Volume 3, Issue 9, September 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Graphical User Interface Design Essentials & Process

Reena Saini
Department of Computer Science and Engineering
B. K. Birla Institute of Engineering & Technology, Pilani, Rajasthan, India

Abstract— GUI (Graphical User Interface) is one of the major achievement on Human- Computer interaction and more acceptable visually for users. This paper explains the GUI design essentials and the process of designing GUI. This paper focuses on eight golden rules of GUI design and discusses how to design good GUI.

Keywords— GUI, User Centered design, Design process, Design essentials

I. Introduction

When designing a Graphical User Interface, it is important that the needs, wants, and limitations of the end users (who finally use the program) are given extensive attention. User Interface is an interface between the user and the computer. The first user interfaces were command-line interfaces where user could interact with the computer by typing commands on the keyboard. Graphical User Interfaces uses pictures and graphics instead of just words to represent the input and output of the program. The program displays certain icons, buttons, dialogue boxes etc. on the screen and the user controls the program mainly by moving a pointer on the screen and selecting certain objects by pressing buttons, etc. The function of a Graphical User Interface is to facilitate the handling of an application by means of graphical elements.

The design of today's graphical user interfaces often uses the so called "desktop metaphor". The “desktop metaphor” is a set of unifying concepts currently used in a number of graphical user interfaces in computer operating systems. The monitor of a computer represents the user's desktop upon which documents and folders of documents can be placed. A document can be opened into a window, which represents a paper copy of the document placed on desktop.

In GUI design we have main consideration on human/user, so it is “User Centered Design” process. "User Centered-Design (UCD) is a design philosophy and a process in which the needs, wants, and limitations of the end user of an interface or document are given extensive attention at each stage of the design process. User-centered design can be characterized as a multi-stage problem solving process that not only requires designers to analyze and foresee how users are likely to use an interface, but to test the validity of their assumptions with regards to user behavior in real world tests with actual users. Such testing is necessary as it is often very difficult for the designers of an interface to understand intuitively what a first-time user of their design experiences, and what each user's learning curve may look like[1,2].

The rest of the paper is organized as follows. In section II & III design essentials are described. In section II a brief discussion of “How to design good GUI” is presented. In section III contains eight- golden rules of interface design. In section IV we explain GUI designing process. Finally conclusion is present in section V.

II. How to design good Graphical user interface

Designing a good user interface is an iterative process. First, we design and implement a user interface using appropriate techniques. Then we evaluate the design. The results of the evaluation feed the next design and implementation. We stop the process when you have met your design goals or you run out of time and/or money. Note that if we have different user communities (or the same user with different jobs), we may need different user interfaces, customizable user interfaces or both. For example, Microsoft Word provides four user interfaces: normal, outline, page layout and master. In addition, Microsoft Word provides a host of customization features for the keyboard, menu and toolbars. While design is important, the real key to creating a good user interface is in your evaluation techniques. Obviously, we should use our own user interface. If we can’t use it, how can anyone else? Next, get feedback from your alpha testers, then from you beta testers. The best evaluations are done by watching over the shoulder of the user [3, 4].

Some tips for design good user interface:

1) *Don’t invite the user to make mistakes*. If a function in you program is not currently available, don’t let the user try to use it. For example, instead of providing an error message that the function isn’t available, gray-out or hide the control that calls the function. Allowing the user to call a function which can’t be used is inviting the user to make a mistake.
2) *Give the user a hint*. If possible, provide the user with some insight into how your program functions. For example, if a process takes a long time, show the user what is actually happening. Providing tool tip help and describing what data fields are for in the status bar can also provide the user with valuable insight.
3) *Don’t write a manual*. A user is much more liable to lose a manual than read it. To avoid this problem, don’t write a manual. Put the effort you would have taken to write a manual and put it into the help system. The help

system will always be there for the user even if the manual isn't. Remember that help will only be consulted when the user has a problem. Your help system should make it easy for users to find information needed to solve their typical problems. (OK, this requires a release of the software to find out what those typical problems are.)

