BUILDING A SIMULATOR CONTROL STATION USING THE TCL/TK LANGUAGE

D. R. LaBelle

April 1998

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States, nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

KAPL ATOMIC POWER LABORATORY

SCHENECTADY, NEW YORK 15701

Operated for the U. S. Department of Energy
by KAPL, Inc. a Lockheed Martin company
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
BUILDING A SIMULATOR CONTROL STATION USING THE TCL/TK LANGUAGE

Dennis R. LaBelle
22 Sandalwood Drive
Clifton Park, NY 12065
e-mail drlabell@albany.net

KEYWORDS
simulator control interface TCL/TK

ABSTRACT
This paper describes the construction of a simulator graphical user interface (GUI) using the cross-platform, public domain programming language TCL/TK. TCL/TK is a high level scripting language for building GUIs. It is freely available for UNIX, Windows and the Mac operating systems. This paper will demonstrate how the traditionally difficult, time consuming aspects of producing a simulator control station are easily overcome with TCL/TK.

Referred to as the Interactive Control Station (ICS), this user interface provides a graphical method for interactive control of real time applications produced with the Real Time Programming Environment (RTProE). The techniques and tools developed for connecting the ICS to real time simulator models are described in detail within the paper. The real time data collection and plotting tool included with the ICS is also discussed.

The ICS uses a client/server design and can provide its features across a distributed computer network. Standard TCP/IP sockets are used as the communications transport medium. Multiple clients on a network may be served by a single server connected to the simulation. The clients may exist on the same or separate computers. Multiple servers, connected to different simulations, can also exist on the same or separate computers of the network.

GOALS
The ICS project goals consisted of building a simulator control station while:

1. Maximizing multi-platform portability
2. Producing freely available source code
3. Generating efficient, compact, low maintenance source code
4. Maintaining costs low for both application development and run time

PERFORMANCE ISSUES
Like other scripting languages, TCL/TK is interpreted. This characteristic results in performance somewhat reduced from a compiled application. However, performance is much more than satisfactory on present day personal computer hardware. Indeed, it is significantly better than control stations delivered just 5 years ago on high end Workstations.

PORTABILITY
Producing a multi-platform compatible product is one of the major goals of the ICS project. The graphical user interface was constructed using the TCL/TK programming language. Interpreters for TCL/TK code are freely available for Windows, UNIX and the Macintosh operating system. Therefore, TCL/TK represents an excellent choice for a multi-platform product.

The ICS is written in TCL/TK. TCL 7.6 and TK 4.2 or later revisions must be installed on the system to run the ICS.

Since the chosen real time environment, RTPoE, is a Windows operating system based product, the ICS was developed under Windows NT. However, the generated TCL/TK code should also run, unchanged, on a UNIX system.

SOURCE CODE
Cost
The TCL/TK language was specifically designed for generating interactive, graphical user interfaces. It contains all the components necessary for producing a modern interactive user application. There are no additional libraries or products which must be purchased to make it useful.
The source code for the TCL/TK interpreter is in the public domain. It is currently being maintained and developed further by Sun Microsystems. There are no development or run time fees for any applications produced with TCL/TK.

**Code Generation**

Most common GUI components are available as objects in TCL/TK. Each object (sometimes called *widget*) can usually be created and configured with a single line of code. The common widget set includes:

1. Button
2. Canvas
3. Checkbutton
4. Entry
5. Frame
6. Label
7. Listbox
8. Menu
9. Menubutton
10. Message
11. Radiobutton
12. Scale
13. Scrollbar
14. Text
15. Toplevel window

A number of GUI building tools for TCL/TK are freely available. For this project, the XF program, a GUI builder for TCL/TK written by Sven Delmas, was used extensively. It proved particularly useful in laying out the visual design of the ICS. XF itself is entirely written in TCL/TK.

