

A New Approach to Development and Support of Real-Time Control Systems

Ronald Maslo, PH D, PE - RMK Engineering, P.C.
William C. Cave - Visual Software International, Inc.

April 17, 2006

SUMMARY

This presentation illustrates new engineering techniques to develop, optimize and test real-time control modules found in automated power plant systems. Our focus is to demonstrate the flexibility and ease of use of a special set of engineering tools developed by Visual Software International, namely the *General Simulation System (GSS)* and the *Visual Software Environment (VSE)*. The combination of *GSS* and *VSE* allows a control engineer to analyze, design, build and test process and control system modules without relying on a programmer to communicate between the engineer and the plant computer. More importantly, this can be done using very modular engineering architectures, so that the time to generate individual modules as well as the cost to support the overall system is reduced dramatically. Time and cost reduction factors have been as high as three to ten for equivalent efforts. It is most notable in reduction of personnel on large projects where strict engineering design rules and standards have been followed.

The example used here was selected for ease of understanding by an engineer familiar with the terminology of power plant control requirements and technology. Our intent is for the reader to gain an understanding of the use of these new development and support tools, without concern for the specific control system design.

Important features to be illustrated are those that allow the control engineer to *optimize* the design of an algorithm, as well as verify its operation, using the special simulation environment provided by *GSS*. The optimized model of the algorithm can then be moved into the real-time environment, directly, using *VSE*. *GSS* can be used to *simulate* the *VSE* real-time control module in a native software environment, outside *GSS*, and allow for its dynamic evaluation and modification if required.

The product of this exercise is an efficient real-time control module that has been designed and tested using a number of operational scenarios. Having designed and verified the algorithms using *GSS*, it is automatically converted to optimized operational software by *VSE* and is ready for operation on the plant control computer. The entire procedure is completed by the design engineer- *without support from a programmer*.

DESCRIPTION OF *GSS* AND *VSE*

The *General Simulation System* is a user-friendly, interactive environment that supports model development, scenario development, simulation and analysis. Using a graphical engineering drawing interface, *GSS* users can quickly build models, run simulations and analyze results.

GSS supports modeling along the physical lines of a system. This implies that one can build models that are in one-to-one correspondence to the physical system. To accomplish this, *GSS* features a unique separation of the overall modeling and simulation environment into five distinct segments:

- ARCHITECTURE ENVIRONMENT - Used to create and maintain hierarchical model architectures based upon graphic symbology and menu driven information. It stores the architectural properties of each element and the structural relationship among the elements. It provides the ability to visually inspect the design directly down to the attribute and rule element level, without software abstractions.
- LANGUAGE ENVIRONMENT - Contains a Resource language for describing hierarchical attribute structures, a Process language for describing English-like rules, and a Control Specification language to specify the top level simulation elements. Anyone who understands the system being designed can read and understand these easily.
- RUN TIME ENVIRONMENT - Contains a fast discrete event simulation run time operating environment, allowing the user to interact graphically while the simulation is running, and to interface with real time devices.
- SUPPORT ENVIRONMENT - Allows the user to create scenario files, analyze output files and review interactive graphic playbacks.
- SET UP ENVIRONMENT - Allows the user to tailor a *GSS* session environment, that is selecting language editors, printers and the like.

GSS is used to simulate or emulate the environment of algorithms that form the basis of real-time control systems inside larger operational systems. *VSE* is used to build complete operational software systems. Once algorithms are tested using *GSS* under various operating conditions, they can be converted directly to *VSE* on the computer platform in which they will operate as part of a real-time system. In addition, the complete control system can be connected directly to *GSS* for testing under both live or simulated conditions.

FOSSIL PLANT FUEL CONTROL EXAMPLE

Figure 1 illustrates the fuel flow control module embedded in its real-time environment in a fossil fuel power plant. The fuel flow control module maintains the proper pulverized coal flow to the furnace in response to the main boiler fuel demand.

