AN INSTRUCTOR-FRIENDLY, OBJECT-ORIENTED APPROACH TO TRAINING SIMULATOR CONTROL

M.T. Matthews, W.T. Sneed, and G. F. Malan
Framatome Technologies
Lynchburg, Virginia 24506-0935
e-mail: simulator@framatech.com

Key Words

Trainers, Real-time, Simulation

Introduction

Most real-time training simulation methodologies allow an instructor to control the training simulator's basic functions such as run, freeze, initialization, snapshot, backtrack, time scaling, and invocation of malfunctions/scenarios. This collection of functions is typically called an Instructor Station and is performed on a dedicated computer system that exchanges data over a local area network with the other simulator system computers. Framatome Technologies has extended this basic concept to enhance the instructor capabilities, productivity, and flexibility to readily adapt functions such as scenarios and malfunctions easily and quickly without any low-level programming and model recompilation. These enhancements have been implemented using recent programming language advances that have become available with the programming language C++ and the Microsoft Windows NT environment. The main program for the Framatome Technologies' Instructor Station is called the Action Center, which interfaces with complimentary instructor station programs to provide features for trending and trainee performance review.

Layered Approach to Model Control

Instructor-friendly model control is essential for effective training. Inadequate tools can impede the training experience by requiring a disproportionate amount of the instructor's attention. In addition to easy model control, the instructor's environment must be dynamic enough to quickly adapt to unforeseen needs without resorting to low-level programming. The Action Center, Figure 1, provides a layered approach that meets both objectives: ease of instructor model control and straightforward extension of model operations (e.g., add new malfunctions, scenarios).

In addition, the layered organization of the model information processed by the Action Center provides a separation of duties that makes a great deal of sense for maintenance and extension of model operations. Three levels of duty result from the Action Center design:

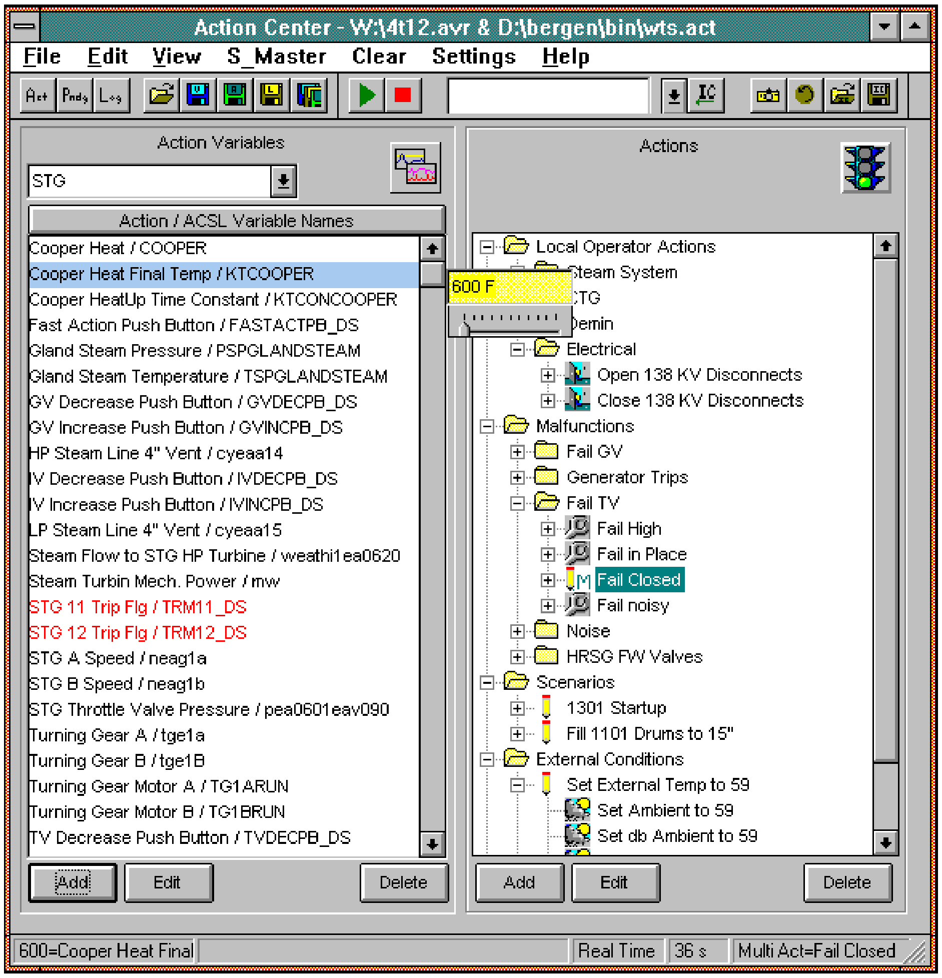
- Action Variable definitions (correlation with model variables)
- Action definitions (malfunctions, scenarios, etc.)
- Instruction in plant operation.

