatically optimized reduced RC-network model is available (default in v. 2.11). Ground heat transfer is currently normally modeled as two 1D heat transfer paths, one to the surface and one to a fixed ground temperature. A uniform grid 3D model is available separately in NMF but a comprehensive ground model which is directly accessible from the Standard level is needed. See also the discussion below on the Femlab link.

Other zone model features. Fanger's models (ISO 7730) for comfort and occupant loads are used, enabling calculation of PPD in multiple zone locations and alleviating the user from having to estimate occupant loads. A simple model (Mundt 1996) for linear vertical temperature gradients is frequently used. However, more sophisticated gradient models are

desired. Room units in IDA ICE are based on manufacturer's data. Radiative/convective split is calculated based on computed surface temperatures and exposed surface area.

Secondary systems. The models from the ASHRAE secondary toolkit have been translated into NMF and complemented with variants using fewer and capacity-independent parameters and with built-in control loops. Models that are suitable for whole-year simulation are currently missing for dry and desicant wheel heat exchangers.

Primary systems. Currently, only a simple boiler and chiller are available with given COP and limited capacity. This is an area in which IDA offers good opportunities for integrating models that are currently unavailable in a whole-building context.

The model library is in the public domain and is currently shipped with IDA ICE as NMF source code. Many users adapt and complement the library to suit their needs. In fact, the main motivation behind DAE based modeling is to provide a radically improved environment for developing, maintaining and enhancing a large set of physical models.

Some questions that should be considered in the evaluation of different approaches are:

- How long does it take a beginner to add a new model?
- How efficient is the model development and testing process?
- What degree of model reuse is possible?
- How well can separate developers benefit from each other's work?

Unfortunately, reports from unbiased users with experience from more than a single development environment are rare. A listing and discussion of pros and cons of DAE based vs. traditional approaches can be found in [11].

2 MODELICA AND IDA SIMULATION ENVIRONMENT

Modelica has proven to be of excellent service to advanced modelers in several domains. However, presently, there is usually close contact between model developers and end-users. In fact, they frequently coincide in a single person. As Modelica uptake evolves, the need to deploy Modelica based simulators among less experienced users is likely to increase. IDA Simulation Environment (IDA SE) has been developed to facilitate this process. Originally based on a Modelica predecessor, NMF [13], IDA SE has been used for equation based end-user application development since the early nineties. Several real-scale simulation applications have been developed, some of which have earned leading roles in their respective markets.

IDA SE is based on the concept of pre-compiled component models, i.e. most IDA application end-users work only with fixed1 component models that may be combined into arbitrary (input-output free) configurations without need for compilation. Simulators do not require a working compiler installation. Encryption is not needed to preserve component model secrecy. The new Modelica implementation retains this structure, separating the typical roles of the model developer and end-user.

A large majority of potential simulation users have little appreciation of the beauty and generality of an advanced modeling language. They have a design problem to solve and want quality answers with minimum effort. Quite often the full mathematical formulation of the problem is of less interest. A good simulation application must communicate in terms natural to the user and in most situations this does not involve any modeling language but rather physical concepts from the target application. Pipes, pumps and valves may well be the optimal elements of communication rather than differential-algebraic equations.

2.1 IDA Simulation Environment

Figure 1 shows the three main software modules of IDA SE:

IDA Modeler: the interactive front-end

IDA Solver: the numerical DAE solver

IDA Translator: the model source code editor and processor

A development version contains all three, while a runtime installation lacks IDA Translator. The developer uses IDA Translator to generate a set of C or F77 routines for each component2, for equation evaluation, analytical Jacobian evaluation and general information about the model. The code is compiled from the translator into a Windows DLL which is then linked to IDA Solver. The Modelica (or NMF) source may or may not be shipped with the

[1] array sizes, including connector arrays, can be modified after compilation

[2] A component or a *compilation unit* becomes an indivisible building block in the end-user application. The Modelica source of a component model may be a composite, hierarchical model. It is also possible to define hierarchical models in IDA Modeler containing multiple components.

application, depending on the desired level of confidentiality. Also generated are native class descriptions for IDA Modeler, containing structural information about the model library. This code may then be complemented by application specific extensions.