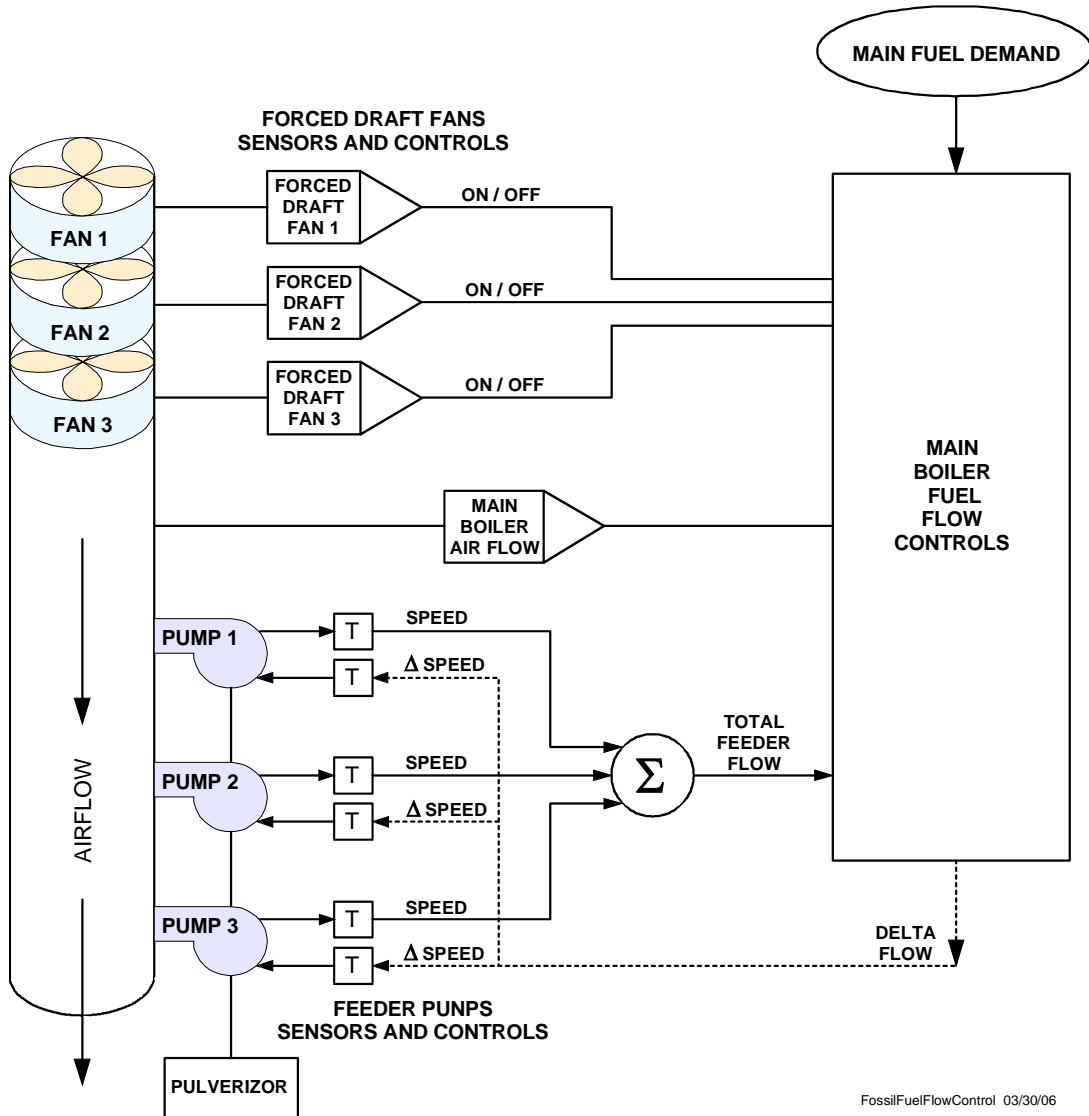


Figure 1. Illustration of the operation of the fuel control system.

In this example, the fuel flow control module compares the main fuel demand with total measured boiler air flow. The lower value is selected as the fuel demand. This selection insures that an amount of fuel equal to or less than the total available air is always selected. Three forced draft fans provide the air flow into the boiler. Should one of the three forced draft fans be out of service, the fuel demand is fixed to a maximum of 5% of the demand value.

