Tcl/Tk — An Integration Vehicle for the Microwave/Millimeter–Wave Pilot Sites (MMPS)

Kevin B. Kenny, Brion D. Sarachan, Robert N. Sum Jr., and Wayne H. Uejio
GE Corporate R&D
P. O. Box 8, Building KW, Room C273
Schenectady, NY 12301
`KennyKB@crd.ge.com`[1]

Abstract. Tcl/Tk has been chosen as an integration environment for workstation software for DARPA's Microwave and Millimeter-wave Pilot Sites program (MMPS). Under this program, GE has developed Tcl wrapper codes for commercial application frameworks and to legacy codes from the MMPS user community. Among the commercial frameworks supported are the STEP Data Access Interface (SDAI), the FrameMaker® document preparation system and the Xess® spreadsheet program[2]. These codes have allowed us to build user interfaces that are beginning to realize substantial productivity gains among the microwave tube designers that use the system.

1. Introduction

The DARPA Initiative in Concurrent Engineering (DICE) is a five-year program involving joint government-industry-academic collaboration to develop automated support for the concurrent engineering of complex products. The initiative's main emphasis is not to conduct basic research, but rather to apply available technology, with small custom increments, to actual problems at industrial pilot sites. One of these pilot projects is the Microwave and Millimeter-Wave Pilot Sites (MMPS), a multi-company effort to demonstrate concurrent-engineering technology in the microwave vacuum-tube industry. The stated goal of MMPS is to replace lengthy and expensive build cycles with computer analysis. In particular, it is to cut the time required for an analysis cycle by 90%, broaden the user base for the analysis codes by a factor of four, and reduce the design cycle time for the test part by 70%.

1. This work was supported in part by the DARPA Initiative in Concurrent Engineering (DICE) through DARPA Contracts MDA972-88-C-0047 and MDA972-92-C-0027, and by the Tri-Services Microwave and Millimeter-wave Advanced Computational Environment (MMACE) program under Naval Research Laboratory contract N00014-92-C-2044.

2. FrameMaker is a registered trademark of Frame Technologies Corporation. Xess is a registered trademark of Applied Information Systems Inc.

In order to achieve these goals, a high degree of software integration is required. Multiple commercial frameworks must be integrated with proprietary analysis codes to present a unified interface to the engineer. Moreover, rapid deployment of the software is essential, since the pilot program lasts only eighteen months. To this end, off-the-shelf software is used as much as possible, with a system of wrappers connecting the various components [LEWI91]. The wrappers belong to four basic classes: tool-code wrappers, which connect user codes into a user interface framework; tool-data wrappers, which present objects from a shared data model into a framework; tool-tool wrappers, which implement communications among different user interface components; and user interface wrappers, which present the engineer with an "electronic design notebook" in which to maintain and manipulate designs. Tcl/Tk has proven to be a valuable framework to aid in building all of these classes of wrappers.

Figure 1 shows a typical MMPS application's user interface, in this case, a design system for helix traveling wave tubes. The user has elected to run a parameter study, varying several properties of the operating regime and of the tube itself. The results are shown in the graph at the lower right. This figure not only shows some of the salient features of the interaction metaphor (for instance, the positioning of components using a mechanism like a word processing tab ruler), but also gives an idea of the number of components involved: the study requires integrating the user interface, a surface plotting program, a database manager, and several legacy codes for performing various sorts of analysis and simulation of the device.

The MMPS team has experimented with a number of commercial and legacy codes as components of the system. Each of the components implements a different interaction metaphor, all of which are required to support the engineer. This presentation discusses our experience with several of these, including the STEP Data Access Interface (SDAI) [ISO89], the Xess spreadsheet program [AIS92], and the FrameMaker [FRAM92] document preparation system.