Although three different skills are needed, it is possible that one individual could carry out two or all three of these duties. The capability to perform one of these tasks is based solely on the individual's model, plant, and instructional knowledge; not on programming ability since setting up the Action Center is quite mechanical and easy to master.[1] The sections that follow contain a detailed description of Action Variables, Actions, and Model Control.

The layered approach discussed above is compliant with the object-oriented design allowed by C++. Framatome Technologies takes full advantage of C++ in its design of the Action Center. The object-oriented design benefits the user in several respects[2]. Most notably, no low-level programming or compilation is needed to develop actions (such as local operator actions [LOAs], malfunctions, and scenarios). However, the Action Center can be programmed in powerful custom applications via OLE (Object Linking and Embedding) Automation (see the OLE Automation section). The object-oriented and layered approach allows the user to hide unnecessary

[1] In fact, all individuals using the Action Center should understand the steps required to perform all three levels of duty.
[2] Additional benefits include easy maintenance and extensions to the Action Center code.

details. This topic is discussed more in the following sections.



Cooper Heat / COOPER
Cooper Heat Final Temp / KTCOOPER
Cooper HeatUp Time Constant / KTCONCOOPER
Fast Action Push Button / FASTACTPB_DS
Gland Steam Pressure / PSPGLANDSTEAM
Gland Steam Temperature / TSPGLANDSTEAM
GV Decrease Push Button / GVDECPB_DS
GV Increase Push Button / GVINCPB_DS
HP Steam Line 4" Vent / cyeaa14
IV Decrease Push Button / IVDECPB_DS
IV Increase Push Button / IVINCPB_DS
LP Steam Line 4" Vent / cyeaa15
Steam Flow to STG HP Turbine / weathi1ea0620
Steam Turbin Mech. Power / mw
STG 11 Trip Flg / TRM11_DS
STG 12 Trip Flg / TRM12_DS
STG A Speed / neag1a
STG B Speed / neag1b
STG Throttle Valve Pressure / pea0601eav090
Turning Gear A / tge1a
Turning Gear B / tge1B
Turning Gear Motor A / TG1ARUN
Turning Gear Motor B / TG1BRUN
TV Decrease Push Button / TVDECPB_DS

- Local Operator Actions
  - Steam System
  - TG
  - emin
  - Electrical
    - Open 138 KV Disconnects
    - Close 138 KV Disconnects
- Malfunctions
  - Fail GV
  - Generator Trips
  - Fail TV
    - Fail High
    - Fail in Place
    - Fail Closed
    - Fail noisy
  - Noise
  - HRSG FW Valves
- Scenarios
  - 1301 Startup
  - Fill 1101 Drums to 15"
- External Conditions
  - Set External Temp to 59
    - Set Ambient to 59
    - Set db Ambient to 59