Fuel demand is compared with speed signals from three coal feeder pumps. The controller output is a feeder speedup or slowdown signal that compensates for a feeder flow-fuel demand mismatch. This signal is transmitted to the feeders that are in service to adjust their speed. This continues until the total fuel flow matches the fuel demand to within a pre-set tolerance.

SIMULATION IMPLEMENTATION

Implementation of models of this system, so that the fuel flow control module is emulated and interfaced as it would be in a real control system, is straight forward using *GSS*. Figure 2 illustrates four *GSS* models used in this example. Three of these models simulated the FUEL_FLOW_CONTROLS module. These are the MAIN_FUEL_DEMAND, FORCED_DRAFT_FANS, and FEEDER_PUMPS. These models can produce different scenarios for testing response of the flow control module under different plant operating conditions. Measured boiler airflow was treated as a constant in this simple example. The instrument module produces an output file compatible with packages such as Microsoft EXCEL for data analysis, plotting, and reporting.

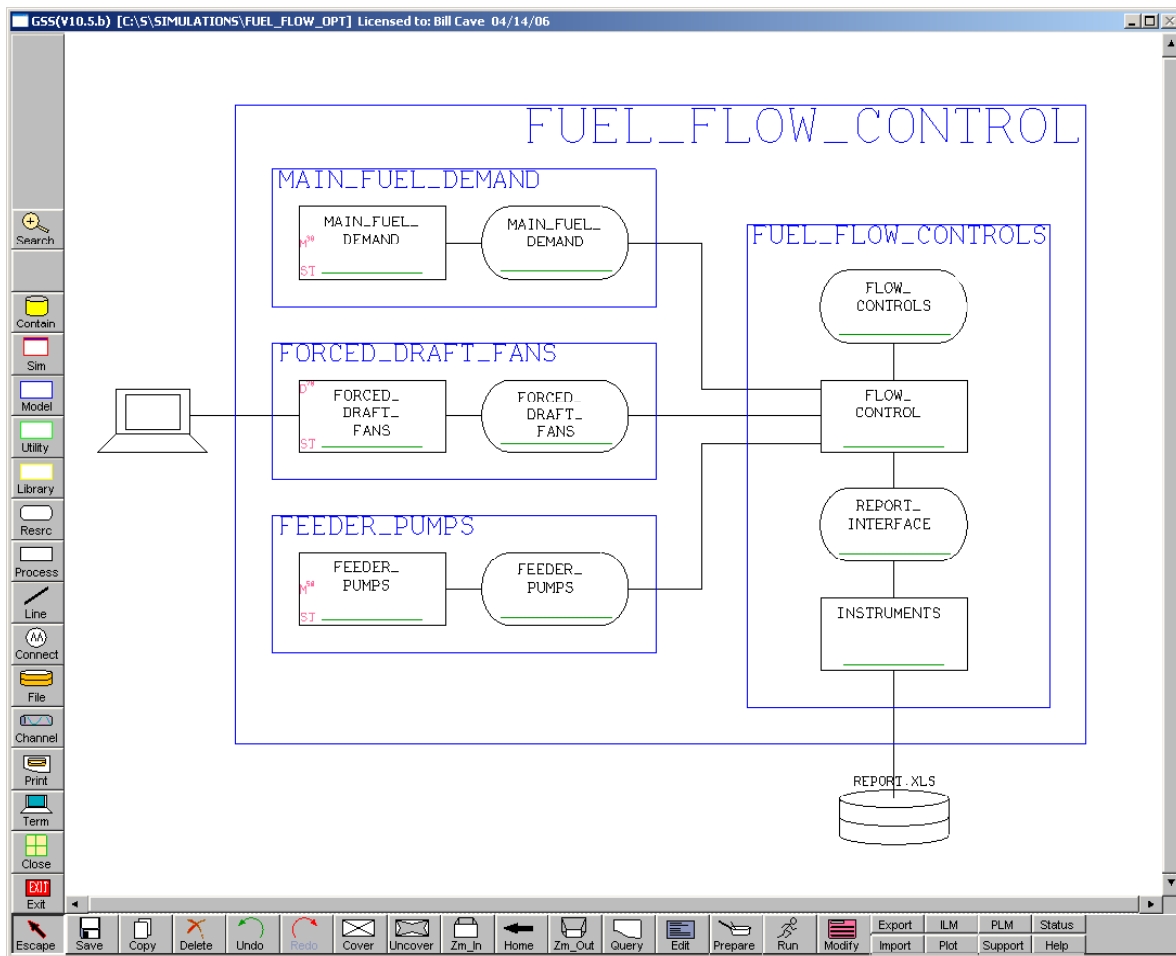


Figure 2. *GSS* Model of the fuel flow control system.

In our example, the FUEL_FLOW_CONTROLS module contains four submodules. Each submodule contains Resources and Processes. Resources are shown as ovals and Processes are shown as rectangles. Resources contain information on the attributes of the entity they represent. An example is shown in Figure 3. They can be thought of as representing distributed subvectors of the state vector of the system. Processes contain rules that sense and change the state of the system attributes. An example is shown in Figure 4.

```

RESOURCE NAME:  FLOW_CONTROLS
CONTROL_FLOW_PARAMETERS
  1  MEASURED_BOILER_AIR_FLOW    REAL  INITIAL_VALUE 25000 *** CF/M
  1  CURRENT_FUEL_DEMAND         REAL
  1  NEW_FUEL_DEMAND             REAL
  1  ADAPTIVE_MULTIPLIER         REAL

```

Figure 3. Resource FLOW_CONTROLS

```

PROCESS NAME:  FLOW_CONTROL
PROCESS_SAMPLE
  CURRENT_FUEL_DEMAND = MEASURED_FUEL_DEMAND
  IF CURRENT_FUEL_DEMAND IS LESS THAN MEASURED_BOILER_AIR_FLOW