4) *Train new users*. Without a manual, you need to be more creative on how to train users. The key here is to *train* users - training materials should not get in the way of (irritating) experienced users. But even experienced users may need re-training or additional training on a previously unused feature. Thus, the training should get out of the way unless it is needed. *Some common training methods are:*

   - *Tip of the Day*- Appears when the program starts. Gives the user information on program features which the user may be unfamiliar with.
   - *Tutorial*- Guides the user through the software using a sample problem.
   - *Tutors*- Provides a detailed list of steps to accomplish some task. A tutor is like a tutorial except that instead of showing the user how to solve a sample problem, the tutor shows the user how to solve the user's problem.
   - *New User Message*- Advice which pop-up when the user tries to do something. For example, the message might pop-up when the user selects a button or enters a data entry field. Users definitely need a way to turn this feature off and on.
   - *Hints*- Hints are reminders which are built into the software in an unobtrusive ways. Tool tips and status bar messages are prime examples of hints.
   - *Wizards*- Summoned to perform some action for the user. Wizards typically ask the user what the user wants and then perform the necessary actions for the user. Wizards are not really training device since the user will have no more idea how to do the job after the wizard is done than before. So use a wizard whenever the user unlikely to ever want to know how to actually use the features hidden by the wizard.

5) *Provide useful interruptions*. When are interruptions ever useful? The interruptions I am talking about are pop-up modal dialog boxes. They interrupt the user and demand the user's attention. They can be used to either give information to or get information from a user. In either event, the most useful interruption is one that does not occur! Here are some specific ideas on eliminating interruptions or at least making them useful:
   - *Just do it.* An error message should never say "You must do X before you can use this Y function." The best way to avoid this is to just go ahead and do X for the user. If X involves substantial time or user interaction, you might ask the user if the user wants to do X now or cancel the action. Finally, you could gray out Y until the user has done X. The problem with graying Y out is that the user may not know that completing X is what turns Y on.
   - *Announce subtly.* Message boxes which say "I'm done" can be very irritating because the user has to get rid of them before going further. Signaling completion can be done in more subtle ways**.** For example:
     a) If a task is really modal (such as copying a file to disk), put up a dialog box showing progress. When the task is done, simply remove the progress dialog box and the user will know the task is done.
     b) Instead of a dialog box, use an hour glass and show progress on the status bar. When the task is done, remove the hour glass and put "Task complete" on the status bar.
     c) Use an in-process icon in a corner of the screen. Both Netscape Navigator and Microsoft Internet Explorer uses this technique.
   - *Give explanations.* Instead of displaying a error message number which is useless to the user, give the user a hint as to what the error is all about. The hint will probably also help your technical support personnel and might even jog your own memory.
   - *Tell user what to do.* If you can determine the corrective actions which the user can take, you may as well let the user in on your little secret.

6) *Satisfy curiosity*. Users will try to learn your software by just trying things out. You should let user satisfy their curiosity without causing problems - for example, by providing an undo feature. Users generally look at the About box out of curiosity (or to find the version of the software). Here is a radical idea for About box: use it to tell the user what your program is about!

7) *Follow standards.* Following standard user interface conventions makes life simpler for the user. For example, which side of the road to drive on is a user-interface convention? It really doesn't matter if people drive on the left or the right hand side of the road, but it is awfully convenient if everyone drives on the same side of the road. In the Windows world, user-interface conventions are set by Microsoft.

III. **EIGHT GOLDEN RULES OF INTERFACE DESIGN**

A number of advocates of user-centered design have presented sets of "golden rules" or heuristics. While these are inevitably 'broad brush' design rules, which may not be always be applicable to every situation, they do provide a useful checklist or summary of the essence of deign advice. It is clear that any designer following even these simple rules will produce a better system than one who ignores them.

