Building Simulation 2005

Ninth International IBPSA Conference
Montréal, Canada
August 15-18, 2005

NEW EVOLUTIONS IN TRNSYS – A SELECTION OF VERSION 16 FEATURES

David BRADLEY[1] and Michaël KUMMERT[2]
[1]Thermal Energy System Specialists, LLC, Madison, WI (USA)
[2]University of Wisconsin - Solar Energy Laboratory, Madison, WI (USA)

bradley@tess-inc.com kummert@sel.me.wisc.edu

ABSTRACT

Throughout its thirty year history, the transient energy simulation package TRNSYS has been under continual enhancement by an international group of developers and users. This paper briefly describes a subset of the features that were added to the simulation package with the release of its 16th version in November, 2004.

BACKGROUND

The TRNSYS Simulation Package [1] traces its roots to a joint project between the University of Wisconsin – Madison's Solar Energy Laboratory and the University of Colorado – Boulder in which a passive solar building was constructed in Boulder and simulated in Madison. Toward the end of the project, the addition of a furnace bypass duct required a half day's work to modify the heating system and two months of reprogramming the simulation. It became apparent (as it did for other groups developing similar simulation engines at the same time) that a more flexible tool was needed. The basic idea of TRNSYS's basic philosophy is to model each piece of an energy system as an individual black box component. Simulating a system entails connecting the inputs and outputs of the black boxes to one another and if certain models are lacking, developing them and adding them quickly to the package.

Commercially available since 1975, TRNSYS was originally developed for the simulation of solar thermal processes. Over its lifetime, the tool has been expanded into a full fledged building energy modeling package; it has been used in projects to study air conditioning systems in cars, combined heat and power systems, and traffic flow to name a few.

In order to insure that TRNSYS would not become stagnant and eventually be relegated to simulating only a narrow range of systems, as few assumptions as possible were built into the components. The multizone building model, for example, models the thermal zones of the building but makes no assumption about the way in which energy is added to or removed from the zones. All systems such as air and water loops are separate entities and indeed are provided not as components unto themselves but as a variety of component parts: pipes, pumps, ducts, fans, supply and return plenums, mixing valves, etc. While this can make the simulation of "standard" buildings more cumbersome, the advantage comes in complete system flexibility from the user's perspective.

As a final point concerning modeling philosophy, TRNSYS often provides multiple components and multiple methodologies for dealing with a given problem. Detailed and simplified models of the same piece of equipment are often available. For example, TRNSYS contains a number of thermal storage tank models that vary in complexity, in the required input data and therefore in the amount of time required to add the component to a given simulation. The advantage is that the user can add detailed models to the system where detail is warranted and can use simplified models where it is not. In comparing two control strategies for a domestic water heating system, the storage tank may remain the same between comparison cases, being of less importance relative to the details of the control strategy. Therefore less time can be spent in specifying building data, provided of course that the selected building model is not so simplified that it does not include required phenomena. In the sections that follow, TRNSYS's multiple approaches to the same problem will be addressed in examining new program features.

TRNSYS's current version (16.0) was released in November 2004. While by no means an exhaustive survey of new features, this paper seeks to present a few of the more significant modifications.

COMPONENT LIBRARY ENHANCEMENTS

GROUND COUPLING

While the energy transfer between a conditioned building and the surrounding ground is of obvious importance, historically, simulation programs such as TRNSYS have used approximative methods for estimating it's effect; analytical solutions are

unwieldy for all but the simplest cases and numerical methods have been too computationally time consuming to be a realistic option. TRNSYS and its peer software have used methods such as defining an effective layer of massless insulation underneath the building and driving the resulting slab heat transfer with ambient temperatures. ASHRAE's effective Heat Flow method [2] has also been implemented in TRNSYS and other simulation packages.

In light of recent interest in better prediction methods, a set of promising new finite difference models have been developed and implemented successfully in TRNSYS [3]. The method relies on defining a volume of soil beneath and beyond the boundaries of a rectangular slab. The volume (called the near field) is broken up into isothermal nodes whose dimensions increase exponentially as they extend away from the slab in the x, y and z (depth) directions. Massless insulation can be inserted around the slab perimeter, underneath the slab and indeed at any node boundary in the near field. The outer edge of the near field can be assumed adiabatic or can be made to interact with an infinite energy source / sink called the far field whose temperature is a function only of depth and time of year. In one version, fluid filled pipes can be defined to follow any path through the slab and near field, flexibly modeling dynamic radiant heating and cooling systems. In third version, basement walls and floors extend into the near field soil volume.