    NEW_FUEL_DEMAND = CURRENT_FUEL_DEMAND
  ELSE NEW_FUEL_DEMAND = MEASURED_BOILER_AIR_FLOW .

  IF FAN_STATE(1) IS OFF
  OR FAN_STATE(2) IS OFF
  OR FAN_STATE(3) IS OFF
    NEW_FUEL_DEMAND = NEW_FUEL_DEMAND / 2.0 .

  DELTA_FLOW = ADAPTIVE_MULTIPLIER *
                (NEW_FUEL_DEMAND - TOTAL_FEEDER_FLOW)
  SCHEDULE OPT_FLOW_CONTROL IN 1 MINUTE

```

Figure 4. Process FLOW_CONTROL

MODEL ARCHITECTURE

The key to building simulations fast is development of independent model architectures. Using *GSS*, this is accomplished by following a fairly simple set of design rules. Note the one-to-one correlation between the model architecture in Figure 2, and key elements in the diagram of the plant in Figure 1. Additionally, a one-to-one correlation of plant to attribute names used in the processes and resources eliminates the need for the engineer to keep track of intermediate or arbitrary software names that arise.

The intent of the simulation is to support development and test a module that is to be incorporated into the real-time control system. Therefore, one must identify the interfaces that the real-time module has to the rest of the system. In our example, we have assumed that the flow control module has interfaces to the forced draft fans, feeder pumps, and main fuel demand information via shared memory interfaces within the computer that controls these components. This is an unnecessary assumption when using *GSS* and *VSE*. However it simplifies our illustration. In a real system, these inputs and control responses could be via signals over communications channels that are interfaced just as easily.

Models, resources and processes are created using the *GSS* Computer-Aided Design (CAD) interface in a user-friendly manner. Resources are used to define the parameters of interest, and their initial conditions and limits. Processes contain the rules and conditions for calculations and data manipulations that constitute the algorithms and describe the dynamic performance of each plant component and process. Descriptions of the FLOW_CONTROLS resource and FLOW_CONTROL process used in the FUEL_FLOW_CONTROLS model are shown in Figure 3 and 4 as examples.