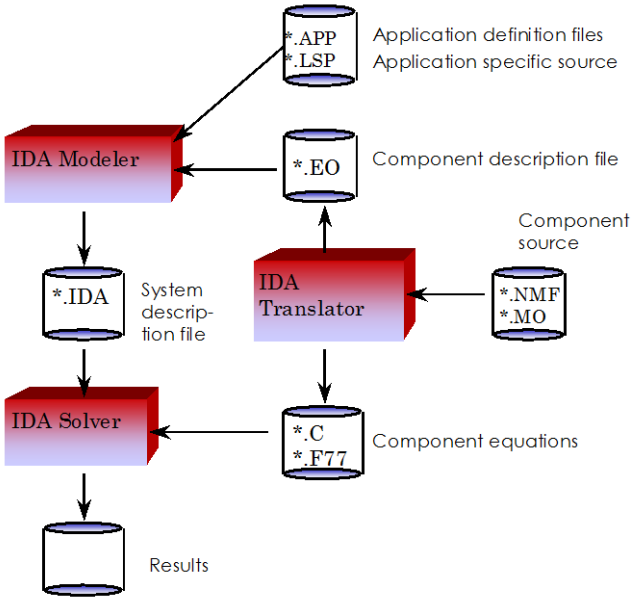

Figure 1: Structure of IDA Simulation Environment

Applications may be shipped stand alone, including an IDA runtime environment or as separate plug-ins for an existing IDA environment. Both the model library and the user interface of an application may be amended and altered by multiple extra separate installations, for customizations and application extensions. This allows efficient management of complex version structures.

The cost of the runtime environment for each installation is significantly lower than that of the full development environment, normally only a small fraction of the cost of the end-user product.

2.1.1 IDA Solver

In tools, such as Dymola, where equations are globally reduced prior to integration, the numerical solver will deal with a fairly dense system of equations but where each equation can be quite complex. One can generally expect equation evaluations to take some time while factorization of Jacobian matrices is likely to be faster due to the dense problem structure. In a pre-compiled setting, the situation is the opposite: functions are rather simple (simple enough to differentiate analytically!) while Jacobians are typically large and sparse.

IDA relies on standard software components for sparse Jacobian factorization. Since large sparse matrices occur in many technical and scientific applications a range of powerful solvers are readily available for scalar as well as parallel architectures. Available solvers for IDA are: SuperLU [14], MUMPS [15] and UMFPACK [16]. The graph theoretical analysis of system structure is done by these external solvers rather than in the context of a global symbolic preprocessing.

There are many implications of this difference in solution strategy. A thorough discussion of this is beyond our current scope and we will merely point out a few aspects:

Component structure is maintained during integration. This allows for example: (1) Exploitation of special component structure by tailored methods. (2) Component level co-simulation with external tools such as FEMLAB (see Figures 2 and 3). (3) Component level debugging.

Equation topology may change during simulation. Since the graph theoretical analysis may be done in each timestep, discontinuities that alter the system structure can be accepted.

For few-timestep simulations, global compilation may take a significant part of the total execution time.

Pre-compiling component models precludes some operations that are natural in a setting where a global symbolic analysis is done. The most serious limitation concerns index reduction. Although index 2 systems generally can be simulated without any problems in IDA Solver, serious high-index problems are most likely better solved by means of global symbolic analysis.

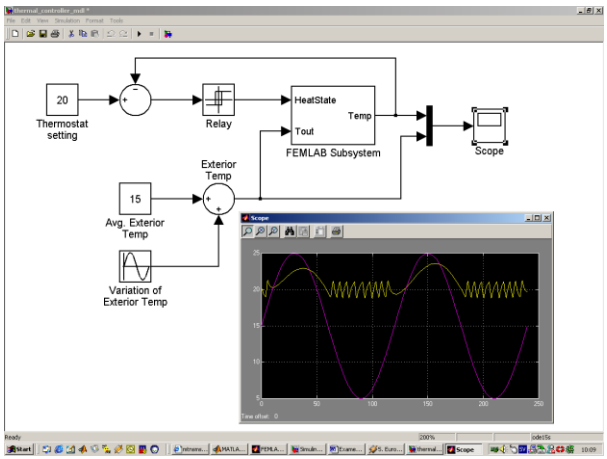

Figure 2: A FEMLAB-Simulink standard case "Thermal controller." A heat source in a 2D region is controlled by a thermostat.

