Development of a Combination Boiler Simulator using General Purpose Simulation Tools

Philip S. Bartells
President
SIMTECH Services, Inc.
1237 Woodbrook Lane
Forest, VA 24551

Joseph S. Gauthier
Director, Applications Engineering
AEgis Technologies Group, Inc.
6703 Odyssey Dr., Suite 200
Huntsville, AL 35806

KEYWORDS

Boiler Modeling, Simulator, Simulation, Software, Open Systems, Application Programming Interface, API, Graphical User Interface, GUI, SAMA control diagrams.

ABSTRACT

This paper describes the methodology with which a detailed simulation of a typical Combination Boiler (multiple-fuel) was developed for use in training and analysis. The boiler that was simulated is a vintage Riley Traveling Grate Spreader Stoker-fired drum unit that is fired by crushed coal and wood bark. It is controlled by a Digital Control System (DCS) and is installed in a paper-mill powerhouse, which is owned by a major paper-manufacturing corporation. Many of the personnel who have operated this boiler for the last 30 to 40 years are nearing retirement age, and the customer felt it was important to have a tool which could be used to familiarize new personnel with boiler operations, and to provide training for current personnel as they worked their way to Operator status. In addition, it could be used to evaluate system design changes prior to them being implemented on the real system.

The customer desired a simulation platform that was "open," that is, not dependent on a control vendor or special hardware. It needed to run in real-time to be useful for training, and it was also required that it be on a personal computer platform utilizing the Windows operating system. The simulator was constructed using general-purpose simulation software and graphic user interface (GUI) tools. The simulator system was delivered on a networked three-PC platform, with touch-screens and a special keyboard for emulating the actual DCS keyboard. The resulting simulation includes very detailed mathematical models of the combustion and steam generation processes. The simulator is used to teach startup, shutdown, maneuvering and the proper response to various system upsets. It was also utilized as the platform for testing design changes which significantly modified the over-fire air, coal and bark fuel systems, and a complete revision to the boiler control strategy.

INTRODUCTION

This paper describes the methodology with which a detailed simulation of a typical Combination Boiler (multiple-fuel) was developed for use in training and analysis. The customer and powerhouse are typical of what is found in the Pulp & Paper industry, and the name is unimportant. The approach taken could be applied to similar situations. This paper focuses on the technical aspects of the development process rather than the use of the completed product.

The boiler that was simulated is a vintage (late '50s) Riley Traveling Grate Spreader Stoker-fired drum unit that is fired by crushed coal and wood bark. This boiler is one of two in the powerhouse, which are used to supply steam to the 850 PSI steam header. It is controlled by a Honeywell (TDC 3000) Digital Control System (DCS), which was initially installed in 1990.

Many of the personnel who have operated this boiler for the last 30 to 40 years are nearing retirement age, and the customer felt it was important to have a tool which could be used to familiarize new personnel with boiler operations, and to provide training for current personnel as they worked their way to Operator status. In addition, it could be used to evaluate system design changes prior to them being implemented on the real system. This last goal turned out to be very significant as the boiler was upgraded two years later.

The customer desired a simulation platform that was "open," that is, not dependent on a control vendor or special hardware. It needed to run in real-time to be useful for training, and it was also required that it be on a personal computer platform utilizing the Windows operating system. This was required so they could maintain it themselves.

A long-term goal was to develop a simulator capability that included all of the major equipment that the operators had to deal with. As such, after the Combination Boiler was completed, the same technology was utilized in developing a simulator for a dual-extraction Turbine-Generator, followed by a simulator for the steam header system.

SELECTION OF SOFTWARE TOOLS

Use of a commercial simulator vendor system was ruled out early due to the cost of a software license, and the requirement to have the simulator vendor perform a significant part of the work. The budget did not allow for this approach and a lower cost alternative was needed.

Review of available simulation platforms was done, and the decision to use the Advanced Continuous Simulation Language (ACSL) was made, for reasons that are discussed below. The ACSL family of software is owned and supported by the AEgis Technologies Group, Inc. (www.AEgisTG.com).

The authors have many years of experience with ACSL, and felt confident that it would be able to handle the detailed model that would be required to realistically simulate the boiler and related systems. In addition, an associate had developed a library of component level models that could be used to simulate the boiler and associated systems, and these were compatible. As it turned out, it is extremely

doubtful that the alternative general-purpose simulation platforms could have handled this very complex model and run in real time and faster.