XF was used to construct all visual aspects of the ICS programs. With XF, the individual components were rapidly assembled and placed together using the mouse. XF is an integrated development environment for TCL/TK. In addition to configuring all visual components of an application, it may also be used to:

1. Define global variables
2. Create and edit subroutines
3. Assign actions to mouse and keyboard events
4. Divide the project into manageable modules

A significant advantage to the TCL/TK language is its interpreted nature. The compilation cycle no longer exists when programming with TCL/TK. Modified source code may be immediately tested. Source code for new procedures may even be generated on-the-fly by the program itself as it executes.

This rapid development environment also makes it possible to test several different programming approaches to determine which one is best. Throwing away old code can actually be encouraged since it takes such little time to replace it.

**Code Size**

As a testimonial to the efficiencies of the TCL/TK language, all current ICS features are provided in approximately 10,000 lines of source code. This contrasts dramatically with the 200,000 lines of code typical of other simulation control stations currently written in C.

**Availability**

All source code for the ICS is included in Appendix A of this document.

**ICS FUNCTION LIBRARY**

A common core of ICS related TCL/TK functions was written. These are shared between the various applications that make up the ICS. This code is accessed by all ICS programs through TCL/TK's support of dynamically loaded TCL/TK code libraries.

This source code includes functions that:

1. Connect to the main central server including verification of computer and server availability
2. Send requests to and receive responses from the icserver program
3. Scan the TCL/TK program at startup and build a list of parameters that must be retrieved from shared memory.
4. Cyclically retrieve the parameter list from the icserver
5. Repeatedly retrieve general simulator status information
6. Retrieve information from the Programmer's Data Base (PDB)
7. Remotely display user diagrams on other machines
8. Perform a proper shutdown of the TCP/IP socket prior to exiting
9. Execute commands received from the central server

**CURRENT FUNCTIONALITY**

**Overview**

The ICS provides the following general functionality:

1. A user interface to request information
2. A method of specifying the desired simulator state
3. Tableaus to display information obtained from the central server.
4. A means of requesting the display of user built diagrams

Specific Functionality

To date the following functionality has been programmed into the ICS.

**Application Status Monitoring of:**
- Real world time
- Percent of the CPU used by the application
- Simulation run time
- The currently loaded Initial Conditions file
- Run status (Freeze, Run, Term, etc.)

**Initial Conditions File Management** - The ICS manages a list of 99 Initial Conditions (IC) files. At any time, the current state of the user application may be saved. Any of the saved IC files may be recalled into memory using the Reset feature. Doing so, will reset the state of the application to that which existed at the time the IC was created. The application will also go to Freeze and reset its virtual time to 00:00:00.

**Application Status Control** - The simulation code can be loaded or unloaded with the ICS control bar. Simulation mode may be changed between Run and Freeze.

**Rate Control** - Simulation run speed can be adjusted between one tenth and 10 times real time.

**Diagrams** - The ICS can be used to display user Piping and Instrumentation diagrams on local or remote screens.

**Alarm Monitor** - The ICS can display currently active and recently cleared alarm conditions. Alarm conditions are the result of parameter values that are outside a range specified in the PDB.

**Automated Procedures** - The ICS can launch TCL/TK scripts that set simulation parameter values and control the behavior of the simulation.

**Backtrack/Replay** - The simulation state and significant events can be saved for later restoration and replay.

**Malfunctions** - Simulation malfunctions can be interactively configured and inserted. Malfunction values may be ramped over time, time delayed or triggered by a user definable event.

**Monitored Parameters** - A user sizable table for displaying or setting the values of up to 300 variables within the simulator is provided. The window may be stretched to see up to 30 parameters at once. A scroll bar is used to position the viewing port anywhere within the list of 300. Individual parameters are entered into the table manually or retrieved from a saved file. The ICS obtains parameter units and descriptions from the PDB and inserts them into the table.

**Plotting** - The icsgraph program is provided for the plotting of simulation parameters. The desired parameter names are specified interactively by the user at run time or loaded from a file.

**SUPPORTING PROGRAMS**

The Interactive Control Station (ICS) provides a graphical interface for interactive control of real time applications produced with RTProE. The author discusses RTProE in greater detail in the companion paper "Real Time Programming Environment for Windows."

