On the Suitability of Tcl/Tk for *SYS*

Wilfred J. Hansen

February 2003

COTS-Based Systems Initiative

Since its initial release, this report has been revised to correct a few inaccuracies. This revision was released March 5, 2003.

Technical Note
CMU/SEI-2003-TN-001

Contents

List of Figures

Acknowledgements

I am grateful for the tremendous cooperation offered by the developers and users of what I have called *SYS*. Only the Software Engineering Institute's need to protect sources prevents me from naming them here. Early reader Bob Binder contributed server crucial corrections.

Abstract

The Software Engineering Institute (SEI[SM]) was called on recently to examine a system, hereafter called *SYS*, written entirely in the Tool Control Language/Toolkit (Tcl/Tk) language. In response to some negative comments in the SEI's report, the developers presented a list of systems purported to demonstrate the viability of Tcl/Tk as a development tool. A review of the 67 listed systems found that Tcl/Tk is indeed practical for developing large systems.

Small systems written in the language often follow a paradigm of "classic Tcl/Tk windows." *SYS* embraced this approach to the extent of involving hundreds of windows. The review showed that *no other* large system written in Tcl/Tk has anywhere near as many such windows. User interviews suggested that the number of different windows was indeed a problem. *SYS* should consider an alternative design, perhaps a Web-based approach. Some design criteria are described at the end of the report.

[SM] SEI is a service mark of Carnegie Mellon University.

1 On the Suitability of Tcl/Tk for *SYS*

Recently, I was part of a team reviewing a Department of Defense (DoD) program that we can call "*SYS.*" This system is written almost entirely in Tcl/Tk (Tool Control Language/Toolkit, pronounced "tickle-tee-kay"). While Tcl/Tk is an unconventional choice for DoD systems, it is not necessarily a bad choice. Nonetheless, as part of our review, we did recommend that another approach be taken for the developmental work planned to supplant additional legacy systems. The development team was not completely sympathetic to our suggestion; they referred us to several Web sites proffering evidence that Tcl/Tk is a widely used platform for commercial systems. This note reviews those Web sites and goes on to consider a number of other factors that should influence the choice of Tcl/Tk as a tool for further development of *SYS*.

At the outset, it is important to make two points:

- This note does NOT suggest that Tcl/Tk is a "bad" environment. On the contrary, the Web review shows the utility of Tcl/Tk for numerous purposes.
- This note does not suggest that Tcl/Tk was a "bad" choice for the current *SYS* functionality. *SYS* works. It is a fielded system that daily satisfies the needs of hundreds of users.

The question that this note does address is whether to continue further development of *SYS* with its current choices for user interface architecture and system architecture. These architectural questions are far more critical to success than the programming language/environment.

2 The Design of Tcl/Tk

Tcl was designed as a control language: a language in which a programmer could quickly write scripts to invoke diverse operations—often operations written in another language. In common with other interpretive languages, Tcl was designed for rapid code-test cycles. The Frequently-Asked-Questions (FAQ) posting for Tcl/Tk [Virden 2002] describes the language this way:

> Tcl is actually two things: a language and a library. First, Tcl is a simple textual language, intended primarily for issuing commands to interactive programs such as text editors, debuggers, illustrators, and shells. It has a simple syntax and is also programmable, so Tcl users can write command procedures to provide more powerful commands than those in the built-in set.

Second, Tcl is a library package that can be embedded in application programs. The Tcl library consists of a parser for the Tcl language, routines to implement the Tcl built-in commands, and procedures that allow each application to extend Tcl with additional commands specific to that application. The application program generates Tcl commands and passes them to the Tcl parser for execution. Commands may be generated by reading characters from an input source, or by associating command strings with elements of the application's user interface, such as menu entries, buttons, or keystrokes.

Tcl/Tk was designed with scripting as its specific area of functionality. As the FAQ goes on to say,

Note that Tcl was designed with the philosophy that one should actually use two or more languages when designing large software systems. One for manipulating complex internal data structures, or where performance is key, and another, such as Tcl, for writing smallish scripts that tie together the other pieces, providing hooks for the user to extend.

Early in its career, Tcl was extended with Tk, a *T*ool*k*it of widgets suitable for making graphical user interfaces (GUIs). Examples of applications using these widgets can be found at incrtcl.sourceforge.net/itcl/mmc/full/electric.gif. One example follows:

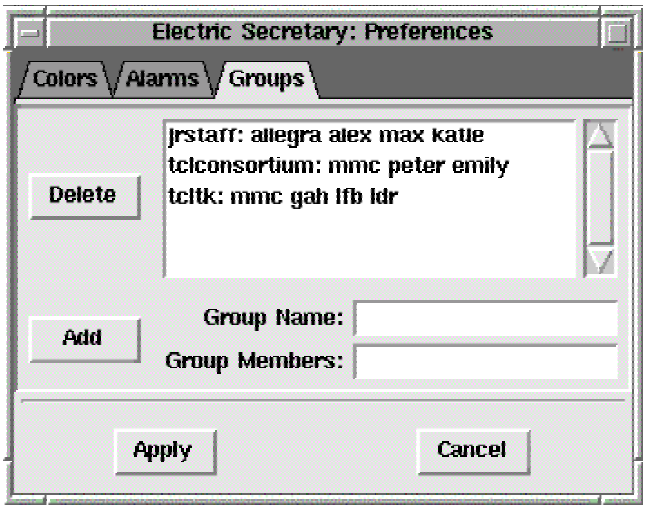

Figure 1: Electronic Secretary: Preferences

Here each box is a separate Tk widget. There are labels (“Group name”), text entry fields (to the right of the labels), buttons (“Add”), a panel (contains two bottom buttons), a list (“jrstaff…”), a scrollbar (to the right of the list), and file tabs (“Colors”). When a button is clicked, the usual implementation is that a Tcl procedure is called. It usually changes the window or opens a new one.

