Rendering Tcl/Tk Windows as HTML

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-002

Contents

List of Figures

Abstract

Tool Control Language (Tcl) is a programming language having a Toolkit (Tk) library that provides a standard set of graphical user interface (GUI) widgets. Since these are aimed at direct presentation via a window manager, Tcl/Tk applications are not compatible with Web-based service delivery environments. Several tools provide some help, but do not provide a migration path for eventual full conversion to Web-based delivery. This note suggests a new approach. For the particular application prompting this note, the GUI consists almost entirely of Tk widgets, especially tables and buttons. Hypertext Markup Language (HTML) offers these same widgets, so it is natural to consider delivering Tk windows by expressing their contents in HTML. To demonstrate this possibility, the Tk library was altered to generate HTML. As described in the paper, this shows that the Tcl/Tk internal data structures are sufficient to generate appropriate HTML commands having the same user interface as that presented by the application. Consequently, it is possible to add to Tk a fourth GUI interface in parallel to the existing ones for Unix, Macintosh, and MS Windows.

1 Introduction

One system recently reviewed by the Software Engineering Institute (SEI[SM]) is a military accounting database. I will refer to it as *SYS* for purposes of discussion. One of its distinctive features is an implementation written entirely in Tool Control Language/Toolkit (Tcl/Tk) [Tcl/Tk 2002]. While this makes for succinct code, it has disadvantages:

- Military users must access *SYS* through special approved hardware-software platforms and these must have an X Windows server installed.
- The user interface is unlike emerging companion systems that are accessed through standard Web browsers.

In answer to these disadvantages, users have created another tool—a browser-accessible tool—that handles an important subclass of *SYS* operations.

Browser-accessible applications and SYS's X Windows application both run the application on a remote computer that communicates to the user's workstation. In the browser-accessible case, this communication is in the form of HTML data. In the X Windows case, communication is via a specialized data stream. Military and other secure systems usually allow communication via HTML, while restricting access via X Windows.[1] Users' systems come with ready-to-run browsers, but require some non-trivial setup to use X Windows. HTML will go through firewalls whereas X Windows often do not.

The *SYS* windows generated by Tcl used primarily the Tk widgets for tables and buttons. These, it seemed to me, could be rendered into HTML and thus presented to the user through a Web browser. It seemed simple enough,[2] so I was encouraged to create a proof-of-concept demonstration to show that it could indeed be done. It can. For the demonstration, I modified the Tcl/Tk processor so that each time it repaints the window it also writes a file containing the HTML that will generate an equivalent window. This shows that the necessary information is present within the Tcl/Tk implementation. A practical approach to using HTML would create a separate GUI interface in parallel with the existing interfaces for Unix, the Macintosh, and MS Windows.

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

1 More specifically, Web access is via HTTP and port 80 while X Windows uses its own transport protocol and port 6000. HTTP has been considered security-benign; usually it is.

2 It might even have turned out to be simple had I been familiar with Tcl/Tk and its implementation. As it was, the three-day exercise served mostly to remind me how bad C is as a programming tool.

There are other alternatives for delivering Tcl applications through a browser.[3] The Tcl Plug-in enables a Web page to incorporate Tcl code to be executed on the client's machine [Demailly 2002]. For a database application like *SYS* this approach entails performance and security drawbacks. For instance, the code will take a long time to be sent, and detail records from the database will have to traverse the net for summation at the client machine. This problem is avoided by Proxy Tk [Roseman 2000]. In this system the Tcl application runs on the server and communicates to a Java plug-in running through a browser at the client end. A deeper option is to incorporate Web service into the application. This can be done with the TclHttpd Web server [Welch 2000] and AOLserver [Davidson 2000], among others. These other approaches do not offer a migration path away from Tcl/Tk into a Web-based application.

[3] Helpful emails came from Michael Schlenker and Jeffrey Hobbs.

2 Example of the Approach

The window in Figure 1 is generated by the Tcl/Tk code in Figure 2. When this code is processed by my modified version of Tcl/Tk it creates the HTML shown in Figure 4. The latter appears in a browser as shown in Figure 3. (I made no attempt to match borders. These can be added to the HTML to make it appear much closer to the Tcl/Tk generated image.)