The next problem was to decide on how best to emulate the Operator Screens that are on the DCS, and how to interface those screens to the simulation. For the screens, the authors were aware of a simulator project that utilized Visual Basic (VB) for screen emulations, so it was decided to try and use VB for that task.

At the time, Dynamic Data Exchange (DDE) was the only method available to interface to an ACSL simulation. After discussing the use of DDE with ACSL developers, it was determined that a better approach would be to use Object Linking and Embedding (OLE2) as the data exchange software. A set of routines that would allow programs written in Visual Basic, C or C++ to control an ACSL-based simulation, and to send data to it and retrieve data from it, were available from the ACSL vendor. This set of routines would later become known as the ACSL Open API. Figure 1 shows the software layout.

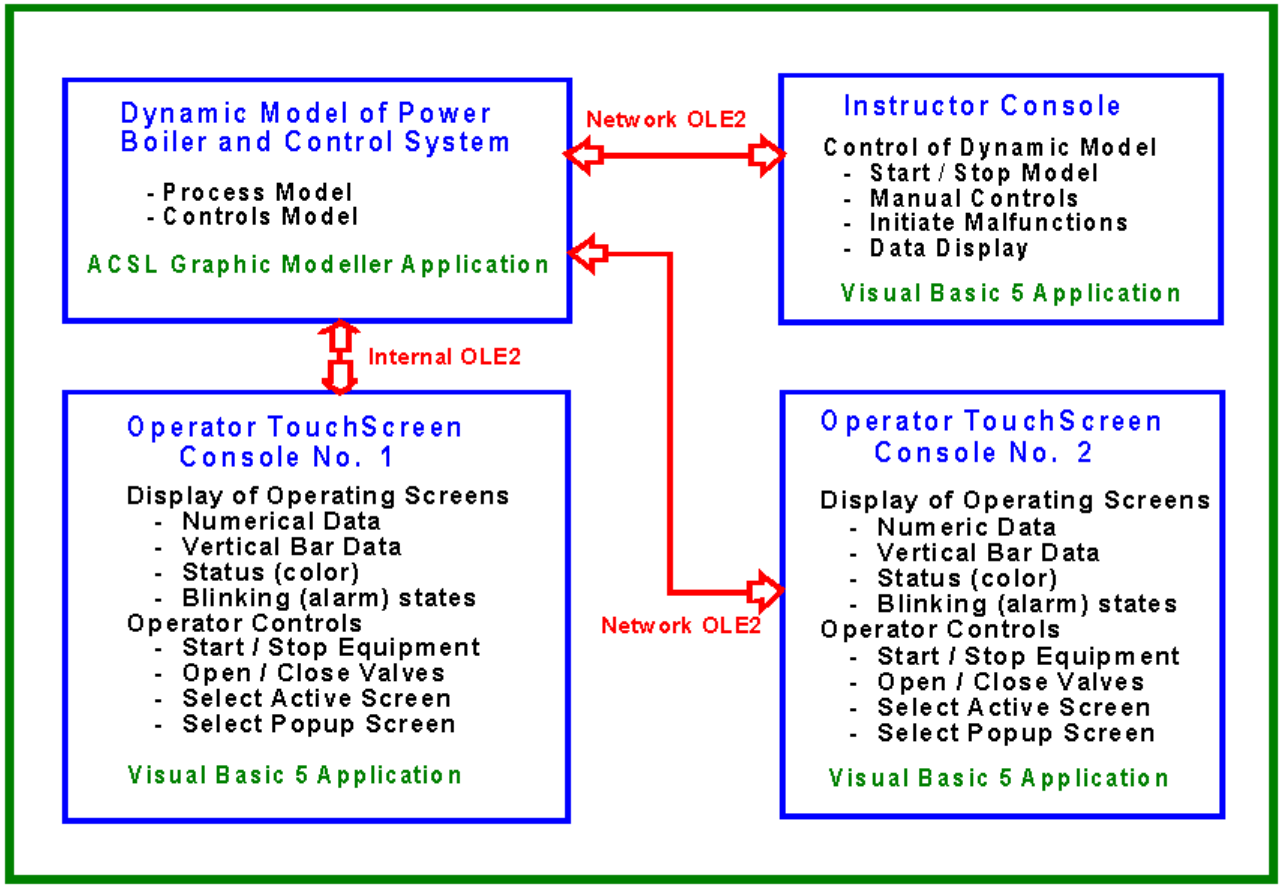

FIGURE 1 – COMBINATION BOILER SIMULATOR SOFTWARE LAYOUT

HARDWARE PLATFORM

The project started with the purchase of two Personal Computers. For the Operator Station a 20-inch MicroTouch Systems touchscreen monitor was purchased. This initial system was delivered in mid-1997 and installed at the mill. As subsequent projects were undertaken, the system grew to include two Operator Stations and one Instructor Station, as shown in Figure 2.

