How to critique an interface

The exercise

Select a device or user interface, preferably one you that is new to you. Start to learn to use it for perhaps an hour, taking notes of problems while doing so. Write these notes up as a report as described below. Suggestions for the interface include: a drawing package, Steve Draper's web pages, the (new) department web pages.

Raw Problem reports

Whenever you notice you are having a problem, write a separate "report" describing the problem. Examples of "problems" are getting stuck for more than a minute; or not being able to do something you want to and you feel the program should let you do; or the method for doing something being unnecessarily troublesome. No-one else needs be able to understand your reports at this stage, and they need be only a sentence or two, but you must be able to remember what you mean when you read them again in a few weeks time. You may want to scribble rough notes only while you are using the program, then fill them out a little when you finish the session.

Reports with analysis

The final stage is to take your raw problem reports, and make up a written report which lists these problems and adds some simple analyses to them.

A. For each raw problem, or "symptom" as I shall call them, put down a rough estimate of cost per occurrence, and frequency.
A1. "**Frequency**" means how often you think users would encounter this problem e.g. "every few minutes", "occasionally", or "once when learning the system". (When we apply this method to observing other users, then frequency will take account of what proportion of users experience the problem.)
A2. "**Cost**" is most often measured in time wasted. However if you think some other measure of cost is relevant to a given symptom then mention this e.g. "causes anger", "wastes 2 sheets of printer paper".

B. For each symptom, first report the raw **symptom** (e.g. user selects the wrong menu item); then suggest an **interpretation** or cause (e.g. the command name is poorly chosen, and reminds you of a different meaning that you confuse it with); and thirdly suggest a **modification** to the program that you think would solve the problem (e.g. propose an alternative name for the command).

You may not be able to decide what the right unit of cost is — I shall have a lot more to say about this later. Total cost of a symptom is the product of frequency and cost per occurrence, but you may find that many things which are clearly design faults nevertheless do not really cost the user much (like supermarket doors). Having to write down an estimate of the cost is a way of selecting out the really serious problems from those which are annoying but not much more.

You may not be able to offer a good interpretation or modification for every symptom. Sometimes you just can't tell why a user is having a problem — they just keep on making some error. Sometimes you may not be able to decide on a modification — they often depend on tradeoffs or other design decisions which you, as an evaluator, do not understand.