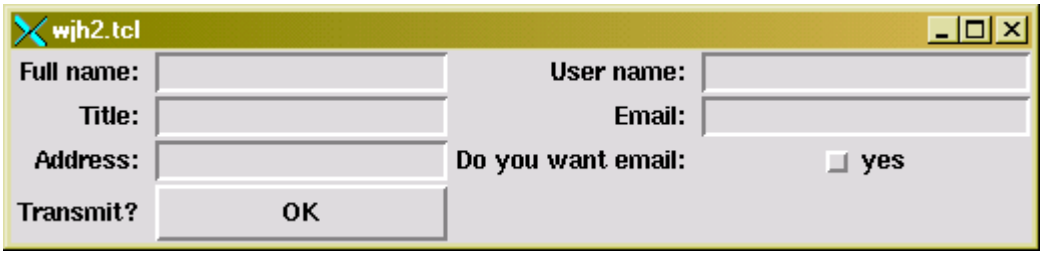

Figure 1: A Data Entry Form with Label and Field Widgets

```
# A data entry form.
#       Eric Johnson 21-Feb-96
# adapted 9 August 2002, Fred Hansen

# Labels for the data entry fields.
# left column
label .l_nm  -text "Full name: "
label .l_titlw -text "Title: "
label .l_addr  -text "Address: "
label .l_ok -text "Transmit? "
# right column
label .l_user  -text "User name: "
label .l_email -text "Email: "
label .l_ yes -text "Do you want email: "

# Data-entry fields.
# left column
entry .e_nm   -width 20
entry .e_titlw  -width 10
entry .e_addr   -width 20
button .e_ok -text OK
# right column
entry .e_user    -width 12
entry .e_email   -width 20
checkbutton .e_yes -text "yes"  -width 20

# Set up grid layout (instead of packing).
# left column
grid config .l_nm   -column 0 -row 0 -sticky "e"
grid config .l_titlw  -column 0 -row 1 -sticky "e"
grid config .l_addr   -column 0 -row 2 -sticky "e"
grid config .l_ok  -column 0 -row 3 -sticky "e"
grid config .e_nm  -column 1 -row 0 -sticky "snew"
grid config .e_titlw  -column 1 -row 1 -sticky "snew"
grid config .e_addr   -column 1 -row 2 -sticky "snew"
grid config .e_ok  -column 1 -row 3 -sticky "snew"

# right column
grid config .l_user    -column 2 -row 0 -sticky "e"
grid config .l_email   -column 2 -row 1 -sticky "e"
grid config .l_ yes -column 2 -row 2 -sticky "e"
grid config .e_user  -column 3 -row 0 -sticky "snew"
grid config .e_email -column 3 -row 1 -sticky "snew"
grid config .e_ yes -column 3 -row 2 -sticky "snew"

# Set up grid for resizing.
grid columnconfigure . 1 -weight 1
grid columnconfigure . 3 -weight 1
```