Because TRNSYS includes more than one kind of building model two sets of ground coupling models are needed. Certain multizone TRNSYS building models take a lumped capacitance approach in which it is assumed that zone energy loss is governed by a first order differential equation. This type of building model provides an air temperature, which is passed to the ground coupling model. The ground coupling model includes information about the capacitance of the slab as well as its top surface convective and radiative heat transfer coefficients. Based on this physical data as well as information about the soil properties, the ground coupling component calculates a slab temperature and the amount of energy transfer through the slab. The second type of TRNSYS building model uses the conduction transfer function method, which includes the concept of a boundary wall. With boundary walls, the building model must be provided with the temperature on the back surface of the wall. This type of building model performs internal radiation and convection exchange calculations and in so doing computes the temperature of the boundary wall's inside surface. The building model passes the interior surface temperature to the ground coupling model. The ground coupling model in return calculates the back side temperature, which it passes back to the building model. In order that the slab can be embedded in the ground and account for perimeter losses, the capacitance of the slab material is specified not in the building but in the ground model. The building's boundary wall is simply the floor covering material (tile, gypcrete, carpet, etc.).

One other feature of the ground coupling model is that a text based output file of each near field node temperature can be generated at any time during the simulation. The same file syntax can be used to provide the ground model with precalculated initial node temperatures at the start of a simulation. Not only is this advantageous for understanding the energy transfer in the ground after running a simulation but it can also be used as a time saving device in later simulation runs of the same system. Normally, the building is run for approximately 5 years to precondition the ground and to reach a periodic steady state solution of energy transfer between the ground and building. At the end of this time, the ground temperature profile is output to a file. Thereafter, annual simulations can be run using the preconditioned ground temperature profile in place of a five year preconditioning period.

Benchmarking models is always of vital importance. To that end, a new series of Bestest [4] test cases are under development at the National Renewable Energy Laboratory. The test cases begin with a simple case that can be solved analytically (GC10) and proceed toward more and more complex cases. Final results from TRNSYS and the other software packages being used to develop the benchmark are not yet published. However, early results show that the TRNSYS ground coupling model is uniquely able to perform all the test scenarios.

COMBINED THERMAL / AIR FLOW MODELING

The TRNSYS Developers have taken three approaches to implementing combined building thermal / air flow simulations. These approaches make use of a software link between TRNSYS (for the multizone building thermal simulation) and either COMIS [5] or CONTAM [6], (bulk air flow modeling software packages) for the air flow simulation.

Where a computational fluid dynamics (CFD) package breaks the building spaces up into finite elements and performs flow calculations based upon extensively defined boundary conditions and driving forces, a bulk air flow modeling package defines isopressure nodes throughout the building. These nodes are connected to one another by a non linear network of air flow paths. The flow of air through any given path is a function of the temperature and pressure difference on either side of the path. Based on zone temperatures output by the TRNSYS multizone building model, CONTAM and COMIS calculate the resulting interzonal air flows and pass

them back to the TRNSYS building model. Using this updated information, the building model recalculates the zone temperatures. Iteration continues at each time step until the air flows, the zone temperatures, and all other connected system output variables change by less than a user defined tolerated amount.

Two methods leave the component programs (TRNSYS, COMIS, and CONTAM) in an unmodified state. The CONTAM solution engine (called AIRNET), whose more recent versions are written in C++ was recoded into Fortran. Its simulation control was removed and it was recast as a TRNSYS component. The user runs CONTAM (as provided free of charge by NIST) to define a multinode air flow network that corresponds to the building. Once the building is created, the user selects an option to generate an *.air file. This text file contains the building description, a list of inputs that must be provided (ambient pressure, wind speed, wind direction, and zone temperatures) and a list of outputs that will be generated by the recast Fortran version of CONTAM. When the user drags the CONTAM link component out into a system being simulated, s/he is asked to provide the location of the *.air file. At this point, the TRNSYS graphical engine reads the *.air file and assembles a list of required inputs and provided outputs that can be linked to and from other components in the system. Figure 1 shows the completed link in a system.

