Usability of Disabled Controls

Summary

The inappropriate use of disabled controls violates the principles and guidelines of good design and decreases the usability of a system.

What is Usability?

What is "usability"? The International Organization for Standardization (ISO) defines usability as, "(the) extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [1].

Furthermore, ISO defines "user interface" as, "all components of an interactive system (software or hardware) that provide information and controls for the user to accomplish specific tasks with the interactive system" [1].

Jakob Nielsen, a renowned usability expert, defines usability more succinctly as, "Usability is a quality attribute that assesses how easy user interfaces are to use" [2].

In summary, the primary goal of usability is to ensure that the information and controls provided to the user can be used to accomplish the user's goals with ease.

What is a Control?

The definition of "user interface" above divides all of the components of an interactive system into two categories: "information and controls". Components that provide information are non-interactive elements of the user interface that let the system tell the user something. Components that provide control are interactive elements of the user interface that let the user tell the system something.

What is Disabled?

The word disabled means that an ability has been lost or taken away. When referring specifically to a control in a user interface, "disabled" means that the control's interactive characteristics have been taken away so that the user cannot use the control. In most user interfaces, the control's visual characteristics are modified when the control is not interactive. The disabled state of a control is a secondary state that the control has. In his book "The Design of Everyday Things", Don Norman calls these states "modes" [4].

Disabled Control Usability

To summarize from the definitions above:

> **disabled** means something cannot be used; and
> **control** means a user interface component that a user uses to tell the system something; and
> **usability** means information and controls can be used to accomplish the user's goal with ease.

Now, let's discuss the usability of disabled controls, or, using the summarization above, let's discuss how controls — that cannot be used — can be used to help the user accomplish their goals with ease. Please read the previous sentence a few more times. After reading it, the conflict between disabled controls and usability should be glaringly obvious. Nevertheless, I will continue illustrating why I believe that disabled controls decrease usability.

I know that I'm being a bit unfair in the previous paragraph. I know that usability refers to the user interface and not just a component of it. Let's look at the role disabled controls play in a user interface and how it impacts usability. If a user interface is "all components of an interactive system that provide information or controls" then which of these categories does a disabled control fall into and how does it impact the usability of the user interface?

A control is used by the user to tell the system something. For example, a user could use a control to tell the system "I want to search for 'cute pictures of kittens'" or "I read the information you sent me and I am ready to move to the next step". Typically, these controls have a label that indicates what the control will tell the system like "Search" or "Next".

A disabled control cannot do these things because it cannot be used. If a disabled control cannot be used to let the user tell the system something, it is not a control at all. If it is not a control, what is it? Since there are only two categories of user interface components, it must be information. If it is information, what is the system telling the user with a disabled control?

Generally, a disabled control tells the user that the system may give the user control if the user does something else first (a precondition). When a disabled control is used, most applications fail to inform the user what the preconditions are. This leaves the user with a mystery to solve — there's a dead control on the page and the user must figure out how to bring it back to life. Providing the user with a mystery creates a confusing user interface. Confusion is contrary to the goal of an informational user interface component and the goals of good design.

In his book "About Face", Alan Cooper said, "Software should behave like a considerate human being" [5]. He goes on to liken a user's interaction with a system to a conversation with a considerate person. Don Norman also compares a user's interaction with a system to a conversation between two people when describing how a system should be designed for errors:

> "Consider a conversation between two people. Are errors made? Yes, but they are not treated as such. If a person says something that is not understandable, we ask for clarification. If a person says something that we believe to be false, we question and debate. We don't issue a warning signal. We don't beep. We don't give error messages. We ask for more information and engage in mutual dialogue to reach an understanding. In normal conversations between two friends, misstatements are taken as normal, as approximations to what was really meant. Grammatical errors, self-corrections, and restarted phrases are ignored. In fact, they are usually not even detected because we concentrate upon the intended meaning, not the surface features" [4].

Think about your conversations with rude people who interrupt you or silence you. Now think about your conversations with considerate people who listen and care what you have to say. Which conversations leave you with a good feeling (a good user experience)?

How considerate is a disabled control? It interrupts the user's flow by causing confusion. It silences the user by taking away the user's ability to tell the system something. It frustrates the user and makes the user feel inferior by withholding information — especially when the user is unable to solve the mystery of the disabled control's preconditions. A disabled control is not considerate at all.

A disabled control is a form of modal interface much like a modal dialog window. Alan Cooper defines modal as, "a special state that must be dealt with before it can return to its normal state and before the person can continue with her task" [5]. Cooper goes on to say, "interrupting the user's flow for no good reason is stopping the proceedings with idiocy and is one of the most disruptive forms of excise." He summarizes this design anti-pattern into a design principle, "Don't stop the proceedings with idiocy."

Because a disabled control is modal and, therefore a user interface containing a disabled control is also modal, it creates the opportunity for what Don Norman calls "mode errors". He writes, "A mode error occurs when a device has different states in which the same controls have different meanings: we call these states modes. Mode errors are inevitable in anything that has more possible actions than it has controls or displays; that is, the controls mean different things in the different modes" [4].

The mode error that can occur with a disabled control is that the user is unable to solve the mystery of the preconditions. If the disabled control is the only control given to the user in relation to the user's task, then the user cannot talk to the system. Therefore, the mode error is silent and the system is completely unaware of the problem. As a result, the user has lost visibility into the status of the system, the system has lost visibility into the status of the user, and the user has lost control — the system's usability has decreased.

To eliminate mode errors, Don Norman recommends removing modes from your user interface [4]. This aligns with Alan Cooper's design principle that considerate software keeps the user informed through the use of "rich modeless feedback about what is going on" [5] — the significant word being modeless, or without any modes. When this recommendation is applied to disabled controls, the recommendation would be to eliminate the disabled mode of a control which leaves you simply with controls that do not have a disabled mode.

Are disabled controls always a bad thing? In a section titled "Deliberately Making Things Difficult", Don Norman says, "some things are deliberately difficult to use— and ought to be" [4]. In this section, he describes scenarios where a good design intentionally makes things difficult for the user. In these scenarios, the goal of the deliberately difficult design is to protect people or things from harm.

Deliberately making something difficult to use makes sense in the physical world where there are often dire outcomes to actions. Outcomes that cannot be undone or reversed. In the software world, however, we have a much greater ability to avoid dire irreversible outcomes. As a result, the need to deliberately make software difficult to use is much less than in the physical world. In my many years of designing and developing web applications, I have not yet encountered a scenario where the use of a disabled control could be deemed appropriate.

By taking away the user's freedom and control, by taking away the user's voice, by withholding information and failing to keep the user informed, and by causing confusion, disabled controls fail to align with design principles and guidelines from leading usability experts. The inappropriate use of disabled controls decreases the usability of a system.

Bibliography

[1] ISO 9241-210:2010. Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems.

[2] Nielsen, Jakob. Usability 101: Introduction to Usability.

[3] Neilson, Jakob. 10 Usability Heuristics for User Interface Design.

[4] Norman, Don (2013-11-05). The Design of Everyday Things: Revised and Expanded Edition (p. 177). Basic Books. Kindle Edition.

[5] Cooper, Alan; Reimann, Robert; Cronin, David; Noessel, Christopher (2014-08-13). About Face: The Essentials of Interaction Design (Kindle Locations 6147-6148). Wiley. Kindle Edition.