Tk was developed just as X Windows was developing the sets of screen widgets that Tk employs to draw on the screen. Since it was one of the few easy ways to build prototypes with X widgets, Tcl/Tk became popular early on. The existence of Tk is one factor that distinguishes Tcl from other scripting languages such as Python and Perl. The Tk widget set is quite similar in capabilities to the sets supported in Motif, HTML, Visual Basic, Java, and most GUI builder tools.

3 Classic Tcl/Tk Windows

Based on Tcl convenience and the Tk widgets there arose a class of programs implemented as "classic Tcl/Tk windows." In the paradigm followed by these programs, the program opens a window, populates it with widgets, and interacts with the user by responding to actions on those widgets. Very often the application has a number of such classic windows; clicks on some buttons may change the window or create a new one.

In this classic Tcl/Tk window paradigm, when a new set of data is to be displayed, a new classic window pops up. When a decision is required from the user, a separate "dialog box" window is popped up wherein the choice can be made, usually by clicking a widget. Since classic Tcl/Tk windows generally do not scroll the entire window, they must survive in constrained screen space. In consequence, there is a tradition of cryptic button labels. Image tools are few, so icons are seldom developed. Altogether, the window image for a classic Tcl/Tk window is functional, but not always inviting.

In programs with classic Tcl/Tk windows, the usual software architecture is one Tcl source file per window. There is seldom any further program structure to subdivide functionality or relate windows to clusters of functionality. The user interface architecture is a branching tree of windows. For small trees, this is great; with experience a user can easily navigate through the tree to get what is needed. Navigation problems arise, as noted below, when the number of windows grows large. This is an example of a technological boon morphing into a liability. Similar user interfaces can be implemented almost as easily with other languages, but their programming becomes more onerous as the size grows. Tcl/Tk continues to offer programming simplicity even after the user interface has begun to degrade.

4 SYS Overview

The system architecture of *SYS* connects user workstations across a local-area network to application servers. These in turn are connected via a wide-area network to remote database servers:

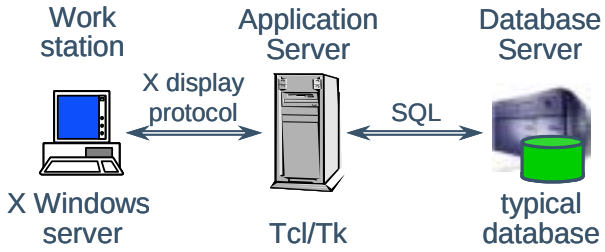

Figure 2: SYS Workstation/Server Connections

The application server is a Unix box and runs the Tcl/Tk application. The application sends commands in the X Windows protocol to the user's workstation where an X Windows server program converts them to screen images and then relays user actions back to the application. On the other side, the application server converses in SQL with a typical database.

4.1 User Interface Architecture

The *SYS* user interface architecture is that of a collection of classic Tcl/Tk windows. The user interface is a large set of windows—over two hundred. Each window is a set of widgets, in a few shades of blue and gray, with rare images and only occasional flashes of color. Users click buttons to progress from window to window through the hierarchy. While this sounds simple, there are several problems:

- Hierarchies this large impede use more than do the small trees typically found with classic Tcl/Tk windows.
- Users become lost, without sufficient cues as to how to return to a window that will lead them forward on a particular errand.
- As the system grows, it becomes ever harder to remember which subtree contains a needed functionality and unused portions of the tree fade from memory.
- A proliferation of windows on the screen requires tricky manual management of physical screen space and mental recall of the purpose of each window.

In interviews we conducted as part of our review, users expressed various reactions to the *SYS* user interface. Many comments were positive. Despite the fact that the screen image of reports is just a rectangular array of fixed-width text, it does provide users the information they need with a minimum of extraneous matter. The screen size limitation is circumvented

for reports by scrolling within the text panel holding the report (as opposed to scrolling the entire image as would happen with a Web browser).

Negative comments, however, suggested that users get lost navigating through the plenitude of options the system provides. They said things like “I don’t know what report to run to get the data I need,” and, “Whenever I venture out of the parts of *SYS* I usually use, I have to review the manual to be able to do the work.” In reviews of many projects, this usage difficulty is reported as a need for “more and better training.” Training will help, but it cannot forever paper over the design deficiencies. Nor will it help when dozens of new users are pressed into service in a time of emergency, the one time when *SYS* most needs to be functioning reliably.

4.2 Software Architecture

The software architecture of *SYS* is also that of a collection of classic Tcl/Tk windows. For most windows a single Tcl module draws the window and responds to user actions on that window. The source code intertwines database access, business rules, and GUI. The result is a “brittle” system, one that is difficult to modify without unintended side effects. Changing the user interface may cause unexpected changes in business rules, and vice versa. Revisions to the database schema may require revising dozens of modules that intersect with the changes.

The code is not entirely Tcl/Tk. Some of the application is written in SQL and stored on the database server. These procedures are executed based on various sorts of data accesses and revisions. In our review, we did not cover sufficient detail to determine whether a clear architectural distinction is made between Tcl code and SQL code. It is this author’s impression, however, that decisions to use SQL were based on ad hoc criteria. If so, the separation offers no architectural clues that will aid in program revision.

4.3 With All These Problems, How Can *SYS* Be Working?

For both the user interface and system architectures, *SYS* has serious problems. Why then is the system in successful everyday use? There are a number of reasons:

- Users have undergone extensive training. This has minimized the navigation problems.
- Each user’s specific job encounters only a small subset of the full *SYS* functionality.
- The system is new and has not undergone business rule changes. Structural deficiencies are not yet apparent.

In other words, *SYS* is working *at its present level*. The problems noted above will become more burdensome as the system grows to encompass new functionality and its code is revised to deal with changes to business rules, database design, or user interface.

