WHAT CAN WE TEACH ABOUT HUMAN-COMPUTER INTERACTION

Terry Winograd

Stanford University
Palo Alto, California

ABSTRACT

This paper is the closing address for CHI '90. It addresses the problem of educating computer professionals in the area of human-computer interaction, arguing that standard approaches within computer science need to be augmented and that new models of education can aid us in producing students with broad competence in the design of computer systems for human use.

THE GROWTH OF HUMAN-COMPUTER INTERACTION

CHI `90 has been an impressive event. It has brought together a tremendous group of people who are concerned with "empowering people," and showed how they are making that slogan a reality. The diversity of topics and obvious intensity of interest demonstrates the strength and vitality of a new and growing field.

And we may sometimes forget how new it all is. This conference and the research community it represents are a product of just the last decade. The first conference on "Human Factors and Computing Systems" was held in 1982 and the Journal "Human-Computer Interaction" began publication a few years after that. Not so long ago, there were just a few odd psychologists concerned with "human factors," who happened to study how people used computers. They probably spent more of their time talking to their counterparts in the automobile industry than to people working in computing, and what they did was certainly not considered a part of computer science.

We've come a long way in a short time. Our field is rapidly both growing and maturing. Now in addition to the CHI conference, there are a variety of conferences, journals, books and funding programs on related topics. In fact it is getting hard to know just what counts as a "related topic," as concerns of human-computer interaction spread throughout the computing world.

I would like to attribute this success to the brilliance and hard work of all of you who have participated in it, and indeed you can be proud of many notable achievements. But being realistic, I have to acknowledge that growth and maturity are being forced upon us by the tide of development–by the persistent flood of hardware and software products in the marketplace, and by the changing nature of how they are created, purchased and put into use. To put it simply, a decade ago the people who designed computer hardware and software were dominated by the question "How do we get it to run?" They had to apply tremendous engineering ingenuity to the problems of getting computational tasks done on machines that could be reasonably built,programmed and paid for.

With the advent of the personal computer, a new trend that had been building for years came to center stage. Processing power was no longer the central arena of competition, and the convenience that new machines and languages could offer to programmers no longer drove the market. The "look and feel" of a product became its dominating characteristic, and fortunes were made and lost because designers understood (or failed to understand) what the computer programs they were writing meant outside of the world of computing: How they were encountered by users in a context of work and equipment for getting things done, not a context of data processing and computation.

TEACHING WHAT WE KNOW

The urgent sweep of technology development has promoted the rapid growth of new paradigms for design, without having built adequate foundations, either conceptual or institutional. We in the research world have been struggling to catch up: to bring some kind of order and understanding to what we see going on in the world of practical development, and to build on this the basis for continuing progress and future invention.

As a university teacher of computer science, I look at this from a particular perspective. Has our research

led to the kinds of results that can be systematically taught and can form the basis for professional competence? Can our area of concerns become a fundamental part of education in a mature computer science?

This is a tall order. Computer science departments are not generally noted for respecting much of anything outside of their disciplinary boundaries. Over the years, computational education and research have focussed almost exclusively on the technical aspects of computing with little attention to the human contextual aspects that are crucial to effective design. Considerations of "human factors," "social impacts," and "design methodologies" have appeared as tangential add-ons to the study of "real computer science." People and organizations are often viewed as an unfortunately unpredictable and messy part of the technological environment. With a few notable exceptions, computer science departments offer at most a token course or two on contextual concerns.

At the same time, thinking about the role of technology within the broader society has moved towards a more cautious and directed approach to the design of new technologies. We have come to recognize that the potential benefits of a technology will come to fruition only when we pay sufficient attention to the context in which it is instituted.

In the public discourse about computing, we can see a number of trends that are illustrative of this shift. For example, as Mike Dertouzos described in his opening talk, we are feeling the impact of the overall national concern with productivity. One of the primary motivations for computer technology has always been the efficiency it can bring to individual and organizational work. The "office of the future" has been a rallying cry of the computing vendors for many years. But the promises have not always been met and there is a large and growing critical literature within the business and management community, questioning the assumptions that have been made about the value of computer tools. We are beginning to recognize that there is an important difference between "efficiency," in the traditional sense it has been used in computer science, and "productivity," in the practical sense of getting more value from the work.