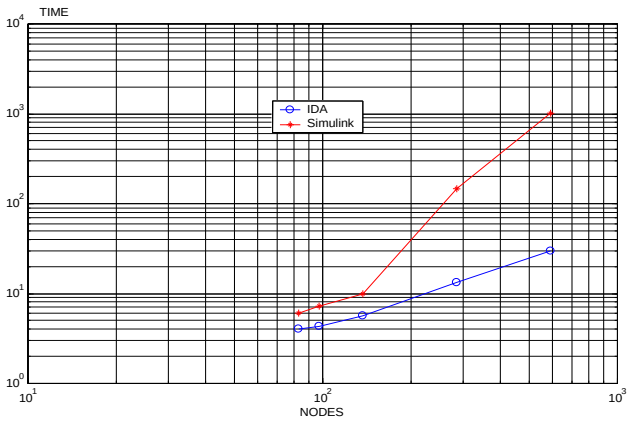

Figure 3: Execution time vs. FEMLAB spacial resolution in the "Thermal controller case". The original Simulink model is compared to an identical FEMLAB-IDA model (from [17]).

IDA Solver is a variable timestep and order solver based on the MOLCOL implicit multistep methods, which include the most common implicit methods such as BDF. Explicit methods are currently not available for the global integrator but may be implemented for individual components.

A selection of methods for initial value computation are available: damped Newton, line-search, gradient and homotopy (embedding) methods

2.1.2 IDA Modeler

IDA Modeler provides a framework for interface development. It may be used to write simulation oriented applications of sufficient quality for competition with tools written from scratch but at a fraction of the cost. IDA Modeler exploits the fact that many tasks are common to most simulation applications: building and presenting models, editing parameters, interacting with a data base, making simulation experiments, viewing results as diagrams and reports, checking user licenses etc.

More elaborate IDA applications, divide the user interface into three levels, to serve users with different needs and capabilities:

Wizard level:	Least demanding. Each required input is presented in a sequence of user input forms.
Standard level	Intermediate. The user is required to formulate a model, but in terms that are natural to the domain.
Advanced level	The user builds a model using equation based objects. Facilities for model checking, automatic mapping of global data, selection of given variables and similar tasks are available.

In such an IDA application, the Advanced level interface offers a model-lab work bench similar to that offered by other DAE environments, providing the user with direct contact with the individual equations, variables and parameters of the mathematical model. However, a great majority of end-users prefer the tools of the Standard and Wizard level interfaces, where the basic mental concept is that of a physical system and not of a mathematical model.

The kernel of IDA Modeler is written in Common Lisp but most application programming is done interactively or by writing native scripts. Extensive facilities are available to simplify common tasks such as: building user interfaces in multiple natural languages; defining a data bases for input data objects; report generation; data mapping etc. Some user interface elements, such as dialog boxes with complex logic, may be written via an API in other languages.

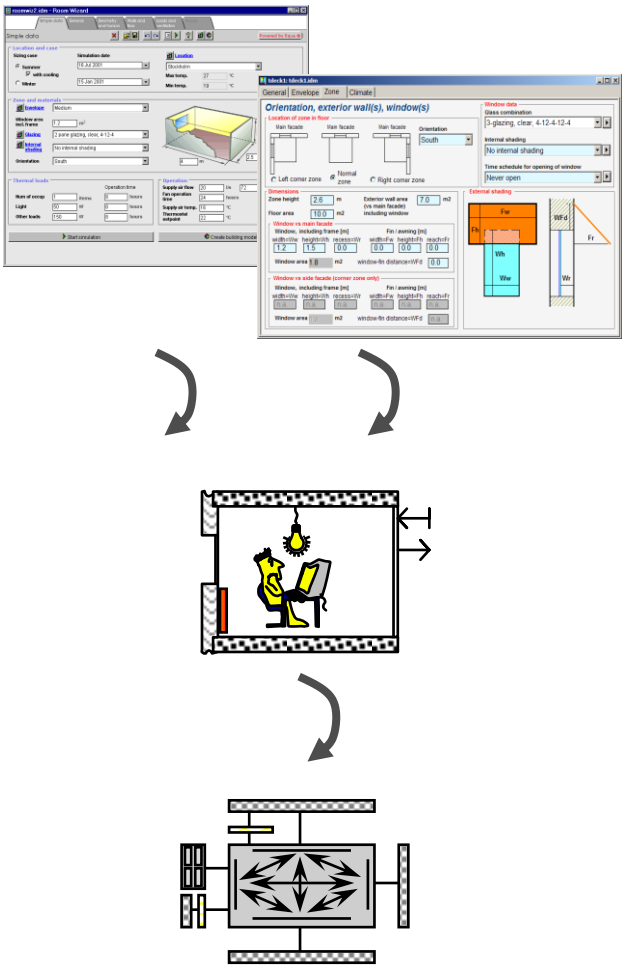

Figure 4: Applications may have multiple Wizard level interfaces for typical simulation tasks. Each interface has a separate data model and a tailored script language for data mapping between levels is provided.

