Chapter 32

User Error Is Design Error

When people fail at using our products, we may be tempted to attribute that failure to user incompetence. It's a "user error," we may say. This response has its own acronym: "PEBKAC," which stands for "Problem Exists Between Keyboard And Chair."

When you do your first usability test, it's natural to experience some amount of denial—surely your product can't be *that* bad. You probably just happened to pick a few truly inept people to test it, right?

This tendency to blame users is often supported by users themselves, who also tend to blame themselves for problems with your product. When they are unable to figure out how to use it and you show them how, they will often respond with something like "Oh wow, I have no idea how I did not see this! It seems so obvious now!" They tend to blame themselves for not seeing something, rather than blaming your product for not making it obvious enough.

Blaming the user for your product's errors does not fix the error. If tens of thousands of people use your product, every problem you see in a usability test will be experienced by hundreds or even thousands of users. Rather than assigning blame, fix the problem. In his book *The Design of Everyday Things* [Nor88],[1] Don Norman puts it like this:

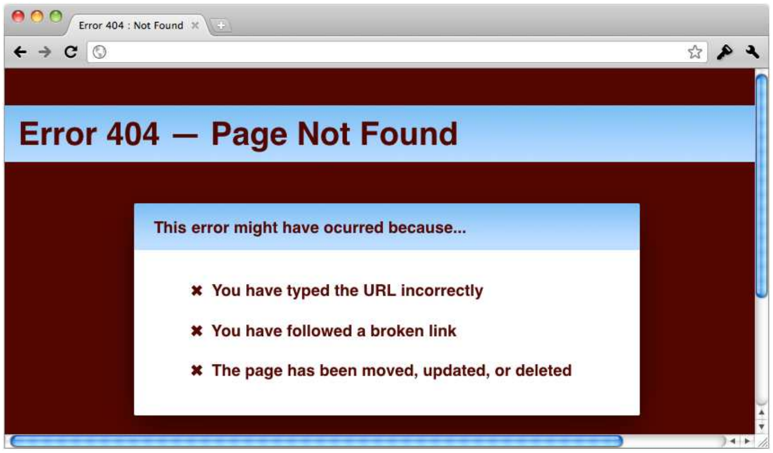
> Don't think of the user as making errors; think of the actions as approximations of what is desired.

People don't make errors. Your product makes errors when it doesn't interpret the user's actions correctly.

1. If you haven't read *The Design of Everyday Things* [Nor88] yet, drop this book right now and go pick up Don's book instead.

32.1 Don't Blame Your Users in Your Error Messages

We've all encountered an error message like this:



Error 404 : Not Found

Error 404 — Page Not Found

This error might have ocurred because...

✖ You have typed the URL incorrectly

✖ You have followed a broken link

✖ The page has been moved, updated, or deleted

The first thing people will read is, "You have typed the address incorrectly." This error message blames the user for trying to access a URL that doesn't lead to a page. But the user very likely didn't do anything wrong. When people encounter "page can't be found" errors, it's more likely that they simply clicked a broken link, probably a broken link on the very website where the address leads to.

Instead of blaming the user, we should start by apologizing for the problem. This is good usability, because it helps people calm down and deal with the issue. In *The Man Who Lied to His Laptop* [NY10], author Clifford Nass writes about an experiment with a computer that provides "emotional support" while the user plays a game. He concludes that "actively acknowledging and addressing people's emotional states alleviated the high negativity and high excitement associated with frustration. In other words, people feel better when you show that you have heard them, understand their feelings, and sympathize."

Another problem with the error page shown earlier is that it uses a lot of jargon to explain what the problem is. Most people don't know and don't care what 404 means or what a "URL" is.

And finally, the error page isn't helpful. It offers reasons for the problem but doesn't offer any solutions. The site could determine where the user intended to go, based on the information available to the site. For example, links to blog posts often contain the year, month, and day that a blog post was written. Even if the last part of the URL got cut off

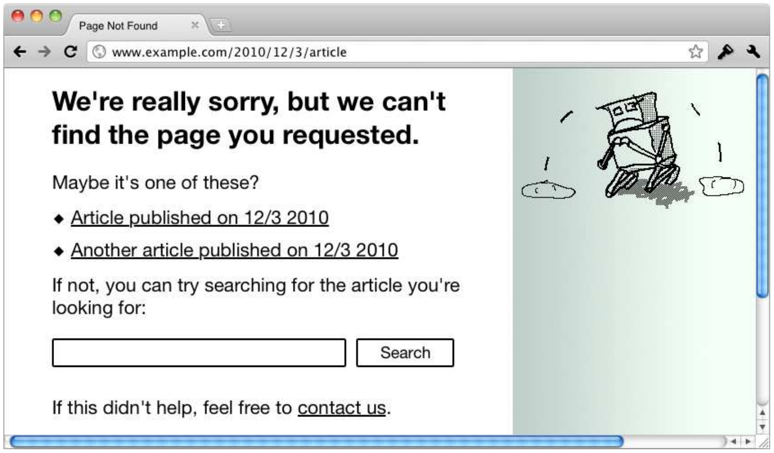
People Don't Like to Be Insulted

At folklore.org, Andy Hertzfeld recounts a funny experience that occurred while doing usability testing for Apple's Lisa computer.[*] He explains that the two default buttons in dialog boxes were originally called Cancel and Do It. In usability testing, however, they found out that a few users regularly clicked Cancel when they should have clicked Do It instead. When they talked to a user who seemed particularly confused by the dialog box, he replied "I'm not a dolt; why is the software calling me a dolt?"