There is also a growing concern about the social consequences of computing–a recognition that along with the potential for improving our work and lives there is also a potential for creating new problems and exacerbating old ones. Along with awe for the benefits of the information explosion we find increasing public concern that large safety-critical computer systems are not dependable and that there are serious potentials for the misuse of information in the myriads of blossoming databases.

What does it mean, then, to educate computing professionals in the discipline of human-computer interaction? We can approach this question in two parts. Clearly we need to articulate what we mean by "human-computer interaction", but first let us consider what we are trying to do in educating our students at all.

The purpose of education is to develop the student's competence to take some kind of action. Often we lose sight of this in our eagerness to "transmit knowledge" or "cover the material," but in the end it is the ultimate justification for teaching anything. However, we must be careful not to identify competence with a narrow vision of action–with the ability to write code in a particular programming language or to design efficient algorithms. We need to consider the broader domains of actions and effects in which our students will work.

There are, of course, different career paths and professions within computing, which demand different kinds of competence. As scientists and researchers, we tend to see our students as young copies of ourselves, and we often focus on teaching the principles and skills that are relevant to doing research. This is indeed one part of the university's role, but it is certainly not the only one. The vast majority of people we train in computer science (even those at the elite schools) will not go into academic research, but will play a variety of roles in the invention, production, implementation and use of new computing devices. In this they need competence in design–the activity of bringing forth new technologies and the practices for using them. To realize the potential of our students in designing computing systems we need to develop their capacity to relate the design of computer systems to the human activities and practices in which they will be embedded.

But in relating humans and design, we must avoid simple answers. This is not just a matter of sensitivities about "listening to the users." To understand what will really work we need to go beyond the superficial aspects of "user-friendly" and "seamless integration", and pursue a deeper analysis of what people are doing in larger human and organizational contexts, and how this is influenced by the use of computers.

And here we get back to the other question, which we put aside a moment ago: "What do we mean by human-computer interaction?" Webster defines "interaction" as "mutual or reciprocal action or

influence." Clearly, humans act on computers and computers influence humans. But how? In what dimensions? We cannot be satisfied with the conventional view that this is a question of designing the "human-computer interface." When we narrow our attention to the flow of activities between an individual and a piece of machinery sitting on his or her desk, we lose sight of the larger picture of mutual influence–the influence that our networks of equipment have on the way we work, live and make sense of our actions.

This broader picture does not lie within the traditional boundaries of computer science or human factors. It touches on many of the disciplines that have dealt systematically with human activity, including philosophy, psychology, anthropology and the social sciences. But it is misleading to see the problem as one of offering an "interdisciplinary" education. We will not succeed at developing competence in design by turning computer students into amateur sociologists, amateur anthropologists, amateur psychologists and amateur organization theorists. Although it is certainly valuable to introduce them to the key insights that each of these disciplines has generated, there needs to be an integration–a way of turning a multi-disciplinary goulash into a background that makes sense in the context of the design tasks our students will encounter in the exercise of their profession.

A MODEST PROPOSAL

As a response to this challenge I want to offer an example of what it might mean to educate students in human-computer interaction. At this point it is not a comprehensive educational program, but a small experiment, which was suggested to me by Mitch Kapor (one of the most successful "self-made" software creators) and which directly embodies an educational approach being developed by my colleague Fernando Flores.

In a letter, Kapor described his own educational quandary:

> "..people, such as myself (at an earlier point in my career) find themselves with a difficult set of choices, no alternative of which is entirely satisfactory. Existing graduate schools of computer science place a disproportionate emphasis on subjects that are not of fundamental relevance to design... . As communities, they lack the human-centered values central to a design oriented approach. But outside of these graduate schools, there is no organized professional training, so one is left on one's own."

His proposed solution is based on an analogy with the education of architects, whose profession stands as an prime example of design-oriented action. A student architect begins with courses in the basics of the engineering disciplines that underlie competence at designing buildings that meet the physical requirements of construction and durability. But that is only a preparatory component of the training. It is accompanied by studio courses in which students are presented with realistic design problems and work out solutions in collaboration with experienced senior architects. These teachers (or perhaps they are better called "coaches") critique the students' work and help them develop their ideas as the project progresses.

