GUIDELINES FOR DEVELOPING A GOOD GUI

I. Antcheva, R. Brun, F. Rademakers, CERN, Geneva, Switzerland

Abstract

Designing a usable, visually-attractive Graphical User Interface (GUI) is somewhat more difficult than appears at first glance. It adds quality to the application only if it reflects the needs and capabilities of the users within the hardware and software constraints. The basic guidelines that constitute an intuitive and good user interface are discussed in this paper.

HUMAN CONSIDERATIONS IN USER INTERFACE DESIGN

Complex relationships exist between people involved in application design and development, and their end-users. Every group has a comprehensive view of aspect of the application goals, as well as the steps that have to be taken to meet successfully the application requirements.

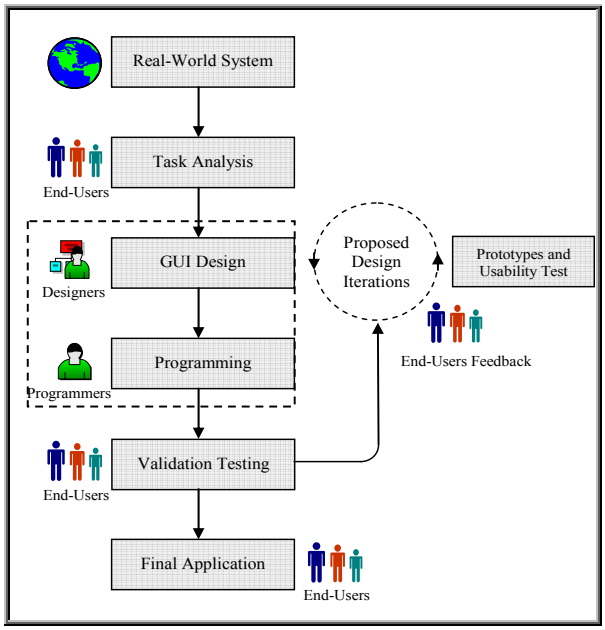

Figure 1: GUI Development Process

To simplify these relationships three generalized models of GUI are defined according to the human roles in the GUI development process: the mental model of users, designers and programmers (see Fig. 1).

Users

The user's mental model is the user understanding of how the application parts work together. It consists of static and dynamic elements. The static elements are user's previous experience, education and knowledge of computer systems. They do not change when the user gains experience with the GUI but they form the base of the dynamic elements that evolve as a spiral:

- The user's perception of the real world system represented by the application.
- The user's formal training with GUIs.
- The user's experience with GUIs.

The primary goal of user interfaces is to ensure that user's mental models are developed correctly in the positive direction of the spiral [1,2].

Key design principles related to this goal are listed below:

Simplicity – only the basic functions are shown in the main application window and the interface is kept simple. The advanced functions, less obvious for new users, appear only if the task analysis requires them. Complex GUIs are split into discrete units, organized according to the user's expectation and needs. The interface simplicity determines whether the application will have a single or multiple document interfaces.

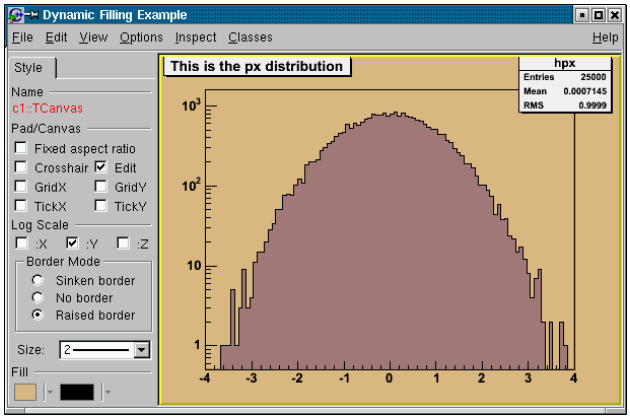

Figure 2: The novice users observe the cause-effect relationships in the application

Consistency – the GUI is transparent, predictable and easy to use; users focus on tasks they perform, not on how the GUI works. Users develop their knowledge gained from experience and use it consistently throughout the interface. The same things work the same way and there is no need to learn something new when performing similar tasks.

Freedom – control is in the user's hands. Users have freedom of movement and freedom of choice that encourages application exploration and experimentation, and enhances development of the user's mental model. The current application shows an appropriate set of

interface elements; potentially dangerous user errors are blocked.

The application supports power users giving them the possibility to make their tasks quickly. These users are permitted to modify parameters directly and to use keyboard shortcuts.

Users can leave the application for a moment or a day and can find the same state when they return.

