Exercise D2: Critique of an interface

(S.W.Draper, disk courses 6, 13 Jan 1998)

This exercise is for 2% of the marks, handed out 15th Jan; due in on Tues.27th Jan.

The exercise consists of (1) learning and using an interface (e.g. ClarisWorks graphics) for about an hour, (2) taking notes of problems while doing so. These notes should be saved and may be used again not only in the rest of this exercise but later in the term. (3) These notes should then be written up as a report on problems with the interface using the concepts mentioned below.

Which software

My recommendation is to use ClarisWorks graphics, and this is the only one I outline suggestions for. However in principle, if you have a strong preference, you could use some other software e.g. Hypercard. It is not essential that you are new to the software, but you will find it easier to find and note problems if you are a novice. (If you insisted on trying to do this exercise with THINK Pascal, you might have to use it for a long time before coming across a new problem. However easy or hard you find the software at your present level of skill, the aim is to collect 4-8 problem reports.

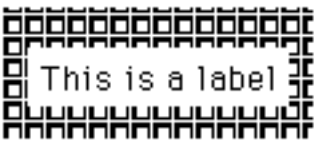
1. ClarisWorks drawing graphics

Fire up ClarisWorks on a Mac in the lab, and select "Drawing" when it asks what type of new document you want to create and work on.

Spend about an hour learning to use it: below is a list of suggestions of things to do. It is not important to finish the suggestions: the aim is to collect 4-8 problem reports.

Things to do in ClarisWorks graphics:

Draw a simple figure with each of the drawing tools.
HINT: there is on-line help available.
There are many different ways to delete something: undo, select object then use the delete key, ...
Try to create the figure below. Find ways of moving text so as to align it with other items.
Copy the figure into a text document.
If you find all this easy, then think of a figure you would like to draw, and see if you can find out how to do it in ClarisWorks graphics.



And this is another

2. 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.

3. 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.