Simulation Synergy: Interconnecting Simulation Programs

David Bradley and Nathan Blair
Solar Energy Laboratory
University of Wisconsin - Madison
Madison WI 53706

ABSTRACT

One of the drawbacks faced in increasing the use of simulation programs for renewable energy system design is that new models must be written in a specific programming language. Often times models developed for one platform must be translated to the language used by the simulation program. As an example, researchers use Engineering Equation Solver to develop steady state simulation models. However, in order to use those models within a larger system or in a transient mode with TRNSYS, the models must be rewritten in FORTRAN, a time consuming process. If better communication between programs existed, the problem would be greatly diminished. This paper describes the creation of a synergy between different simulation tools using three methods of linking programs together: calling other Windows programs, calling Dynamic Link Libraries, and using Dynamic Data Exchange.

1. BACKGROUND

Many of the current energy simulation programs such as DOE2, BLAST and TRNSYS saw their first versions released as many as 25 years ago. At the time, individual researchers crafted simulation tools to meet their own needs. FORTRAN was used extensively in these tools as it was well known and was independent of computer platform. A great deal has changed in the computer world and in the engineering industry over the intervening years and developers of simulation tools are now working to find new ways to adapt.

One of the most important features of the above mentioned simulation tools is their modularity. In the case of TRNSYS, individual component models (solar collectors, pumps, tanks, etc.) are represented by separate FORTRAN subroutines. New subroutines can be written (in FORTRAN) by users and can be incorporated into simulations.

However, far fewer researchers study programming currently and if they do, they rarely study FORTRAN. In order for researchers to expand the capabilities of a simulation tool, they must first learn FORTRAN. Furthermore, since the researcher's ultimate goal is not the creation of a new FORTRAN subroutine, the routines tend not to be rigorously tested and often contain bugs. These bugs diminish the reusability of the models by future users.

Instead of forcing researchers to learn an outmoded programming language, it would be preferable to either allow them to program in any language, or to incorporate familiar tools such as Engineering Equation Solver, MatLAB, Mathematica and Excel into the simulation tool.

Another trend affecting the world of renewable energy simulation is the overwhelming acceptance of the Microsoft Windows operating systems on desktop computers. The prevalence of Windows has made it possible to add graphical tools to simulation programs, including plotting tools, graphical user interfaces, and CAD based tools. Furthermore, the convergence on a single operating system has made development and use of protocols for communication between programs much more worthwhile.

2. SOLUTION METHODOLOGIES

By taking advantage of the MS Windows operating system, various methods have been used to create a synergy between different, commonly used simulation tools. The methods used can be broken into several categories: calling and being called by other Windows-

based programs, calling Dynamic Link Libraries, and using Dynamic Data Exchange to send and receive messages.

2.1 Calling And Being Called By Other Programs

Perhaps the most basic mode of communication between two programs is to allow one program to "call" another. In other words, while one program is running, allow it to launch a second program, wait for completion of a task and then continue its calculations. A distinct advantage of this solution is that the second program need not be modified. However, the process of calling and waiting is slow, especially in simulations where the second program must be called at each timestep.

TRNSYS [1] currently contains a number of utility routines: subroutines that perform common tasks such as the calculation of thermodynamic properties of fluids. These utility routines are called from within component models such as a pump or a chiller. In order to allow calls to external programs, a utility subroutine named "CALLPROGRAM" has been created. CALLPROGRAM uses several Win32 Application Programming Interface (API) commands to start the external program. API commands directly control the Windows operating system. Depending on the mode specified in the call statement, CALLPROGRAM will either wait for the second program to finish its task before proceeding or will merely start the second program or then proceed with other TRNSYS calculations. The call statement for CALLPROGRAM is:

CALL CALLPROGRAM(CMDLINE,bwait,prochand, thrdhand)

Where:
CMDLINE is the command line text string containing the path and name for the executable program
BWAIT is a logical variable that determines whether TRNSYS will wait for the second program to finish running or not.
PROCHAND and THRDHAND are Windows variables that may be useful in some cases.

In order to make use of the information generated by the second program, it was necessary to develop a method for inserting data into the TRNSYS input file. Until now, the TRNSYS input file has been one continuous text file containing all the information that describes the system being simulated. However, in certain situations it would be advantageous to have part of the TRNSYS input file (for example, equations passed from another program) in a separate file and reference it with an INCLUDE statement from the main file. In this way, it is possible to change items or rewrite the include file without changing the main TRNSYS file. The syntax would look like:

INCLUDE c:\trnwin\file.inc

TRNSYS treats all the statements in the include file as though they were in the main file.

2.2 Calling Dynamic Link Libraries

Because of the slow speed, calling an external program and waiting for it to finish is often an unacceptable solution. However more and more often, Windows based programs are comprised of both an executable, and a dynamic link library (DLL). The executable contains all the program's peripheral tools as well as the interface and calls the DLL, which is essentially the compiled source code. The DLL can be accessed quickly because none of the program's overhead actions, such as screen refreshing, are required. Another distinct advantage to interacting directly with DLLs is that since a DLL is compiled and linked, its original language is irrelevant. Simulation programs that can access external DLLs can use component models written in any language.