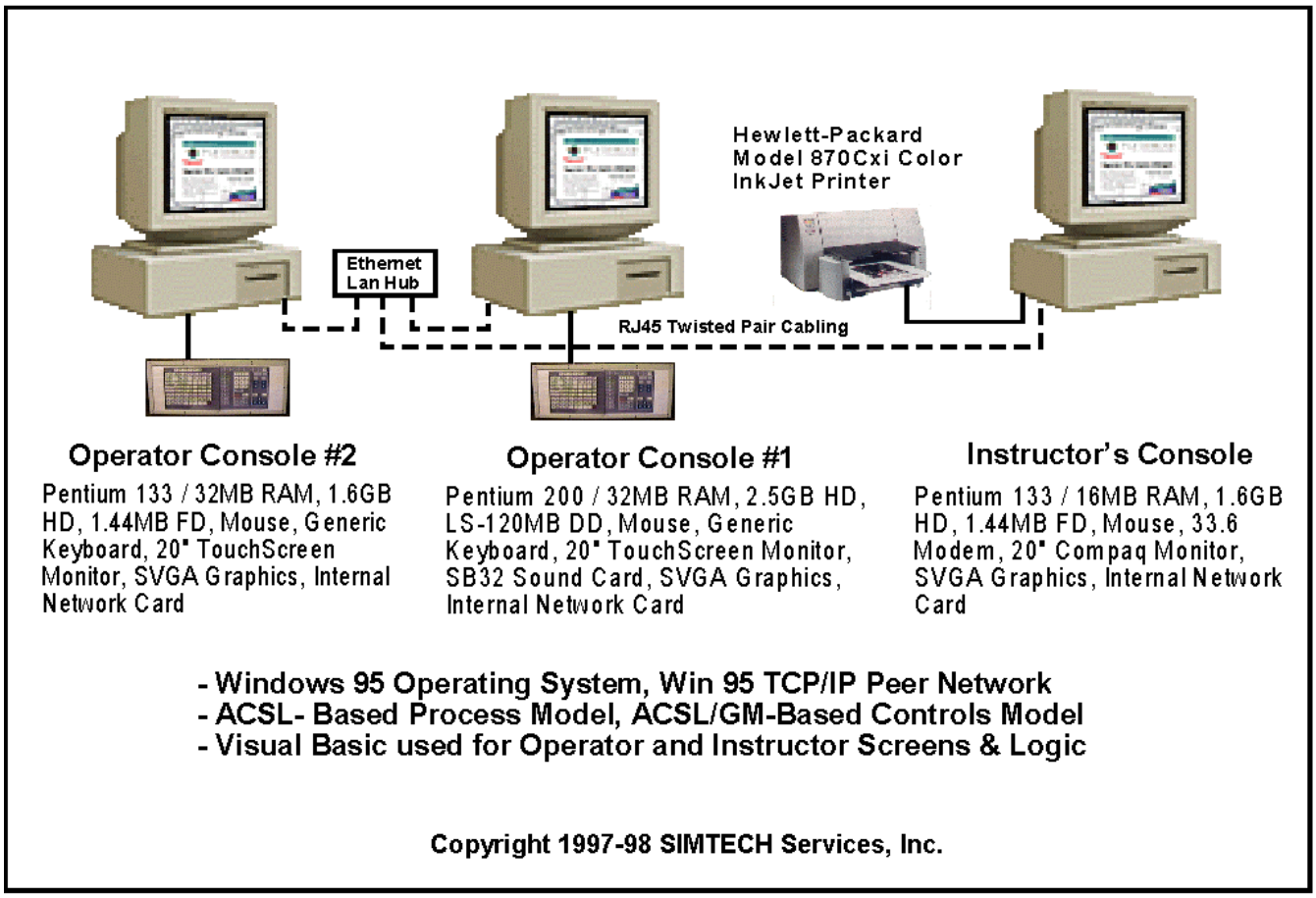

FIGURE 2 - COMBINATION BOILER SIMULATOR HARDWARE

Each Operator Station is comprised of a PC, a 20-inch touchscreen monitor, a mouse, and a special keyboard used to emulate the DCS keyboard. This special keyboard is an array of miniature pushbuttons, and when a key is pressed, a signal is sent to the PC via the serial port. Each key has its own identifying number, so the software reads the number and performs the operation assigned to that number. The Instructor Station is comprised of a single PC with a 17-inch monitor and mouse. From this station the Instructor can start and stop the simulation, change to fast time or real time, initiate malfunctions, etc.