The ICS consists of several supporting applications. A summary description of each follows.

**User Diagrams**

Any TCL/TK script may be incorporated into the ICS. A single procedure call from the ICS function library can connect a program to the central server and start a cyclic data transfer process. This capability can be used to generate the normal Piping and Instrumentation (P&I) diagrams most often necessary for a simulator control station. The ICS includes a few such diagrams to serve as examples of proper construction.

P&I diagrams may be easily and visually constructed with the XF program. They are generally created using TCL/TK's canvas component. This component serves as a background for geometric shapes, images and other TCL/TK widgets. XF can be used to quickly place and configure items on the diagram. Any of the items may be activated to perform actions or send requests to the central server.

The most common task of P&I diagrams is the dynamic display of simulation parameters. Under the ICS, this is done by placing a Label widget on the diagram and...
assigning it the name of a simulation variable. The ICS function library will detect the parameter name assignment and cyclically retrieve the parameter value. TCL/TK Labels automatically display the value of their assigned variables. There is no other work required. It is not necessary to know the data type or byte size of the variable. The value is returned from icserver as a text string properly formatted based on information entered into the Programmer's Data Base.

The ICS provides the ability to remotely display user diagrams on any network machine running a copy of the icserver program. This is most easily done by dragging the name of the diagram from ICS' Diagram Index window and dropping it onto the Virtual Control Station (VCS) window. The VCS automatically configures itself to display a drop target for each icserver on the network.

Ics server

The interactive control server (icserver) provides a set of centralized services for controlling the real time application locally or over a network. It uses RTProE's rdata program for communication with simulator models. The icserver is written in TCL/TK and communicates with clients using TCP/IP sockets.

As the central server for the ICS, icserver contains the bulk of the ICS source code and functionality. icserver maintains centralized information on:

1. Alarm conditions
2. Application run state
3. Automated procedures
4. Initial conditions
5. Notable events
6. Malfunction status
7. Switch positions

Plain ASCII text is used in all communications between icserver and its clients. This provides an easily understood interface during any system trouble-shooting. All messages sent to icserver can be displayed in a scrollable list box which makes up the majority of the icserver visual screen presence.

Normally, there is no direct interaction required with the icserver. It must simply be started before its clients.

Ics server is also used to display all user diagrams. It must be running on all machines that will display diagrams. When executed for this purpose, the icserver program will automatically detect which copy of icserver is controlling the main simulation and behave as a client to it.

Ics Client

Running the ics client program provides the user with an initial control bar, shown in Figure 2, from which the previously described ICS functionality may be accessed.

The ics client communicates with the icserver to process user requests for information and control the simulator. Communication is done through TCP/IP sockets. The ics client is written in TCL/TK.

Icsgraph

There are several free extension libraries available for TCL/TK. One such library, BLT, was used to provide a graph component for development of the icsgraph program. BLT's graph object has well over a hundred configuration options available.

In order to graph a parameter, an array variable is first assigned to a graph element. The plotting of data points is
performed by simply adding new data pairs to the array. The graph object will plot the data contained within the specified array.

The icsgraph program plots parameter values contained within the simulation code. This plotting is performed in real-time, while the application runs. The variables must be defined using RTProE’s pdm program. Any of the variables defined in pdm’s Programmer’s Data Base (PDB) can then be specified by their text label during run time.

Figure 3 shows icsgraph displaying 1 of 4 possible simultaneous graphs.

The following features are available with icsgraph:

1. Selection of up to 4 plotted parameters from the PDB
2. Automatic value axis scaling
3. Automatic adjustment of data collection rate based on simulator run speed.
4. Sample rate adjustment
5. Display range selection
6. File SAVE and LOAD of parameter sets
7. Display of the current, minimum or maximum parameter value.

Multiple copies of icsgraph may be run concurrently. The icsgraph program is written in TCL/TK and communicates with the icserver using TCP/IP sockets.