A component was added to TRNSYS that makes calls to DLLs possible. Type 61, allows a user to write a component in any language (as long as the language supports DLLs) and then link that component into TRNSYS. The code of Type 61 handles the standard TRNSYS initialization routines and then calls the DLL called "EXTDLL". The calling command for the DLL, within Type 61, is:

CALL EXTDLL(Spass,Sarraypass,simarray,xin,out,t,dtdt, par,info,icntrl)

TRNSYS passes its standard arrays of information to the DLL (parameters, info, inputs, etc.) but also passes several other arrays. In this manner, EXTDLL has every capability available to a normal TRNSYS component. The three additional arrays contain information accessible to other TRNSYS components through FORTRAN common blocks. The SPASS array is a character string that can be used for passing either error or text messages back to TRNSYS from the external DLL. The SARRAYPASS array is a 100-place array set aside for values that need to be saved from one timestep to the next. This capability is normally provided to standard TRNSYS components through the S-array. The SIMARRAY array passes the current timestep value, the simulation start time, stop time and timestep values to the DLL.

As an alternative to using Type 61, it is possible to write a DLL using the standard TRNSYS call statement, place the DLL in the TRNSYS "userlib" directory and call it as though it were a normal TRNSYS component. No matter what language the component is written in, it can be

directly used within the TRNSYS simulation. Unlike the EXTDLL call described above, the SPASS, SIMARRAY, and SARRAYPASS arrays are not included. The advantage of this method is that adding components becomes extremely simple; it suffices to place them into the proper "userlib" directory.

The capability of loading and using external DLL's has been successfully applied to REFPROP [2], a program designed to calculate the thermodynamic properties of pure refrigerant fluids as well as those of user defined mixtures. The connection to REFPROP can replace the TRNSYS fluids subroutine, which makes the same calculations for a limited selection of refrigerants.

2.3 Using Dynamic Data Exchange

The third method used to interconnect simulation programs is Dynamic Data Exchange (DDE); a method of sending messages to between programs through the Microsoft Windows operating system. DDE is a special set of Windows API commands that allow two programs to communicate. Many programs support DDE, including the applications within Microsoft Office and Engineering Equation Solver [3]. Engineering Equation Solver (EES) is a non-linear equation solver that incorporates two distinct advantages for energy simulation work. First, EES allows equations to be entered in any order with unknown variables placed anywhere in the equations. Second, EES provides numerous built-in mathematical and thermophysical property functions. For example, the steam tables are implemented such that any thermodynamic property can be obtained from a built-in function call in terms of any two other properties. Unfortunately, researchers wishing to use EES models with TRNSYS need to rewrite them in FORTRAN. Using DDE, however, removes this necessity.

A TRNSYS component (Type66) performs all the necessary DDE message handling between TRNSYS and EES as shown in Figure 1.

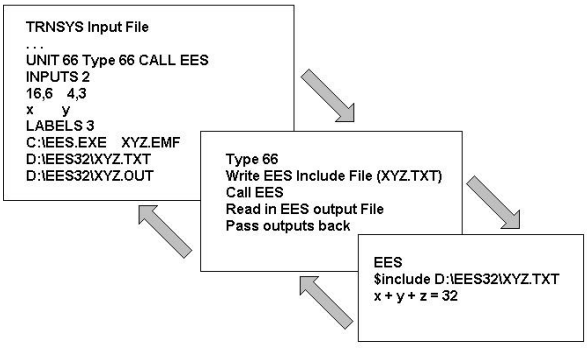

Fig. 1: Type66 DDE Message Handling Between TRNSYS and EES

Type 66 allows the user to call an EES file, pass it inputs and receive outputs from EES using DDE and text files to exchange the information and commands. To begin, the user creates a model in EES, then decides which variables should be inputs and which should be outputs. The variables chosen to be inputs are placed in an include file automatically. The EES program is set up to look for the include file, manipulate the inputs and then set the outputs as shown in Figure 2.

```
$include E:\EES32\XYZ.INC
x^2 = Out[1]
x + y = Out[2]
```

Fig. 2: A sample EES file

The user then enters information into the TRNSYS input file in the form of a standard type description. An example is shown in Figure 3:

```
UNIT 66 TYPE 66 CALL EES XYZ
INPUTS 1
xvalue yvalue
x y
LABELS 3
*the command line to run ees
c:\EES32\EES.EXE c:\EES32\XYZ.EES
*the include file to write
c:\EES32\XYZ.INC
*the output file that ees writes
c:\EES32\XYZ.OUT
```

Fig. 3: TRNSYS input file call to EES

The above type description is similar to a standard TRNSYS type description with the exception that there are no parameters, only inputs. The first line of the inputs specifies either the variable names or the TRNSYS component outputs that contain the values of the inputs at each time step. The second line of inputs contains the variable names that are to be printed in the include file by Type66. The assignment of variable names is necessary because EES is expecting them to have specific names. There are also three LABELS associated with this type. The first label is the command line that runs the EES file. This tells TRNSYS where to look for EES, where to find the EES file to be run and to start EES if it is not already running. The second LABEL is the name of the Include file into which TRNSYS will place the input values and their labels. The third LABEL is the name and location of the output file from which TRNSYS will read the EES solution.