In this studio work, teachers and pupils look beyond the basic engineering domain, bringing in questions that distinguish architecture from engineering. They bring to the forefront concerns with the requirements people have in terms of space (both physical and psychological), comfort and productive action. To design a building well, the student needs to understand the activities people intend to perform in it and the problems they are likely to encounter. Knowing how to find out these things in a particular case is dependent on having a strong general understanding of how people are constituted and how they act, both individually and as part of a social organism. The studio courses serve to develop overall professional sensibility: a way of looking at the world that knows how to perceive the relevant features of a situation, of knowing the right questions to ask, of sensitization to a certain realm of concerns.

This kind of education does not fit well with the models held by most people in scientific and engineering disciplines. Instead of having a well-defined set of theories and techniques that can be put into a book and mastered through a set of progressively more difficult exercises, the student is developing a vaguely-defined "professional sensibility." But if we adopt a different approach to education, as advocated by Flores, this practice is not only reasonable, but takes on a central role. I will not be able to elaborate his educational theories here, but will use the example of human-computer interaction to illustrate their basic structure.

The starting point is to connect learning to action, approaching it not as the accumulation of knowledge but as the process of becoming effective in a particular domain of action. The student achieves effectiveness with the help of teachers who introduce the relevant domains and "coach" the student as he or she experiences acting within it. This coaching is not simply a matter of giving tips and hints, but plays a fundamental role in building the

students' ability to recognize and create in the relevant space of actions.

The learning process can be described in several basic steps:

1. Announcing a New Domain

Initially the teacher introduces to the student the potential for learning something in a new domain. There is no way to develop competence without first recognizing your incompetence–without recognizing that there is some area in which your capacity for action could be expanded or developed. Of course the teacher is not operating in a vacuum. Students are enmeshed in a variety of other activities and experiences which lead them to recognize breakdowns and the potential for overcoming them.

The emphasis here is on creating a recognition of possibilities that are relevant to action–in our case the actions associated with the design of computer hardware, software and systems architectures. In cases where such a connection is not evoked (as in "Take Latin, it's good for your mind!") the student does not enter with the kind of openness that is needed in a field which demands thoughtful exploration rather than rote mastery.

In the case of human-computer interaction, there is not a single domain but a variety of domains in which we can articulate the effects of computers on humans (the "interactions"). Looking through the program for this meeting one can identify a number of different perspectives:

- **Interfaces:** The largest single group of papers and sessions is focussed on the question that first gave the field shape: the design of interfaces through which individuals interact with machines. Design must both take into account the potential for new technologies in a variety of media and the psychology of the human actors, with their limitations, preferences, aptitudes and failure modes.

- **Work Structure:** In the domain of concerns that has at times been labeled "computer-supported cooperative work," the user appears not as a face and pair of hands in front of a workstation, but as a participant in organized social activity, in which the use of computers can play a role. The question for design is to understand how different possibilities for computer augmentation will enhance, modify, or possibly detract from the effective action of the computer-using community.

- **Supporting Technologies:** Many of the sessions and papers deal with the details of a wide variety of technologies which can be used in the design of systems with which people interact. They range from hypermedia and video simulations to object oriented programming languages, window management systems and structure editors. The domain of technical possibilities is a cornerstone on which design proceeds, but of course only one of the corners.

- **Learning:** In designing a computer system it is not sufficient to produce software that can only be used by the person who built it, or by someone who is able to magically duplicate his or her understanding. Many otherwise excellent systems have been useless because they weren't sufficiently "learnable" to the appropriate audience. This of course touches on the work structure questions as well, since it matters who is expected to learn what and how their overall work situation creates a situation that supports such learning.

- **Social Processes:** Some of the most significant impacts of computers systems stretch beyond the walls (even the electronic network walls) of the people and groups that use them. There are broader social, legal and political questions that are both driven by and shape the technology. As an obvious example not so far from our field, look at the disputes over digital audio tape, with its potential for distortion-free copying. The relevant design questions cannot be understood in terms of the technology, but must address the larger social interpretation of rights of ownership. Similarly, questions of automation and the role that computers play in changing the nature of work processes (either "deskilling" or "informating") are fundamental to computer design.

- **Design Processes:** Another domain of concern is the self-reflective one, in which we observe our own activities as designers and then attempt to better design those activities. This includes both the self-reflective examination of what the designer does and the broader social analysis of the design process, and the many parties involved in it,

including designers, users, purchasers of workplace systems, etc.