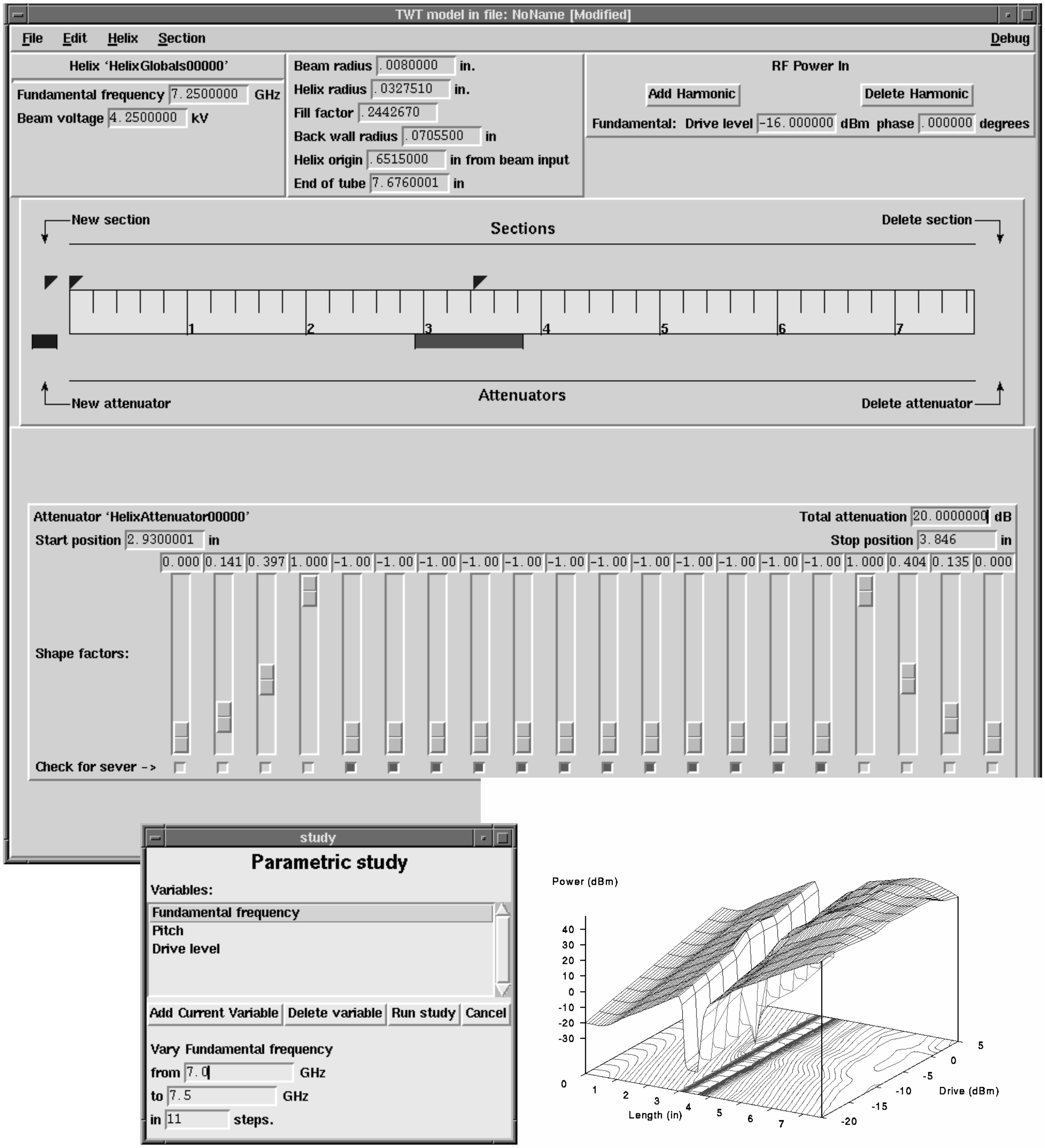

Figure 1. User Interface to Design Codes

2. Internals of the wrappers.

Tcl provides a common scripting and control facility to which all the chosen application frameworks can connect, so each of the wrappers has a component that is a Tcl script. In most cases, the Tcl code is object based, in that the commands that it executes identify first an object and next an operation to execute upon that object. These object commands are similar to Tk's widget commands, where one writes, for example, '`widgetName configure`'. The commands that are implemented in C and linked with the Tcl interpreter are, in fact, implemented in the same way as the widget commands. An object creation command invokes `Tcl_CreateCommand` giving it the name of a new object and a `clientData` argument that describes the object. The Tcl objects represent objects in the framework; for example, an object might represent a spreadsheet, a document, a menu, or a dialog box.

