

**Where Do GUIs Come From?
And How Can We Make Them Better?**

Course Project

John C. Giordano

Department of Computer Science

University of Virginia

April 30, 2004

“On my honor as a student, I have neither
given nor received aid on this assignment”

Executive Summary

With the advent of window based operating system interfaces, Graphical User Interfaces (GUIs) have emerged as the de facto means by which users employ computer programs. Nonetheless, GUIs represent only a single interface – that between the user and other software. In the field of Computer Science, designers, developers and programmers are schooled primarily in the implementation of algorithms in order to solve problems. In many instances, the concerns of the users are considered only as an afterthought to the development of a code base designed to address a problem or provide utility to users. At best, user interfaces are addressed by programmers or designers other than those responsible for core algorithms or processes.

We explore two prominent platforms by which many computer scientists gain their first experience in developing a GUI – JavaSwing and Tcl/Tk (pronounced *tickle/tee-kay*). Sun Microsystems has provided an extensive, web-based tutorial for JavaSwing in order to assist programmers and developers in the design and implementation of GUIs for their Java code. On the other hand, Tcl/Tk is not a proprietary product, except as distributed by vendors, therefore there is no foremost authority to provide a tutorial for it. A handful of tutorials have emerged as a primary means for developers to obtain elementary skill in employing Tcl/Tk, as well as a freely distributed computer-based training application called TclTutor.

We assess how well the JavaSwing and Tcl/Tk tutorials and freeware address GUI design principles, demonstrate where this foundational instruction falls short of addressing critical human factors concerns, and discuss some of the fundamentals of GUI design. We propose how these tools can be improved to include elements of design crucial to users.

I. Introduction

With the advent of window based operating system interfaces, Graphical User Interfaces (GUIs) have emerged as the de facto means by which users employ computer programs. Nonetheless, GUIs represent only a single interface in complex software system – that between the user and other software components. Today’s sophisticated software has numerous interfaces between program components and modules. Arguably, the GUI is the most critical interface from the software users’ perspective.

In the field of Computer Science, designers, developers and programmers are schooled primarily in the implementation of algorithms in order to solve problems. Their focus is on designing algorithms that are efficient, robust, and scalable. Accordingly, the skills to build or develop proficiency in crafting usable GUIs often come from freely available tutorials and how-to’s, independent learning and self-study on the part of programmers. In many instances, the concerns of the users are considered only as an afterthought to the development of a code base designed to address a problem or provide utility to users. At best, user interfaces are addressed by programmers or designers other than those responsible for core algorithms or processes.

In this work, we explore two prominent platforms by which many computer scientists gain their first experience in developing a GUI – JavaSwing and Tcl/Tk (pronounced *tickle/tee-kay*). JavaSwing is an extension of the Java programming language that enables developers to create GUIs for Java source code, which can subsequently be employed on various architectures like PCs (running Linux or Windows), Macs, Sun Workstations and other computing platforms. Java and JavaSwing are products of Sun Microsystems. Tcl/Tk, which stands for Tool Command Language (Tcl) and its associated toolkit (Tk), is a vendor-independent scripting language and development environment created at UC Berkeley and intended as a library package for the C

programming language. Our hypothesis before beginning this analysis is that these platforms and the tutorial resources associated with them provide a sufficient background regarding programmatic concerns, but despite their intended use as GUI development tools, the material for introducing programmers to JavaSwing and Tcl/Tk falls short of addressing user concerns at any length.

II. Background

JavaSwing is a component of the Java Foundation Classes (JFC) for the Java programming language. The Java language was first introduced by Sun Microsystems in 1995 and is somewhat unique relative to other high level object oriented languages. Programs written in the Java language are first compiled into Java bytecode, and then this code is run through a Java interpreter which must be installed on the platform on which the code is to run. This allows programs written in Java to be compiled once, distributed, and then run on the various computer architectures like those mentioned in the Introduction. In order to make Java a more compelling choice for program development, Sun introduced the Abstract Window Toolkit (AWT) comprised of classes and methods that programmers could use or extend in order to create GUIs. Although there were a number of problems with the AWT, Sun introduced JavaSwing and the JFC in 1997 to replace AWT technology, with the intention of enabling easy GUI development as well. The JFC is comprised of five other components in addition to JavaSwing, yet JavaSwing is the most sizable element of the JFC. JavaSwing is a collection of 17 public packages of custom classes and methods for creating GUIs. This means that programmers have a rich set of tools for creating GUIs that suit their needs, as well as the ability to extend the classes and methods provided by JavaSwing for customization.