People simply didn't notice the space between "Do" and "It" and read it as "Dolt." As a result, the Lisa team changed "Do It" to "OK."

*. Read the full story, and many other interesting stories about the development of the Mac, at http://www.folklore.org/StoryView.py?project=Macintosh\&story=Do_It.txt\&sortOrder=Sort%20by%20Date\&detail=medium\&searc

or was entered incorrectly, the remaining information can be used to determine likely candidates for the specific page the user was trying to open. How about something this:



We're really sorry, but we can't find the page you requested.

Maybe it's one of these?

- Article published on 12/3 2010
- Another article published on 12/3 2010

If not, you can try searching for the article you're looking for:

If this didn't help, feel free to contact us.

Rather than putting the blame on the user, this screen acknowledges that the site itself is more likely at fault and apologizes for the problem. It explains what went wrong in reasonably understandable language, but it also offers possible solutions to the problem. If nothing helps, it offers a direct way for people to contact you.

32.2 No Error, No Blame

It's good practice not to blame users for the errors they make. It's even better practice to include error messages that help users figure out how to fix the problem. But it's best not to let the problem occur at all. If something goes wrong, it's usually not the user's fault; it's your fault. If your product worked differently, then the problem might never have occurred.

You can prevent a number of common errors from occurring simply by changing the user interface. Let's look at two sources of such errors.

Mode Errors

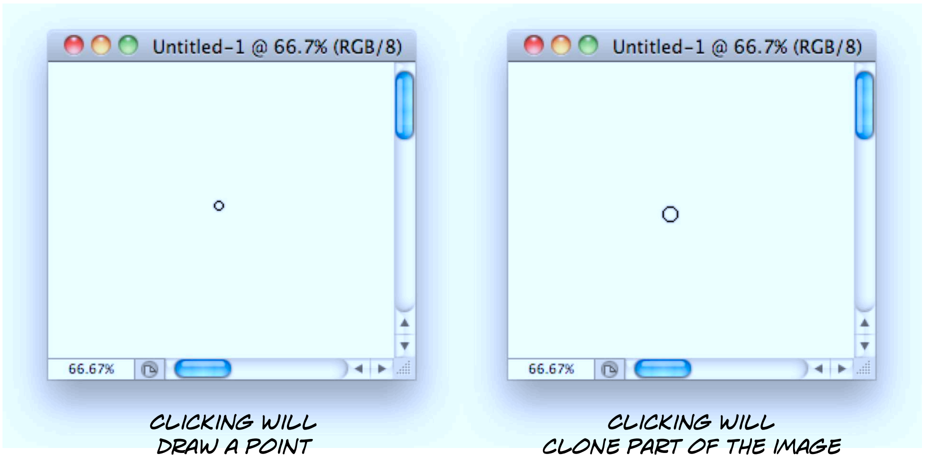
Problems we perceive as "user error" can often be attributed to products not clearly indicating their current state or what specific action they expect from the user. I've written about this in Chapter 20, *Modes*, on page 175, but since modes are such a common source of errors that we often perceive as "user errors," let's quickly recap.

A typical example of a mode error is a cell phone that goes off during a movie; most cell phones don't have obvious "silent" modes. A cell phone's current state is not clear just by looking at it. It's too easy to forget whether it is currently in "silent" mode.

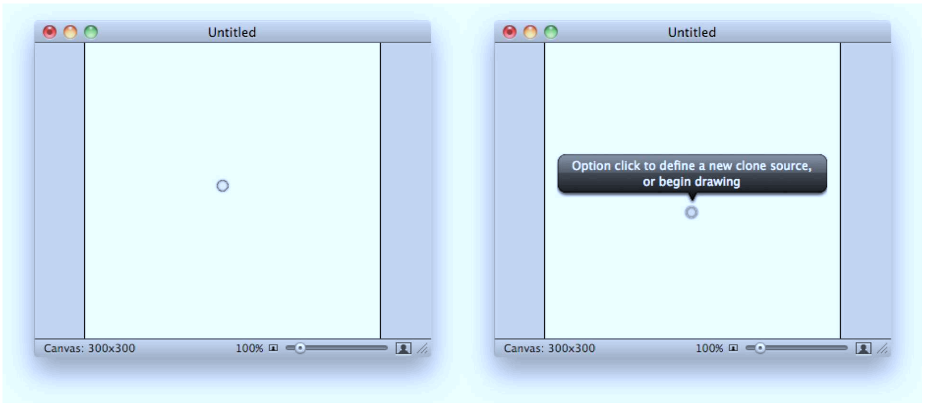
30 MINUTES INTO THE MOVIE, ONE OF THESE TWO CELL PHONES WILL MAKE A LOT OF NOISE. CAN YOU TELL WHICH ONE?