SDAI. Central to the entire system is the concept of a shared information model describing the product under development. The Express language was chosen to express the conceptual schema for the information, and the STEP Data Access Interface (SDAI) [ISO89] is used to access the data objects. The same SDAI interface can be used with a variety of data bases, ranging from a simple object system implemented using Tcl variables with stylized names like

    className:instanceName.fieldName

to an elaborate object system called ROSE provided by STEP Tools Inc.

SDAI is a simple enough interface that no real benefit was seen to breaking its objects out separately in the Tcl interpreter. Instead, the wrapper provides a single `sdai` command that performs all the data access, using subcommands such as "`sdai make_instance`" and "`sdai set_field`". The data types in the interface are simple enough that a very straightforward set of C functions encapsulate all of them into Tcl. The system also provides a Tk-based browser that allows the user to peruse various components of the schema and the objects in the repository.

Xess. The team's experience is that spreadsheet programs provide a very natural interaction metaphor for the engineering user to experiment with alternative designs, perform trade studies, and study "what if" scenarios. The data that appear in the product model and the data that an analysis code uses can be mapped into spreadsheet cells, and the analysis codes and data accessors can be represented as spreadsheet functions or menu-driven operations. The MMPS trial system has used the Xess [AIS92] spreadsheet in this capacity. Tcl provides an important capability, since Xess has an elaborate C-language API but no scripting capability; providing Tcl bindings for the Xess API provided the missing functionality. The scripting capability has proven to be so valuable that Applied Information Systems, the manufacturer of Xess, has announced that a forthcoming release of the product will support scripting in Tcl.

The Xess API is elaborate enough that Tcl object commands are used to group API functions and manage the API data (as `clientData` structures in the C functions). Object commands are provided that represent spreadsheets, menus, dialog boxes, graphics, and so on. The operations on these objects correspond to the functions in the API. Argument conversion functions are also provided in the C code so that items such as cell addresses, cell ranges, and printer fonts can be represented in their natural forms. Several of the API functions, that configure the user interface, accept elaborate data structures describing the configuration. Option tables using `Tk_ParseArgv` were designed so that the object commands could use the Tk `configure` mechanism to represent these.

Some of the object commands have many subcommands (the one representing a spreadsheet has almost eighty different functions). To streamline implementing these commands, a Tcl script was written to preprocess the C source code of the object commands, and generate a simple table-driven recognizer for the command names. This technique resulted in simpler and faster code than a series of `strcmp` operations.

Many of the operations in the Xess API involve callbacks, where Xess causes some operation to be performed in the integrated application (that is, the Tcl script). A mechanism analogous to the Tk `bind` operation is provided to process these callbacks. The callbacks are requested by means of `PropertyNotify` events in the X event stream. In order to process these, a generic event handler is established when the Xess interface is initialized to dispatch them through the Tk event manager.

The right-hand windows of Figure 2 show a sample user interface integrated with the Xess spreadsheet, again showing a front end to a helix-tube analysis code. The spreadsheet presents a simple form that the engineer can fill in with the most important design parameters. Menus allow connections to the various analysis codes ('LINKS'), analysis of the design using the selected small signal analysis code ("SMALL_SIG"), and saving the data in an electronic design notebook ("EDN").

FrameMaker. The FrameMaker [FRAM91] document preparation system plays an important role in the prototype system. Electronic design notebooks [UEJI92], in which designs are documented, are implemented in it. It provides interactive help for several components of the user interface, organized as hypertext, and accessible directly from the interface (for instance, clicking the right