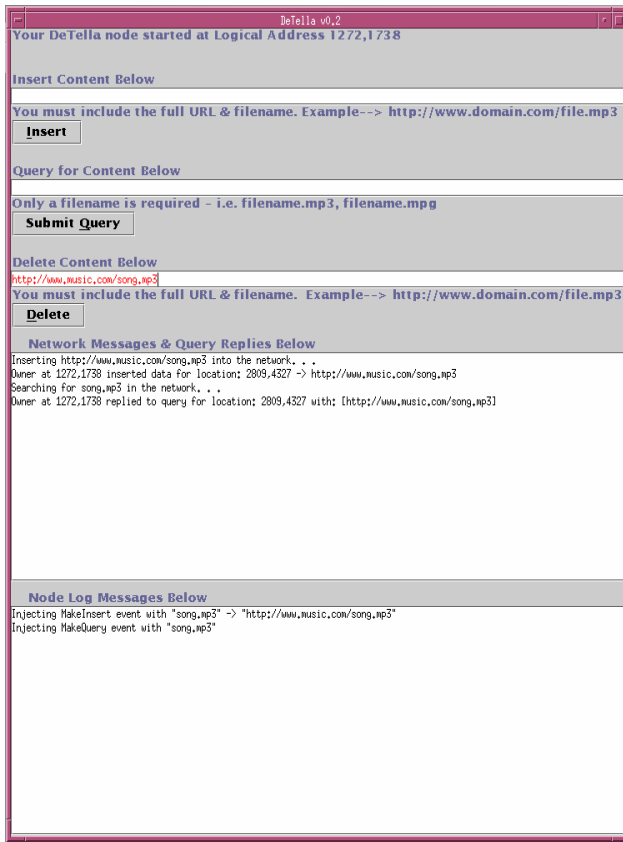


Figure 1 – An Example JavaSwing GUI

In order to become familiar with the capabilities and components of JavaSwing, Sun provides an extensive web based tutorial for programmers through their corporate web site. This tutorial addresses most of the commonly used classes and methods of JavaSwing – at least those that are required in order to program fully functioning GUIs. Even though there is a wealth of information available to the user, only a few pages are required in order to create a functional GUI. By way of example, Figure 1 shows a GUI that was created in less than a day by a programmer who had never used JavaSwing before or programmed a GUI.

The JavaSwing tutorial provides discussion of the components that are used to create a GUI, example code segments, fully functional code that can be copied, compiled and run by the user, and screen shots of what the provided code should look like when it is run on a native interpreter.

Unlike the relationship between Sun and Java, Tcl/Tk does not have a single, corporate entity to control its distribution and oversee its use. Perhaps this stems from the fact that Tcl, the precursor to Tk, was created by John Ousterhout while he was on sabbatical from the University of California – Berkeley in 1988. Tcl differs from Java in that it is not viewed by the community as a programming language. It is more like a scripting language and was originally intended by its creator as a library for the C programming language. Ironically, after working at Sun for a

few years upon leaving Berkeley, Ousterhout spun off his own company, called Scriptics, which maintains a Tcl/Tk developers exchange web site. Notwithstanding, Tcl is vendor-independent, and is updated and re-released by a consortium of developers. Like Java, Tcl is distributed at no cost, and has a graphics capability as well, known as Tk. Tk is an extension of the Tcl language and provides widgets, which are similar to objects in object oriented programming, for programmers to use in creating GUIs. It was developed after Tcl's initial release and first became available in 1991. Currently, the Tcl/Tk distribution is available through the Tcl/Tk developers web site mentioned above. This web site, though, does not provide the same kind of comprehensive tutorial that Sun maintains for JavaSwing. It does, however, provide pointers to numerous tutorials and discussion forums hosted by other developers or those interested in Tcl/Tk. Also, there is a freely available computer-based training application that provides users with a self-paced tutorial in a standalone setting.