DEVELOPMENT

To have the simulator result in an effective training tool, it was necessary to perform the following tasks:

- Model the boiler in great detail;
- Duplicate the control system logic;
- Duplicate the Operator screens;
- Interface all the pieces and get them working together;
- Test the result against actual data and/or operating experience.

BOILER MODELING

It was necessary to model the boiler in great detail if it could be expected to react to upsets or operator inputs in a realistic manner. A library of FORTRAN-based subroutines was utilized to develop the model of the boiler and related systems. This library is proprietary and was the only piece of software that was utilized that is not (currently) commercially available.

System descriptions and operating procedures were gathered, and together with test data, was used to build the model and test the resulting simulation. This involved development of an ACSL-based model, written in Continuous Simulation Language (CSL) code. This model code contains calls to the appropriate subroutines and includes the data necessary to make the subroutine into a mathematically equivalent representation of the real equipment. The resulting code was documented with a block diagram.

Figure 3 shows some of the water and steam circuits and is just one of several diagrams. Flow stream variables are defined on the block diagram and it is used extensively at run time for debugging purposes.

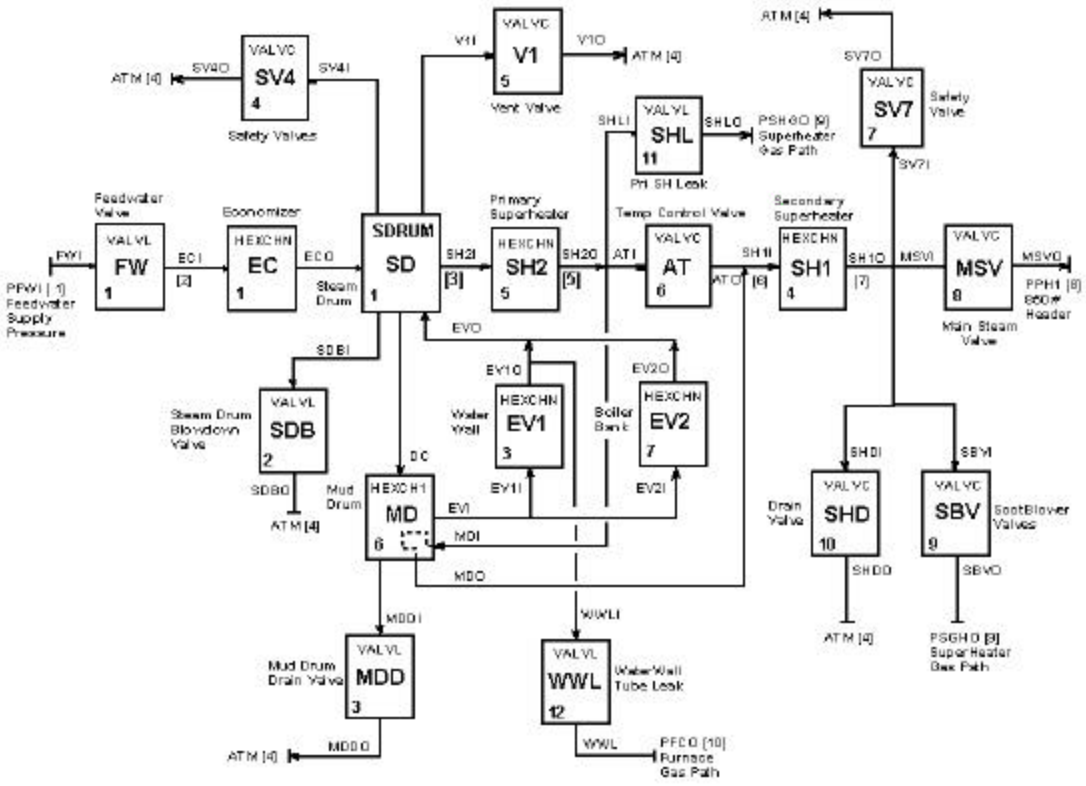

FIGURE 3 – EXAMPLE OF MODEL BLOCK DIAGRAM

Also required to be used with the modules are routines needed to calculate steam and water properties, combustion properties, gas properties, etc. The model inter-connection network is solved using a pressure-flow matrix solution technique, which utilizes the mathematical system of equations and partial derivatives. It is beyond the scope of this paper to describe this solution technique in detail, but it is very similar to that used by the commercial simulator vendors.