4.4 Legacy Systems

SYS is not now complete. The system it replaced interfaced with a dozen other systems, all of which are antiquated and difficult to maintain. Plans call for them all to be incorporated into the *SYS* framework. In some cases this will mean that a button click in *SYS* will launch another application. In most cases, however, the existing functionality is to be recast in a new user interface, possibly incorporating revised business processes. Extending the current *SYS* architecture will present problems in managing the development of extensions, maintaining the extended system, and in using the system without considerable training and handholding.

The existence of these legacy systems is the imperative driving the need for flexibility and extensibility in the design of *SYS.* The problems explored above will make it difficult to extend the existing *SYS* to encompass these legacy systems.

4.5 *JSYS*—Progenitor of *SYS*

The *SYS* database augments the data in *JSYS,* a database maintained by a parent organization. Before *SYS* was released, many of its current users interacted directly with *JSYS,* so that system was an excellent model to follow in the design of *SYS.* Copying the system architecture—*JSYS* is itself a collection of classic Tcl/Tk windows—meant that about a third of the *JSYS* code could be re-used in *SYS.*[1] Copying the user interface meant increased user acceptance and decreased training effort. Indeed, our interviews of users elicited favorable comments about the similarity of the two user interfaces.

Over time, however, the value of *JSYS* as a model has faded. The operational environment is such that users should work exclusively through *SYS* because it enforces the right set of business rules for its users. When the users no longer encounter *JSYS,* there is no longer any merit in following it as a model. Presently, plans for revision of *JSYS* may well be in progress, although nothing has been announced. Any change to *JSYS* could move it away from *SYS.* In addition, there is more pressure on SYS to supplant legacy systems. *JSYS* may not need to reconsider its architecture to evolve, but *SYS* must.

[1] Among the advantages is increased reliability. As the *SYS* developers work with the *JSYS* code they sometimes discover and fix deficiencies that might otherwise have caused problems. The *JSYS* developers are occasionally able to return the favor.

5 Tcl/Tk in Commercial Systems

In support of the continued use of Tcl/Tk on the *SYS* project, a consultant and the development manager sent references to Web sites listing Tcl/Tk projects in commercial use. I have reviewed the listed systems, with the results shown in the Appendix and the Type codes listed in Table 1. After eliminating duplicates, some 67 systems remain. Of these, sixteen (marked *defunct*) no longer have a detectable presence on the Web[2] and nine (marked *dist*) are distributions of Tcl/Tk rather than applications.

One system, AOLserver, appeared on several of the lists. It merits a closer look. A presentation by Jim Davidson describes the adaptations they made to Tcl, the architectures they used, and a few examples [Davidson 2000]. One example, the Digital City Movie Guide, might appear to a user like this:

Figure 3: AOL Digital City Movie Guide

As shown for a user in Austin, Movie Guide offers a synopsis of a movie, ratings by members, and the times the movie is showing at Austin theaters. Among other features, another page shows the "Top Ten" movies that have been clicked on "recently" by other AOL users.

[2] That so many listed systems are defunct need not indict Tcl/Tk. The lists include systems five and more years old, a time frame in which most systems become defunct. Rather, the invalid references suggest that the lists are passé because those who would be swayed have been already and those who would not are already committed to other tools.

Type	#	Description
web envt	6	These systems are environments that create Web sites. Each has been used as the basis for multiple Web sites by multiple different customers. In general, these systems generate HTML to create pages; that is, they do not use Tk, except possibly for a central control console.
web app	8	These are Web sites where page generation is coded in Tcl. The code generates HTML and uses Tk, if at all, only for a control console. Unlike the *Web envt* category, these systems generally are used for only one Web site.
small	8	These applications appear to the user as a small number (say under two dozen) of classic Tcl/Tk windows, although the number of lines of code is not necessarily small. Several of these systems are targeted at developers, who will appreciate the opportunity to revise them. There were NO corresponding systems with large numbers of classic windows.
other	3	Two of these entries are statements to the effect that one or more people use Tcl/Tk frequently; they use it as the hacking language of choice for whatever tasks arise. This is commercial use of Tcl, but does not result in deployed applications. The third system, NBC's digital broadcast control system, has had its operational code ported to C++; it is no longer strictly Tcl.
script	3	In these systems Tcl is used for its scripting advantages. Users are expected to add Tcl code to tailor the system for their purposes.
ctler	14	Here Tcl code controls hardware. Many of these systems are used in hardware development where engineers do Tcl scripting in order to create tests by piecing together operations from a library.
dist	9	These are distributors of Tcl/Tk or libraries.
defunct	16	Systems no longer having a discernable Web presence.

Figure 4: Type Codes and Their Meanings
("#" is the number of systems of that type.)

There are *no* similarities between this user interface and that of classic Tcl/Tk windows. Rather than a collection of widgets there is a document image. Rather than static, space-constrained window contents, the entire document scrolls. Rather than buttons, there are links. Rather than just text, information is augmented with images and icons, such as the stars for the movie rating. Rather than shades of blue, the full rainbow is exploited. Rather than window proliferation, the usual result of an action is to replace the window contents. Rather than a succession of user decisions, the information flow has been worked out so that

common operations are done in simple fashion. Rather than extensive training, there is NO training. Not all is perfect, of course; the system is expensive, so advertising is rampant.

