LECTURE 3

HUMAN-COMPUTER INTERACTION IN THE COMPUTER SCIENCE CURRICULUM

In this final lecture we shall consider three things:

1. *Why* computer scientists need to know about human-computer interaction,
2. *Where* it fits in the ciurriculum, and
3. *What* might be taught.

Why does a computer scientist need to know about human-computer interaction?

This question is easy. It is difficult to escape the conclusion that computer interfaces are becoming an increasingly important part of computer systems for the reasons we listed in Lecture 1: the availabilty of more computer power, bitmapped graphics and other graphical hardware innovations, increased subtlety of computer interaction, the possibility of "intelligent" interfaces. Beyond these technical reasons, and partially because of them, organized groups of users (unions and user groups) guarantee that computer scientists are going to hear a lot about usability in the future. There may even be attempts at regulation, so the computer scientists of the future would do well to be informed. But the real reason is that greater understanding of human-computer interaction will be necessary for the computer scientist to do his work.

Where does human-computer interaction fit in the curriculum?

Let us rephrase this question as a set of nine computer science curriculum design issues:

1. *Should human-computer interaction be in the computer science curriculum?*

Our answer, perhaps not surprisingly, would be yes, that a number of topics in computer science are heavily influenced both by the structure of computers and by the structure of humans, that currently virtually all the study is on the computer side, and that this should be balanced by some on the human side.

2. *Is it peripheral or central to computer science?*

Examples of peripheral topics would be social responsibility or databases. Human-computer

interaction belongs first on the periphery as the subject of graduate seminars and small parts of courses, moving in towards the core as it gains adequate content.

3. *Should it be oriented towards evaluating systems or designing systems?*

Our choice is design. Design is a key activity in computer science. Design time is the only time in which there is enough leverage to really make a large difference. This implies an emphasis on *theory*, when available; *tabulated facts*, when usable; *guidelines*, when possible; and *design methodologies*, again, when usable.

4. *Who should do the human engineering? Human factors specialists or computer scientists?*

The emphasis on design implies that the computer scientist is going to be the major player in the human engineering of computing systems. Interface considerations must be traded off against other factors such as machine speed, memory, display technology, and data structures. An example (due to Robert Sproull) is when a designer of a drawing system chooses between using a geometric model or a pen model. This choice of abstraction by itself determines that some tasks will be easier and some tasks more difficult; it determines whether the storage requirements will be large or small; it determines some of the errors users will have and some of the difficulties there will be in training. These issues must be traded against each other directly. By the time the designer has made this decision, the major interface decisions have already been made.

Another reason computer scientists will be the ones to do human engineering is that it is the computer scientists who will be the first ones to see/invent many technology opportunities. A related reason is that much human engineering may get done through the creation of user interface software packages. The creation of these is a computer science task.

Finally, the history of having human factors specialists in other engineering domains shows they tend to play minor roles and have little impact on the overall system. (They get to choose the fonts, the color of the case, and to write the instructions).

Of course I have talked as if people came in certain stereotyped packages. In fact, we are beginning to see more students who have acquire respectable backgrounds in both computer science and the relevant psychology as well as professionals who have cross experience. This should warn us to judge research contributions directly, rather than by primary professional affiliation. A related qualification is the real lack of a laboratory culture in computer science. This means that while computer scientists are preeminant in design, they often do not do particularly well when it comes to making measurements and working out theories of user behavior. If computer scientists are going to do *reasearch* on human-computer interaction as well as building

systems, it would be nice to ensure in their training that they have some background in laboratory science, including at least some in the natural sciences.

5. *At what course-level should human-computer interaction enter the curriculum?*

Ideally, human-computer interaction would enter the curriculum at the undergraduate level for those things that can be established. Presently, however, the most appropriate place for most of the material in this area is in graduate seminars.

6. *Should what is taught be human-computer interaction or human factors?*

We think it should be human-computer interaction. The issues are closely bound up with software issues. Splitting off of the software issues and leaving only the human factors considerations is likely to result in the isolation and ignoring of human factors by computer scientists.

7. *Should the sort of material in these lectures be a course or part of one?*

For similar reasons as above, we think it preferable that this material be part of a course on the interface. The human part of programming languages should go with the programming language courses, where appropriate, and the human part of interfaces should go with interactive systems and the human part of graphics should go with graphics. This especially should be a consideration for text-book writers.