CONTROL SYSTEM MODELING

The DCS was documented with the Scientific Apparatus Makers Association (SAMA) standard for diagramming such systems. The documentation supplied for the DCS defines how each control module operates. To provide a realistic representation of the control system, it is necessary to replicate each of these modules as accurately as possible.

Since the SAMA-based control diagrams existed, it was decided to take a graphic modeling approach to this part of the development task. One of the software tools provided with the ACSL SIM Suite is a package called the ACSL Graphic Modeller. It allows the use of pre-defined graphic icons, which can be dragged and dropped and then inter-connected to construct a model.

To utilize this tool, an equivalent graphic module for each control block was developed. As it turned out later, this was a good decision, as the control system was modified dramatically during an upgrade in 1999. A typical example of part of a graphic model screen is shown in Figure 4.

Another advantage to this graphic model approach is that the control blocks can be labeled with the same nomenclature as on the original SAMA diagram.

The graphic model also allows the viewing of the outputs of each control block while the simulation is running. This is a feature that provides significant user feedback when tuning and debugging.

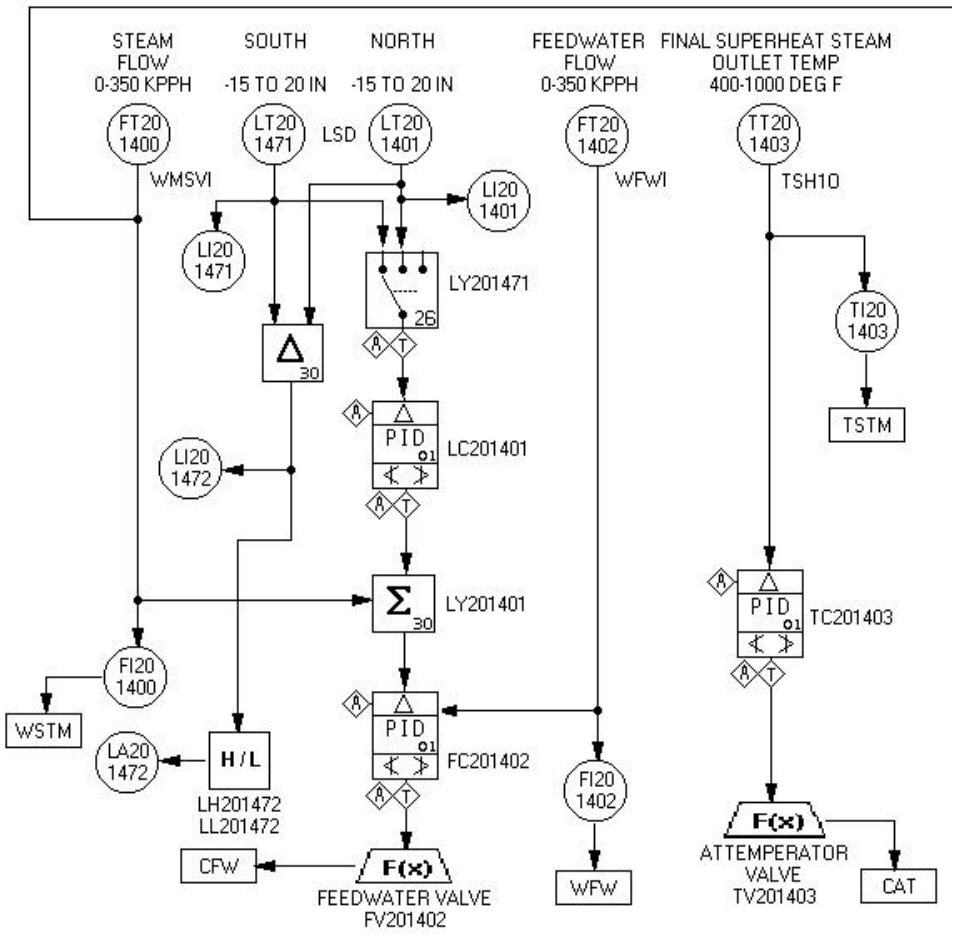

FIGURE 4 – CONTROL DIAGRAM EXAMPLE

OPERATOR SCREENS