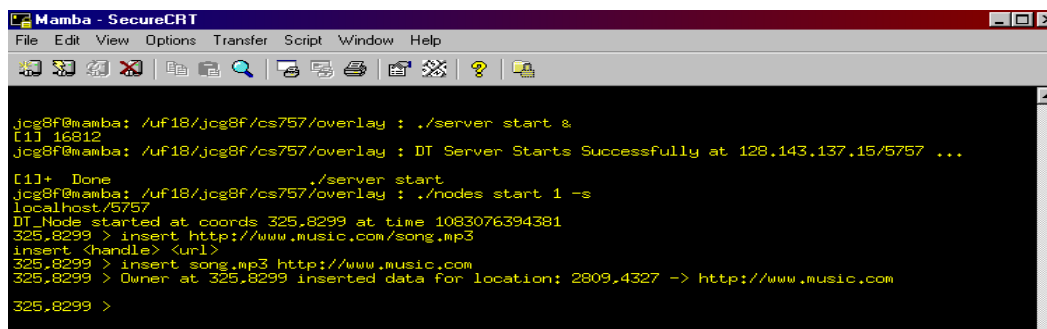
With an abundance of material that provides programmers with an introduction to both JavaSwing and Tcl/Tk, we took the opportunity to assess these resources in how they address concerns regarding GUI development beyond programmatic issues. We have explored and assessed the JavaSwing tutorial provided by Sun on their website, four of the many tutorials available for Tcl/Tk, as well as the computer-based training application known as TclTutor. Based on our hypothesis, we turned a critical eye to any discussion of the principles of GUI development included in the materials, and accordingly, discuss some of those principles in detail. Following our assessments and review of GUI design principles, we propose suggestions for improving the existing tutorials and offer concluding comments. Throughout this work, we offer examples and demonstrations of what the tutorials are capable of and the usefulness they offer to the programmer.

III. Assessing the JavaSwing Tutorial

Aside from books or independently provided courses or courseware, all with an associated cost, Sun provides numerous tutorials on their web site covering a variety of topics, particularly those related to its Java programming technology. These tutorials are provided at no cost and are readily accessible to anyone with a connection to the Internet. The Sun website is easily searchable and a query for JavaSwing on it returns the tutorial as one of the most relevant hits. Like other tutorials on the Sun web site, the one for JavaSwing is referred to as a “trail”. This trail is organized into seven individual lessons and over 125 individual web pages, with content ranging from an introduction to JavaSwing to detailed explanations of how to use many of the available components. While some of the pages are quite brief, presenting only a single screen of information, others are extensive, thorough and require repeated scrolling. If a programmer is completely unfamiliar with JavaSwing, they may work through all of the lessons sequentially in order to gain a working knowledge of how to employ the GUI classes and methods. For those interested in immediate functionality, expertise can be gained by focusing on a few key components of a GUI. If one is returning to the tutorial to gain more advanced or specific information, a table of contents is provided so that the desired content can be quickly located and accessed. By their own calculations, Sun estimates that the entire JavaSwing tutorial could be completed in less than ten hours.

As Figure 1 demonstrates, a programmer possessing intermediate knowledge and experience with Java can work through the initial lesson material rapidly and deploy a functional and practical GUI in a matter of hours. For reference, the GUI in Figure 1 serves as an interface for a functional code base of about 8,000 lines of Java and was developed in ten person-hours, including time spent reading and referring to the tutorial and developing the required JavaSwing