In the course of a simulation timestep, new values for the inputs (xvalue and yvalue in this case) are set by other components in the simulation. These new values, along

with the labels ("x" and "y") are written to the EES include file. Then, Type 66 sends a DDE message to EES to open and calculate the EES file (c:\EES32\XYZ.EES). EES does this calculation and writes the results to an output file (c:\EES32\xyz.out). Type 66 then opens the output file, reads in the values for the OUT array (OUT[1] and OUT[2]) and passes these values to the TRNSYS solver to be used as inputs for other components.

3. SPECIFIC EXAMPLES

Simulation programs often have specific requirements as to the presentation of their data. The three basic methods described above require slight modification for individual cases.

3.1 Using TRNSYS Components in EES

EES has the ability to use external procedures and functions contained in DLL's. A method has been developed on the TRNSYS side, which allow standard TRNSYS components to be called from EES. The format for the call statement that EES requires is:

SUBROUTINE MDASF(S,MODE,NINPUTS,INPUTS, NOUTPUTS,OUTPUTS)

Whereas, the call statement for a TRNSYS component is:

SUBROUTINE TYPE144(TIME,XIN,OUT,T,DTDT, PAR,INFO,ICNTRL,*)

Evidently, there is a mismatch between the two call statements. A "wrapper" subroutine is written for each TRNSYS type that is to be called by EES. The wrapper subroutine rearranges the input variables coming from EES and then calls the TRNSYS component with the proper call statement. In this way, TRNSYS components can be used without modification in EES programs.

3.2. Energy-10 – TRNSYS Interaction

Energy-10 [4] is a simulation program geared towards smaller buildings and specifically designed to evaluate energy-efficient building features (including solar thermal and photovoltaics) in the very early stages of the design process. Since there are already simulation programs specifically designed to analyze renewable energy features, it would require a great deal of redundant development to add these capabilities to Energy-10 as well. Instead, Energy-10 will call TRNSYS to simulate the performance of the renewable energy feature under consideration. Specifically, Energy-10 will create a text file containing a few critical design parameters such as PV array size and orientation, and inverter efficiency and then call TRNSYS. TRNSYS will then run a ready-made simulation and pick up the information in the text file, inserting those parameters into the input file at the proper location. Once its simulation is complete, TRNSYS leaves behind an output file containing the performance data, which is incorporated into the Energy-10 simulation.

3.3 EnergyPlus – TRNSYS Interaction

Recent work has gone into developing a connection between EnergyPlus [5] and TRNSYS. EnergyPlus is a new simulation tool based on the marriage between DOE2 and BLAST. EnergyPlus will include its own capability to simulate building energy use but will rely on TRNSYS to simulate the impact of renewable energy systems. The first generation of the EnergyPlus-TRNSYS connection will be quite similar to the Energy-10-TRNSYS connection. One disadvantage to this method, however, is that TRNSYS can be called only once per simulation. Further developments are planned to fully integrate the two programs so that TRNSYS can be called at each time step.

4. CONCLUSIONS

Throughout most of their history, simulation programs have been written in a programming language appropriate to the situation, most often FORTRAN. Any subsequent developments had to be made in the original language. However, with the dominance of Microsoft Windows in the computer market and the ever-decreasing number of developers fluent in FORTRAN, an opportunity exists to break free from dependence on FORTRAN. Recent developments in one simulation program, TRNSYS, have centered on the creation of new methods of communication between applications. The use of predefined Windows operating system tools has replaced the need for translating existing models into FORTRAN. Major techniques employed include automatically calling other programs, calling Dynamic Link Libraries written in any programming language, and using Windows operating system commands. Combinations of these techniques have been employed to allow TRNSYS to interact with EES (Engineering Equation Solver), Energy-10 (architectural energy tool), REPROP (thermodynamic fluid properties), and EnergyPlus (building simulation tool in development). The techniques employed are quite general and can be applied to any FORTRAN based simulation tool. The use of these interaction techniques should reduce the amount of "re-coding" or re-inventing the wheel that is a current disadvantage of concentrating use on any single simulation tool.

5. REFERENCES

[1] Klein, S.A., et al. TRNSYS vers. 14.2. Energy Simulation Software. Solar Energy Laboratory, University of Wisconsin-Madison, 1996
[2] REFPROP. vers. 6.0. Software for the calculation of thermodynamic and transport properties of refrigerants and refrigerant mixtures. National Institute of Standards and Technology, 1996
[3] Klein, S.A., Engineering Equation Solver. vers. 4.93. Equation Solving Software. F-Chart Software, 1998
[4] Energy-10. Software for the design of low-energy buildings, The Passive Solar Industries Council, 1997
[5] EnergyPlus. Building Energy Simulation Software. US Department of Energy, 1998