Part of the strength of the *GSS* approach lies in the fact that, once defined, reusable modules are stored in a library for future use in other applications. Modifications, if necessary, are simple and straight-forward using engineering vernacular. Renaming of modules is easy, giving the user the opportunity to quickly generate submodules of specific modules.

SIMULATION RESULTS

The fuel flow control simulation was developed on an Intel workstation using UNIX as the Operating System. *GSS* and *VSE* are usable on many PC or workstation platforms, using various Windows, UNIX or Linux based operating systems. *GSS* output files can be generated as Data Interchange Format (DIF) or Standard File Interface files that are compatible with Microsoft EXCEL.

Using EXCEL, graphic results are easily obtained as shown in Figure 5. This plot shows Total Feeder Flow approaching and equaling Measured Boiler Air Flow. This was the objective of the control scheme under development. This result was obtained using a fairly simple but conservative algorithm (slightly more complex than the one shown in Figure 4) for controlling the flow of fuel into the boiler furnace. This algorithm used the difference between NEW_FUEL_DEMAND and TOTAL_FEEDER_FLOW to set DELTA_FLOW to 5% of the difference. With this algorithm, one observes a slow response from the model.

After review of the results of the first design, it was determined that a more efficient algorithm could be build to adapt to the changes in main fuel demand. This was accomplished using the simple adaptive multiplier in the algorithm shown in Figure 4. The ADAPTIVE_MULTIPLIER on DELTA_FLOW in this simple algorithm was optimized using the *GSS* optimization facility. The results are shown in Figure 6, and described in the next section. Thus, using optimization, the adaptive multiplier gave a sufficiently good response with a constant with a value of 0.2.

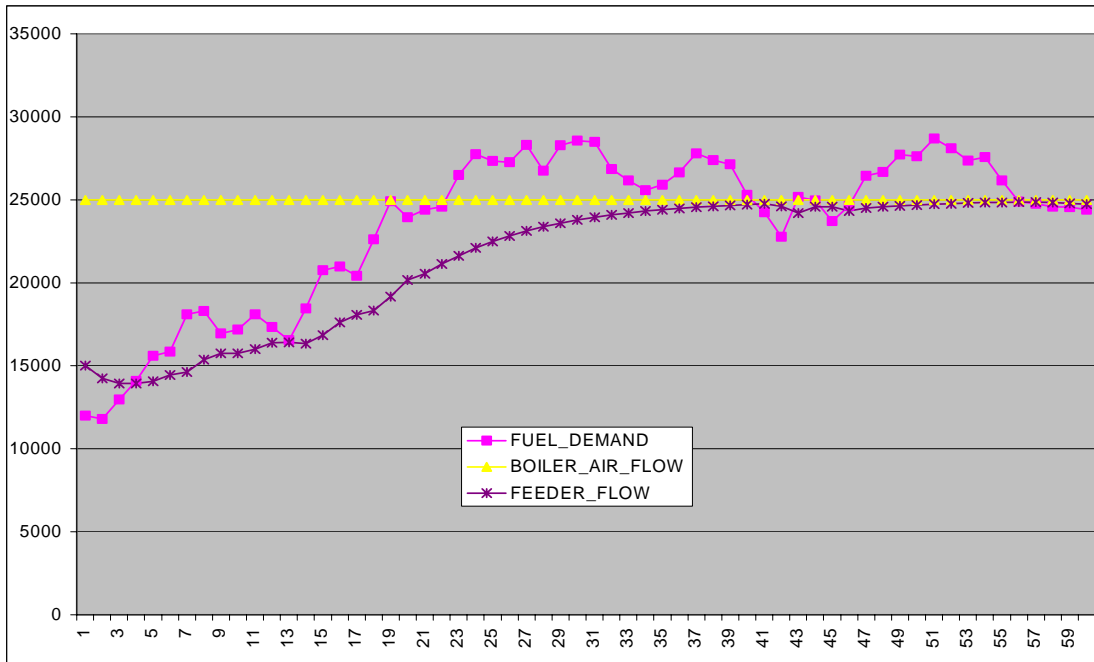


Figure 5. Output results of the fuel flow control simulation.