Direct manipulation and feedback – when operating with the GUI elements, users observe the cause-effect relationships between their actions and application outcomes. Immediate feedback has the key role because it makes users feel confident to explore the application. When they do something, they can view the result and undo the action if it is unacceptable. A general rule to follow here is to make the user action independent, not bundled with another action. Only information relative to the user's task is presented on the screen ensuring that everything that must be compared is visible at the same time. Users need to be informed for any operation longer than 5 seconds (e.g. changing the mouse cursor, progress bar indicators or a message dialog).

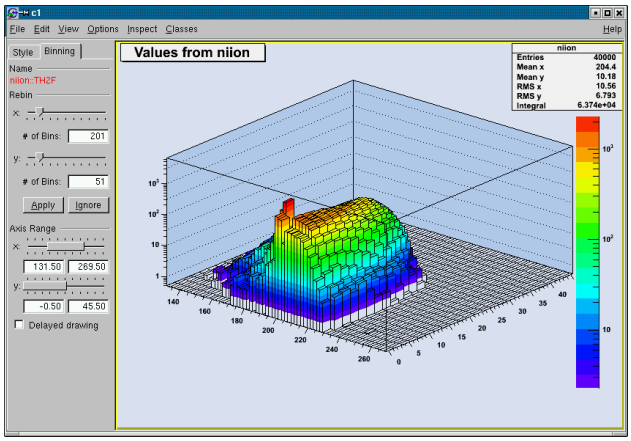

Figure 3: The advanced users are able to map GUI paths to the task flow

Forgiveness – the GUI should protect the applications from user mistakes when an inappropriate or dangerous step is made. Even if users try to make the right choice; incomplete mental models can lead to wrong steps and erroneous actions.

Intuitiveness – whenever possible, the real objects are presented adequately in the software and all users understand them. All GUI elements and metaphors in use have an intuitive 'look and feel'. They are arranged and ordered to reflect the natural visual scanning: from left to right, from top to bottom. Any visual element that does not contribute directly to the visual communication is hidden. GUI layouts create a visual hierarchy, drawing attention to the most important first. User actions and tasks lead naturally from one sequence to the next. The trash can icon on any desktop environment is a simple and very good example of how this principle works.

GUI Designers

The role of GUI designers is to determine what the application will look like and how it will work finally. For this purpose a number of complex, arbitrary, and sometimes conflicting requirements should be broken down into a sequence of user's actions. These actions need to be classified according to their priority, sequence, and structure of the performed task. Lastly, the designer's role is to present them in an intuitive, memorable and visually pleasant way, using a set of window's gadgets (widgets), layout managers and graphics design principles.

The main factors determining the mental model of GUI designers are listed below:

- Knowledge of real-world system requirements and task analysis that should be applied to it.
- Understanding users.
- Visual presentation and interaction abilities.
- Ability to meet programmer's needs.
- Ability to use metaphors.

As a real bridge between users and programmers the GUI designers must have a broader view than both groups. The key GUI design principles listed for users can be completed with the next group related to the interface look and feel:

Presentation – decide whether the application has a single or multiple document interface, and accord the application composition and layout to that. Complex GUIs should be split into discrete units that are organized according to the user's expectation and needs.

Power and flexibility - only information relative to the user's task should be presented on the screen ensuring that everything that must be compared is visible at the same time. Possible ordering schemas of GUI elements may include the importance, sequence or frequency of use. Everything depends on how the GUI elements are organized; what metaphors are in use, which part of the GUI is visible and which is hidden; how do users get help and what documentation is provided.

Alternate interactions - each interaction device must be optimized and users are able to make their choice as individuals with different abilities. The GUI should allow different ways of interactions. A complete and equal keyboard and mouse support gives the possibility to switch between them when performing a single interaction. Including accelerator keys for repetitive actions enhance the efficiency of any application and the user's productivity.

Color assignments and their usage – colors are a powerful tool for informing users, to alert them to a critical situation, to insert additional information or to underscore relations between objects on the screen.

Clarity – use up to three fonts and a list of reserved words with a clear meaning; provide short error messages and on-line help tool tips.

Programmers

The programmer's role is to convert a detailed GUI design plan into a fully working user interface. The conceptual user tasks, discrete GUI elements, and the hidden functional parts of the software are put together into the application to serve the user needs effectively.

The key elements that make up the programmer's mental model are:

- An accurate and complete understanding of the designer's model.
- Understanding the specific requirements of the real system.
- Knowledge of GUI programming and interactions.
- Hardware, platform specific and operating system knowledge.
- Networking protocols.
- Inter-application communications and links to the external code, libraries and programs.

The key design principles created on this level are:

Responsiveness – the speed of the application is a determining factor of the application acceptance. There are different ways for giving the appearance of speed: avoiding screen repainting unless it is absolutely necessary; providing complete and equal keyboard and mouse support; including accelerator keys for efficient access to repetitive actions.