code. Compared to the code base, the GUI required only about 250 lines, or 2.5% of the entire project. We leave it to the reader to determine if working with the simple yet effective GUI in Figure 1 represents an advantage over working with the series of command lines that would be required to garner the same functionality, as shown in Figure 2.



```

Mamba - SecureCRT
File Edit View Options Transfer Script Window Help

jcg8f@mamba: /uf18/jcg8f/cs757/overlay : ./server start &
[1] 16812
jcg8f@mamba: /uf18/jcg8f/cs757/overlay : DT Server Starts Successfully at 128.143.137.15/5757 ...

[1]+  Done                  ./server start
jcg8f@mamba: /uf18/jcg8f/cs757/overlay : ./nodes start 1 -s localhost/5757
DT_Node started at coords 325,8299 at time 1083076394381
325,8299 > insert http://www.music.com/song.mp3
insert <handle> <url>
325,8299 > insert song.mp3 http://www.music.com
325,8299 > Owner at 325,8299 inserted data for location: 2809,4327 -> http://www.music.com
325,8299 >

```

Figure 2 – A Demonstration of the Command-Line Sequence Correlating to Figure 1

Along with discussing the functional components of JavaSwing, the tutorial addresses such concerns as layout and screen geometry management, yet it does so in reference to programming concerns and not what the user of the GUI may require. As we see in Figure 3, the JavaSwing tutorial address the layout of a window, yet it offers no suggestion as to when one layout would be preferable to the others or what feedback the programmer should seek from the user with regards to potential use. This represents an opportunity for Sun to provide some guidance for programmers and developers who may be seeking such direction. This could easily be done with pointers to design standards or organizations

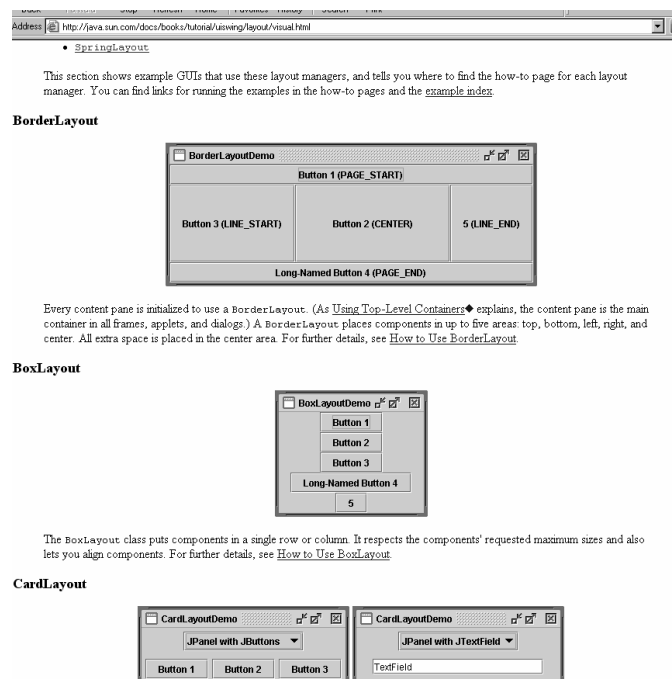


Figure 3 – JavaSwing Layouts

which advocate good usability design. Also, the overwhelming balance (over 60%) of web pages in the tutorial are explicit “how-to’s” which include very little discussion other than implementation details. This further demonstrates the sparse nature of material concerning the usability of the interfaces developed with JavaSwing. While JavaSwing provides a rich set of tools for developers to quickly and easily produce functional GUIs, and a detailed tutorial that minimizes the time required to attain proficiency, we note that Sun could easily integrate design guidance into the tutorial. This would serve to not only enhance the quality of the interfaces developed with JavaSwing, but also broaden the skills of those programmers who learn to develop GUIs through Sun’s tutorial.

IV. Assessing the Tcl/Tk Tutorials and Computer-Based Training

While JavaSwing and Tcl/Tk can both be used to quickly develop user interfaces, there are many stark differences between these two platforms. Primarily, they represent functionally diverse programming paradigms. Whereas JavaSwing is a component of a high level object oriented language, Tk is an extension to a scripting language intended to complement a procedural programming language, and neither Tcl nor Tk apart from each other would be highly useful in today’s graphically driven, desktop visualization operating systems. Even though they are both distributed freely, Java and JavaSwing are in much wider use than Tcl/Tk. There is a vibrant community of Tcl/Tk users and developers; however, as evidenced by the consortium behind the Tcl/Tk developers exchange (<http://www.tcl.tk>) and the various web sites that offer tutorials covering Tcl/Tk. Herein, we assess the usefulness of the material covered in these tutorials, as well as the TclTutor application.