Probably the single most important feature for a training simulator is to maintain the look and feel of the actual operating environment. Several graphics packages were considered, but since the package needed to be programmable as well as provide graphics capability, and Visual Basic (VB) had been used on another simulator project, it was decided to utilize it. Another key consideration was the fact that an API between VB and ACSL was being developed. C or C++ could have been utilized as well, but the ability to make changes quickly and see immediate results with VB was the deciding factor.

In order to develop the screen images, photographs of the actual screens were taken, then cropped and enlarged, scanned, and edited. All static parts of the screens became a bitmapped image that was placed on the VB form in the background. All dynamic parts of the screen were added in the foreground, and consisted of bitmapped images or label or text boxes for the most part. This was a fairly labor intensive part of the development, but the resulting screens are virtually indistinguishable from the real ones. Adding up all the main screens (forms) and pop-up screens (used mainly for controls) there are a total of 42 forms used in this simulator; an example is shown in Figure 5.

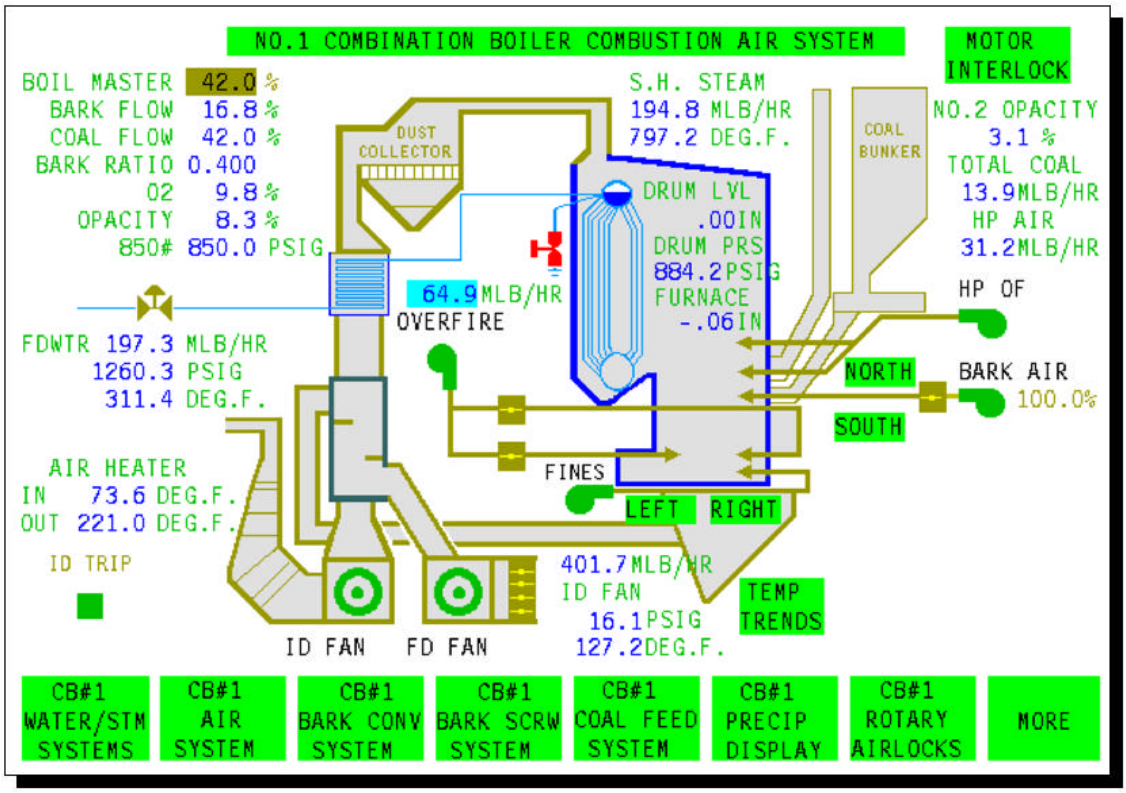

FIGURE 5 - EXAMPLE OF OPERATOR SCREEN

INTERFACE SOFTWARE

Tying all of the pieces together was done by utilizing the Application Programming Interface (API) software, which had been developed for use with ACSL. The way this software operates is by exposing all of the objects in an ACSL-based simulation so values can be read from or written to variable locations in the simulation. It also allows program control for starting and stopping, plotting results, etc. Essentially, it converts a simulation into a server application by using OLE2.