The “Top Ten” list offered challenges in getting acceptable performance. Two architectures were implemented, deployed, and discarded before hitting on a successful approach. The final Movie Guide architecture looks like this:

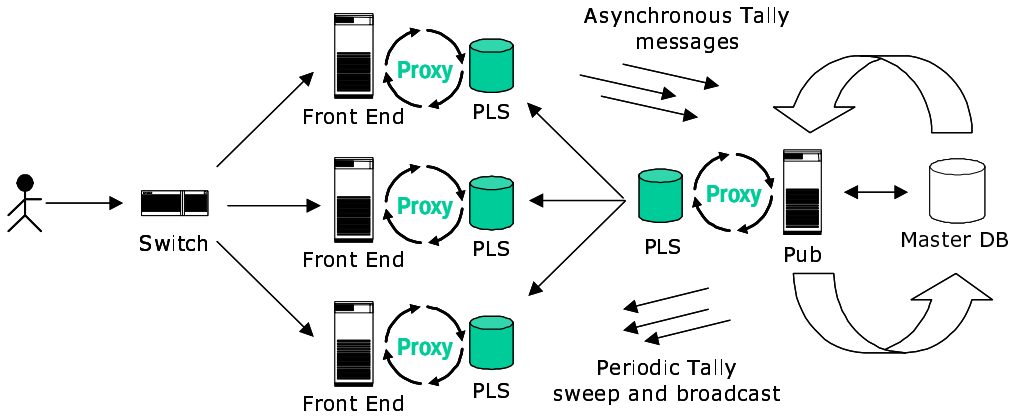

Figure 5: Digital City Movie Guide Architecture

Each user’s page request is distributed by a switch to one of the myriad front-end servers. These communicate via a “Proxy” mechanism that serves as a lightweight process running a script; in this case the script accesses a “PLS” database of movie data. This database is replicated from a master database maintained by a publisher server “Pub” reading from a database. As each front end gets a click on a movie, it accumulates tally counts and sends them to pub. In turn, pub periodically collates this data and sends new top ten lists to the front ends.

From this example, we see that AOLserver has nothing in common with systems composed of classic Tcl/Tk windows. In fact, the Tk GUI toolkit is used only marginally to provide some operator functionality. Nor is the software architecture that of a classic Tcl/Tk window. Many layers are involved in providing service. Moreover, AOL is not reluctant to rewrite Tcl procedures into other languages to improve performance. Indeed, Davidson comments that they have an *nstats* command that “returns the usage counts of individual [Tcl] commands [and is] a great way to look for fat procedures which could be moved to C.”

6 Issues in Choosing Architectures for *SYS*

The preceding section reviewed a number of commercial systems put forth as examples illustrating the applicability and viability of Tcl/Tk. Strikingly, *not one* of these systems adopts either the GUI or software architecture of *SYS*. The *Web envt* and *Web app* systems generate Web pages instead of classic Tcl/Tk windows. The *small* systems do generate classic windows, but not nearly as many as *SYS*. The remaining systems use Tcl for its scripting capabilities; users are expected to write Tcl code as part of their work.

Since the case has not been made that Tcl/Tk and the current architectures should be retained for future development of *SYS*, it is appropriate to consider alternatives. This section does so and goes on to explore a number of ancillary factors that also bear upon the decisions that must be made.

6.1 Alternatives for the User Interface Architecture

SYS's present user interface architecture is classic Tcl/Tk windows. The major alternative is to interact with the user by presenting Web pages through a browser. Leaving aside for now the issue of generating these pages, we can ask here whether they offer advantages. I believe they do:

- The overall effect of a Web page is that of reading a document. Thanks to the built-in tools of HTML, the document can have fonts, colors, and images without undue effort on the developer's part. Users have considerable experience in perusing Web documents. They are accustomed to scrolling, searching, and cut/paste within them. None of these operations is as intuitive or as familiar in classic Tcl/Tk windows.
- Navigation in a Web-based version of *SYS* can easily offer more options than those of a window hierarchy. For instance, browser text search can be exploited by providing a large page listing all available *SYS* reports. To find one a user could scan this list or interrogate it with text search. Another page can offer the present hierarchy of menus. By exploiting free-text format, the menu options can have much more descriptive text than the current buttons.
- The distinction between buttons and links deserves comment. Each can, of course, be used to emulate the other. However, the tradition with buttons is to place them at a specific location with a cryptic word or two to suggest their function. Links, in contrast, are usually embedded in explanatory text. What's more, links are often associated with pictures to make their meaning even clearer.

6.2 Alternatives for Software Architecture

The critical goal of a software architecture for *SYS* is to isolate changes in one area from the other areas of the system. To this end, business rules, user interface operations, and database schemas should each be handled in distinct portions of the code. Often this is accomplished by separating the code in "tiers" with a layer for each set of functionality. A complimentary mechanism is object-oriented programming. Each entity—rule, data item, or user interface artifact—is represented by an object. The code implementing a class of objects is designed so that any operation that changes attributes of an object will make corresponding changes to other objects as necessary to maintain consistency.

Tcl code can easily by stratified into tiers. It is enough to define the layers and specify how each will interact with the others. The code is then written according to these conventions. In the past, Tcl was intended for small tasks and lacked the usual constructs that support object-oriented programming. Since version 8.0, the Tcl core has been augmented with such support in the form of "[incr Tcl]." That capability could be made part of the architecture for SYS's future.

If Web pages are chosen as the user interface architecture, that does not mean that a Web server must be used. Tcl could be used to create the pages; it is a general-purpose language and anything can be coded. However, it makes more sense to use a Web-server environment since many of the necessary tools will be provided. There are a number of options:

- Choose one of the systems in the *web envt* category of Table 1. Perhaps even AOLserver.
- Choose J2EE (Java 2 Enterprise Edition). This is an open framework for which pieces have been constructed by a number of vendors.
- Choose .NET. This is similar to J2EE, but does have the disadvantage of further concentrating the marketplace.
- Choose one of several proprietary Web portal frameworks.

Whether or not Tcl remains the development language, any of these choices will result in a considerably revised software architecture. The new architecture can be expected to be more flexible and resilient.

6.3 Development Tools

The choice of development language must be influenced by the support for that language available in the market place. A support environment provides code-generators, program editors, configuration management, debugging, analysis, and other tools that reduce development efforts. In general, the number and quality of environments available depends on the popularity of the language. There are more and better environments for C, C++, Java, and Visual Basic than there are for Tcl. (But that would not justify the use of C or C++.

Neither has automatic storage management and both are notorious for harboring difficult-to-find errors.)