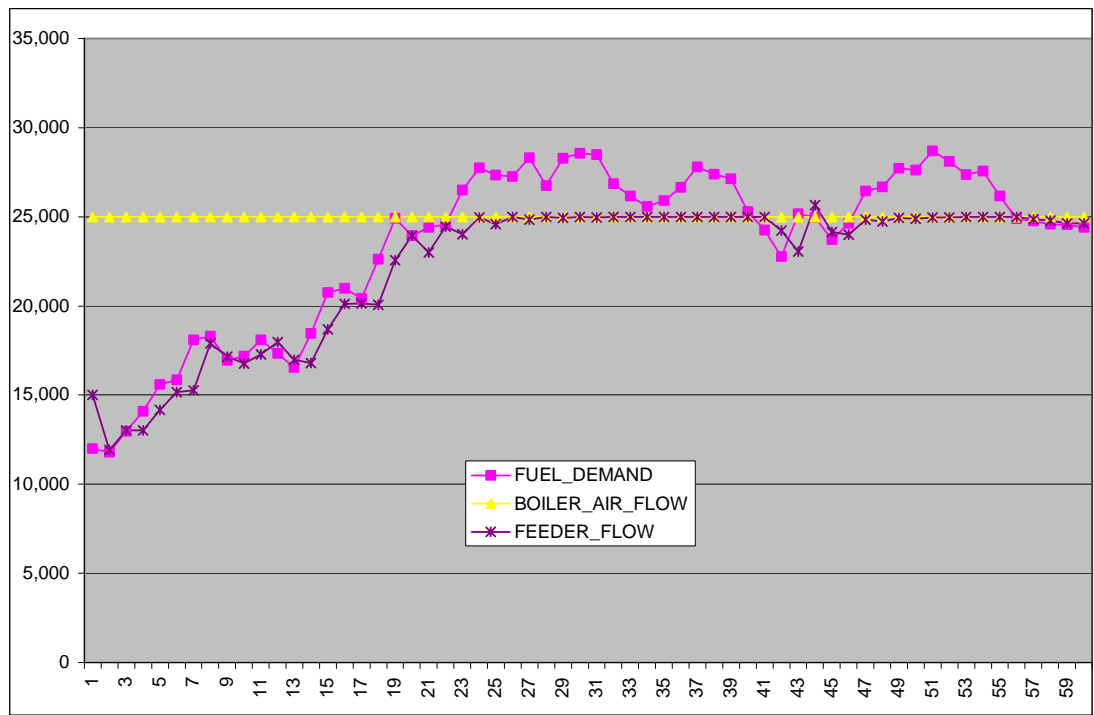


Figure 6. Output results after optimization of the adaptive multiplier

OPTIMIZATION

One of the most powerful facilities of *GSS* is the ability to solve very general constrained nonlinear optimization problems. The *GSS* optimization system and simulation speeds are the basis for this extremely flexible and easy to use facility. Once a simulation is running, it is a simple matter to enter any optimization criteria, including inequality constraints and an objective function. These do not have to be formulated mathematically, one need only assign numeric values to these functions, possibly using conditional statements. In this problem we selected the adaptive multiplier as the unknown parameter to minimize the sum of the absolute deviations between fuel demand and feeder flow. Measured boiler flow was treated as an upper limit. The results are shown in Figure 6.

REAL-TIME TESTING

Having decided upon the design of the flow control module, it remains to incorporate it into a real-time computer environment and test it. This was accomplished by converting the flow control model out of *GSS* and into *VSE* using a set of simple menus. This results in Figure 7.