Special emphasis has been laid on tools for development of web clients, running in a browser, powered by an IDA based simulation engine on the (Windows) server. A large portion of the native data structures have been mapped to Java script, facilitating advanced web development with minimum effort.

2.2 Present state of Modelica in IDA

The design of the Modelica language itself has for natural reasons been centered around the presently leading implementation by Dynasim. In this section, we will outline a few issues where the present Modelica design is less well suited to usage in the pre-compiled setting of IDA and where Modelica extensions have been introduced. Present shortcomings of the implementation are also discussed.

2.2.1 Interpretation of Modelica code

The IDA Translator compiles classes, not complete systems. Compiled models normally contain:

- public connectors
- more variables than equations
- outer elements
- arrays with non-constant sizes

All public non-partial and non-local classes declared with keywords class, model or block are compiled to IDA components. Blocks are presently compiled to IDA algorithmic models. Public non-partial and non-local atomic types and connector classes are similarly compiled to IDA quantity and link types.

A compiled model may be extended after compilation by inserting and connecting submodelssub models.

Public top-level connectors in compilation units are preserved by the compiler available for connections.

Compilation units may contain unresolved outer components. Such compiled models should be used only as elements of models that contain corresponding inner components. Unresolved outer classes are not supported.

For each compilation unit, a symbolic analysis is performed where as many variables as possible are solved for symbolically, effectively removing them from the global system of equations. Resulting equations are differentiated and code for evaluation of analytical Jacobians is generated. Although principally possible, no index reduction is currently done at this stage.

It is possible to allow the IDA Translator to process entire simulation problems, resulting in just a single compilation unit. However, this is not the intended usage of the tool since the topological flexibility of being able to re-configure pre-compiled units is an essential feature of most IDA applications.

2.2.2 IDA driven Modelica extensions

Events in functions and pre operator

The previous IDA language, NMF, supports events in functions, also in foreign functions. This is possible because the variables that monitor events are explicit in NMF models. In Modelica, these variables are automatically generated and not available for the programmer.

We have implemented events using the special function mo_event(var, expr). The variable var is a special kind of variable (called assigned state in NMF) that keeps its value from the previous timestep. The function modifies the value of var and generates an event whenever it changes sign. In order to be used in a function, the previous value of var should be passed to the function and the modified value should be returned. To make this possible, we have changed the semantic of the pre operator. In our implementation, pre(v) is always the value of v at the previous successful time step; this is also valid for non-discrete variables.

The modified pre operator may also be used for several other purposes, for example:

- To calculate a maximum value during the simulation:
  ```
  xMax = max(x, pre(xMax));
  ```

- To break an algebraic loop in order to simplify solution of an equation with weak dependences:
  ```
  RhoAir = 1/287.0 * pre(PAir) / Tair;
  ```
- To implement local integration methods, for efficency or for limiting numerical dissipation in PDE:s

A full account of the arguments for the extension of the pre operator is beyond the scope of this paper. However, uncontrolled numerical dissipation due to large and variable timesteps is a fundamental problem for many Modelica applications that should be further discussed.

Conversion to strings

In Modelica 2.1 there are functions that converts scalar values to strings, but there are no functions for converting arrays and matrices. We have implemented automatic conversion of non-strings to strings. Example:

```
assert(x>0, "x = "+ x + " should be positive")
```

Graphics

- More named colors
- Arrow: Closed, Left, Right, {type,side}. The size may be a vector
- lineThickness=0 - non-scaled minimal thickness
- Transformation: negative scale and aspectRatio may be used instead of flip.

2.2.3 Features yet to be implemented

The following list is intended to give a flavor of the present state of development.

Available variable and parameter types

- All variables and non-scalar parameter declared as Integer or Boolean are converted to Real. These variables cannot be used as arguments of function calls.
- Boolean scalar parameters are converted to Integer.
- String variables are not implemented (string parameters are supported)
- Attributes (except value and start) should be constant. They cannot be used in expressions.
- Attributes displayUnit, fixed, enable, nominal, stateSelect are not used.

Connections

- Connection of subconnectors is not yet supported

Modification and redeclarations

- Modifications of class elements are not supported (i.e., when instantiating or extending a class, it is not possible to modify local classes in that class).
- No subtype checking in redeclarations. The constraining clause is ignored.
- Choice annotations not supported.

Expressions

- Record constructors are not supported.

Iterations

- Multiple iterations (separated by ",") not yet supported.
- Ranges with step from:step:to are not supported.
- Vectors in indices only partially supported.
- The index **end** is not supported.
- Deduction of range is not implemented.