An obvious hardware button on the device that toggles between the two modes, like the one found on iPhones or webOS devices, would help prevent this problem.

A similar issue can occur in modal applications when the application doesn't clearly indicate what mode it is in or when it switches the mode unexpectedly. For example, picture-editing applications often have a modal tool selection that indicates the currently selected tool by changing the cursor. It's easy for users to miss this hint and, say, paint a line by clicking an image when they thought the clone tool (rather than the paint tool) was currently active.


Untitled-1 @ 66.7% (RGB/8)
66.67%
CLICKING WILL DRAW A POINT
Untitled-1 @ 66.7% (RGB/8)
66.67%
CLICKING WILL CLONE PART OF THE IMAGE

Using different cursor images for different tools would help prevent this problem. Here's how the same two tools look in Acorn:[2]



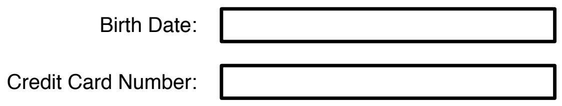
Option click to define a new clone source, or begin drawing

Acorn reminds users of the tool they're using. The chance of being confused about the active tool is much smaller.

2. Learn more about Acorn at http://www.flyingmeat.com/acorn.

Input Errors

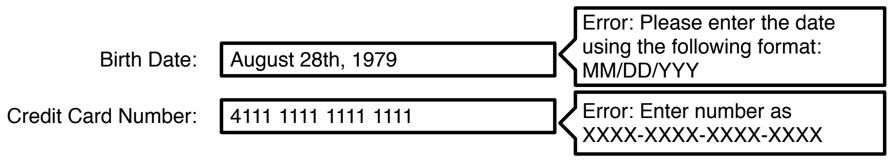
Another common example of "user errors" occurs when products don't clearly explain what they expect users to do, for example in overzealous and nonobvious form data validation schemes:



Birth Date:

Credit Card Number:

It's not clear what format the product expects for the date and credit-card number. As a result, users will try to enter all kinds of different formats. This often doesn't work properly.



Birth Date: August 28th, 1979

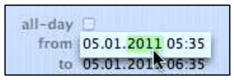
Error: Please enter the date using the following format: MM/DD/YYY

Credit Card Number: 4111 1111 1111 1111

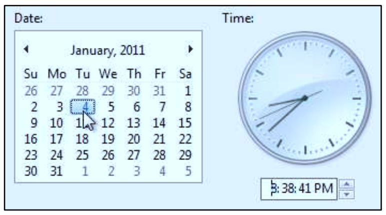
Error: Enter number as XXXX-XXXX-XXXX-XXXX

Apps and websites either should explicitly state what they expect the user to do (for example, by entering example data into the fields) or should be liberal in the kinds of user input they accept.

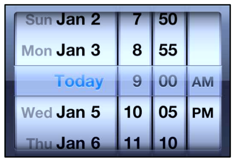
Another way of solving this problem is to provide a user interface that doesn't leave the input format up to the user. For example, rather than asking people to enter a date into a text field, the user interface could show a calendar.



ICAL ONLY LETS YOU CHANGE ONE NUMBER AT A TIME, MAKING SURE THE FORMAT IS ALWAYS CORRECT.



WINDOWS USES A CALENDAR THAT LETS YOU PICK A DATE BY CLICKING ON ITS DAY AND ONLY USES TEXT ENTRY FOR THE TIME OF DAY.



THE IPHONE USES A DATE ENTRY INTERFACE THAT AVOIDS TEXT ENTRY BY THE USER ALTOGETHER.

Takeaway Points

- Take responsibility. "User errors" are really user interface design failures; they are *designer errors*. Simply assuming that the user is responsible for a problem doesn't make the problem go away.
- Don't blame the user in your user interface. It makes your customers feel bad, and the problem was probably your fault to begin with.
- Present useful error messages. If something goes wrong, explain what went wrong, but more importantly, explain how the user can fix the problem.
- Prevent the problem. It's always best to avoid errors altogether. If users encounter errors, think about how you can change the product so that the error doesn't occur again.
- Prevent mode errors by avoiding modes or by clearly indicating your product's current mode.
- Prevent input errors by clearly showing what you expect users to do or by preventing invalid input from ever occurring.