600=Cooper Heat Final | Real Time | 36 s | Multi Act=Fail Closed

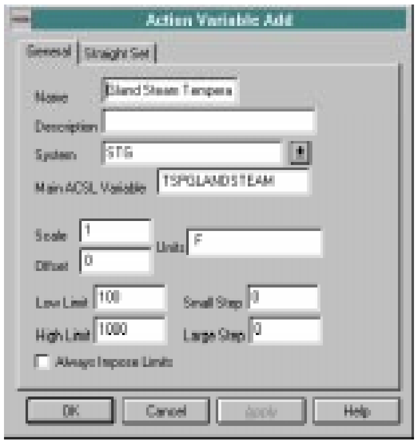
Figure 1. Action Center Main View

Action Variables

Each action variable provides monitoring and direct control of a simulation variable. An action variable represents the lowest level of model information in the Action Center; it is a direct tie to a variable in the simulator model. As a result, defining and modifying action variables should be carried out by someone with a good understanding of the simulation model itself. This person is referred to as a “model expert” throughout this paper.

For real-time simulation, larger plant models are contained in an optimized executable without the overhead of supplementary functionality. The simulator server maintains variable offsets in a memory region shared by both the simulator executable and the simulator clients. Clients (such as the Action Center and MMIs) access this information as needed through an inter-process communication (e.g. using TCP/IP and OLE Automation) with the server.

Action variables provide a wrapper around the variable defined in the simulator model. The wrapper adds attributes to produce instructor-friendly, descriptive variables without any impact on real-time simulation performance. Equally important is that these variables and their attributes can be changed interactively in the midst of simulation control (or without being connected to the simulation model).



Name: Gland Steam Tempera
Description:
System: STG
Main ACSL Variable: TSPGLANDSTEAM
Scale: 1
Offset: 0
Units: F
Low Limit: 100
High Limit: 1000
Small Step: 0
Large Step: 0
☐ Always Impose Limits

Figure 2. Menu for Defining Action Variable

Variable name syntax is restrictive in programming languages that are typically used to develop large plant models. An action variable eliminates this restriction by allowing multiple word names (spaces are allowed) without length restrictions.

By providing a scaling function, action variables make it easy to change the desired units viewed by the instructor at anytime. The instructor will interface with the new units with all indications of the variable value in the desired units. All simulator model variable values are automatically converted to the simulator model units before being set in the model.

Improperly setting a variable often results in undesirable consequences. Action variables contain certain attributes that significantly reduce the possibility of this occurring. Range validation (if defined for the action variable) occurs during an attempt to set an action variable. Additionally, an action variable can be flagged as non-setable, which grays out the display of its value and prevents any attempts to set its value.

Setting a simulation variable (directly for the simulator model) sometimes requires more than just a direct set to the variable. The main variable may require other peripheral variables or toggling. These set requirements dictate the different types of action variables that can be defined by the model expert at the time they are added. In effect, the steps and conditions required to set a variable are hidden behind the different types of action variables.

Grouping capability is provided for the action variables. Although not restricted to plant systems, most grouping will be based on these systems. The action variable grouping provides an aid throughout the user-interface when action variable names must be selected.

Typically, all action variables will be permanently defined for all training sessions. The instructor will benefit from the action variables without having to understand the sometimes intricate details of how they were defined in the simulator model. These benefits are highlighted in the section titled "Total Model Control in an Instructor-Friendly Environment."

Actions

Although the action variables discussed in the previous section provide monitoring and direct control of variables, actions provide the instructor with the functionality needed for controlling training scenarios. Actions are categorized as local operator actions (LOAs), malfunctions, and external conditions, for example. Access to all variables allows the instructor to introduce actions in any plant system/equipment supported by the simulator's scope of supply.

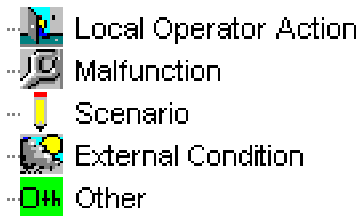
Typically, actions are generated by an "action programmer" who understands both instructor training needs and the action variables. Like the action variables, actions are permanently stored and retrieved for instruction.