Arrays

- Array expressions (not instances) may not be used as arguments to non-built-in functions.

Functions

- Optional arguments are not supported (except in some built-in functions)
- Record arguments are not supported.
- Protected variables in functions are not supported.
- The annotation derivative is not yet supported.
- Some restrictions on external functions.
- Not all Modelica utility functions are implemented.
- External objects are not implemented.

Initialization

- Initial equation/algorithm not implemented

Built-in functions and operators

- Not implemented functions:
  `initial, terminal, smooth, sample, edge, change, reinit, terminate, div, rem, integer, cardinality`.

Graphics

- Attribute visible and smooth is ignored.
- Cylinders and Sphere fill patterns are not supported.
- BorderPattern shown as rectangle with 3D border
- No line pattern if lineThickness >= 0.375
- Text rotation is not implemented
- Filled text is not implemented.
- Bitmaps: may be rotated by 90 degrees only, imageSource not implemented, fileName just copied (no directory information added).

3 SUMMARY

A brief presentation of IDA ICE and the underlying IDA Simulation Environment has been given. IDA will be the base of simulation work to be carried out within I3CON and with the planned extensions, it is intended to be one of the key deliverables of the project.

The implementation work that is reported here concerns a very fundamental change to the existing simulation tool: the inclusion of the new simulation language Modelica. This work has been brought to a state where it is useful for practical modeling work.

4 REFERENCES

[1] Felgner F., Agustina S., Cladera R., Merz R., Litz L. Simulation of Thermal Building Behaviour in Modelica. 2nd International Modelica Conference, Proceedings, pp.147-154. 2002.

[2] Musy, M., Wurtz E., Sergent A. Buildings Air-Flow Simulations: Automatically-Generated Zonal Models. Proceedings of Building Simulation '01, Rio de Janeiro, Brazil. 2001.

[3] Nytsch-Geusen C., Bartsch G. An Object Oriented Multizone Thermal Building Model Based on the Simulation Environment Smile. Proceedings of Building Simulation '01, Rio de Janeiro, Brazil. 2001.

[4] Jeandel A., Boudaud F., Larivière E. ALLAN.Simuation release 3.1 description. M.DéGIMA.GSA1887. GAZ DE FRANCE, DR, Saint Denis La plaine, FRANCE, 1997.

[5] Dynasim Dymola --- User's Manual, Dynasim AB. 2001.

[6] Tiller, M., Bowles P., Elmqvist H., Brück D., Mattson S. E., Möller A., Olsson H., Otter M. Detailed Vehicle Powertrain Modeling in Modelica. 2nd International Modelica Conference, Proceedings, pp.169-178. 2000.

[7] Bowles P. Feasibility of Detailed Vehicle Modeling. SAE-2001-01-0334, Society of Automotive Engineers. 2001.

[8] Sahlin, P., Sowell E.F. A Neutral Format for Building Simulation Models. Proceedings of the IBPSA Building Simulation '89 conference, Vancouver, Canada. 1989.

[9] Sahlin P. Development of a Component Model Translator for the Neutral Model Format. Final Report ASHRAE 839-RP, Dept. of Building Sciences, KTH, Stockholm, June 1996.

[10] Bring, A., Sahlin P., Vuolle M. Models for Building Indoor Climate and Energy Simulation. Report of IEA SHC Task 22 Building Energy Analysis Tools Subtask B Model Documentation, KTH, Stockholm, Sweden, 1999.

[11] Sahlin P. The Methods of 2020 for Building Envelope and HVAC Systems Simulation - Will the Present Tools Survive?. Proc.of the joint ASHRAE and CIBSE conference Dublin 2000, Dublin, Ireland, Sept. 2000.

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

[13] P.Sahlin, E.F.Sowell, „A Neutral Format for Building Simulation Models", Proceedings of the IBPSA Building Simulation '89 conference, Vancouver, Canada, 1989

[14] J.W. Demmel, J.R. Gilbert and X.S. Li, "SuperLU User's Guide", Technical Report, UC Berkeley, USA, 1997

[15] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, "MUMPS Multifrontal Massively Parallel Solver v. 2.0", Technical Report, CERFACS, France, 1998

[16] T.A. Davis, "UMFPACK v. 4.0 User Guide", Technical Report, Univ. of Florida, Gainesville, USA, 2002

[17] C. Panagiotopoulos "Finite element models in a lumped model simulation environment. An interface between FEMLAB and IDA S.E." Technical Report, KTH, Stockholm, 2001