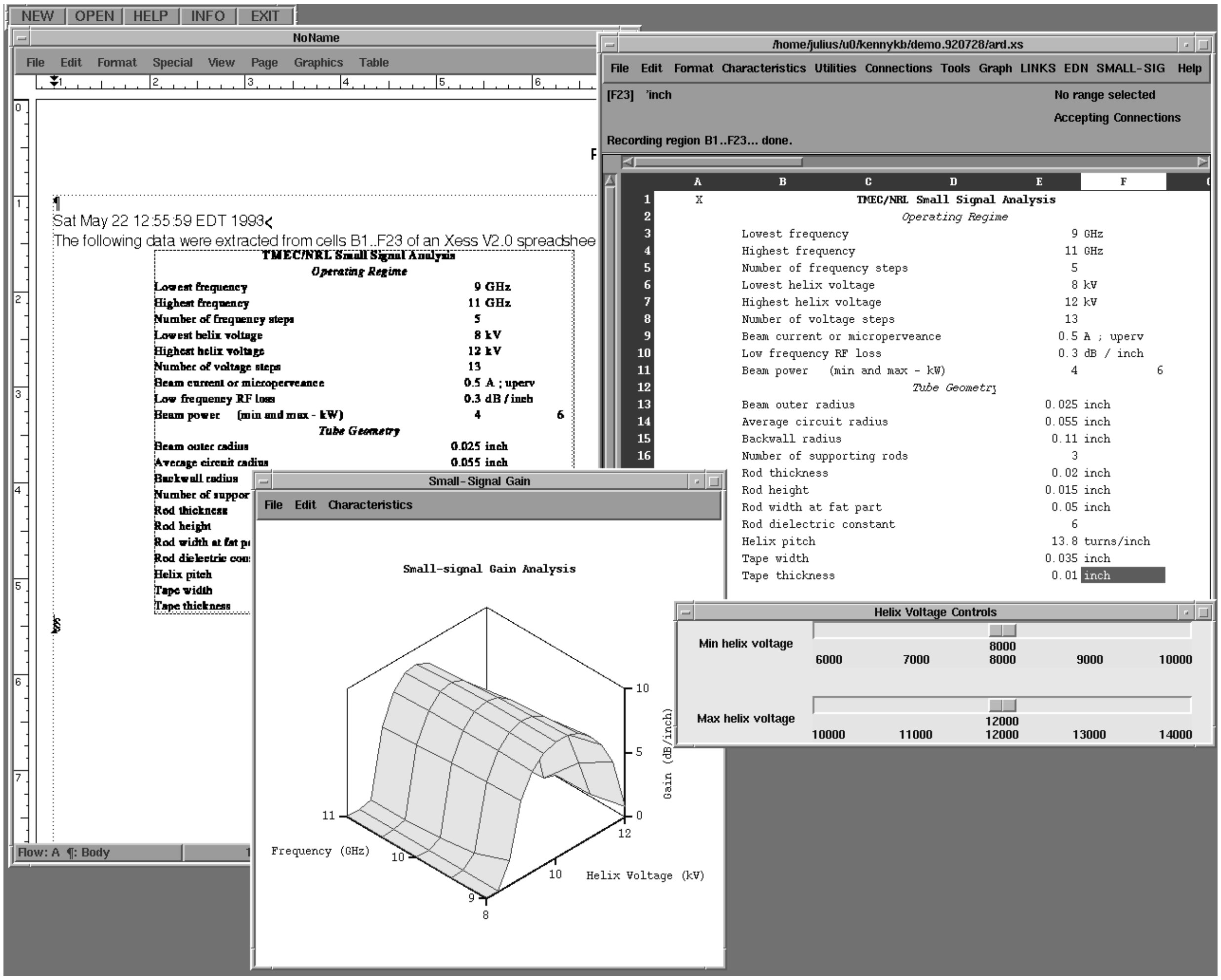

Figure 2. User interface based on commercial frameworks.

mouse button on a widget positions the document to the description of that widget). Finally, it provides an unusual means of data definition, in which the data description is a document providing a 'fill-in-the-blanks' form that the user edits, and a button embedded in the document launches a conversion from that form to any of several schema definition languages. The leftmost window in Figure 2 shows an EDN in operation, having just recorded data from the spreadsheet on the right-hand side.

Just as with Xess, the FrameMaker product provided a reasonably complete C API but no scripting capability, and once again Tcl came to the rescue. The API was encapsulated in a Tcl command called '`fm`', with object commands corresponding to the RPC channels which the interpreter used to communicate with FrameMaker. The FrameMaker data were simple enough that simple conversions sufficed and no special parsing was necessary; the only exception was that a huge (500-case) enumerated type was represented in the Tcl script by symbolic names rather than integers.

Event management posed a greater problem than with Xess. Just as with Xess, callbacks were required into the Tcl script from the FrameMaker program; these callbacks came via RPC operations. The RPC system is normally driven from its own event loop, but allows the user to substitute another event manager. One was written that allowed establishing a file handler for every socket in the RPC system with the Tk event manager.