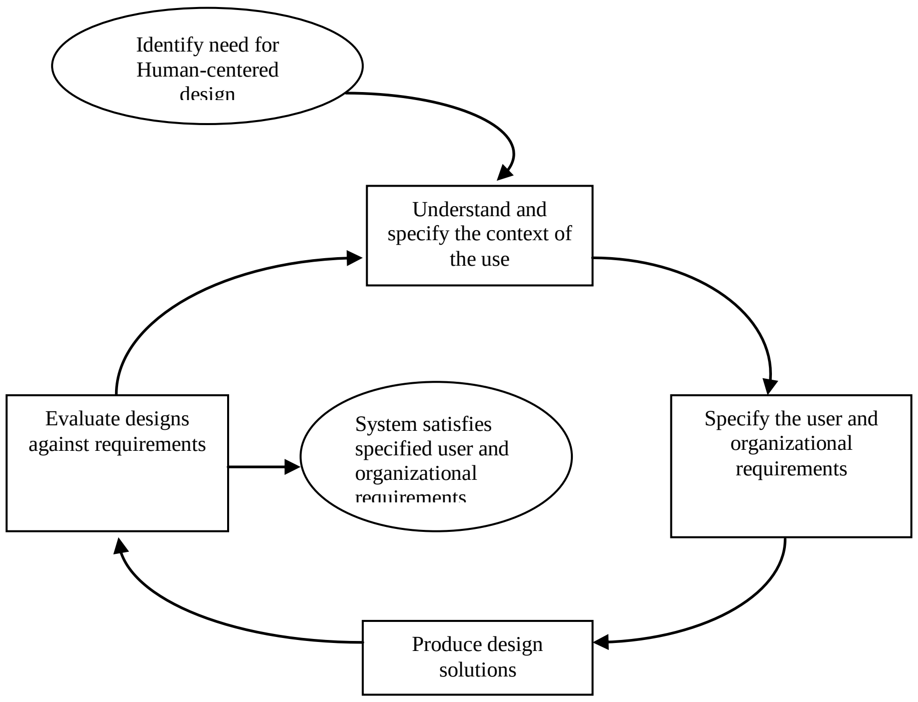
Shneiderman's eight golden rules provide a convenient summary of the key principles of interface design. They are intended to be used during design but can also be applied to the evaluation of systems. These are:

1) *Strive for consistency* in action sequence, layout, terminology, and command used and so on.

2) *Enable frequent users to use shortcuts,* such as abbreviations, special key sequences and macros, to perform regular, familiar actions more quickly.
3) *Offer informative feedback* for every user action, at a level appropriate to the magnitude of the action.
4) *Design dialogs to yield closure* so that user knows when they have completed a task.
5) *Offer error prevention and simple error handling* so that, ideally, users are prevented from making mistakes and, if they do, they are offered clear and informative instructions to enable them to recover.
6) *Permit easy reversal of actions* in order to relieve anxiety and encourage exploration, since the user knows that he can always return to the previous state.
7) *Support internal locus of control* so that the user is in control of the system, which responds to his actions.
8) *Reduce short-term memory load* by keeping displays simple, consolidating multiple page displays and providing time for learning action sequences.[2, 5, 6]

IV. PROCESS OF GUI DESIGN

In figure1 shows a simple view of GUI design process.



Identify need for Human-centered design
Understand and specify the context of the use
Specify the user and organizational requirements
Produce design solutions
Evaluate designs against requirements
System satisfies specified user and organizational requirements

Figure 1

The first stage is establishing what exactly is needed. It is usually necessary to find out what is currently happening. Identification of needs performed through observation and interviews. The result of observation and interview needed to be ordered in some way to bring out key issues and communicate with later stages of design. Design is a central stage, move from what you want to how to do it. There are numerous rules, guidelines and design principles that can be used to help with this. Humans are complex and we cannot expect to get designs right first time. We therefore need to evaluate a design to see how well it is working and where there can be improvements. Most user interface design involves some form of prototyping provide early versions of system to try out with real users. Finally when we are happy with our design, we need to create it and deploy it. This will involve writing code perhaps making hardware, writing documentation and manuals everything that goes into a real system that can be given to others.

IV. CONCLUSION

In this paper we have reviewed that designing GUI is not just about creating software but instead is about the whole interaction between people, software and their environment. The GUI design process starts with understanding the situation as it is and the requirement for change. We have seen how GUI design rules can be used to provide direction for the GUI design process.

REFERENCE

[1] White, E.A., Wildman, D. M., and Muller, M.J. Practicum in Methods for User Centered Design. Workshop at Human Factors Society 35th Annual Meeting (San Francisco CA,1--4 September 1991).

[2] Shneiderman, Designing the user interface, 3rd edition, Addision-wesley, 1998.

[3] Alan Cooper, The essential of User Interface design (IDG Books worldwide, Inc.1995).

[4] Brenda Laurel, Art of HCI design (Addision- wesley, Pub. Co 1990).

[5] J.Mayhew, Principles and Guidelines in Software User- Interface design, Prentice Hall, 1992).

[6] Alan Dix, Janet Finalay, Gregory D. Abowd, Russell Beale, Human Computer Interaction, 3rd edition, pearson.