**Rtdata**

Ics server runs RTProE’s rtdata as a child process and communicates with it through standard input (stdin) and standard output (stdout). The rtdata program acts as a data communications path between the icserver and the simulation code’s shared memory. It is used to control the application run state and read or modify values in shared memory. Rtdata can directly access and interact with the variables in the simulation code.

The rtdata program uses the PDB to determine the location of variables within shared memory data structures used by the simulation. The user, however, refers to the data location by its normal symbolic name in the application.

The main work performed by rtdata for the icserver is the repetitive retrieval of current simulation values from shared memory. Rtdata’s list management features are employed for efficient retrieval of this information. For infrequent, non-list based, data retrieval rtdata’s GETVAL command is used. Shared memory parameter values are modified using rtdata’s SET command.

**INTER-PROCESS COMMUNICATION**

**Overview**

```
+----------------+      +-----------------+      +-----------------+
| ICSERVER       | TCP/IP| ICS client      |
|                | stdin/stdout|            |
| RTDATA         | P&I diagram|
| SIMULATOR CODE | Shared memory|
| ICSGRAPH       |            |
```

Figure 4
Figure 4 outlines the inter-process communication pathways used between the user and the simulator models. Three different communication techniques were used. Each one has specific advantages that influenced its selection during system design. Reasons for their use are described below.

Shared Memory

Typical of most real time simulation environments, RTProE based simulator code communicates with other programs through shared memory. Shared memory uses little CPU overhead and is extremely fast. Most importantly, it allows inter-process communication without using any Windows Application Programming Interface (API) function calls. This avoids Windows’ annoying tendency to steal the processor away from an application in order to service low priority events such as mouse movements. An application’s use of any other type of inter-process communication under Windows severely degrades its real time response.

RTProE provides a general utility, rdata, for accessing the simulator code’s shared memory.

Stdin/Stdout Pipes

The rdata program is designed to accept user commands from the normal standard input (stdin) stream and display any responses on standard output (stdout). This use of stdin and stdout is normal for most interactive, console oriented programs. The advantages for this application are:

1. Commands are easy to test from the keyboard
2. Command responses are easily viewed
3. Real time application can still be controlled from systems with text based interfaces
4. The communication protocol is in a form readily interpreted and debugged by humans.
5. Stdin/stdout interfaces are available on almost all computer operating systems.

Icservers runs rdata as a child process using stdin and stdout for communication between the two processes. This use of stdin/stdout allowed interactive verification of proper command syntax and response prior to implementation within icserver. Commands were first tested by running rdata from a console shell. Code to send the same command string to rdata was then added to icserver.

TCP/IP Sockets

Today, TCP/IP sockets are the de-facto standard for inter-process communication over most networks, including the Internet. Sockets are used to connect all ICS client applications (ics, icsgraph, P&I diagrams, etc.) to the central server program. Each client has its own independent socket connection to the server. Failure or stalling of a client does not affect the server’s response to the other clients.

The TCL/TK language has high level support for connecting applications via sockets. A TCL/TK program can establish itself as a TCP/IP socket based server with a single command. Likewise, other TCL/TK programs can connect to this server with a single line of TCL/TK code. Communications between the client and server is then as simple as writing to and reading from the socket stream.

CONCLUSIONS

The 4 main goals of the project were reasonably achieved. It is now possible to quickly write a compact, full featured, low cost simulator control station. This is due to the availability of a free, high level graphical development language along with recent hardware speed improvements.

The development of the ICS is an important step in the expansion of the real time simulation community. The financial costs associated with using the software are minimal. The ICS is based on the free, cross-platform programming language TCL/TK. All source code for the ICS has been made publicly available. Using the ICS, along with RTProE, any person with access to a personal computer, Windows 95 or NT can quickly produce a real time simulator of professional quality.

ABOUT THE AUTHOR

Dennis LaBelle is currently employed as a software engineer for Lockheed Martin. He holds a B.S. in Chemical Engineering from the University of Maine and received his M.S. in Computer Science from Rensselaer Polytechnic Institute. The author has 16 years programming experience in a wide variety of mainframe, Workstation and personal computer environments.