- **Assessment Techniques**: Lurking within the whole discussion of competence in design is the question of assessment–what constitutes "good design"? There are various measures that can be applied, including the judgment of recognized "experts", the results of controlled experiments, and the success of a design in the marketplace. Each of these has its strengths and weaknesses, and part of the discourse within our field is an attempt to better design our tools for assessment and to understand the potentials (and limitations) for their application.

This list is not a systematic taxonomy–it was generated in a fairly quick scan through this week's program–but I hope it will evoke for the listeners a sense both of the diversity of concerns and their importance in the activity of design. The first step, then, is to evoke the student's awareness that what appears to be a simple act of engineering or programming can be construed as an act in each of these domains, with consequences and possibilities that are uniquely visible within it.

2. Showing the Domain

The teacher has not really introduced a new domain until the learner sees it in the real world–seeing conversations that are meaningful to people seriously engaged in their fields of activity. The aim here is not to ensure that the learner understands what he is shown. It is only to make certain that he or she sees a new, concrete domain of action.

In the case of teaching about human-computer interaction within the university, this is far from trivial. Some domains of design, such as those associated with command languages, are straightforwardly visible to students as users of computer systems themselves.

But other domains, such as those dealing with the social structures of work, are not familiar and do not show up in the kind of classroom computer use that makes up the bulk of their training. For computer science students, it is an eye-opening experience to talk open-mindedly with "real users" in a "real-world setting." This need not be in an exotic place–often it is no farther away than the administrative personnel sitting in the next-door offices. But it takes a conscious effort to ground the recognition that there are whole domains of concern about how the specifics of design can affect people's work, which do not show up in courses on compilers, networks or programming languages.

3. Constructing an Ontology of Distinctions

These first two steps are focussed on the experiential–on opening up possibilities in the student's understanding. But there is much more to education than pointing out a problem and tossing the student in to sink or swim. The teacher provides what Flores calls an "ontology of distinctions," or what we might more conventionally call the "conceptual framework" in which to observe and act in the domain. He says "effectiveness in action is grounded in linguistic distinctions that make action possible... to learn, to acquire knowledge, is to become proficient in a language."

This talk is not the place to go into an extended discussion of what he means by a "language" and how it compares to more traditional ways of talking about "theory" and its role in practice. Let me just observe that this concept of language is quite broad, and encompasses both the learning of formal systems (such as the base of mathematics and physics that underlies computer engineering) and the less formalized but still rigorous languages that appear in the discourses of philosophy and social inquiries. The teaching of these distinctions need not be organized around the specific activities of the learner, but rather provides a background on which further learning can rest. The point is that the ontology (the framework) must support the relevant domain of action. For example, the formal knowledge of computation theory underlies the design of efficient algorithms and is useful for this, but at the same time it is not the framework for understanding the design of effective programs–those that help the user get something done.

Looking again at the program of this conference, it is clear that a number of different ontologies are relevant to the design of human-computer interaction. This is where the corollary disciplines, such as psychology, anthropology and organization theory, come in

But as I mentioned above, this does not mean that we are training our students to be professional researchers in these disciplines. Rather, they need to have the ability to recognize and enter into the discourses within those professions that are relevant to questions of design.

We face a challenge here, as there is no consensus in the field as to what the relevant domains of distinctions are or what bodies of existing discussion are most relevant to the student. The texts that exist are not oriented to the questions our students face, and there are still few people who have made the connections by addressing the concerns of the designer within the ontology of other disciplines. As the years go on, we can expect such writings to appear, and although there is unlikely to be anything like the near-universal consensus about what should go into the elementary training of physicists, there will be reasoned and productive discussion leading to much-improved possibilities.

4. Ground the Distinctions in History

Computers are new in the world, and for educating the people who work with them this brings both good news and bad news. The good news is the excitement of so much to be learned and explored. In any period of the modern history of science and technology, there are a few "emerging areas" which capture the imagination of a generation, entice the best minds, and attract substantial resources for development.

But along with this there goes a kind of arrogance and blindness to the ways in which all that is new is also a reflection of what has gone before. I know that in my own training, there was an almost complete disdain for any disciplines that had gone before (except, perhaps mathematics). All of those old linguists, philosophers, psychologists and even electrical engineers were simply irrelevant as the march of computerized progress swept their old-fashioned concerns and theories into the dustbin of history.

