Use & Misuse of Principles of Usability: Examples in Commercial Applications

Hooman Baradaran
Department of Computer Science
York University
Toronto, ON
cs213240@cs.yorku.ca

ABSTRACT
Principles of usability are defined in three categories of learnability, flexibility and robustness where each category includes several principles to support usability of applications. It is easy to find examples of applications that apply this rules, both correctly and incorrectly. Consistency of interfaces and adaptivity of toolbars in OpenOffice show correct application of learnability and flexibility while bad choice of defaults in some cases show incorrectly application of robustness. Mozilla Firefox supports flexibility in many parts of the application while the inconsistency of interfaces between platforms shows an incorrect application of learnability. Windows Media Player incorrectly applies flexibility by allowing novice users to select completely different interfaces that are very hard to use for them.

KEYWORDS
Usability, principles, learnability, flexibility, robustness, consistency, Firefox, OpenOffice

INTRODUCTION
The *principles to support usability* as defined by Dix et al[1] are the more general and abstract rules in design of software applications. These rules are applied to many commercial software applications, however, this does not mean that they are always applied correctly. I will look at a few examples of correct and incorrect application of each of these principles. One of the applications considered is Mozilla Firefox, the next generation browser from the Mozilla foundation that now has the largest market share after Microsoft's popular Internet Explorer, while at the time of this writing, it is still in the final stages of preview testing. The other application considered is OpenOffice.org, the open source productivity suite commercially sold as StarOffice Suite by Sun Microsystems. Since StarOffice and OpenOffice have exactly the same interface, and since this paper is written using OpenOffice, from now on I will refer to these sister applications as OpenOffice. Sun has always been an example of a company that applies usability to the applications and OpenOffice is one the open source examples of such applications from Sun[2]. The reason for this selection of applications is that web browsers and desktop publishing applications are the two most popular software applications for novice users. You can hardly find a personal computer without Internet access and a web browser and if you do find such a system, chances are it is used for desktop publishing. The other application considered is Windows Media Player which also falls in the category of applications used by most novice users.

PRINCIPLES TO SUPPORT USABILITY
Principles of usability are general and abstract rules that can be applied, in repeated occasions, to the design of software applications. These fall into three categories of learnability, flexibility and robustness where each category includes several principles to support usability for the novice and even advance users.

Learnability
Learnability deals with initial understanding of the system by the novice users as well as achieving a maximum performance once they have learned how to use it. Novice users of a software application should be able to get some level of *familiarity* with the system based on other real world or computer systems and the application has to be *general* enough with respect to other applications and situations to allow the user to apply the interaction techniques from similar situations. The applications must be *predictable* and show the effect of future interactions based on the previous interactions and also *sythesizable* to allow the user to assess those future effects based on the current sate of the system when there is no history of previous interactions. Above all the others, there must be a lot of *consistency* within the system and with respect to other systems. Consistency is so important that it should have been given a category of it's own. The examples chosen in this paper are the software applications used most often, specially by novice users, and learnability is very important for such applications.

Flexibility
Flexibility means that there must be more than one way for the user and the system to exchange information during the interaction. When dealing with *dialog initiatives* the application must avoid *system pre-emptive* dialogs, those initiated by the system, and allow *user pre-emptive* dialogs as long as the number of of user-initiated actions are not too much for the user to lose track of tasks started. Humans need to do several tasks at a time and the software application should support this *multi-threading* and allow multiple tasks at a time but restrict the communication at any given instance to a single task; the tabbed-browsing feature in most browsers is a good example. *Task migratability*, the ability to transfer the control of task

between the user and the system, as well as substitutivity, the ability to substitute equivalent input values, are two very important principles of flexibility specially for applications such as desktop publishing suites. And last but not least, the system should be *customizable* and allow both the system or the user to change the interface based on the context of the task in hand if necessary.

