Homework Set 2
COSC552 HCI

Byron Heads
E00062946

December 9, 2010

A

Development House

A commercial development house can have the resources to hire personnel that are knowledgeable in the field. This gives them the ability to add knowledge in different areas. Many development houses also employ requirements engineers to gain knowledge of their potential user base. This level of detail can also lead to problems; software produced maybe too specialized and become difficult to maintain or adapt to changes in customer needs. When dealing with tool selection a development house often chooses to work with a specific technology set. This limitation may lead to using a language, library, or technology that may not be a good fit for the problem domain. Though this may have limitations, it can still be a positive situation for the development of specific software. The developers will know what their tool selection can do and how to do it.

When designing a user interface the development house often has in house design rules and guidelines that its developers follow. The large number of experienced developers will also be able to apply their own experience to the user interface design and to the software prototype. Though the development house may be able to design a consistent and usable interface, they rarely are able to develop outside of these constraints. This can be due to the libraries and tools used, or company rules and policies.

College Students

A group of college students will have less development experience then then the development house. This will lead to less effective design and code, but the college students are not limited to tools and technology that maybe used

at a development house.

College students will have less experience in user interface design and will draw design rules from their own experience in using other interfaces. College students will be focusing on the knowledge and experience they will gain by developing a large commercial project. Students often use trial and error when writing software. This will lead to more experimentation in the interface design.

B

There are many tool to use in designing a system to support users with different disabilities. Before adding these tools you need to have user controls that enable or disable different tools that depend on the user that is logged in to the system. This prevents the user from having to use the program's options menu to activate or deactivate the different tools.

To assist a deaf user the interface needs to ensure that any audible signal has an equivalent visual signal on screen. This can include flashing icons, shaking windows, or in important situations, a pop-up window can be used. These visual signals can be used with the audible signals for all users.

Users with manual dexterity problems can be helped with a tool that increases the size of on-screen controls and fonts. The can make clicking buttons, selecting menus, or selecting text easier. A system can also be deployed with a touch screen which may make it easier for some users.

The interface can also include tools to assist users with vision impairment by giving the user the option to increase the size of the font and controls. The interface can use specific operating system set color scheme, such as high contrast colors to help users. The interface can take advantage of these settings.

C

Visual consistency: All menus, windows, and controls should have a consistent look and feel to them. Continues across all minor versions of the interface.

Interface should respond quickly: Any long running or inconsistent time operations should be run on a separate thread with an interface control indicating the progress of the operation.

Operations should be stoppable: Operations should be able to be stopped and any changes should be rolled back to before the operation was started.

Confirm dangerous operations: Operations that cannot be undone should be confirmed with the user prior to being executed.

Do not crowd the screen: Do not fill the screen with a large number of controls or information. This can confuse or overwhelm the user.

Do not punish users for mistakes: Users will make mistakes. Design the interface to recover gracefully from these mistakes. This can include backups and saving or allowing users to edit information they have invested time inputting into the system.

Reuse as much as possible: Reusing interface menus and components makes the users more comfortable and increase the ease of use and memory retention of how to use the interface.

Use the users language: The interface should be in the users language. This includes language related to the field or business it is designed for.

Design for recognition and not recall: The user should be able to recognize how to use the interface and should not have to remember a set of complex steps to perform an operation.

Access to help and documentation: The user should be able to access a searchable help system from within the interface. The user should also be able to access documentation outside of the interface.

Help users understand and recover from errors: Errors should be express in a plain and understandable fashion. It should indicate the problem and give possible solutions.

Do not cause the user pain or stress: The interface should not induce pain or stress on the user.

Many of these rules and guidelines are useful in the proposed interface. One of the most useful is "do no punish the user for mistakes". This interface may require the user to spend time inputting data into the system. If the interface punishes the user for mistakes the company loses money due to redundant work, and the use will become frustrated and stressed from using the system. "Design for recognition and not recall" will be important to the proposed interface. It reduces the amount of training time, and reduces stress on the users which leads to more productive users.

D

The answer to this question is based on the paper *Small Business Storage Management and Gate Control System.*

The metaphor used for the storage site is based a 2D map of the facility. The map shows the location of all storage units, parking spaces, and the gate. Units on the map are color coded to give the manager a quick view of the status of the facility. Rented units and spaces would be in blue, renters that have not paid are in red, unrentable units would be in orange, and units that are ready to be rented are in green. The color coding can be configured by the manager. Each unit is marked with a number and the size of the unit. This screen also includes options to change the color coding based on other attributes. This can include unit sizes, how long a renter has had a unit, unit income over time, and how often a unit is late on payments.

The site map can be useful for many other operations. If a costumer wants to rent a unit the manager can quickly see on the map which units are ready to be rented. They can also view rentable units by size. The manager can then click on a unit and select "Rent out unit" from the pop up menu. From my own experience many customers forget which unit number they have, but often can remember where their unit is on site and can find it on a map. This can help the manager by keeping them from spending time searching for the customer in the database. The manager can right click on a unit on the site map and from the pop-up menu, select several options. These options include "Apply a payment", "Move out", "Contact customer", "Change Rate" or "View details".