HCI introduction: some notes

Preface

These are outline notes on points made in the first two lectures on HCI. You should be able to illustrate the points with examples (not given here).

Why study HCI?

1. Bad design is very common (there is a problem)
2. But it is not universal, so it can be done better (It's soluble)
3. Economics: machines used to be more valuable, now humans are.
4. Potential for theoretical advances: theories of interaction.

- Review / summary:
  The bad news: Even when we try, we don't usually get it right first time.
  The good news: but iteration makes it a lot better.
  The "system" being (re)designed includes both machine AND the human user.

Towards a constructive approach

- The first step is to learn to complain, to adopt the users' interests and convenience as a primary goal of design.

- But good intentions are not enough: we just don't know how to design right first time (the bad news). So we must test on users and iterate the design. This leads to a simple 2 step cycle: implement and test.

- What do we mean by "testing"? There are 2 kinds. When an area of engineering is well understood, testing may still be important because of some recognised unpredictability e.g. iron castings sometimes contain air bubbles or voids or cracks. So they are X-rayed to check, and any that are found are discarded immediately, while the rest are used with high confidence. Thus an understanding of what the possible problem is, leads to a focussed and specific test.

Unexpectedness

But a second kind of problem is when the area is not well explored. A key feature of HCI, unlike mature areas of engineering, is that a major part of testing is looking for problems that are unexpected. That is why **open-ended methods** are so important, and are often the most important contribution of HCI evaluations. In other words, the success of some interaction depends on many, many assumptions working, and the most important information is as to which (if any) have broken down.

So: We want to test prototypes on users for two reasons:
a) because humans are unpredictable; b) to discover unexpected issues.

Tradeoffs [Text: Norman & Draper pp.54-59.]

1. If you can always have the best, then may not need to do tradeoffs. But this is not general: other design problems may require revisiting "old" areas e.g. central heating controllers limited by display hardware.

2. Kinds of tradeoff

2.0 Between the implementation cost of making a change, and cost to users of not.

2a Conflicts within one task
Between cost types e.g. learning time (or menu search time) vs. expert performance time.
Conflicts within one small task e.g. consider the task of scanning a list on a screen to find a particular item (small print size best), and copying its details down on a piece of paper (big size best).

2b Conflicts between tasks (different parts of the program): e.g. in a word processor between text editing and file handling.

2c Conflicts between different users' requests (e.g. choice of command names)

2d Conflicts between different user groups or types e.g. handicapped, librarians and readers as users of library system.

3. The nature and consequences of tradeoffs for HCI
a. Look out for tradeoffs: naive pro-user attitudes don't directly lead to design solutions. Cost matters; and even if/when it doesn't, there are conflicts within user needs.
b. Don't assume you needn't make tradeoffs: technology cannot always save you.
c. Don't assume you must compromise: new hardware and new solutions are constantly appearing, often allowing you to have it both ways.
d. Unexpectedness is so common in HCI that you usually need to test any new adjustment, to make sure that you have not disturbed some other factor you weren't thinking of: i.e. that the modification you have made is not subject to some other tradeoff as well as the one you were considering.

The progression (Earlier approaches)
1. Machine centred: blame the user.
2. Learn to complain, blame the designer. Result: designers who want to help the user,
(UCSD type 1). The attitude that we are better off (as designers, as researchers, as business people) trying to make things as good as possible for users ("the customer is always right").

At this point we have established a user-directed attitude i.e. designing FOR the user. However how to act on it is not entirely straightforward.

3. Good intentions are not enough: we must iterate, must relate the design and the design process to users somehow — hence various kinds of UCSD (see below).
4. But eventually interaction centered?? the decision/remedy stage is a tradeoff, and not only the user but also the designer's time and costs must be considered.

Senses of "user-centered"
[UCSD = User Centered System Design]
1. Intend to benefit users.
2. Ask users: whether they like it; What they *would* like.
3. More sophisticated methods of "asking users", of eliciting their needs etc.
3a. Observe users: their opinions not necessarily what you need
3b. Give them the experience of the proposed machine, and task. The need for prototyping even in requirements gathering.
4. But tradeoffs: designer motivation and opinions from users are still not enough to determine design decisions.
5. Involve users in design.
6. Take the development of practice during use in the workplace seriously. Studying work.

So we saw that in moving from a machine centered view it was not enough merely to change attitudes and good will, but that we must involve users for two basic kinds of reason: a) we cannot predict users well enough (what they will do, what they want) so we must discover this empirically b) Unexpected issues. Thus in developing a workable UC approach, we see a range of senses of "UC", and all but the crudest involve working with users. They also all involve or at least can be combined with, a prototyping cycle. Users are needed at least in the use and observation steps; and perhaps can be made part of the design team and so help with interpretation and particiapte in the decision (remedy) stage.

Use / practice development

Three basic reasons for taking the Use step seriously:
1. Users invent methods not anticipated by designers: must observe what these are. (Could do that in a lab. however: a simplified Use step.)
2. Users eventually may employ the machine for higher level tasks not anticipated.
3. User's goals and methods are sometimes strongly affected by "context" in the workplace, as are unexpected interactions (both support and difficulties); not by the simplified single task taken into the lab.

Note, then, that the prototyping cycle still serves these concerns, though in new ways. For instance, as a requirements elicitation prompt; as something to take into the workplace. It is the Use stage, the considerations of use and practice, that motivate more advanced forms of "user centeredness": hiring users on the team, studying the workplace, learning about the task domain from users.