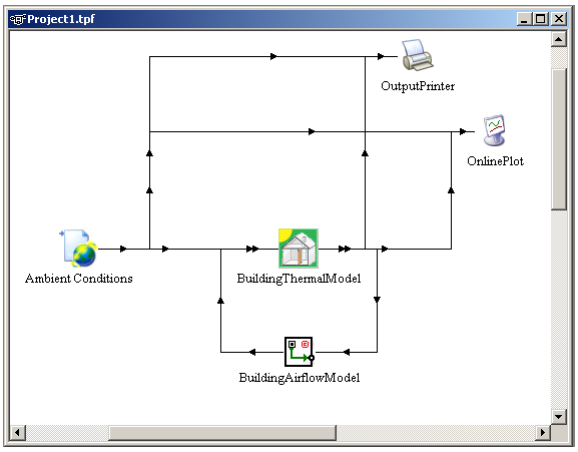

Figure 1: Completed TRNSYS / CONTAM Link

The link between COMIS and TRNSYS is fundamentally the same as that between CONTAM and TRNSYS. The only real difference is that COMIS (freely available from LBNL) does not have a graphical front end of its own. However, the same group that develops the TRNSYS front end has also developed a commercial add-on front end for COMIS. As such the add on COMIS front end and the standard TRNSYS front end have much the same look and feel. The other difference in the COMIS / TRNSYS link is that COMIS's native form is Fortran. This avoids the disadvantage of having to take a periodic "snapshots" of the air flow modeling package, then converting those snapshots to Fortran for TRNSYS's sake. It should also be mentioned that a modification has been made to the TRNSYS kernel, allowing not only components written in Fortran but components written in any language and compiled as Dynamic Link Libraries (DLLs) to be loaded and used TRNSYS during a simulation. A detailed discussion of this feature is included in the "Simulation Engine Modifications" section of this paper.

There are two significant disadvantages in the links described above. First, CONTAM includes a multiplier on each of the air flow links. In the native program, these multipliers usually have a value of 1 or 0 in order to activate or deactivate a given link. They are either set by the user or are modulated internally based on defined control strategies. Unfortuantely, the multipliers are not included in the list of available model inputs and so while they can be scheduled inside CONTAM, they cannot be triggered by events in the TRNSYS solution. From a user's point of view, this makes natural ventilation studies particularly cumbersome since there is no way to open or close a window when certain indoor conditions occur. It is worth noting that this disadvantage is a purely technical issue that will no doubt be resolved in future versions of the link. It is also worth noting that the COMIS can read a triggered event (called an "actual value" input) from TRNSYS.

The second, somewhat more fundamental disadvantage is that while a significant amount of information is shared between TRNSYS and either CONTAM or COMIS, because the air flow modeling programs have their own front ends, the user is obliged to enter the information twice, once for the thermal model and once for the air flow model. The solution to this problem lies in the third link in which the informational screens required by the COMIS air flow model have been added to the thermal model definition interface, TRNBuild. This preprocessing program is the standard interface that allows users to enter information about the layers comprising the building walls but adds screens for entering information about wind pressure coefficients on the buildings façades and about the air links connecting zones. This last link methodology is again an optional, commercial add-on to the standard TRNSYS package.

This third link method also differs from the other two in that instead of the air flow solution being contained in a TRNSYS air flow component and the thermal solution being containing in a TRNSYS multizone building component, the two pieces of Fortran code are merged into one integrated solution.

The disadvantage of the TRNBuild integrated method is that while it increases ease of use, it also decreases the flexibility of the link. Necessarily, in the integrated method, the air flow pressure nodes and the thermal zones of the building coincide. With the other two link methodologies, the air flow model and the thermal model can make assumptions that are most appropriate to themselves while in the integrated method, care must be taken to make assumptions that are appropriate to both models at once.