8. *Should all the human related aspects be in one course or separate courses?*

This is a version of the previous question. The human-related aspects of computer science ought to be separated into different courses because the focus in computer science is not on humans per se; it is on whatever (human aspects included) is relevant to the various topics of computing.

9. *What should be the objectives of such a course (or part of one)?*

The objectives of the part of the course that teaches about human factors should be: (1) to establish the goal of taking the user into account; (This has been the lament in the human factors discipline all these years.) (2) to give students a realistic operational picutre of the user; and (3) to give the students some techniques to use in design and research.

There is one additional comment that should be made. The sorts of interfaces that are becoming possible and that are much in need of university-based research depend on modern hardware. Yet the equipment available to most users in universities is such as to make training and research on modern interfaces almost impossible. Students trained on interfaces using 80 character x 24 line displays running on a time-sharing mainframe will be largely unaware of and unprepared for the rich possibilities for modern interfaces. *Probably the most cost-effective expenditure that any government funding agency could make toward raising the general level of the state-of-the-art in human-computer interaction would be to reequip the universities through equipment grants.*

What would be the content of a course in interactive systems?

To explore this issue, let us propose a straw-man syllabus for a course in interactive systems to see where things might fit. Our suggestion is as follows:

1. *Overall interaction model.*

Computer scientists will program not just the type of systems they commonly use at universities, but also devices such as aircraft flight systems, power plants, space telescopes, automated bank tellers, and other systems that have embedded computers. Students should be given an overall model, general enough to embrace all these contexts. The Sheridan model presented earlier has some attractions as does recent work by Rasmussen and others.

2. *Characterization of the human.*

Computer scientists need some approximate characterization of human capabilities to structure their knowledge of humans so as to be able to predict the gross reasonableness of various proposals. (This is an intended role of the Model Human Processor.) They also need to be taught what applied models there are for predicting user responses.

3. *Dialogue styles and task characteristics.*

Putting these all in order is one of the tasks in this area that needs tidying up. Students should have exposure to a systematically collected sample of styles, since at this stage in the field, concrete cases are one of the most important design influences. These, of course, are the building

blocks out of which the students will later build systems. Suggested topics include users' models, windows, menus, alphameric dialogues, and devices. The topics would also include some parts of computer graphics, data structures, and similar topics. But different interaction techniques are appropriate for different tasks, so this activities also leads to some taxonomizing of the tasks people actually do at the interface.

4. *Problems.*

There also needs to be exposure to the challenges in this area and the means that exist for fielding them. Examples: the integration problem, controlling detail on the display, increasing functionality without proportionally increasing complexity.

5. *User interface management systems.*

How can the complexity of programming the interface be reduced by making packages that support components of the interaction, much as is done for graphics? The supposed benefits of such packages include reduced costs of constructing interfaces, smoother interaction, higher reliability, and increased interface consistency across applications and within applications.

6. *Intelligent interfaces.*

Using an interface is a communicaiton process. Artificial intelligence techniques can be applied to the interface to produce different types of interfaces: For example interfaces can be made that coach their users by giving them hints (Sleeman and Brown, 1982). Interfaces can be built that tailor their interaction based on dynamic models of the user. Systems can be made that make use of the "conversational postulates" of human discourse.

Conclusion

Let us recapitulate our theme:

Now (in the next five years) is the time to deal with the interface. There are finally adequate computational resources beyond the needs of the subject program such that we can afford to spend code and memory on the interface. A lot is at stake, since the usefulness of many programs depends critically on the interface.

We would suggest requirements for making progress in this area as follows: (1) The human side of human-computer interaction must be taken seriously. (2) This must happen within

computer science (not within just psychology or human factors). (3) It must happen by building an engineering theory of human behavior that encompasses task analysis, calculation, approximation, and is theory-based. These are necessary so that they can aid system building by computer scientists at design time.

We have outlined some progress in satisfying these requirements: The Model Human Processor as a base and models such as the GOMS model and the Keystroke-Level Model as examples. The Model Human Processor is as yet very crude. It gives some useful results, but there is much that is still missing in terms of its coverage. It is a synthesis of various results that are around, not a new theory on its own. It tries to make it possible for computer scientiststo work with a reasonable model of what the user is all about.