6.4 User Platform

In a software architecture with application servers, each user's workstation must have appropriate software to connect to the server. Installation and upkeep of this software can be an issue. There are three general connection types: Web browser, X-windows as at present, or an application-specific interface program. Since browsers are universal on workstations, that option minimizes maintenance effort. X-windows is also generic, but must be installed and tailored for access to the application server. Application-specific programs are the most problematic; they must be specifically installed on any workstation that needs to access *SYS* and they will usually need to be re-deployed as the system evolves.

Like most organizations, the one using *SYS* has chosen a set of standard hardware/software platforms that are supported, supplied to users, and expected to be the systems they use. Among the problems are these:

- The standardization process may delay deployment of new versions of commercial software. When new *SYS* versions have been built in anticipation of the new environment, the standardization delay will defer *SYS* deployment. The new functionality will be that much longer in reaching the users.
- In emergencies, users may want to use non-standard workstations. These will have to be fitted out to have whichever access software has been chosen.
- Non-standard workstations may not offer the same level of security as is offered by the standard package.

Altogether, the issue of server connection is best met by browser-based delivery of *SYS* to the user. However, the other alternatives are not bad enough that the choice of delivery method should depend on this factor alone.

6.5 Scripting

Many of the commercial systems reviewed expose Tcl scripting as a tool for users. For the *ctler* and *script* categories, this is a primary motivation for Tcl. For instance, engineers using systems in the *ctler* category will create numerous scripts to test each new product. Few of the *SYS* users, however, are qualified for or interested in writing scripts to augment or modify the system. Indeed, allowing them to do so could result in violation of the business rules incorporated in the system and possible corruption of the database. Thus scripting, a key advantage of Tcl, is not a factor in the choice of architectures for *SYS*.

6.6 Development Personnel

The reservoir of developers for Tcl/Tk systems is small and relatively stagnant. Schools are not turning out developers trained in Tcl/Tk, so they are harder to find and more expensive when found. On the other hand, they tend to be more experienced and introduce somewhat fewer bugs than recent graduates. Tcl/Tk programmers, being more mature and less mobile, have lower turnover; at least that has been the experience of the *SYS* project. However, they are sometimes passionate about their chosen tool and may not be amenable to transfer to sibling projects in other languages should funding vagaries so dictate. Current and anticipated availability of development and maintenance personnel must be a factor in architectural decisions.

Conclusion

The basic claim questioned in this note is that Tcl/Tk and an architecture of classic Tcl/Tk windows is an appropriate path for the future of *SYS*. Examination of the Web site lists offered as evidence in support of this claim showed that those lists are, at best, irrelevant. No projects similar to *SYS* appear. In fact, projects similar to *SYS* generally use enterprise framework tools like J2EE and .Net.

Two architectures pervade classic Tcl/Tk windows, a user interface architecture and an application architecture. Both are limited to small systems because of the increasing complexity that arises as their paradigms are extended to bigger systems. There is no question that the current *SYS* implementation is working and can continue to work. It should probably be retained as a component of any future development. However, the present architecture should not be permitted to dictate the architecture of extensions.

Even if the present user and software architectures are abandoned, Tcl/Tk could still be retained as the development environment. AOLserver would be one model that could be followed. Adoption of this model should not be made lightly or without considering the many factors explored above.

Appendix: Commercial Uses of Tcl/Tk

Column "**S**" – Source Codes

A www.aolserver.com
C stage.caldera.com/Technology/tcl/Tcl.html#Commercial Systems
F www.tcl.tk(see section "Tcl Features")
S www.science.uva.nl/~mes/tcl/tclFAQ/tcl-commercial/Commercial-Tcl.html
T www.tcl.tk/customers/success
M mini.net/tcl/1887

See Table 1 for codes in the "**Type**" column

S	Type	Co. / Product	Excerpted from Website and other sources	Website
A	*web envt*	AOL / AOLServer	Web server using TCL as scripting Language. See text.	www.aolserver.com
C	*web envt*	Ariadne / Sibylla	Application Devt. Framework for WWW with access to various databases. The English language Web site says: "Sulla macchina server risiedono gli script Tcl, interpretati da Sibylla, che costituiscono, assieme ai documenti HTML, l'applicazione utente." (A developer said there is 12 KLOC of C code. The Tcl code was "offline" and could not be counted.)	www.ariadne.it/english/products/sibylla
C	*web envt*	Cuesta / Web-N-Able	Tool for implementing commercial Web sites with Tcl. All client sites are complex, commercial Web sites with searchable databases, secure online transaction processing, etc. All are built completely with Tcl and HTML -- no CGI, no Perl, etc.	www.cuesta.com (click "Our Clients" in upper right)

S	Type	Co. / Product	Excerpted from Website and other sources	Website
F	*web envt*	InfoCetera	"Infocetera is a stand-alone Web server/application that provides a rich suite of tools for information management and group collaboration." It uses TclHttpd for its Web-based information management system. (18 KLOC Tcl/Tk)	www.infocetera.com
C	*web envt*	Patzschke + Rasp Software / GIPSY	Gipsy is a "tool for the cross-platform development of interactive dynamic user interfaces. It comes with a pre-built display executive, visual editors and a powerful 4GL." VisualGIPSY is a drag-and-drop visual GUI builder for TclTk.	www.usersgroup.quovadx.com/tracks_session02.htm
M	*web envt*	Quovadx / QDXi	Uses Tcl extensively as an internal scripting language. From a course description: "Tcl is one of the strongest features of the QDX Integrator (QDXi) engine. Tcl allows QDXi implementers to add features that are specific to interface needs. ... just about any interface ... can be implemented using the combined features of Tcl and QDXi." (This product was once called "Neotool.")	www.usersgroup.quovadx.com/tracks_session02.htm
M	*web app*	BAE Systems, Business Systems Group (BSG)	BSG uses Tcl extensively in-house and on projects. ... Their biggest Tcl use involved building the USPS mailing online system. There are over 125k lines of Tcl driving the system (Web site, document manipulation, scheduling, distribution, high volume printer controlling, postage mgmt, address standardization, mail prep and envelope inserting).	www.usps.com/mailingonline/letters/faq.htm
M	*web app*	Career Demon	As (co-?)owner Steve Blinkhorn observes, "www.careerdemon.com instantiates a browserless secure client-server model for individual assessment and guidance wrt careers, using scripted documents, a modified tclkit and tclhttpd."	www.careerdemon.com/about_01.htm
T	*web app*	Credit Suisse First Boston / Global Talk	Using Tcl and TclPro as its software development technology, Metronome created Global Talk for CFSB. The system is a reliable, real-time, whiteboard-style messaging system for CSFB's Trading Division. Embedded C-based programs provide additional functionality	www.csfb.com
F	*web app*	Intelliclaim / COPS	Uses Tcl in Claims Overpayment Prevention Service. "It is built on a Tcl interpreter and includes object-oriented and database Tcl extensions [incr Tcl] and Oratcl."	www.intelliclaim.com
T	*web app*	Lyris Technologies / MailShield	Used Tcl and the TclHttpd Web server to write a Web interface for their MailShield anti-spam software; users write scripts in another, ad hoc, language, not Tcl.	www.lyris.com/products/mailshield