HYDROGEN SYSTEM COMPONENTS

In 1998 a research group at the Institute for Energy Studies (IFE) in Norway developed a series of TRNSYS components called Hydrogems [7] for modeling hydrogen power systems. Included were models of an akaline electrolyzer, a battery bank, a Proton Exchange Membrane (PEM) fuel cell, an akaline fuel cell, an electric generator with various fuel sources, a gas storage tank, and a gas compressor as well as models of wind turbines, photovoltaic arrays, power converters, and controllers. Originally developed as a freely available add on to the standard TRNSYS v. 15 package, the Hydrogems library has been completely integrated into the standard TRNSYS package. Technical documentation on the models and their assumptions are also included.

This new library of power production components uses a slightly different approach than do many of the other TRNSYS components. While they fit in well with the rest of the package, this difference bears mentioning as it impacts the way in which the components interact with other parts of the system. In most of the standard TRNSYS models, control of a given piece of equipment occurs outside the component model. A pump for example, is equipped with a control signal input. When the control signal is set to zero, the pump is off and when it is set to one, the pump is on, providing its rated flow and drawing its rated power. When set between zero and one, the pump (provided that it is a variable speed pump model) provides some fraction of its rated flow and draws some fraction of its rated power. The pump does not have any control algorithms (intelligence) built into it. By contrast, many of the Hydrogems components do have such intelligence built into them. Perhaps the simplest example is the photovoltaic (PV) array. The power produced by a photovoltaic array depends not only on ambient conditions but also upon the voltage of the load to which it is connected. Were the model written without any internal intelligence, the user would be obliged to provide the load voltage and the PV model would calculate its corresponding performance. More often than not, photovoltaic arrays are connected to semiconductor devices called maximum power point trackers (MPPT), which artificially adjust the electrical load the the array sees in order to keep the array producing the maximum possible power for the current ambient conditions. In order to model such devices, the Hydrogems PV model includes not only a mode of operation to account for direct connected PVs but also contains modes for built in intelligent controls such as would be needed to model a PV with an MPPT. In essence the PV model is not just a PV model but a PV / MPPT / controller model. It is worth noting that not only the Hydrogems PV model but also the standard TRNSYS PV model contain this internal control feature as MPPTs are so common in solar electric installations.

A more complex example, and one unique to the Hydrogems Library is the alkaline electrolyzer model. Electrolyzers have strict power requirements; their cells cannot be operated above a certain current density or below a certain voltage. While these requirements could be met in modelling by creating an external component that took input power and broke it up into appropriate current and voltage, this becomes cumbersome because as with the air flow / thermal simulation there is a great deal of overlap between the information required by the electrolyzer itself and the information required by the controller if it is to correctly maintain the electrolyzer's inputs within bounds. Instead of writing two models, the power requirement calculations are performed within the electrolyzer component itself; the user provides input current and input power and the electrolyzer calculates the voltage at which the input power must be provided in order to operate within its requirements. It is then up to the user to include a power converter model between the power source and the electrolyzer in case the input voltage needs to be stepped up or stepped down based on the electrolyzer's calculations. Figure 3 shows the required informational flow for the Hydrogems electrolyzer model.

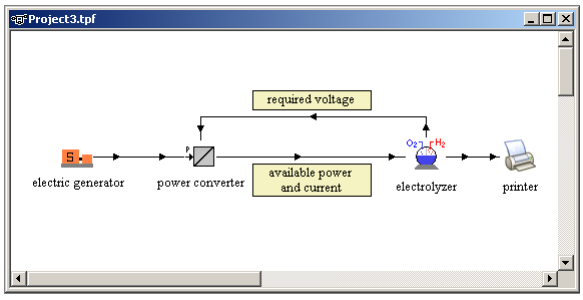

Figure 2: Electrolyzer Informational Flow

While the Hydrogems components internalize micro controls as discussed above, they also necessitate macro controls. In this, the Hydrogems library lends itself exceedingly well to the ease with which new TRNSYS components can be written and incorporated into a simulation. Imagine a power system that includes photovoltaics, wind turbines, a battery bank, an electrolyzer, an electrical load and a

fuel cell as backup power. The issue at hand in such a system is not only the performance of the individual pieces but dispatching the components, deciding what to do with excess power and from where to draw backup power under given sets of circumstances. Such systems are so unique in nature that writing a generic macro controller / dispatcher is almost impossible. While a few dispatcher components are included as examples, it is more often necessary to create a new component in TRNSYS and to write specific controller components for these power generating systems.