Well, it isn't quite that simple, and I think that after a few decades we are beginning to acquire a bit more perspective and a bit more humility. Computers may be millions of time faster and larger and cheaper, but people are still people, organizations are still organizations, and work is still work. When we look at domains outside of the narrowly technical, much of what we need to be concerned with is not unique. It has its own special shape due to the particularities of computing, but everything from user interfaces to legal ramifications have clear correlates in many other technical (and even non-technical) enterprises.

As part of introducing the student to new domains, the teacher makes the domain available as a historical conversation which distinctions have been invented and in which that learner can become a participant. This is not just a matter of "history appreciation" but plays a direct role in the student's potential for contributing to the discipline. The historical discourse both provides a map of the paths and blind alleys that have been trodden before and reveals the ways in which progress is made.

5. Practice for Competence

The key to the learning process is that the student has the opportunity to make sense of new distinctions in the practice of action. People are not able learn by simply hearing the relevant distinctions and theories, but need to ground them in practice. Even the most traditional modes of teaching are based on "exercises."

The idea that learning how to design requires practice is not very surprising or novel. It is common in computer science curricula to find "project courses" in which students take on individual or group projects of relatively large size (relative to the exercises in other course work). The aspect I want to emphasize is the way that teaching is structured to relate the project experience to the relevant domains of distinctions.

It is all too easy in teaching a project course to simply extol the value of experience or some kind of "mysterious" transfer of expertise. It can be quite a bit harder to design the experience and the associated coaching to develop competence in a more systematic and rigorous fashion. The teacher is doing much more than simply providing an experiential opportunity and then acting as a cheerleader and final evaluator. In working with the learner, he or she is building the learner's capacity to be an observer of his or her own action. This requires using the specifics of the project to trigger opportunities to introduce distinctions, make new concerns visible, and demonstrate the appropriate use of formal tools. In the architect's studio course, the teacher works with the students in an ongoing way, in the context of the designs that they are evolving.

To do this, of course, the teacher must already have had the experience and teaching to be a competent actor in the domains being taught. Someone who has designed software with a narrow concern for algorithmic elegance and

efficiency is unlikely to be an adequate coach in questions concerning the way people in a work setting will use a given tool. There is clearly a bootstrapping problem here–the kind of proficiency we are trying to develop is not one that is widely shared or recognized within the profession. There are seas of good sensibility and peaks of brilliance here and there, but it will take a good deal of collective work before this can be gathered into a coherent curriculum and made widely accessible.

The opportunities for providing a suitable practical experience themselves are at times limited within the university context. The highly bounded nature of academic coursework makes it very difficult to set up a situation in which the students are committed to practical results for the users. But without this, it is hard to justify the kind of effort required by the potential user community to interact with the designers sufficiently to reveal the less superficial aspects of the design. If the outcome is simply a student project, the tendency is to pick projects that are grounded within the student's own life and experience, which is far from representative of the larger arena of computer use.

The experiment which I am advocating, then, is to create a course on human-computer interaction in which students are challenged to develop competence in design through a process of guided learning. There are many difficulties to be overcome, and many creative possibilities. Some departments (for example in Scandinavia) already offer courses that put students into a computer application context outside the bounds of their academic world. Others, such as a course at Carnegie-Mellon, ask the students to find a "real user" for whom they will build a small system. There are also rich possibilities for cooperation between the university and the computing industry, in which the actual design enterprise becomes a resource for teaching, and experienced designers outside of academia can participate in the coaching. The exact form of such courses is yet to evolve, but the framework presented above offers a sense of direction.

I see this, then as the challenge for the second decade of the field of human-computer interaction. In order to bring our concerns and our understanding into the center of computer science education, we need both to get our own act together and to get computer science to shift its center. In doing this we have one tremendous ally–the external pressures that society and the computing industry will be generating to educate our students to a level of design competence that is all too rare today. But pressure alone doesn't make a dent, unless it has a sharp point on which to push. You here at this conference are on the cutting edge and are honing that point. I look forward to the breakthroughs that will come.

BIBLIOGRAPHY

Mitch Kapor, Personal Communication.

Fernando Flores and Michael Graves, Education, Emeryville CA: Logonet, Inc., 1986.