The first thing one notices when browsing the tutorials available for Tcl/Tk is the heterogeneity in the sources of these materials. While most appear to be associated with academia, the geographical dispersion of the institutions that host these web sites demonstrates the wide spread acceptance of Tcl/Tk. The tutorials we reviewed are:

1. The Tcl/Tk Tutorial, hosted by Kansas University (<http://hegel.ittc.ukans.edu/topics/tcltk/tutorial-noplugin/index.html>)
2. The Tcl/Tk Cookbook, hosted by the Council for the Central Laboratory of the Research Councils in the United Kingdom (<http://www.dci.clrc.ac.uk/Publications/Cookbook/index.html>)
3. Tcl/Tk Made Easy, hosted by the California State University, San Luis Obispo (<http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/tcl/>)
4. Graphical Applications with Tcl and Tk, hosted by an independent developer and publisher (<http://www.pconline.com/~erc/tcl.htm>)

The depth of material presented in these tutorials ranges from very broad and topical, as is the case with *Graphical Applications with Tcl/Tk*, which is little more than an advertisement for a Tcl/Tk book authored by the developer who created the tutorial. *The Tcl/Tk Tutorial* and *The Tcl/Tk Cookbook*, which are the most expansive of those reviewed, are not as sizeable as the JavaSwing tutorial. They do, however, present sufficient treatment of Tcl/Tk, which is a more limited tool relative to Java and JavaSwing. These tutorials incorporate Tcl as well as the graphical extensions provided by Tk. *The Tcl/Tk Tutorial* consists of 13 lessons or chapter, with embedded pointers to more explicit discussion, code segments and screen shots. Unfortunately, most of the material addresses the basics of Tcl, and very little Tk. *The Tcl/Tk Cookbook*, however, devotes most of its ten chapters to Tk in addition to the basics of Tcl, and provides the same kind of embedded pointers to screen shots and expanded discussion as those found in *The Tcl/Tk Tutorial*. Strictly in terms of GUI development, *The Tcl/Tk Cookbook* is superior. Neither of these tutorials, however, devotes any discussion to the principles of GUI design and is mostly programmatic in nature.

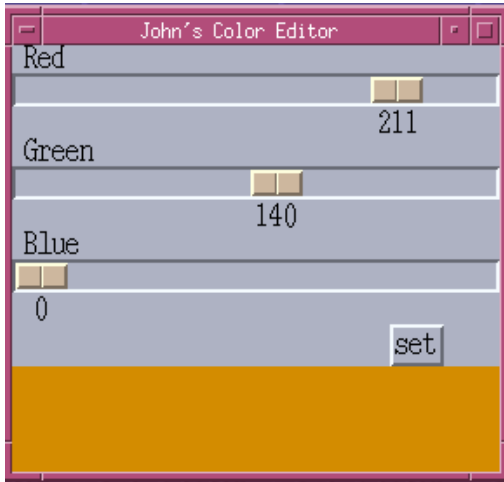


Figure 4

In order to gain the most rapid understanding of how to deploy Tcl/Tk, *Tcl/Tk Made Easy* presents material that guides users through crafting a working GUI and application in less than 30 minutes. Although not highly functional or useful, this demonstrates how quickly a novice can adapt to these tools and begin developing a GUI. An example of the work described above is shown in Figure 4.

Along with these tutorials, TclTutor, a freely distributed computer based training application, is available for download and self paced learning. It is expansive, consisting of 43 lessons with material presented in one of three levels of verbosity (novice, intermediate, expert) which can be toggled by the user at any time. Sample code including comments, and a Tcl interpreter with output display are integrated into the application, thus allowing the user to modify or write example code, run it, and view the output all on one screen at the same time. This learning and development environment is depicted in Figure 5.

Despite all of its rich features and capabilities, TclTutor contains no material discussing or employing Tk – all 43 lessons are devoted strictly to Tcl, hence there are no graphical user interfaces or components that can be created with this tool.