S	Type	Co. / Product	Excerpted from Website and other sources	Website
T	*web app*	NationsBank / GAS	"The company uses Tcl as part of its Global Administration Services (GAS), a computerized management framework located at multiple sites throughout the company." Marc Rossi: "Users at different sites can write Tcl scripts to customize the system for their needs and add functionality locally. We use a Tcl-based Web server, so functions can be invoked remotely from Web browsers." (NationsBank is now part of Bank of America. No word on whether the Global Administration Services is still in use.)	www.Tcl.Tk/customers/success/nations.html
M	*web app*	Re-route	Re-route ... forwards email from your old ISP to your new ISP. ...The backend code that processes retrieved email (the address correction, change-of-address notice generation and SMTP communication) was all recently re-done in Tcl (relying heavily upon Tcllib's mime and smtp libraries). The Tcl code handles 10,000 -> 20,000 emails per day and sends out double that amount via a local SMTP daemon.	www.re-route.com
T	*web app*	Tcl Developers Xchange	The Tcl Developers Xchange uses the Tcl Web Server to maintain the common look and feel of its pages and to handle dynamic page generation.	www.Tcl.Tk
S	*small*	ANSYS / ICEM & ICEPAK	Uses TCL/TK for Computational Fluid Dynamics Applications (35 KLOC Tcl; 35 KLOC C++; many KLOC FORTRAN).	www.icemcfd.com/offices.htm
M	*small*	BitMover / BitKeeper, et al.	Enterprise-level development tools for software and Web developers. BitKeeper, a powerful replicated and distributed configuration management system. … These tools are written in Tcl/Tk.	www.biTkeeper.com
T	*small*	Cygnus Solutions / Source-Navigator	Source-Navigator aids source code comprehension ... It extracts information from existing C, C++, Tcl, [incr Tcl], FORTRAN, Cobol, and assembly programs to build a project databases. Via internationalization with Tcl, Source-Navigator also ships in a completely localized Japanese edition. Source-Navigator is developed in Tcl, with about 95% of the code in Tcl and the rest in C Last release: April 2001.	www.redhat.com/solutions/embedded
C	*small*	DUX / SimCity	A platform-independent version of SimCity written in Tcl/Tk. (10 KLOC).	ftp://ftp.uu.net/vendor/dux/SimCity/README.SimCity

S	Type	Co. / Product	Excerpted from Website and other sources	Website
M	*small*	Ideogramic / Ideogramic UML	Ideogramic's proprietary Gesture Recognition algorithms convert sketches into UML diagrams. The results are captured and rendered into XMI, for export to tools like Rational Rose ... The engine for Ideogramic is tDOM, a high-performance XML processor. Unlike most XML products, tDOM supports Tcl rather than Java. Ideogramic codes its UML metamodel in [incr Tcl].	www.ideogramic.com ; http://www.ibm.com/developerworks/xml/library/x-xmi/
F	*small*	INRA / Famoz	Uses Tcl/Tk in parentage studies (genetic matching).	www.pierroton.inra.fr/genetics/labo/Software/Famoz
M	*small*	The Tolis Group / BRU	The graphical user interface for BRU is built with Tk. BRU stands for Backup and Recovery Utility.	www.tolisgroup.com/bp-qfl.html
F	*small*	Xconq	Game engine uses Tcl/Tk.	sources.redhat.com/xconq
M	*other*	Jet Propulsion Laboratory	"People all over JPL use Tcl." For example, the JTCCS scripting library: While the user interface is entirely written in Java, its server side includes a Tcl/Java interpreter. This exposes the server's Java classes as scriptable elements.	www.jpl.nasa.gov
T	*other*	NBC	NBC uses Tcl/Tk in its digital broadcast control system. Running 24x7, it controls 18 tB of digital video feeds to affiliate stations. ComputerWorld Article: "The application is now Windows NT-based and most of the code has been ported to C++ because of C++'s huge performance advantage. About half the code is still TCL, especially the GUI."	www.computerworld.com/news/1999/story/0,11280,33629,00.html
M	*other*	Pixar	Dr. Michael B. Johnson, Pixar Animation Studios: "I've used Tcl since it was first publicly available, around 1989. ... in practically every software system I've built, from image processing systems to fault tolerant agent based planners, and a number of 2D and 3D computer animation systems."	www.usenix.org/events/tcl98/brochure/TechProgram1.html#keynote
M	*script*	Electronic Data Systems / Slate	SLATE is part of EDS's "product life-cycle management" suite . … "Slate Sentry is Slate's open Tcl/Tk API, which allows direct access to Slate information via a programmable interface."	www.eds.com/products/plm/teamcenter/slate/products.shtml
T	*script*	Healtheon / Benefit Central	Healtheon uses Tcl in its Benefit Central application. Benefit rules are described with a collection of Tcl scripts that determine such things as an employee's eligibility for particular benefits. Customers can modify or extend the basic rules with a forms-based template interface, or by writing Tcl code.	www.healtheon.com