EXTERNAL PROGRAM CALLS

Another area of development has focused on adding new content to TRNSYS not by redeveloping native versions of features already well integrated into other software packages but by providing flexible links between TRNSYS and external programs. This concept is not new to TRNSYS; the release of TRNSYS 15 saw the addition of a link to Engineering Equation Solver (EES) [8] and a generalized methodology for calling external programs during simulations as though they were TRNSYS components. With the release of version 16, new links have been added between TRNSYS and Microsoft Excel [9], Matlab [10] and GenOpt [11]. While the strengths of Excel and Matlab are fairly well know, GenOpt is less so.

GenOpt was developed at Lawrence Berkeley National Labs as a generic optimization engine for complex simulation packages such as TRNSYS, DOE2, EnergyPlus, etc. The user defines a set of simulation variables (component parameters or other simulation constants) and an error function that is to be minimized. In TRNSYS's case, this error function can be a user defined equation or can be a component output. GenOpt then repeatedly runs TRNSYS simulations of a user selected duration (a year, for example), collects the results and continues to modify the set of simulation parameters until the results reach a minimum.

The links between TRNSYS and EES, Excel and Matlab are essentially transparent to the user. Once the link component is set up, the simulation proceeds normally. The TRNSYS GenOpt link is somewhat different in that GenOpt actually controls the simulation and the user sets up the optimization beforehand using a preprocessor program called TRNOPT. TRNOPT's interface is shown in figure below Figure 3.

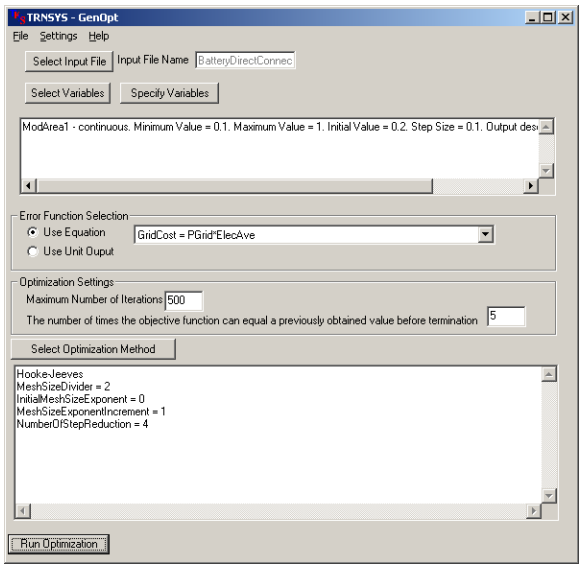

Figure 3: TRNOPT Input Window

SIMULATION ENGINE MODIFICATIONS

In addition to new component content, the TRNSYS simulation engine (called the kernel) has undergone important changes as well. Foremost, with the release of TRNSYS 16, the simulation starting time was redefined; in previous versions, the user specified the simulation starting time as the time at the end of the first timestep. Each component was called for initializations prior to the simulation beginning then again at the end of the first time step. With version 16, the users specifies the simulation start time as the hour of the year at which the simulation is to begin. Each component is called once for initialization, once before time has progressed, then once at the end of the fist time step. Because of this additional call and the component level changes that it entailed, an opportunity to largely rewrite the kernel presented itself. Driving the rewrite was the goal of simplifying the process of adding new components and in giving users better access to functions and features already present in the kernel.

DROP-IN DLLS

Perhaps the most evident new feature in the TRNSYS kernel is that of drop-in Dynamic Link Libraries (DLLs). A DLL is a piece or pieces of source code that are compiled and linked into a single file in a standardized Microsoft Windows [12] form. Because of the standardized form, DLLs, regardless of the programming language used to create them, can communicate with one another through the Windows Operating System. During its initialization phase, the TRNSYS kernel searches a particular folder within the TRNSYS directory structure for files with the extension *.dll. It opens each of these files and looks for subroutines that are

exported with the name TypeXXX where XXX is a number. Since all TRNSYS components have a name of this form, the kernel is able to load components that have been written in any programming language and saved in the correct format and location.