General Action Attributes

All of the action types have the same general attributes that define the basic functionality needed by the instructor. Actions may be activated by some combination of:

- Model timed activation - The action is scheduled to be activated at a particular model time.
- Keyed activation - The action is activated when the instructor types an assigned function key.
- Remote control activation - A hand-held remote control allows the instructor to invoke actions when he is not near the instructor station, freeing the instructor from the keyboard. General model control (e.g. run and freeze) is also provided by the remote control.
- Delayed activation - The action is activated at a time delayed relative to the current time.
- Triggered activation - The action activation occurs when conditions specified by the instructor are satisfied. This type of action provides the instructor the use of operators such as AND, OR, and NOT and relational operators such as EQ, NE, GT and LT.

It is up to the "action programmer" to categorize the action according to the list below. All actions are listed with the icons/descriptions shown below so that the instructor immediately recognizes the category.


Local Operator Action
Malfunction
Scenario
External Condition
Other

The most powerful attribute of an action comes through the specification of its activation state[3,4]. By highlighting an action and hitting go, the action is initiated. Once initiated, an action is placed in a pending state and not activated until the condition specified by the activation set is met. The trigger syntax are relational operators such as EQ, NE, GT, LT that are used for comparisons and operators such as AND and OR that are used for combining conditions. The ability to control precedence is done parenthetically with no limit on the level of complexity. If a delayed activation state is chosen, the action is not activated until the specified time span since initiation elapses.

While the general attributes apply to all actions, each type contains individual characteristics as described in the sections below. There are four basic types of actions: Simple, Rampable, Multiple, and Scenarios. The first two of these are centered around the controlling of one action variable, while, the last two are groupings of actions.

Simple Actions

The details of a simple action specifies an action variable and a corresponding value to be set at the time of activation.

Rampable Actions

An action variable can be linearly changed at a specified rate or with some duration. Once activated, the ramp remains active until the target value or duration has been met.

Multiple Actions

Multiple actions provide containers for a group of simple, rampable, and even other multiple actions. It still contains the same general attributes that all actions have. For example, a triggered state can be assigned to a multiple action and it will be placed in a pending state upon initiation. When activated, a multiple action will initiate all of its child actions unto the pending list and they will be handled as if they were initiated directly.

Scenarios

Typically, an instructor will initiate a scenario that will completely define the training session. Like a multiple action, a scenario will contain actions to be initiated. However, scenarios contain details needed for controlling the entire training session[5].

Unlike other actions, scenarios are immediately activated upon initiation and remain active until the end condition is met or the specified duration expires. At the time of activation, an initial condition (IC) according to the scenarios specification is loaded and the model is started. All of its member actions are immediately initiated just as they are for a multiple action activation and the scenario begins. Essentially, a scenario is completely automated for the instructor with the click on one button.

A scenario is ended according to one of three conditions: 1) it is manually ended by the instructor, 2) the ending condition is satisfied, or 3) the specified duration expires. Regardless of the reason, at the end of a session, an IC file is automatically stored with a filename containing a date and time stamp, and the instructor's name.

Total Model Control in an Instructor-Friendly Environment

After initiating a scenario, an instructor is free to evaluate a student's actions without interruption. However, total model control is provided for those instances when it is needed. In addition, the Action Center can be viewed as more than an Instruction tool. It offers a large number of utilities for controlling the model.

Model Control

[3] Initiation refers to the act of placing the action on a pending list, while activation refers to the act of executing the action.
[4] Scenarios are an exception, instead of an activation state they use similar information to define an ending condition. See the scenario sub-section.

[5] Only one scenario can be active at a given instant.

Figure 1. illustrates an instructor interface for a typical model. A break-away toolbar provides easy access to starting/stopping the model and the loading/storage of ICs. An IC file viewer (not shown) displays three action variable values as the user scrolls through a list of the available IC files. The user can change the displayed action variable values at anytime and the last choices are persistent between sessions.