S	Type	Co. / Product	Excerpted from Website and other sources	Website
M	*script*	Oracle / Oracle Enterprise Manager	Scripts for Oracle Enterprise Manager are written in TCL and OraTcl. … OraTcl is an extension to the Tcl/Tk language (a package) that utilizes Oracle OCI calls to access Oracle DBs from TCL.	http://technet.oracle.com/products/oem/content.html
T	*ctler*	ADC Telecomm. / Cellworx	Uses Tcl/Tk and its object-oriented extensions [incr Tcl] and Expect to create the GUI interface for its ATM network ring management system, Cellworx (a hardware box).	http://www.tcl.tk/customers/success/adc.html
M, T	*ctler*	Alcatel	Rumor has it that Alcatel uses Tcl/Tk for data acquisition and text processing. (NewBridge Networks used Tcl/Tk in testing. That company is now part of Alcatel.)	www.alcatel.com
M	*ctler*	Boeing Satellite Systems	Boeing Satellite Systems (El Segundo California) uses [incr Tcl/Tk] as the core language, with extensions written in C/C++. Applications include Ground and Test Equipment operations and other complex products for both internal and external customers.	mini.net/tcl/2895
T	*ctler*	Cisco Systems / ATS	A Tcl-based extensive Automated Test System (ATS) is used by 600+ Cisco engineers to meet their testing business requirements … Cisco's Automated Test System (ATS) is composed of several tools and libraries based on Tcl that provide support for writing and executing automated test scripts.	www.cisco.com
S	*ctler*	CPU / RTAPPlus SCADA	Supervisory Control And Data Acquisition System (SCADA) (250 KLOC) "CPU has embraced Tcl/Tk technology [as] the basis for its large system SCADA technology, called 'SCL.' By integrating Hewlett-Packard's RTAP and CPU's SCL, CPU has engineered large SCADA systems without resort to C/C++ programming. CPU can develop all GUI and application programs in SCL."	www.cpu.com/PIMS
M	*ctler*	GMV / ORATOS	mini.net/tcl/1887: "GMV is [in] the aerospace and defense markets. GMV has been using Tcl extensively from 1994, [for] a data entry and manipulation system [for] a spacecraft flight dynamics support system … issues were integration with legacy programs (FORTRAN and C) and the ability to build user interfaces on-the-fly." (No other information available through Google. The GMV "News" section is empty!)	www.gmv.com

S	Type	Co. / Product	Excerpted from Website and other sources	Website
M	*ctler*	iDirect	mini.net/tcl/2892: “Currently, at iDirect, we are extending (also means rewriting) our network building desktop app with Tk. This means lots of MFC/Tk interaction (embedding Tk windows into MFC Views), SWIG, iTcl, BWidgets and other stuff. We are migrating from MFC/Stingray and find that with a few custom-written widgets (network diagramming, date chooser combobox, ip entry widget, etc) we can do just about everything MFC/Stingray can do (and it even ’looks’ better ;-). Visually (because of BWidgets and some custom tweaking on our part), you can’t tell the Tk stuff from the MFC stuff.”	www.idirect.net
T	*ctler*	Landacorp / XAct	Landacorp has been using Tcl for four years in the XAct Integration Engine and the JCS (Job Control System) Workflow/Rules Processing Engine. Rules are stored in a database, and are run from XAct, JCS, or direct user input. By using Tcl, rules can be changed “on the fly” for testing purposes without recompilation.	www.landacorp.com
T	*ctler*	Model Technology / ModelSim	The ModelSim simulator from Model Technology has used Tcl as its scripting language since 1994. The user interface is in Tcl/Tk and is promoted as permitting user tailoring of the interface.	www.model.com/products/modelsim_value_tcltk.asp
M	*ctler*	Motorola	Motorola has hundreds, or likely thousands, of Tcl coders. All (?!) Motorola semiconductor fabrication facilities are ... individual workstations ... GEM ... recipes coded in Tcl. Motorola generally takes the position that it sees no advantage in confirming any of its reliance on Tcl.	www.motorola.com
T	*ctler*	NetTest / InterWatch ATAK	“The Automated Test Authoring Kit (ATAK) available in both UNIX® and Windows® versions, automates the InterWatch and InterEmulator platforms through industry standard Tcl/Tk scripting.”	www.nettest.com/products/automated_test/literature.php
M	*ctler*	Pinebush Technologies / HyperPlot	Pinebush Technologies makes HyperPlot and HyperXpress raster image processing software.... print really big files in various formats to virtually any printer or plotter really fast. ... “We use Tcl and TclX as a cross-platform scripting framework and Tcl/Tk with BWidgets to build our user interfaces.”	www.pinebush.com
T	*ctler*	Synopsys / Formality	Synopsys offers PrimeTime, the market-leading static timing analysis tool, and Formality, a state-of-the-art formal verification tool, both with Tcl as the scripting language.	www.synopsys.com/products/products.html