Figure 2: Tcl/Tk Code for the Data Entry Example

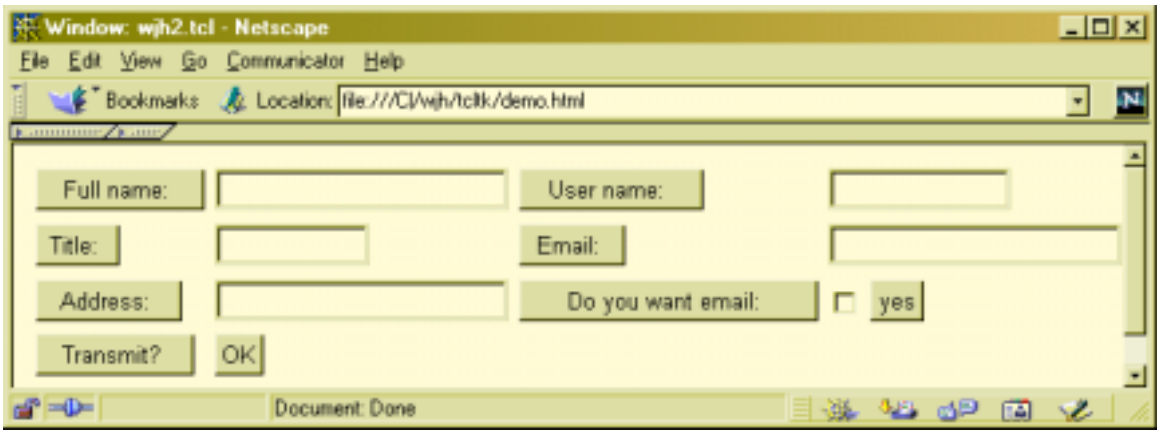

Figure 3: The Image Displayed by a Browser

```
<html>
<head>
<title>Window: wjh2.tcl</title>
</head>
<body><form>
<table>
<tr><td><INPUT type="button" disabled value="Full name: "></td>
<td><INPUT type="text" value="" size="20"></td>
<td><INPUT type="button" disabled value="User name: "></td>
<td><INPUT type=”text" value="" size="12"></td>
</tr>
<tr><td><INPUT type="button" disabled value="Title: "></td>
<td><INPUT type="text" value="" size="10"></td>
<td><INPUT type="button" disabled value="Email: "></td>
<td><INPUT type="text" value="" size="20"></td>
</tr>
<tr><td><INPUT type="button" disabled value="Address: "></td>
<td><INPUT type="text" value="" size="20"></td>
<td><INPUT type="button" disabled value="Do you want email: "></td>
<td><INPUT type="checkbox"  value="yes">  <INPUT type="button" disabled
value="yes"></td>
</tr>
<tr><td><INPUT type="button" disabled value="Transmit? "></td>
<td><INPUT type="button" value="OK"></td>
<td></td>
<td></td>
</tr>
</table>
</form></body></html>
```

Figure 4: The HTML Generated by the Modified Tcl/Tk Processor

3 Implementation

Tk is a collection of widgets, each of which is implemented with one routine in the main Tk source. The central routine calls on a display-specific routine for whichever of the native GUI interfaces is in service (Unix, Macintosh, or MS Windows). Rather than create an entire GUI interface, this demonstration was coded by adding one line of code to each of the central Tk widget routines. This line of code calls on a routine coded for the demonstration. These are mostly trivial text generation. For instance, here is the code that generates HTML for the Tk "entry" widget:

```
char * htmlEntry(win) Tk_Window win; {
        Entry *entPtr = (Entry *)win->instanceData;
        int nChars = {… compute field width …};
        catData *cat = catNew("<INPUT type=\"");
        char *type = {… type is "text" or "password" …};
        catAdd(cat, type);
        catAdd(cat, "\"");
        catSPFs(cat, " value=\"%s\"", entPtr->string);
        catSPFd(cat, " size=\"%s\"", nChars);
        catAdd(cat, ">");
        return catDone(cat);
}
```

An entry widget provides for entering a line of text. The corresponding HTML is an <INPUT … > with appropriate type and value attributes. A string containing the tag is generated with the catXxxx calls, which are a set of functions emulating the Java StringBuffer class. The returned string becomes part of the HTML.

4 What Has Been Demonstrated

This demonstration shows that

- A Web server can be connected to a Tcl/Tk application so that a user can connect through a browser and see the same screen images that the Tcl/Tk user would see at a display console.
- Tcl/Tk has enough information to generate the required HTML.
- Since most sites leave the http port open, it will be easier to access the Tcl/Tk application through the browser than through the X Windows server as required by unmodified Tcl/Tk.
- Any Tcl/Tk code involved in generating an image as a collection of widgets can be reused when converting a legacy system to a Web-based approach.

Is there a performance hit for this approach? It is unlikely. Very little computation is required to produce the HTML. If we assume that the Tcl/Tk application and the Web server are on the same host computer, negligible time is needed to transfer the HTML to the server. Network traffic—the biggest bottleneck—will be roughly the same because the generated HTML text is comparable in size to the corresponding X Windows protocol.

5 Why a Full Conversion Will Be Harder

Please note that this demonstration does not show that it is trivial to convert a Tcl/Tk application so that it can be presented via a Web browser. There are a number of problems.