Finally, we have tried to show where this sort of effort would fit into a curriculum concerned with more than just the user per se.

References

Averbach, E. and Coriell, A. S. (1961).

Short-term memory in vision. *Bell System Technical Journal 40*, 309–328.

Card, S. K.; English, W. K.; and Burr, B. J. (1978).

Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics 21*, 601–613.

Card, S. K.; Moran, T. P.; and Newell, A. N. (1983).

The Psychology of Human-Computer Interaction. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Conrad, R. and Hull, A. J. (1968).

The preferred layout for numerical data-entry keysets. *Ergonomics 11*, 165–173.

Darwin, C. J.; Turvey, M. T.; and Crowder, R. G. (1972).

An auditory analogue of the Sperling partial report procedure: Evidence for brief auditory storage. *Cognitive Psychology 3*, 255–267.

Kurke, M. I. (1956).

Evaluation of displays incorporating quantitative and check-reading characteristics. *Journal of Applied Psychology 40*, 233–236.

Michotte, A. (1946/1963).

The Perception of Causaility. New York: Basic Books, 1963. Originally published as *La Perception de la Causalite.* Louvain: Publications Universitaires de Louvain, 1946.

Murdock, B. B. Jr. (1961).

Short-term retention of single paired-associates. *Psychological Reports 8*, 280.

National Research Council 1982).

Automation in Combat Aircraft. Washington, D. C.: National Academy Press.

National Research Council (1983).

Research Needs for Human Factors. Washington, D. C.: National Academy Press.

Peterson, L. R. and Peterson, M. J. (1959).

Short-term retention of individual verbal items. *Journal of Experimental Psychology 58*, 193–198.

Siewiorek, D.; Bell, G.; and Newell, A. (1981).

Computer Structures. New York: McGraw-Hill.

Sleeman, D. and Brown, J. S. , eds. (1982).

Intelligent Tutoring Systems. London: Academic Press, 1982.

Sperling, G. (1960).

The information available in brief visual presentations. *Psychological Monographs 74* (11, Whole No. 498).

Welford, A. T. (1968). *Fundamentals of Skill.* London: Methuen.

DISCUSSION

Professor McCarthy pointed out that when a designer makes a system for other people to use, this tends to make a "dictator" out of the designer. In other words, there is a difference between designing for yourself and designing for other people. The results of designing for others can be disastrous. For example, the U.S. trade unions are concerned about the fact that computerizing a job often results in a "de-skilling" of that job. They would like to see an increase in the skill required for a job, once that job has been computerized. In other words, a computerized system should not be simply a "data entry system", where the system is the master and the user its slave; a computerized system should be the "tool" of its user.

Dr. Card remembered that, when deciding what sort of system to have in the office, the argument ran that if everybody was put together in the office on the word processor, it would put the office staff on an assembly line. Job satisfaction from people-contact argues for distributed power. When people don't design systems for themselves, there is often a lack of feedback from users (and also often a lack of laboratory testing.) A modest amount of observation can make a great difference in design (e.g. the video showing a secretary doing pointing to the screen: firstly, she was hampered by long fingernails, secondly, she thought she had to trace whole lines instead of single words).

Professor Randell recalled the time when he wrote compilers, in an office he shared with the users of those compilers (although he himself was not a user). In this situation the "feedback loop" was very intensive.

Dr. Scoins asked how many lectures would make up a course on 'Interactive Systems'. Dr. Card replied that he was thinking in terms of 2 - 3 hours per week, for a 10 - 13 week semester. **Dr. Spence** pointed out that an important element in both the teaching and the advancing of a subject is appropriate notation. An appropriate formal language is needed in which to describe 'Interactive Systems'.

Dr. Card went on to say that his team had tried to develop formal languages for the description of 'Interactive Systems', but had experienced difficulties. You can try to describe a system using a formal language that can be compiled. However, too many details about a system are needed in order to get its description to compile. The important aspects of a system become obscured amongst the details. On the other hand, you can describe a system more informally. Such a description may prove useful. However, it has the disadvantage the it cannot be compiled.

Dr. Spence wondered how many pages would be needed to describe, say, a cash dispensing machine, in such a formal language. Dr. Card thought that it might take as many as 10 pages.