S	Type	Co. / Product	Excerpted from Website and other sources	Website
M	*ctler*	TiVo	TiVo apparently uses Tcl internally. There is a claim that the prompt is tclsh, but there is no verification of this on tivofaq.	www.tivofaq.com
M	*dist*	ActiveState / ActiveTcl, ASPN Tcl	ActiveState “productizes” two bundles for Tcl developers: ActiveTcl (no charge) (without Komodo or Tcl Dev Kit, but “batteries-included”); and ASPN Tcl, which includes Komodo, TclDevKit, and support.	www.ActiveState.com
M, C	*dist*	Eolas Technologies / OpenTcl, zMap CS	Eolas is currently constructing the Web site to support its OpenTcl project. zMap CS is a set of tools (in Tcl) for developing and publishing animated imagemaps in Web pages.	www.eolas.com/tcl/index2.html
F	*dist*	Equi4 / MetaKit	MetaKit is an embedded database library with a small footprint. It lies between flat-file, relational, object-oriented, and tree-structured databases, supporting relational joins, serialization, nested structures, and instant schema evolution. There is a C++ API, a Python binding called Mk4py, and a Tcl binding called Mk4tcl.	www.equi4.com/metakit
S	*dist*	NeoSoft other	Other NeoSoft products	From source S
S	*dist*	NeoSoft Tcl	NeoSoft Tcl	From source S
M	*dist*	Procplace	Tcl training, support, and consultation	procplace.com
F	*dist*	PROSA / ETLinux	Embedded Linux features Tcl	www.prosa.it/etlinux/papers/~linuxandc.en.html
M	*dist*	Software AG / Adabas	AdabasTcl is an extension of Tcl that provides a set of commands for communication with an Adabas database.	www.softwareag.com
M	*dist*	Zveno	Specializes in XML, XSLT, DOM and other Tcl/Web-specific things	www.zveno.com
M	*defunct*	arsDigita / ACS	There was an ACS system written in Tcl by arsDigita. The company was absorbed into Red Hat and the successor system is written in Java: “CCM Core, formerly known as ACSJ, is an open-source Java platform for Web application development.”	ccm.redhat.com
M	*defunct*	AsiaInfo / Kinetic Application Processor	A Tcl based Web server. Incorporates a template processor to serve dynamic content and to allow imposition of a site-widget look and feel.	www.asiainfo.com

S	Type	Co. / Product	Excerpted from Website and other sources	Website
M	*defunct*	Belgian Graphic Interface	Provides a variety of Tcl services and products. No information on its Web site. The link provided does not lead to the download claimed.	www.bgi-sa.com
T, C	*defunct*	Binary Evolution / VelociGen or VET	Binary Evolution provides a high-performance Tcl application server, VelociGen, that works with all major Web servers. (www.binevolve.com now takes one to www.bluetitan.com. VelociGen may be defunct.)	www.bluetitan.com
S	*defunct*	CyberTerm	Cyberterm—a Networked 3D OS for PCs	From source S
C	*defunct*	GoAhead / Go Ahead Doctor	Distributed Systems management Framework Product uses Tcl in its Decision and action Inference engine.	www.goahead.com/products/docarch.htm
S	*defunct*	KIS	KIS—Kernel Information System	From source S
C	*defunct*	LPI Voice Modem	Handles multiple open modem connections simultaneously where each connection executes in it's own thread.	netrunner.net/~mersan/tel
F	*defunct*	McLennen / Who Wants to be a Tcl Hacker	Game script (no longer available)	www.tcl.tk/community/features/millionare.html
S	*defunct*	NANNY	NANNY—CPU-time balancer for UNIX compute servers	From source S
C	*defunct*	Netsonic / GotIt!	An offline browser using an intelligent agent to give automatic Web page delivery. Written entirely in TCL.	www.netsonic.com/press/pr_9-17-98.asp
M, C	*defunct*	Sonexis / LearningVoice	TeamWave developed a variety of collaboration tools, using Tcl and Tk extensively (source C named one as “Workplace”). The company was sold to Boston-based Sonexis in late 2000, where the technology was rolled into their LearningVoice and netCollaborator products. (Sonexis now lists only one product called ConferenceManager.)	www.markroseman.com/teamwave
S	*defunct*	System 9 GIS	SYSTEM 9 Geographic Information System (TM)	From source S
C	*defunct*	VHDLcapture	A “general schematic capture package (with a VHDL focus).” (The GUI uses Tk.)	www.tmx.com.au/softsmiths
C	*defunct*	WebBox (UK)	The company manufactures a (solid state) Web site Server utilizing a httpd server written in Tcl.	www.Webbox.co.uk

S	Type	Co. / Product	Excerpted from Website and other sources	Website
S	*defunct*	Westmount / I-CASE	CASE Tool Code Generation "The code generation of all Westmount products is based on parsing diagrams and generating code from that. The conversion is written in C++, the generation in TCL."	From source S

References

URLS are valid as of February 2003.

Davidson 2000 Davidson, Jim. "Tcl in AOL Digital City—The Architecture of a Multithreaded High-Performance Web Site." *Proceedings of the 7th USENIX Tcl/Tk Conference*, Austin, TX, February, 2000. Berkeley, CA: USENIX, 2000. <http://www.aolserver.com/docs/intro/tcl2k/>.

Virden 2002 Virden, Larry. comp.lang.tcl Frequently Asked Questions (November 28, 2002) (Part 1 of 6) Version 8.202, November, 2002. <http://tcl.sourceforge.net/faqs/tcl/part1.html>.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	February 2003	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
On the Suitability of Tcl/Tk for *SYS*	F19628-00-C-0003

6. AUTHOR(S)

Wilfred J. Hansen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2003-TN-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The Software Engineering Institute (SEI[SM]) was called on recently to examine a system, hereafter called *SYS*, written entirely in the Tool Control Language/Toolkit (Tcl/Tk) language. In response to some negative comments in the SEI's report, the developers presented a list of systems purported to demonstrate the viability of Tcl/Tk as a development tool. A review of the 67 listed systems found that Tcl/Tk is indeed practical for developing large systems.

Small systems written in the language often follow a paradigm of "classic Tcl/Tk windows." *SYS* embraced this approach to the extent of involving hundreds of windows. The review showed that *no other* large system written in Tcl/Tk has anywhere near as many such windows. User interviews suggested that the number of different windows was indeed a problem. *SYS* should consider an alternative design, perhaps a Web-based approach. Some design criteria are described at the end of the report.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Tcl, Tk, GUI branching, window hierarchy depth	39

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

[SM] SEI is a service mark of Carnegie Mellon University.