Besides allowing users to develop components in whatever language they prefer, drop in DLLs have a great advantage in distributing TRNSYS. Since there are so many optional add-ons of component libraries available from different developers, the onus has historically been on the user to have a Fortran compiler and to be able to recompile one DLL that contains all of the required components. With drop-in DLLs, the requirement is shifted to the component developers. Provided that the developers create a DLL, the user can place that DLL in the appropriate folder and make use of its contents without having to recompile standard TRNSYS or their existing component libraries.

Equations

One of TRNSYS strength lies in the user's ability to define equations directly within the input file. These equations have historically had to be fairly simple in nature: one unknown variable on the left set equal to some combination of known variables on the right. If the variable "A" is to be used in an equation, it had to be defined earlier in the input file. In TRNSYS 16, however, the equations are preprocessed and blocked during a simulation initialization step. By blocking the equations, the TRNSYS kernel is able to determine the most efficient order in which they should be solved.

GRAPHICAL INTERFACE ENHANCEMENTS

For a time, TRNSYS users had the option between two possible graphical front ends to the package, IISiBat and Presim. Based on an iconographical representation of the system, the front ends generated the text based TRNSYS input file and managed simulations. With the release of TRNSYS 16, only one of the two graphical interfaces has been continued; IISiBat has become a broader program called the TRNSYS Simulation Studio. In order to increase TRNSYS's usability, the launching and manipulation of TRNSYS's peripheral pre and post processing applications such as PREP and TRNBuild have come under the umbrella of the Simulation Studio. Furthermore, new project wizards have been added to help users create new building-based simulations, create new solar water heating simulations and to create new TRNSYS components. While by no means an exhaustive list of graphical interface enhancements, this section highlights three specific improvements.

The Output Manager

The TRNSYS Developers have historically and intentionally shied away from adding a great deal or post-processing content to TRNSYS. Their reasoning has been that they did not want to limit the kind of output that users were able to get by predefining reports and that there are innumerable spreadsheet and data processing programs available to convert raw data into publishable results; reinventing these features would not be development time well used. That said, TRNSYS 16 includes simplified methods for creating output data. Previously, any component output could be sent to a printer or integrator / printer combination that the user had already defined. Defining a printer entailed selecting a model, setting up its parameters and inputs, then linking the outputs to the appropraitely defined inputs. With TRNSYS 16, each output field is followed by a check box. A user has simply to click on the check box whereupon a printer component will automatically be created. When the simulation has completed, the user will find a text based output file containing the value of each checked output at each time step of the simulation.

If data printed at each time step is not desired, the user has free reign over all of the printers, integrators, online plots, and other output devices using the output manager shown in Figure 4.

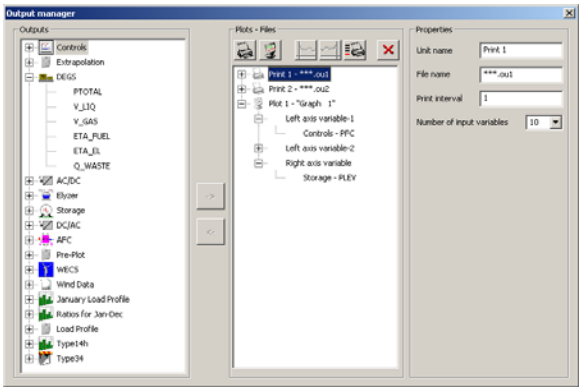

Figure 4: Simulation Studio Output Manager

Plugins

The standard method of defining a component in a TRNSYS simulation is to drag out an icon representing the component, then to double click it and specify the values of the model's parameters (time independent inputs) and input initial values. The iconographic representation of the component is called a Studio proforma. While filling in numerical values is ideal for many components, it can become burdensome for more complex models or for models that simply require a great deal of input information. To facillitate using these more complex components in simulations, an information exchange method called a Plugin has been developed and implemented.

Plugins are mini applications designed to simplify the data entry for one particular TRNSYS component. Once a component is defined in a simulation, the user double clicks it to open its proforma. If a Plugin is available, an icon will appear in the lower left hand corner of the window. Clicking this icon launches the Plugin and passes the current parameter and input initial value information to the Plugin. Once the user has finished entering information into the Plugin screen or screens, information is automatically passed back to the proforma through the "OK" button. An example Plugin for specifying a time dependent forcing function is shown in Figure 5.