Many widget attributes like borders and colors have not yet been converted. These should be straightforward, if tedious. Converting the widgets I have not tackled will require some effort, but since HTML and Motif share widget concepts, this should not be hard. It may take a sizeable effort to convert a drawing widget used for pictures, but I have not seen any such widget in *SYS*. The present implementation only handles the “grid” geometry manager. It will be a bit harder to handle the “pack” manager and quite a bit harder to deal with the “place” manager. I don’t think *SYS* uses these, either. Some design effort will be needed to resolve the issue of returning information from the Web page so as to drive the application.

The existing *SYS* application is almost entirely widgets of the sort implemented by this demonstration. The few exceptions can be dealt with by special-purpose coding. There is one relatively frequent exception: mouse clicks in some non-widget locations are used to invoke new windows. It may be possible to create a general solution with JavaScript programming. Otherwise it may be necessary to revise the GUI design. In the worst case, a user would have to click on a new button instead of on the entire field.

The current approach of writing HTML to a file is ad hoc. The “right thing to do” is to make the HTML/Web-server appear to Tk as a fourth window manager on a par with Unix, Mac, and MS Windows. If done thoroughly, it is conceivable that minimal change would be needed to the *SYS* code.

A major problem is implementation of user navigation. Browsers are ill suited to immediate interaction between a system and a user. Immediate feedback to user actions requires an exchange across the Internet and the attendant delays destroy any illusion the user might have that the application is a tool with the same immediacy as a pencil. Web-based applications typically ask the user to fill in data fields and press a button before getting any feedback (although Java applets and other scripting tools are mitigating this problem). Those parts of *SYS* that I witnessed are, however, much like the typical Web application. The user clicks and another screen comes up. So the conversion of *SYS* to the Web might not be as bad as for other systems. What will have to change is the relationship between the various parts of the application. They must be torn apart so that information exchange with the Web server can be inserted appropriately. Unless the application was originally built in a highly structured fashion, it will be a challenge to dissect it as necessary.

Additional development hassles will arise in having to deal with a server system in addition to Tcl/Tk. There will be additional tasks in infrastructure maintenance, configuration management, developer training, and a host of other considerations. Testing will be harder.

Conclusion

This note demonstrates a possible path for transition of *SYS* and similar applications from Tcl/Tk plus X Windows server to a Web-based approach. The existing application would remain under the covers and the user would interact via a browser. This is thus an alternative to the existing options of retaining Tcl/Tk or total rewrite. The HTML approach can also serve to allow continued use of the existing code as portions of the applications are converted over to a true Web-based approach.

References

All 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/>.

Demailly 2002 “Welcome To The Tcl Plugin.” <http://www.scriptics.com/software/plugin/> (2000).

Roseman 2000 Roseman, Mark. “Proxy Tk: A Java applet user interface toolkit for Tcl,” *Proceedings of the 7th USENIX Tcl/Tk Conference*, Austin, TX, February 2000. Berkeley, CA. USENIX, 2000. <http://www.usenix.org/publications/library/proceedings/tcl2k/roseman.html> (2000).

Tcl/Tk 2002 “Tcl Developer Site” <http://tcl.activestate.com/> (2002).

Welch 2000 Welch, Brent. “The TclHttpd Web Server,” USENIX Technical Program, 7th Tcl/Tk Conference, February 2000. <http://www.tcl.tk/software/tclhttpd/> (2002).

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
Rendering Tcl/Tk Windows as HTML	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-002

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)

Tool Control Language (Tcl) is a programming language having a Toolkit (Tk) library that provides a standard set of graphical user interface (GUI) widgets. Since these are aimed at direct presentation via a window manager, Tcl/Tk applications are not compatible with Web-based service delivery environments. Several tools provide some help, but do not provide a migration path for eventual full conversion to Web-based delivery. This note suggests a new approach. For the particular application prompting this note, the GUI consists almost entirely of Tk widgets, especially tables and buttons. Hypertext Markup Language (HTML) offers these same widgets, so it is natural to consider delivering Tk windows by expressing their contents in HTML. To demonstrate this possibility, the Tk library was altered to generate HTML. As described in the paper, this shows that the Tcl/Tk internal data structures are sufficient to generate appropriate HTML commands having the same user interface as that presented by the application. Consequently, it is possible to add to Tk a fourth GUI interface in parallel to the existing ones for Unix, Macintosh, and MS Windows.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Tcl, Tk, Web portal, Web-based, HTML, GUI Interface	22

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