This is now old technology, and is being upgraded to use the newer COM technology. The computers are connected to a TCP/IP Local Area Network (LAN), which uses the standard Windows 95 TCP/IP protocol. Use of this networking required the purchase of the Enterprise Version of Visual Basic.

Referring back to Figure 1, The Instructor Console is connected to the LAN, and is used to control the simulation, which is running on Operator Console 1. Data transfer to and from the Instructor Console is via TCP/IP. Operator Console 1 also has the Operator Screens. Data transfer to and from the simulation is handled by internal OLE2 in this case. An identical set of screens is loaded on Operator Console 2, without a simulation, and it receives and sends data over the LAN via use of TCP/IP. The beauty of this method of data transfer is that multiple Operator Screens can be connected to the LAN, and operate independently of each other. Speed of response is limited only by the LAN speed and the API interface routines.

The Instructor Console (see Figure 6) was developed to allow the trainer to start the simulation, stop it (Freeze), save conditions for use as initial conditions (ICs) for future runs, save conditions for review and evaluation, restore saved conditions, insert malfunctions to see how the trainee responds, change the speed between real time and fast time for situations where things are moving slowly (such as a cool-down), etc. It also allows for use in debugging by allowing the trainer to interrogate the simulation for values, or to set a value in the simulation.

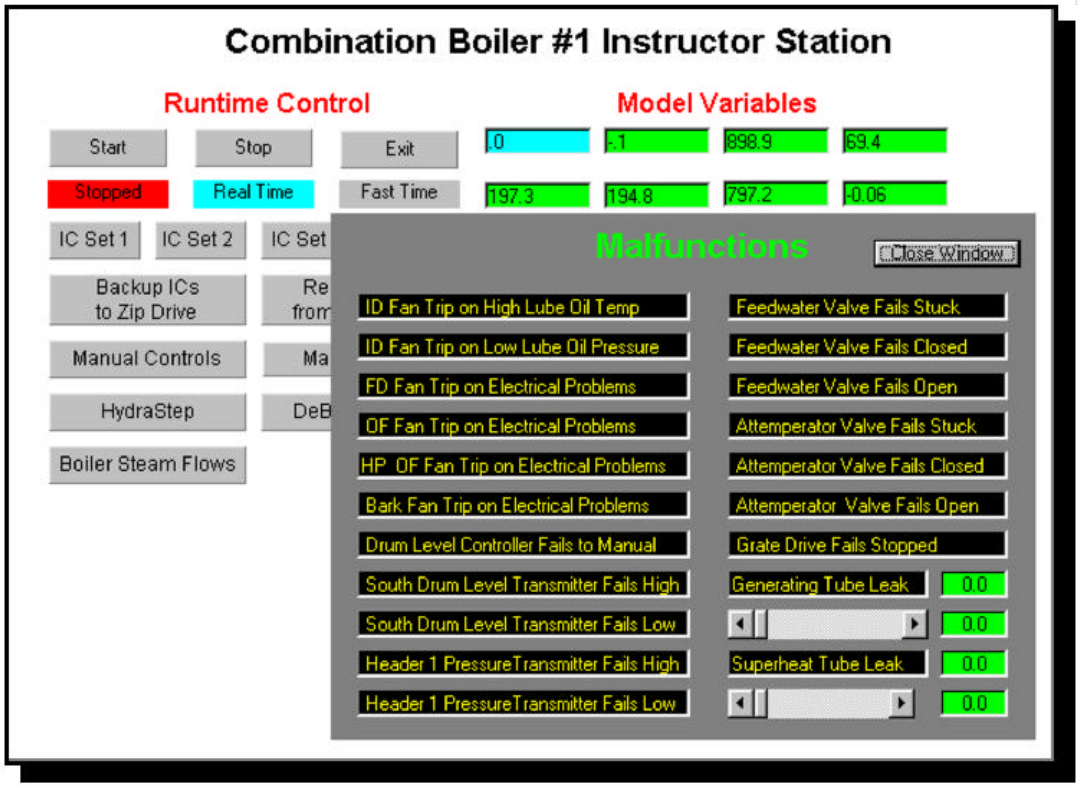

FIGURE 6 – INSTRUCTOR STATION

SYSTEM CHECKOUT AND USE

An extensive checkout of the completed system was made at the mill over a period of several weeks. Changes were made as needed, and the simulator was accepted in the fall of 1997. Elapsed time for the project was approximately one year. The resulting simulation includes very detailed models of the combustion and steam generation processes. The simulator is used for new employee familiarization, to train on startup, shutdown, maneuvering, and system upsets. It is extremely representative of the "look

and feel" of the real control system interface to the real boiler, and includes alarm sounds and other audio cues that are utilized in the real system.