Legacy codes. Most of the engineering analysis codes were 'dusty decks' written in Fortran. A few simple readers and writers were implemented in Tcl to process Fortran `NAMELIST` files, and these allowed the (mostly batch-oriented) Fortran codes to appear as Tcl functions. A few particularly intransigent codes are handled using the Expect [LIBE91] system from NIST.

Callbacks are less of an issue for most of these codes, but in a few cases the team wished to present partial results of a long-running code in the user interface. This is accomplished by having a separate process contain a Tk script that reads successive lines of output from a program and sends them (using the Tk `send` command) to the parent application.

3. Future directions for Tcl and Tk — a wish list

The experience gained with the above wrappers suggests some issues for future enhancements to Tcl/Tk.

Event management — "Whose process is this, anyway?" Virtually all of the wrappers described required peculiar code to interface to the Tk event loop. Two simple enhancements would have greatly simplified the implementation. The first is a mechanism, since implemented in `tclX` as the `addinput` command, to execute a Tcl callback when a particular file becomes ready. The second is a special case in the event loop for file descriptors that represent RPC channels; many APIs use RPC, and the code to interface them to the Tk event manager is cumbersome and slow.

Binding. Callbacks from external applications often require argument substitution. There is, at present, no standard way to do this; even the bindings for X events are becoming awkward because of class bindings. It would be desirable for any improved binding system to provide support for arbitrary argument transmission, and to be accessible to user event procedures.

Object management and security. The wrappers described here all represent various real objects being manipulated by the applications. As object brokers such as CORBA [OMG92] and DDE [MICR91] become prevalent, it would be desirable to have the send and bind mechanisms use them. Not only would this improve data interchange, but it would also enhance security by allowing the object brokerage system handle the authentication and privacy issues.

Dynamic linking. The wrappers often require extensions to the core Tcl language. Because of various limitations of the operating systems and the Tcl interpreter, it is not possible to connect to these at run time, requiring fairly elaborate tricks to fabricate interpreters containing all the required components. Even if dynamic linking should prove impossible, an easier means of integrating user-written extensions would be appreciated.

References

[AIS92] *Xess Users' Guide*, version 2.1, Chapel Hill, NC: Applied Information Systems, 1992.

[COST93] Coston, Arthur, personal communication.

[FRAM91] *Integrating Applications with FrameMaker.* San Jose, CA: Frame Technologies Corporation, publication 41-01245-00, June 1991.

[ISO89] *External Representation of Product Exchange Data.* ISO/TC 184/SC N38, 1989, available from National Institute of Standards and Technology, Gaithersburg, MD.

[LEWI91] Lewis, J. W., *et al.,* "Wrappers: integration services and utilities for the DICE architecture." *Proc. Natl. Conf. on CALS/CE,* Arlington, VA, June, 1991.

[LIBE91] Libes, D. "Expect: scripts for controlling interactive processes." *Computing Systems 4*:2 (1991).

[MICR91] *Microsoft Windows Software Development Kit: Guide to Programming.* Redmond, WA: Microsoft Corporation, 1991.

[OMG92] *The Common Object Request Broker: Architecture and Specification*. Document 91.12.1, Object Management Group, 1992.

[OUST93] Ousterhout, J. K. *An Introduction to Tcl and Tk.* Reading, MA: Addison-Wesley, to appear 1993.

[UEJI92] Uejio, W. H., *et al.,* Capturing the corporate memory of a product." *Enabling Technologies: Proc National Conf. on Concurrent Engineering.* Morgantown, WV, April, 1992.

Appendix: Availability of the codes.

The various codes mentioned in this paper are available for other investigators to use and modify. The Xess interface is planned for release by AIS later this year. The interfaces to FrameMaker and to SDAI are available by writing to the author at the postal or E-mail addresses given.

An interface to TCP streams that allows for Tk-like event management and Tk-like 'send' operations in non-Tk-based applications is available from the author or by anonymous FTP at `harbor.cs.purdue.edu` in the file

`pub/tcl/extensions/tclTCP.1.0.tar.Z`

This interface is used to interconnect the various Tcl interpreters in the systems. This practice simplifies fabricating the interpreters, since each interpreter requires only a few extensions rather than the complete set.