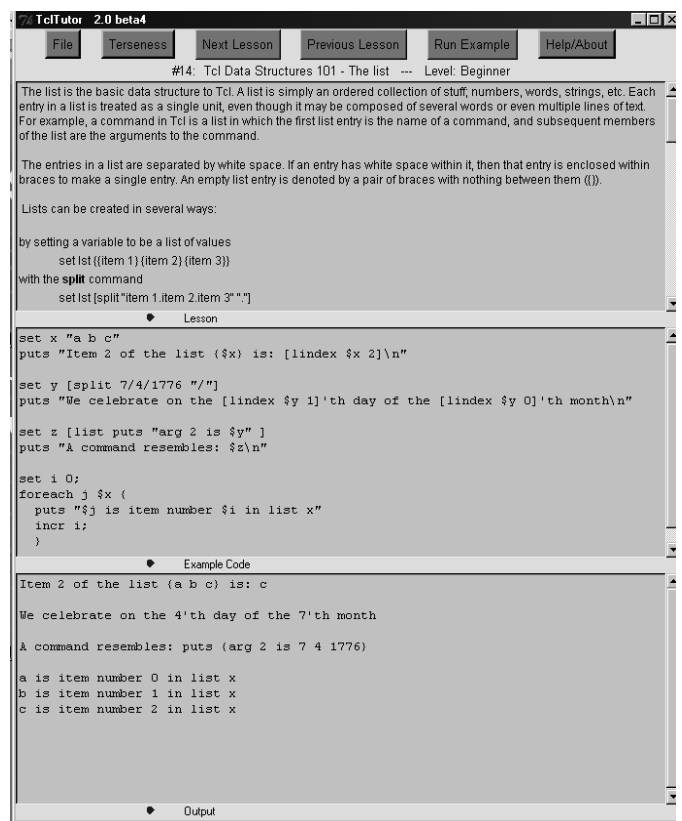


Figure 5

Unfortunately, there is no Tk counterpart to or module for TclTutor currently available from the application provider. As a tool for learning Tcl, however, it is highly useful.

As with the JavaSwing tutorial, any discussion of GUI design principles is noticeably absent from the Tcl/Tk resources. Even though there are a variety of stakeholders associated with Tcl/Tk, the available tools limit their focus to programmatic concerns and getting users to program GUIs as quickly and easily as possible. With no theoretical underpinnings, one can see how GUI development could easily diverge from a sound basis of design and result in hard to use or inadequate interfaces. It would be quite helpful for Sun and the various Tcl/Tk supporters to consider some of the GUI design principles discussed below for inclusion in their respective material.

V. Discussion of GUI Design Principles

Since GUIs first became prominent not only as a way for users to interact with programs, but also in desktop themed operating systems, a significant body of literature has developed addressing the concerns that designers must consider with regard to usability as well as functionality. One of the seminal texts introducing the notion of usability along with discussion of testing methodologies, heuristics, evaluation and implementation is *Usability Engineering* (Nielsen 1993). In it, the author proposes some of the ideal characteristics of a GUI, including:

- speaking the user's language
- using natural dialogue
- maintaining internal and external consistency
- clearly marking shortcuts or exits
- preventing errors as opposed to simply reporting them

Speaking the user's language means portraying the user as the reference point for any exchanges of dialogue between the application, the GUI and the user. This can be demonstrated by not accusing the user of making mistakes or using the software improperly, but rather suggesting what the user could do to correct an error or guide them to a more likely approach. At design time, programmers should consider the context in which messages will be displayed to the user and assume nothing about the level of expertise or proficiency of the user. Comments should be helpful rather than critical, and should correlate to the needs of the user. Using natural dialogue means avoiding technical terminology simply because the programmer is more comfortable with precise technical terms as opposed to conversant dialogue. GUIs should be consistent, both within themselves and with other GUIs or applications. This means that if a particular key stroke or menu item refers to a certain function when a GUI or its application is in a given state, then that key stroke or menu location should refer to the same or a similar function in all of the other states that the software can be in. In the same manner, symbols should be used and displayed in a consistent fashion as a program transitions from state to state. Various applications, while offering different functions, should adhere to the same mappings between symbols, functions and meanings across platforms and implementations. Shortcuts, exits and other capabilities should be marked in manner corresponding to their function. For example, an icon of a printer would clearly indicate what button should be clicked in order to print what the user is interested in, but icons of a broom or a scissor may not be clearly understood by novice users who are unfamiliar with an application or computers in general.