CONTROLS RETROFIT

The actual boiler underwent a complete retrofit of the control system in 1998-99, as part of a project to decrease emissions. A task in the controls upgrade was to modify the simulator so it reflected the new system and would still be useful for training. The details of the boiler upgrade are confidential, but it did involve a new over-fire air system, some new major components, and a controls system design that is based on energy-flow considerations, rather than the more standard mass-flow design.

The new control system design was documented using the SAMA block approach, which was discussed earlier. So to model the new control scheme it was only necessary to redraw the control diagram by using the Graphic Modeller. It was possible to reuse many of the previously developed graphic modules, and just enter new data for gains and other constants, etc. It was also necessary to develop some new modules that had not been used in the previous control design. The new control design was programmed on the simulator at the same time it was being installed on the actual control system.

When it came time to run the new controls on the simulator, some problems were discovered, and after fixing these problems on the simulator, the same revisions were made on the real control system. Once again, the ability to debug the control system programming by displaying values on the SAMA control diagram, was extremely helpful in identifying and correcting problems. The simulator was also utilized to "pre-tune" the actual control system. Proportional Gains and Integral Reset values were tried out on the simulator prior to being used on the actual control system. This saved a significant amount of time when it came to tuning the real control system, since the simulator results provided excellent "ball-park" values with which to start.

The new design and control philosophy necessitated that some changes be made to the operations screens. These new screens were designed by control engineers, and installed on the simulator. After asking the Operations Staff to review the changes, it was realized that there was more information than was needed for the operators to be able to operate efficiently. The screens were simplified and the information needed by the controls engineers was placed on screens other than those used for control.

SUMMARY

This paper has described the use of the general-purpose simulation tools ACSL and ACSL Graphic Modeller, and Graphical User Interface Software (Visual Basic), and a new Application Programming Interface (ACSL OpenAPI) as the basis for construction of a unit-specific training simulator for a multiple-fuel power boiler. Used together with a proprietary set of FORTRAN-based modeling routines, the result was an economical alternative to the use of commercially available simulator software. It satisfied the customer needs, and set the foundation for follow-on work with regard to other mill powerhouse systems.

The existence of the simulator was fortunate when it came time to upgrade the boiler system. Use of it for testing of the new controls resulted in significant savings for that project. The technologies employed to construct this simulator can also applied to non-simulator applications, such as are used for dynamic system analysis for many engineering disciplines.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the contribution of Mr. David S. Weber, P.E, to the early stages of this project. Mr. Weber created the first version of the boiler model, and licensed the use of his software library for this project and the subsequent projects.

REFERENCES

1. Bartells, Philip S. and Weber, David S., Process Modules Library - Developers Manual, SIMTECH Services, Inc., 1237 Woodbrook Lane, Forest, Virginia 24551, August 1998

2. Advanced Continuous Simulation Language Reference Manual, The AEgis Technologies Group, Inc., 6703 Odyssey Dr., Suite 200, Huntsville, AL 35806, Published in 1999.

3. ACSL Model User's Guide for Windows, The AEgis Technologies Group, Inc., 6703 Odyssey Dr., Suite 200, Huntsville, AL 35806, Published on CD December 2000.

4. ACSL Graphic Modeller User's Guide for Windows, The AEgis Technologies Group, Inc., 6703 Odyssey Dr., Suite 200, Huntsville, AL 35806, Published on CD December 2000.

5. ACSL Open API User's Guide, The AEgis Technologies Group, Inc., 6703 Odyssey Dr., Suite 200, Huntsville, AL 35806, Published on CD December 2000.

6. Microsoft Visual Basic Version 5.0 Programmers Guide, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, Copyright 1991-1997.

7. SAMA - Functional Diagramming of Instrument and Control Systems, Scientific Apparatus Maker's Association, 1101 16th Street, N.W., Washington D.C., 20036, September 1981.