Professor Whitfield thought that there might be a danger that this emphasis on "human factors" could cause the designer to concentrate on things that are

non-essential. He recalled the time when he was writing a compiler. The users used the compiler for 10 days without noticing a major (in his opinion as the designer) "bug" in it - he found the "bug" himself. On the other hand, the users noticed less important (in his opinion) things about the compiler. In other words, users may be sensitive to non-essential aspects of a system, yet insensitive to essential ones.

Dr. Card replied that this is why he has tried to emphasise the balance between "computing science" issues and "human factors" issues. At the moment all the emphasis is on the Computing Science issues. He said that we are not trying to detract from Computing Science, just to get human factors in the picture as well.

From Dr. Card's lecture, **Professor Whitfield** came to the conclusion that, when he comes to design a computer system, he will need to know some basic information about human capabilities. For example, he might need to know the answers to questions such as, "What is the resolution of the human eye?" or, "How fast can a person type?" He wondered whether there was any sense in attempting to collect together this sort of information, in some form of reference document.

The speaker replied, "That is the activity I'm trying to do." Also there is another attempt "The Handbook of Chemical Perception and Human Performance" (approximately 50 chapters). It is difficult to extract the required information from this book, for the following reason. The book describes various experiments on human perception and performance. However, when results on the same aspect of human perception and performance are provided by more than one experiment, no attempt is make to compare these results. The airforce found they could not make use of all the information on perception, because it was so spread around, so they commissioned chapters by leading exponents of different fields, then abstracted and tabulated the data. One problem is that some of the "facts" are not established, therefore some of the material is contentious. Work is under way to produce a more useful reference manual.

Dr. Larcombe agreed that it would be useful to have a computer science course on "human factors", yet pointed out that there are many other topics that might also be useful in a computing science course. Furthermore, there are no text books which can be used as background reading for a course on "human factors".

Dr. Card replied that, when he was a graduate student taking courses in Computing Science there were no textbooks on Computing Science. Lecturers strung together material from research papers. The text books followed, and now we are overwhelmed by authority. He continued, "I'm sure others here can remember the wild and woolly days of Computing Science. It seems very tidy now. This happens in areas which are new and difficult."

Dr. Card was asked how he would characterize the book mentioned in his abstract. He explained that the book is a research monograph, yet it could be used as a text book. This is being used for a number of courses at research seminar level. It is machine readable (because it was prepared on a machine.)

It is a recursive book, in that it was generated using the systems which it describes.

It was pointed out that there is a difference between "computing science" and "human factors" - computing science is "synthetic" whereas human factors is "analytic". You can write a program from nothing, but you can't invent data about how people behave.

Someone was unhappy with the term "human factors". For example, in the U.S. there are two journals: one entitled "Ergonomics", which deals with the "hard" aspects of the subject, and another, called "Human Factors", which deals with the "soft" aspects. **Professor McCarthy** requested an explanation of the terms "hard" and "soft" in this context. Dr. Card replied to the effect that "hard" aspects are those that deal with things like display layout or the hardware provided for the user. "Soft" aspects deal with the way in which the user interacts with the system. Someone suggested that "hard" aspects are "static" while "soft" ones are more "dynamic", while Professor Randell ventured the suggestion that "hard" and "soft" aspects are at different "cognitive levels".

Professor Randell wondered whether the subject of "human factors" would move into other computer science topics. For example, this happened with the subject of "compilers". Early text books on compilers simply deal with the details of how to write a compiler. Later books, on the other hand, attempt to put the subject into the larger context of "computing science". Dr. Card agreed that instead of simply concentrating on "human factors", we must blend the study of "human factors" with the wider area of "computing science", by combining it into computer science courses on other topics.

Professor Randell pointed out that this approach was the same as that of these I.B.M. seminars. The goal of a seminar is to synthesise where a particular subject has "got to so far", and try to put that subject in the wider context of "computing science". In this way, people from all areas of computing science will come to know about the subject and then use this knowledge in their particular field. This year's seminar, more than previous years', has achieved this goal. One problem may be that, by concentrating on human factors as a separate field, it never quite gets integrated into the discipline. When a designer designs, whatever he knows or DOESN'T know in his head about the different aspects of human factors is going to determine how and what he designs.