Along with underlying theory, there are many offerings that discuss practical implementation issues regarding GUIs. For example, *Developing User Interfaces* (Olsen 1998) serves as an intermediate text for computer programmers in order to develop an awareness of design

considerations, and interleaves them with demonstrations of how those principles apply on various technological platforms and in different programming languages or technologies. Sufficient treatment is given to both the technical and visual elements of GUI design and development. Also, *The Essential Guide to User Interface and Design* (Galitz 2002) is a mammoth volume integrating theory, design principles and practical advice for the programmer and designer, yet it does so in a manner independent of any single platform, technology or language.

VI. Suggestions for Improvement

Clearly, the opportunity exists for the stakeholders overseeing JavaSwing and Tcl/Tk to combine the material in their tutorials with some of the required background understanding and theoretical issues pertaining to GUI designs. Both of these platforms purport themselves and demonstrate to be ideal for rapid GUI development. Along with swiftness of implementation, those administering JavaSwing and Tcl/Tk tutorials should consider, at a minimum, reference to some of the seminal work that has been done in GUI design. Along with pointers to some of this material, it would not require a significant amount of resources to include tailored discussion of some GUI design principles which correlate to the software components that each platform offers. Sun, which has already placed considerable investment of time and resources into its expansive JavaSwing tutorial, would serve the programming community well to extend their materials to include references or additional material with the existing tutorial. While the Tcl/Tk development community is more dispersed and not unified as a single corporate entity, the Tcl/Tk developers exchange could include such pointers along with links to the various tutorials and work to advocate usability concerns along with the Tcl/Tk tool itself. From a more general

perspective, those who manage GUI development projects should ensure that their programmers have some background in GUI design principles or acquire it before coding begins.

VII. Discussion and Conclusions

For some, using a GUI represents a trade-off. It abstracts the user from the processes that the underlying software is performing and can sometimes complicate accomplishing what could otherwise be done directly. Many users, however, prefer the convenience and reliability of a GUI for performing tasks or interacting with software. In developing usable, functional GUIs, JavaSwing and Tcl/Tk are just two of the numerous technologies available to programmers. The readily available tutorials and other materials associated with them make them an easy choice for the programmer that has never worked in GUI development or who wishes to do so quickly or almost effortlessly. What we have assessed, however, has proven to be more science than art – that is, the focus of the material is on the technical or programmatic issues of developing a GUI, and not the design considerations that should be taken into account well before development begins. Those concerns should not stop at design time. They should be integrated into an iterative design, development, and improvement process, corresponding to whatever software development methodology is employed, and can hopefully contribute to better user interfaces.

Lastly, we note that while the resources we have examined for this work are all freely available over the Internet at no cost to the user, most of the materials related to GUI design principles are proprietary and stem from commercial entities and activities. While there may be a monetary cost associated with obtaining these skills, programmers who seek to develop GUIs would benefit from adding these skills to their repertoire. An investment in such at an early stage would likely contribute to a return on that investment at some later point. Managers and

programmers should develop an awareness of GUI design principles and the shortcomings of the available tutorials. This would likely contribute to better results and fewer resources wasted on the development of poorly designed or inadequate GUIs.

References

<http://www.java.sun.com>

<http://www.tcl.tk>

Galitz, Wilbur O. (2002) The Essential Guide to User Interface and Design. New York, NY. John Wiley and Sons, Inc.

Nielson, Jakob. (1993). Usability Engineering. San Diego, CA. Academic Press, Inc.

Olsen, Dan R. (1998) Developing User Interfaces. San Francisco, CA. Morgan Kaufmann, Inc.