Action variables provide the ability to monitor and directly set variables. When an action variable is highlighted, its value is displayed to its right in an edit box. If the value can be set (based on the action variable's attributes - see "Action Variables") the edit box will have a yellow background. If the value cannot be set the edit box background is light gray. For action variables that can be set, the user can quickly tab to the edit box, change the value, and hit enter to perform the set. Action variables might also have a tracking control that provides a slider representing the current value relative to the range defined for that action variable as well as the ability to drag the slider to a new position (value). In addition, the model variable control allows for incremental step increases via the keyboard.

Views

During model control, the user will toggle between three separate views: Main Action View, Pending View, and Logging View.

Most of this paper's discussion centers around the Main Action View (Figure 1.); it provides a list of action variables and a Tree Control for maintaining and displaying the hierarchical nature of the actions. For model control, action variable values can be set and actions are initiated from this view.

Once initiated, actions are placed in a pending state and then later activated (as discussed in the "Actions" section). Monitoring the current state of this pending list occurs in the Pending View. Typically, this view is for monitoring the current state of a scenario. However, actions can be cleared from the Pending list.
The Logging View automatically provides a log entry for every instructor action; this log can be saved. Each log entry includes a date/clock time stamp and the model time.

Undoing Actions

For certain actions, the instructor must reverse (or undo) the action at a later indeterminate time. For example, the instructor has caused an oscillating control signal malfunction during a training exercise. With the trainee's response and the resulting plant dynamics not being pre-determined, the instructor might decide when to correct the noisy signal based on the response. The instructor is permitted to have other actions that may be treated in a similar manner.

By placing an undo action for each of these types of actions that have been activated in a special list, the instructor can quickly identify and undo earlier actions. Each simple action has an attribute flag that states if the action can be reversed. If the action is reversible, the action programmer also specifies the value being reset during the undo operation.

Scripting with OLE Automation

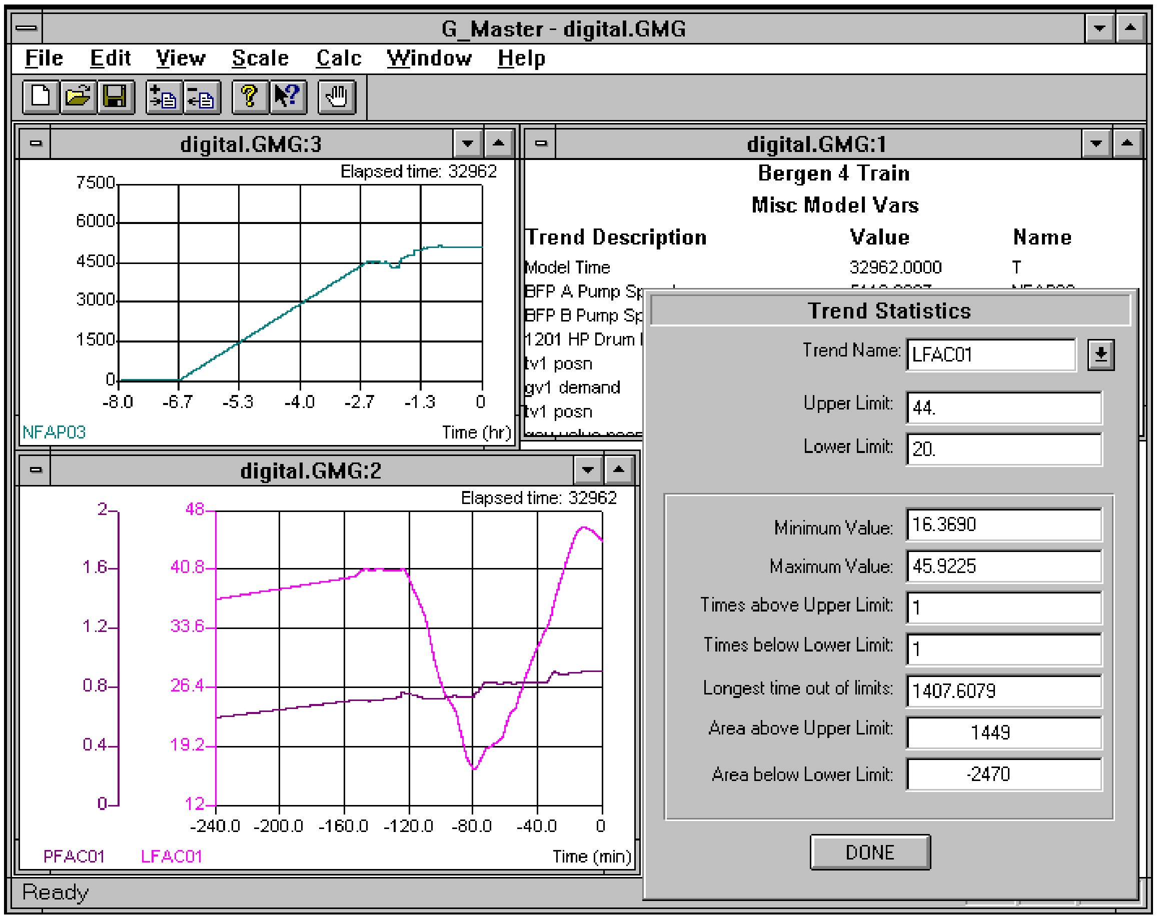
OLE Automation is a technology that lets programs expose their features to scripting tools and other applications. This technology being included in the Action Center provides the ability to write powerful scripts to perform tasks that would otherwise be unthinkable; complex scenarios, parameterization studies, what-if analyses, and tutoring programs to mention a few.