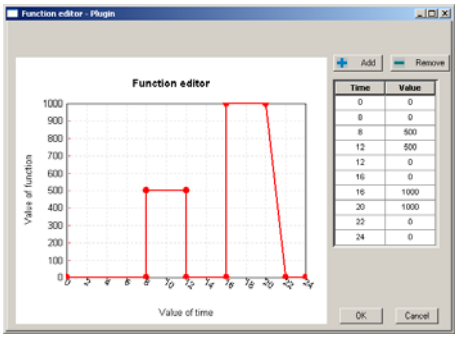

Figure 5: Simulation Studio Plugins

New Component Generation

Under normal circumstances, the proforma presents a graphical representation of a component's parameters, inputs, outputs, to the user. While the component's source code can be developed first and afterwards be added to the Studio by filling out a blank proforma, it is now possible to go the other way thanks to the standardized format of a TRNSYS component. The user first fills out a proforma, specifying the parameters, inputs and outputs for the model. After assigning the proforma a Type number (the component's name as far as TRNSYS is concerned), the Studio is then able to insert the user provided information into a component template. Furthermore, because the workspace and project files that specify the compiler settings necessary to generate a fully functional DLL that can be dropped into the userlib folder and be loaded and used in a simulation are text based, the Studio creates them as well. All that is left for the user to do once the new component has been exported to either Fortran or C++, is to add the equations that calculate the output values based on the parameter and input values. This is often no small task; however, the Simulation Studio is able to take care of a significant part of the procedure.

CONCLUSION

As simulation tools evolve, not only must they rise to meet the challenges posed by new technologies but they must also look back on existing solutions and update them to make use of better algorithms and updated computing resources. This paper has presented some of the solutions to those challenges as proposed by the latest TRNSYS version. In this latest version, particular attention has been focused on adding new component models to the program and on increasing the package's ease of use, continuing the trend of moving TRNSYS from an academic research engine to a manageable commercial tool.

ACKNOWLEDGMENTS

While the efforts of the TRNSYS Developers Group (The Solar Energy Laboratory at the University of Winconsin – Madison, USA, the Centre Scientifique et Technique du Bâtiment in Nice, France, and Transsolar Energietechnik GmBH in Stuttgart, Germany) remain central to TRNSYS's improvement, the importance and generosity of TRNSYS's international user base cannot be understated. Were it not for user contributions of components, methods, and expertise, TRNSYS would be a less extensive and less flexible simulation tool.

REFERENCES

[1] Klein, et. al., TRNSYS – A Transient System Simulation Program User Manual, The Solar Energy Laboratory, University of Wisconsin – Madison., 2005.

[2] 1997 ASHRAE Handbook – Fundamentals, Chapter 27

[3] Bradley, D.B., T.P. McDowell, and J.W. Thornton, TESS Libraries version 2.0, November 2004

[4] Judkof, R. Ground Coupling Bestest, National Renewable Energy Laboratory, Golden, CO

[5] Feustel, H., COMIS – An International Multizone Air-Flow and Contaminant Transport Model. LBNL Technical Report LBNL-42182. Lawrence Berkeley National Laboratory, 1998

[6] Dols, W.S. and G.W. Walton, CONTAMW 2.0 User Manual, NISTIR 6921, 2002

[7] Ulleberg, Ø. Hydrogems Users Manual, 2003

[8] Klein, S.A., et al. Engineering Equation Solver version 6 Professional, *f*-Chart Software, 2004

[9] Microsoft Corporation, Microsoft Excel, Microsoft Excel is a registered trademark, 2004

[10] The Mathworks, Inc., MATLAB and Simulink are registered trademarks, 2004.

[11] Wetter, M., GenOpt® Generic Optimization Program User Manual version 2.0.0, LBNL

Technical Report LBNL-54199, Lawrence Berkeley National Laboratory, 2004.

[12] Microsoft Corporation, Microsoft Windows, Microsoft Windows is a registered trademark. 2004.