3 CHI Workshop Position Paper Workshop on "The Many Faces of Consistency in Cross-platform Design"

Chip Alexander, Experience Design Group, Sun Microsystems

In this position paper, I will be addressing two questions from the Workshop Summary:

- What aspects of consistency matter in cross-platform design? What evidence has been gathered to inform us? What are the biggest unknowns?

- What are the unique problems for ensuring consistency in application interfaces that span multiple platforms?

INTRODUCTION

First, a bit about myself as background. I have over 19 years of experience designing user interfaces and leading user interface design teams. Most of it has included defining overall style for a product line and making guidelines, at ASK Computers, at Ross Systems, for about 6 years at Oracle Corp., at a startup Bidcom Corp., and for the last 4 years as an architect at Sun Microsystems -- where I focus almost exclusively on the overall style of Sun products.

As both a consumer and someone passionate about usability design, I am constantly evaluating usability issues across different types of products -- PDA's, automotive controls, phones, and of course "full size" computers and applications running on them. In particular, I find that in my avid use of a PDA, I have a great tool for also studying how user experiences can migrate among vastly different platforms. This is especially true when similar applications are available on both, such as the Palm Desktop vs. PDA versions of the calendar, address book, etc. and the different versions of web pages on a full-size computer vs. a PDA.

One of my first projects at Sun was the Java Look and Feel: Advanced Topics book published by Addison and Wesley, as one of the two lead designers, which put me firmly in the fat-client world of design and its constraints. Since that time, I have transitioned to focus particularly on the overall look and feel of Sun's web applications, but have done UI reviews of products across many platforms on Sun's UI review board.

Further, as one of the senior staff in the Experience Design group at Sun, I coordinate with the branding team and with designers working on projects on other platforms (desktop apps, thick client apps like Java-based ones, web apps, etc.). My experience in the branding aspects has particularly expanded in the last year, as I have become a brand representative within the software group, and participated in an international conference specifically on corporate branding issues.

Based on all these experiences, I believe the choice of where to follow native look and feel vs. where to have a consistent experience across platforms that may also express the corporate brand is a complex one which cannot be addressed with a simple overarching rule. Instead, it depends on the specific element of the user interface. Below, I've listed a few categories of elements that require varying degrees of consistency, and some examples of each.

ELEMENTS RESTRICTED TO PLATFORM-CONSISTENT LOOK AND FEEL

The first category is the set of widgets whose appearance and behavior are both restricted, usually

by platform guidelines, so cannot follow a consistent branded look of the company creating them.

Buttons are a great example of this. While designers can make buttons which look different as part of expressing their brand, the primary button (such as OK on a dialog box) needs to always be on the right on the Mac and in Gnome, and needs to always be on the left in Windows, Palm, most phone applications, and many other platforms. Doing otherwise will lead to users rejecting the product as being simply wrong, or clearly ported without consideration to a new platform, and will definitely not lead to a good impression.

Moreover, if the visuals are changed, they need to be changed significantly, to clearly show a brand. If buttons look like they are a small deviation from Windows buttons on a product running on the Mac, many Mac users will reject it as not truly a Mac product. So the designer either needs to use native buttons or go to something dramatically different in order to have the buttons seem acceptable on all platforms.

Another critical requirement is to observe the needs of those with physical challenges who need accessibility aids, particularly on "full sized" computer platforms. Many of these homegrown elements seem to just ignore such needs, and leave an important segment of their user groups unable to use the application as a result. It can also lead to legal issues and product acceptance criteria issues, particularly with government-purchased software.

A further set of elements that fall into the category are those restricted to be the native widget due to security issues. For example, to open a local file on a PC, one needs to use the Windows Local File Selection dialog. Accessing local files without doing so is difficult or impossible, due to security issues. This forces these elements to follow the native look and feel, no matter how much it contrasts with a branded look and becomes inconsistent.

ELEMENTS THAT CAN CHANGE LOOK BUT NOT FEEL

The second category of elements is the set of standard, basic widgets that can change appearance for branding, but should never change their behaviors.

A great example of this is scroll bars. While most people would have no objection to an application which changed some appearance aspects of a scroll bar such as its color, what I often see instead in "branded" scroll bars is something quite foreign that was made from scratch to look unique and stylish to the designer or developer's preference. What I then typically find is that the control covers the basics such as scrolling the page if you click the arrows. But some of the more subtle behaviors I have come to expect, such as clicking in the scroll bar to move by a page, dragging the elevator to move quickly, or even scrolling from the keyboard, all don't work correctly, or at all.

Had these scroll bars carried all the behaviors I reasonably should expect as a user, not just the fundamental up and down, a visual re-branding of the scroll bar would have been fine (presuming it is easily recognizable as a scroll bar of course). But often these things are overlooked in favor of just making something new and cool. And of course, like the first category, all the standard behaviors must still work if they are customized, including keyboard access – both for accessibility and to satisfy non-challenged users accustomed to operating them that way.