Robustness

Robustness as a principle of usability is the robustness of a system to help achieve a user's goals; this is not to be confused with robustness from the software engineering point of view, which is also equally important in an application. The interface of the application must be *observable* and allow the user to evaluate the internal state of the system given that limited view on the screen. This includes *browsability* of the internal state, setting *defaults* for choices given to the user, *reachability* of any state from another, *persistence* of information given to the user and *operation visibility* to make the choice of actions to be performed visible to the user; which is also a factor for predictability of the system. The application must allow *recovery* from both system and user errors alike; this is also another principle that is extremely important for productivity suites and similar applications. *Responsiveness* of the system is better when response time is short and unchanged for the same situation. An application is built to support a user's *task* and must be successful to allow the users to achieve their goals; this includes new tasks that are introduced due to the use of the software application itself.

THE USE

There are many example of correct use of the principles of usability in both Mozilla Firefox[3] and OpenOffice. I will give one detailed example of the correct use of these principle for each of the described categories.

Learnability

Consistency, probably *the* most important principle of learnability, is very well applied to the interface of OpenOffice in general. Not just OpenOffice keeps the same consistent interface within the application and across every platform it is built for, but also the interface is very similar to other office suites. While many users use office suites for the first time, there is a very large number of novice computer users who use office suites on daily bases. OpenOffice is used in many places as an alternative to Microsoft Office either because of the cost or because it runs on more platforms; for this reason it is extremely important to have consistency with MS Office or other office programs and for the most part OpenOffice is very consistent. Certain thing are of course different but that is a reasonable choice as OpenOffice is easier to use in many ways. In comparison of OpenOffice on multiple platforms, namely Linux and Windows, the interface is exactly the same except the toolbar items in Linux are larger and clearer while the Windows version, perhaps due to limitations of the SDK used, uses the default toolbar icons from Windows. Within the application, all similar functions are very similar and consistent; for example formatting of objects, from a single character to the whole page, is done via similar interfaces that are accessed either via the "Format" menu or the right-click context menu. In fact, formatting of sub-sections of a page like columns and headers are all access via tabs in the same window as page formatting itself.

Flexibility

OpenOffice is also a very good example of correct application of flexibility. It is *adaptable* as it allows the user to completely modify the toolbars; both the usual toolbar on the top and the toolbar on the left, exclusive to OpenOffice, which is used for most frequently use functions; or create macros, like other similar applications. But it is also very *adaptive* as it automatically changes the toolbars and menus based on the context. For example while editing text, the toolbar for font and paragraph settings is visible, but as soon as you click on an image or another object the toolbar changes. This also makes the toolbar icons easier to access as there are only two dynamic toolbars visible, rather than five or more toolbars where most of them are hardly ever used. This can be ambiguous, for example when selecting an image which is inside a frame the user may want the image toolbar to change contrast of the image or the frame toolbar to change the alignment. To avoid the ambiguity, the toolbar for the item on top of others - the image in this example - is initially shown and the user can change the toolbar by a right-click. The subsequent times the same group of object is selected the last type of toolbar selected is shown. Another example is editing formulas in a text document: after selecting a formula, the interface changes to that of OpenOffice Math, the formula editor module of OpenOffice to allow easy insertion of symbols. There are also more familiar examples of flexibility in OpenOffice, for example *substitutivity* of units of measurement is supported and *task migrability* is supported in the automatic spell checker that underlines incorrect words and allows the user to correct them.

Robustness

Mozilla Firefox is meant to be very easy to use for completely novice users and has many useful *defaults*. For example when downloading files, all files are downloaded and saved to the desktop by default; this can be changed to save files to another pre-defined location, or ask for a location for each download. Default location is a great feature for novice users because they do not need to know about the hierarchical file system and downloaded files simply show up on the desktop where the user can see them. The download manager in Firefox is very *persistent* and shows a list of files being downloaded as well as those that have finished downloading, giving an option to either open or remove each file. Firefox, like other popular browsers is a clear example of *reachability*: user can go back and forth between web pages one by one using the forward and back buttons or jump to a page by using the little menus next to these buttons. There is a "Go" menu which shows these pages as well as an item to open the history sidebar for a larger selection of previously visited web pages.

THE MISUSE

Unfortunately applying the principles of usability does not guarantee usability because these principles can be applied incorrectly. Ironically, both applications described as examples of correct use of these principles also use them incorrectly in some occasions.