A model expert or action programmer can create client applications using tools such as Microsoft's Visual Basic or Excel spreadsheet. The programming required to control the model through OLE automation is quite simple. The actual scenario being programmed will determine how complicated the client application is.

Trending and Trainee Performance Review Capabilities

Variable trending and trainee performance review are two important complimentary functions to the Action Center and are included in the instructor station as separate, but integrated programs. The simulator model variable trending functions provide the instructor the ability to display simulator variables in a trend chart format as illustrated in Figure 3. The instructor station provides a very flexible, real-time trending capability for process model variables. The user has complete control over what variables are to be trended and how they should be displayed. There are no limits on the number of variables that can be displayed or the number of graphs that can activated at one time. Each variable on a graph is plotted in a different color, and the size and proportions of each graph can be sized interactively by the user with the mouse pointer. Scaling can be set by the user or dynamic auto-scaling can be calculated by the program. The dynamic scrolling of the trend graph is smooth and provides for ease of viewing. Trend information can be stored in a data format for post-transient analysis by other programs. Dynamic digital update of trended variables are another option.

Trainee performance review during or following a training session is accomplished by examining the Trend Statistics (sample shown in Figure 3.). The instructor can interactively establish the trainee performance criteria through upper/lower limit selections and receive immediate trend statistics evaluation of these selections for the trainee’s current training session.



Bergen 4 Train

Misc Model Vars

Trend Description	Value	Name
Model Time	32962.0000	T
BFP A Pump Sp		
BFP B Pump Sp		
1201 HP Drum l		
tv1 posn		
gv1 demand		
tv1 posn		

Trend Statistics

Trend Name:	LFAC01
Upper Limit:	44.
Lower Limit:	20.
Minimum Value:	16.3690
Maximum Value:	45.9225
Times above Upper Limit:	1
Times below Lower Limit:	1
Longest time out of limits:	1407.6079
Area above Upper Limit:	1449
Area below Lower Limit:	-2470

Figure 3. Trend Displays with Trainee Performance Review Results

Conclusion

Advances have been made to provide an instructor station environment that not only uses the latest and most predominant operating system (Microsoft Windows), but also provide the instructor significant enhancements to increase productivity when using the simulator as a training tool. These enhancements include the ability to use descriptive model variable naming conventions, provide complex conditional malfunctions and scenarios without having to modify the simulator model coding or model recompilation, and permit merging the simulator model with third-party software for model control and presentation of results through OLE technology. The object-oriented approach used for this development acts as a catalyst for future advances because of the ease in which the capabilities can be extended to new instructor station functions.