Table 1: Using Widgets

Widget	**Number of Choices**
Menu bar	Maximum 10 single word menu titles in one line
Tool bar	Maximum 15 buttons
Popup menu	Maximum 12 entries
Cascaded menus	Up to 5 entries, 1 level deep
Context menu	Maximum 10 entries
Text button	Maximum 6 per dialog
Check button	Maximum 10-12 per group
Radio button	Maximum 6-8 per group
List box	10-20 displayed entries
Text button	Maximum 6 per dialog

Appropriate dialog use – use modal dialogs (e.g. File Open dialog) for presentation of finite tasks; modeless dialogs for presentation of ongoing tasks (e.g. Search dialog).

Use widgets correctly – a revision of the GUI widgets in use should be made at the programming stage. Table 1 shows basic guidelines that ensure user's comfort in the application. As soon as the behavior of listed widgets is changed, there is a risk that users will feel lost [5].

USER INTERFACE USABILITY

The 'Look'

The user interface look is important to the initial acceptance of any application. Three components form the visual presentation of the GUI elements and create the visual language for users: typography, layout and color.

The *typography* deals with the text readability of the application interface. The summarized key principles for good typographic design are listed below:

- Use a maximum of three different fonts and three different font sizes in an application.
- Avoid use of all capital letters because users read mixed case much better.
- Limit text lines to between 40-60 characters to enhance the readability.
- Use appropriate formatting, like text fields aligned to the left, number fields aligned to the right and centered text in lists.

The *layout* creates the symmetry and balance of the application. Techniques for screen layouts include the use of horizontal and vertical alignments for logical grouping of interface elements. This allows users a better understanding of the relationships between objects and facilitates the development of the user's mental model.

The correct use of *color* can only add benefits to the application usability. The main guidelines are as follows:

- Use a maximum of seven colors to avoid "visual chaos" in the application.
- Use color for group related items.
- Any color keeps the same meaning throughout the interface, e.g. to draw the user's attention.
- Make the foreground color different in brightness or hue to the background color to improve the contrast.
- Avoid use of blue for text and small objects.
- Use contrasting colors to express differences.

The 'Feel'

The feel describes the interaction behavior of the visual interface elements. It defines paths mapping the GUI tasks to the data structures.

Any application provides an obvious starting point in the upper-left screen corner and the most important and the most frequently used GUI elements are located at the top left.

The ‘look’ and ‘feel’ are closely related because the appearance of any GUI widget gives users clues how to interact with it.

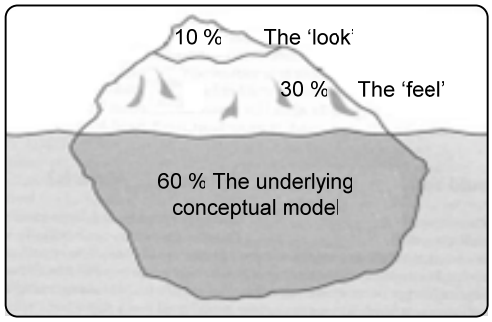

Figure 4: The Iceberg Metaphor

The Conceptual Model

The user’s conceptual model of how an application works is formed upon his or her mental model of the application, i.e. the model of how the application parts work together and how the user can combine them to map his or her goals to the available application capabilities. This model is based on application objects and their relationships, on how they behave when users interact with them, what metaphors are used to apply knowledge about the whole structure of objects and tasks that can be performed [3,4].

It is very important to have users represented from the very early stages of design. To understand who they are in terms of their skills and experience, and the tasks they perform, we classify them the following way:

Novices (for a short time) are users with theoretical understanding, but no practical experience with the application. They are impatient with learning concepts; patient with performing tasks.

Advanced beginners are users that focus on a few tasks and learn more on a need-to-do basis. They perform several given tasks well. Many people remain at this level.

Competent performers (fewer then previous class) – they know and perform complex tasks that require coordinated actions; interested in solving problems and tracking down software errors.

*Experts (identified by other*s) have ability to find a solution in a complex functionality. They are interested in theories behind the design; also in interacting with other expert systems.

The user-task knowledge is very important in the process of modeling the application concepts, as well as the application look and feel. User models are the foundation of any application design [6].

CONCLUSION

The presented design principles will have effect only if user’s tasks and requirements are well analyzed and understood. The application’s interface will lead to efficient and satisfying results only if users take their central place in the design process.

ACKNOWLEDGEMENTS

This work has been supported by PPARC (Particle Physics and Astronomy Research Council), Swindon, UK.

REFERENCES

[1] http://www.well.com/user/smalin/miller.html
[2] http://www.asktog.com/basic/firstPrnciples.html
[3] www-106.ibm.com/developerworks/library/w-berry/
[4] J. Hackos, J. Redish, User and Task Analysis for Interface Design, John Wiley & Sons, 1998
[5] W. Galitz, The Essential Guide to User Interface Design, John Wiley & Sons, 2002
[6] R. Eberts, User Interface Design: Prentice Hall, 1994.