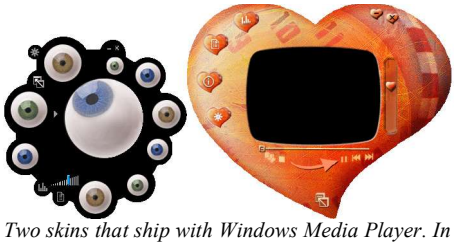
Learnability

While Firefox is my favourite example of good user interface design[3] and the developers spend a great deal of effort on making sure this browser is simple and easy to use for novice users, in a few situations *learnability* is incorrectly applied to the Linux version of Firefox. This version follows the usability guidelines for GNOME desktop environment to keep Firefox consistent with applications of this environment, however, this breaks the consistency with Firefox on other platforms as well as applications on other desktop environments in Linux. For example in this version, to change the settings the user must choose "Preferences" under "Edit" while in the Windows version to perform this, "Options" under "Tools" must be selected. Another example is having the "Cancel" button on the left side of the "OK" button. This is inconsistent with the other versions of Firefox, as the figure shows, both by incorrect position of buttons with respect to each other and different alignment of buttons on different operating systems. This problem has not gone unnoticed and many developers have complained about it, however, due to the open source nature of Firefox it was decided to keep following these standards and allow modified versions to be released by other developers.

Authentication dialog box in Firefox: Linux (left), Windows (right)

Flexibility

A very common example of incorrect use of flexibility is the option to change skins in Windows Media Player or similar media players. In the older versions of WinAmp, a famous MP3 player for Windows, the users could change the skin, the look of the player, while keeping all the controls in place. Unfortunately, the future versions of WinAmp as well as Windows Media Player and many other media players went too far and allowed the user to use skins that would completely change the interface. These skins would look beautiful, however, it was absolutely hard to find the controls even those used most often. In this case *adaptability* cost the application its learnability. In Windows Media Player these alternate interfaces mimic real world examples too closely and are more like "interactive images" than interfaces. In many media players these skins do not come with the main program, but in Windows Media Player for example, these alternate interfaces are included by default in the program and some even chosen as the default interface when a new version is released. It is less of an issue when the alternate skins are not included with main program because only users who want this option, younger users in my experience, can *choose* to download and use them. However, in Windows Media Player when the users want to change to the mini view, they have to pick one of these alternate skins and having no prior experience, they pick the one that looks best rather than the one that is easier to use.



Two skins that ship with Windows Media Player. In either skin some control buttons are hidden.

Robustness

Perhaps *static defaults* are the most incorrectly used principle of robustness. Many applications have set defaults that are not what the majority of users would want as the default. Unfortunately OpenOffice is one of those programs and the most obvious example is spell checking not being active by default. OpenOffice has the option to use the normal find and replace spell checker as well as the auto spell checker that underlines incorrect words as you type. The problem is not that the auto spell checkers is not the default, but it is that no spell checking is selected by default at all. Both spell checkers exist in the left toolbar and are easy to select, but many times I forget to select the auto spell checker and it is only after typing a number of pages that I realize the spell checker was off. Another annoying default in OpenOffice is the auto capitalize feature that capitalizes the first letter of a sentence. When using Draw, the flowcharting module of OpenOffice, the user may need to enter a single word in lower-case in the diagram but single words are considered sentences and the first letter will be capitalized. The default is to have this option enabled and let it repeat itself no matter how many times the word is changed back to lower-case.

CONCLUSIONS

Principles of usability, although very general and abstract, can greatly benefit a software application and following these principles has greatly improved the usability of many applications, however, if these principles are applied, it is extremely important for the designers to make sure they are applied correctly; otherwise using them can do more damage than good.

REFERENCES

[1] Dix, A. J., Finlay, J., Abowd, G., Beale, R. Principles to support usability, In *Human-Computer Interaction*, 260-273, Third Edition, Prentice Hall, 2004

[2] Benson, C., Matthis, M.P., Mzourek, J. Professional usability in open source projects: GNOME, OpenOffice.org, NetBeans. In *Extended Abstracts of CHI 2004*, 1083-1084. ACM, 2004

[3] Baradaran, H., A Review of Firefox for Newbies, OSNews, URL: http://www.osnews.com, [September 15, 04]