As a contrasting example, I have multiple times seen re-branded check boxes that work just fine, on both full-size computers and handheld PDAs. They often have a fun unique look that gives character to the site and furthers its unique branding. But there is no question they are check boxes, and carry all the behaviors users expect, down to clicking on the label to change the value, or tabbing to them and using the keyboard to change the value. Creating a new custom element even allows for creating additional states beyond what is provided by the native widget, such as showing the values are mixed for the selected set ot items. But again, this is acceptable because all the standard behaviors work too.

Additionally, some behaviors are larger than individual elements, and absolutely must follow the standards of platform rather than be consistent for a company. A great example of this is the 'Save' methodology. The Palm platform's Save model is one of everything you do (save for in a brief dialog box) being saved without specific user request. If a company introduced a Palm version of their Windows application and included the Windows behavior that your work is lost if you shut the device or close it without saving, users would quickly stop using the product as they would lose work when following the normal behaviors they have come to rely upon on that platform. Yet even the Palm

Desktop which comes with a Palm PDA follows the "save it or lose it" methodology, and works fine, as users expect that behavior on the Windows platform, so are not hurt by it.

ELEMENTS THAT CAN INNOVATE AND DEVIATE IN THEIR LOOK OR FEEL

The third category is those elements that are collections of small widgets into more complex elements. These seem to me to be great areas to create a consistent product-specific design that improves the user experience.

A great example of this is the wizard. There are of course some basic elements users expect in a wizard, such as Previous and Next buttons to guide them through. But beyond those basics, this is a great place to create easy-to-use designs, innovate, and express branding.

In the Java Look and Feel, for example, we have wizards that include tabs on the left panel of the wizard to choose between viewing the set of steps or help about the current step. Another nice feature is a Last button to allow the user to optionally jump to the summary step once the minimum amount of information has been entered. Such features can certainly be used across platforms without issues, and the innovations can become part of the company's brand and even their competitive advantage without problems. This again is possible because instead of being a small basic component, it is an assembled element and can therefore take on personality without causing problems, so long as the elements inside follow the standard rules of the platform.

Another good example of branded design is the choice of graphic style used in an application. When Yahoo introduced their My Yahoo Portal for PDAs and cell phones, the page had to be completely redesigned for the substantially smaller presentation space. But it was still immediately recognizable as Yahoo, not just from the logo but also from the appearance of the round News, Mail, etc. icons on the opening page. That visual design portion of Yahoo's experience can be carried across platforms of different sizes and helps convey brand and provide consistency, without a loss in the user experience.

ELEMENTS THAT DON'T FALL NEATLY INTO ANY OF THE ABOVE CATEGORIES, AND CAUSE ISSUES

The last category is of elements that don't fall into any of the buckets above. It includes elements like file hierarchy trees that can pose real challenges.

Trees are difficult enough elements that if the user does not see controls to which they are accustomed, they may not figure out how to operate the control. At Sun we have seen exactly that issue in usability testing our web application tree controls. In our web applications, we originally designed our own branded tree controls, which looked like circles with lines coming out to the right or down depending on state (some deridingly called them flushers). But in usability testing these applications, we found consistent issues with people not recognizing them as active controls that could be used to open branches.

We therefore switched to the standard triangles/arrows, which are used by Netscape and the Mac alike. But while the results improved, many of our usability testing subjects accustomed only to Windows still did not recognize these as controls to open branches. However, unlike buttons and checkboxes, simply going with the platform look and feel is not easy, as there is no native "tree control" to call. That leaves us trying to improve our control (which is our latest plan) or hand coding different trees for each platform, and copying the look and feel of that platform despite some legal concerns in doing so. I would be very interested in hearing how others have addressed the problems in this last category in their applications.

Additionally, not all widgets fall neatly into one or more of these categories. For example, are tabs something which can be branded and improved upon, especially given the real usability issues in the Windows model when more than one row of tabs is required? Or, like scroll bars, will users be confused when a standard behavior of a basic element is changed, even if for the better? This decision may also differ when moving from the full-size computer space to a PDA, as what fits within the screen can drastically change what is possible in such navigation controls. Dealing with such widgets which do not fall neatly into clear categories is another topic I would love to get input from

others on, as well as whether other designers agree with my categorization conclusions in the first place.

CONCLUSION

Many standard elements fall nicely into one of three categories – those which can change their appearance to innovate and express brand but are restricted to an expected behavior due to platform standards and user's behavioral expectations, those which are restricted by platform standards and user's behavioral expectations in both appearance and behavior, and those which are flexible for innovation and branding without such restrictions. I have given examples of each, and hope readers can use those examples to extrapolate to similar elements and situations.

However, the situation is not so simple on all elements, and some can cause real issues. There are some elements which users will be confused by if they do not have the standard appearance of the platform, but where providing those individual appearances can be challenging due to platform differences and no available platform-provided support to do so. Others can be challenging because which category they fall into is unclear and subjective. I hope that other participants in the workshop can help provide their own viewpoints to compare and contrast with mine, and that through discussion the workshop can help build consensus in those areas.