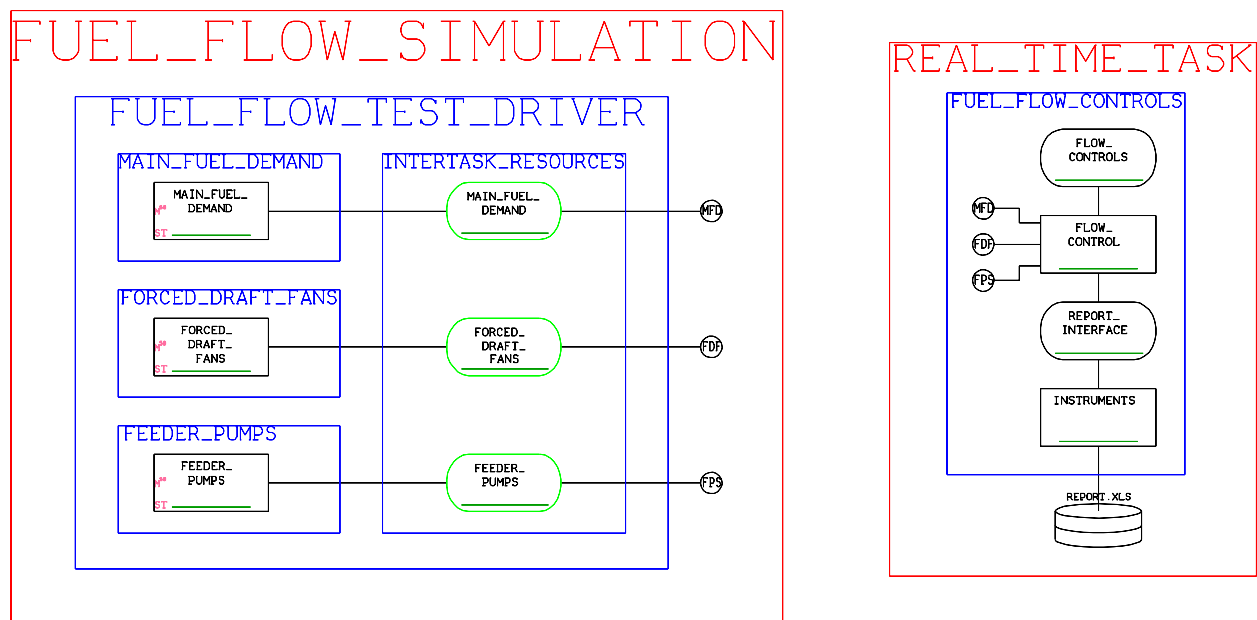


Figure 7. The real-time fuel flow control environment.

With *GSS*, the three models *MAIN_FUEL_DEMAND*, *FORCED_DRAFT_FANS*, and *FEEDER_PUMPS* declare their resources as *intertask*. This is done using the menus of the *GSS* Architecture Environment. The *FUEL_FLOW_CONTROL* module was exported from the *GSS* environment and moved to a *VSE* environment on same the computer where a real-time task was generated. This task could then be moved to a real-time environment. When these tasks are started (there are a number of options for starting multiple tasks in *VSE*), they will run and communicate via the intertask resources.

The results are the same when driven by the same simulated inputs. However, if the intertask resources are the same as those provided by the actual real-time system, the module can now be put into production testing with the actual system simply by sharing the actual intertask resources in the real system.

CONCLUSION

A simple example of the development of a fuel flow control module for a real-time fossil fuel plant control system has been given. This example demonstrates the ability to build and test real-time control systems without the aid of computer programming knowledge. To illustrate the development procedure, a simple flow control algorithm was built and tested, and then replaced by a simple adaptive algorithm that was optimized in the *GSS* simulation environment. The algorithm was tested by simulating the inputs that would stimulate the control module. The plot of Total Feeder Flow and Boiler Air Flow versus Time indicates that the flow control module brings the two flows into an acceptable balance for the given test scenario.

Once the control module is performing to the expectation of the design engineer, the *Visual Software Environment (VSE)* is used to generate a real-time module that can be tested on the actual plant process control computer. This assumes that *VSE* can be installed on the host machine. *VSE* requires a C compiler and Open-GL and currently runs on various Linux, UNIX and Windows platforms. Once *VSE* is running, *GSS* is installed directly on the same computer, since it has been developed in *VSE*. This provides for a real-time test environment on the native control system computer. Using this approach, engineering personnel can build and test real time system modules without worrying about the many sophisticated programming details that such an effort otherwise requires.