Human-Computer Interaction in the School of Computer Science

Edited by
Bonnie E. John, Philip L. Miller, Brad A. Myers,
Christine M. Neuwirth, and Steven A. Shafer

October 1992
CMU-CS-92-193

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

This document represents a collaborative effort on the part of the SCS faculty interested in Human-Computer Interaction, with major contributions from David Garlan, Wilfred J. Hansen, Michael L. Horowitz, Bonnie E. John, Roy A. Maxion, Philip L. Miller, James H. Morris, Brad A. Myers, Christine M. Neuwirth, Dean A. Pomerleau, Steven F. Roth, Steven A. Shafer, Mary Shaw, and Maria G. Wadlow

Abstract

The SCS faculty interested in Human-Computer Interaction (HCI) presents its position on what role HCI can play in Carnegie Mellon's School of Computer Science. We present a short description of the need for HCI research and recommend a task/human/computer approach to satisfying that need. After presenting illustrative research scenarios, we draw implications of adopting this approach for our research and educational programs. SCS is well positioned to implement this approach, given the interests and skills of our faculty and faculty in other organizations at CMU. We recommend that the Computer Science Department form a new area in HCI.

Keywords: Human Computer Interaction (HCI), strategic plan, curriculum

Executive Summary

The Problem. Computers have the potential to revolutionize many areas of human endeavor--delivering memory, data processing, communication, visualization, and control power for performing all manner of tasks. But many systems fail to reach their full potential because they are daunting or difficult to use. For every successful computer system (e.g., automated teller machines) there exist scores of unsuccessful systems ranging from the comic (e.g. the international joke of programming VCRs) to the potentially tragic (e.g., nuclear power plant control systems).

An Approach. We argue that a fundamental tenet of computer science is that, ultimately, *people* use *computers* to accomplish *tasks*. This implies that an understanding of human capabilities and tasks is as important to the design of effective computer systems as an understanding of computer technologies. The pursuit of understanding the triad of task/human/computer is the purview of the relatively new, but active and growing field of Human-Computer Interaction (HCI).

Recommendation. We recommend that SCS include an area in HCI that, within itself and in cooperation with the other SCS areas, performs research on HCI issues, develops systems using HCI methods of design and evaluation, and trains students in the theory and skills necessary to become HCI professionals.

Probability for Success. We believe SCS can contribute extensively in the HCI arena for both historical and current reasons. Historically, we have had success in many of the relevant subtasks (e.g., building and deploying real systems, which can demonstrate the effectiveness of HCI techniques; training students who have gone on to be leaders in the field of HCI; and fostering good working relationships with other departments at CMU that will contribute to our understanding of HCI issues). Currently our faculty (both SCS and elsewhere at CMU) have strong interests in various aspects of HCI and most of the requisite skills necessary to do important and influential research in HCI.

Structure of this Document. The first section details the problem, including a few short examples of effective and ineffective systems. The second section asserts the task/human/computer triad as an approach to solving this problem that SCS could adopt and describes an implementation of this approach through example research scenarios. The third section makes recommendations for fitting the training of HCI professionals into the existing structures of graduate and undergraduate education in SCS. The last section presents evidence that adopting this philosophy has a high likelihood of success in SCS, including a taxonomy of HCI research, placing ongoing work at Carnegie Mellon within that taxonomy.

1. The Problem

While a few computer systems are designed to perform tasks autonomously, the overwhelming majority have an interface through which humans interact with the computer system to accomplish a task. A computer system's interface is one of the most important factors in determining a system's success or failure (*cf.* Markus, 1983). Moreover, the interface is a relatively costly factor: estimates concerning the fraction of code required to support a system's interface range from about one third to one half (Bobrow, Mittal, & Stefik, 1986; Myers & Rosson, 1992).

A good example of a successful interface between humans and computers is the automated teller machine (ATM). For the most part, these machines are tremendously successful at handling a number of banking tasks. An example of an unsuccessful interface is the VCR: the average person's inability to program one has become an international joke. If ATMs have been so successful, why are so many other user interaction systems so abysmal? Two reasons suggest themselves. First, perhaps it's because we don't understand the implications of "scaling up" in the domain of human-computer interaction (HCI). Certainly part of the reason for the success of ATMs is that their tasks are few and of limited scope. User goals are usually well defined, and information and instructions displayed on ATM screens can be readily matched to users' goals. Moreover, most people are familiar with the common banking transactions handled by ATMs, so training is not a significant issue. As a result, users find it easy to make an educated guess at how to use ATMs to accomplish their goals. As tasks become more complex and less familiar, as the amount of information to be absorbed increases, and as decision making encompasses more and more variables, the best ways to facilitate interaction between humans and computers are little understood.

However, the task of recording TV shows is only barely more conceptually complex than withdrawing money from the bank, so scale-up cannot be the source of the VCR's legendary problems. A second reason this and so many other interfaces are abysmal, may be that designers need more awareness of HCI concerns and better training in the theory, research, and practice of HCI. As long as the speed, size, and cost were the limiting factors in computer systems, the emphasis of computing practice and training was placed on developing faster and cheaper computers and algorithms to process data. Programs were written, and devices designed, with little or no regard to the needs of the people who were to use them (or their needs could be anticipated through introspection because most of the users were technical people, just like the programers). Programs were also conceived as individual stand-alone entities rather than as pieces of an interwoven fabric of people, equipment, information, and data processing tasks. Yet, such a fabric is actually the characteristic of real workplaces, in which computers are part of larger integrated systems of people and machines. Because of these historical forces, computing is still viewed as a somewhat exotic technology, with only the most mundane computing tasks accessible to the general user population, and more sophisticated tasks accessible only to a few highly technical "gurus." Computing has not realized its full potential to aid all people in all activities and the reason is the lack of attention to the interaction of humans, computers, and task requirements in workplace systems. In the last decade, some very fundamental advances have occurred such as graphic interfaces to computer programs, WYSIWYG word processing, crude speech input, and sketch interpretation, which suggest the direction that human-machine interfaces might evolve, but do not by themselves definitively solve the problems. There is a need for substantial further research in HCI to unlock the potential of computers to contribute to all aspects of human activity, and a need to integrate existing results into the computer science curriculum.

This is the time to reverse the historical pattern of computing development. Today, computing hardware and software have advanced to the point that computing power in massive amounts is available cheaply and in small packages that can appear at many places in the office, home or other environments. With this ubiquity, speed, and capacity, vast amounts of raw data can be collected, generated, and presented to a user, and the limitation on completing a complex task has become the user's ability to sift through and interpret this data to produce useful information, rather than the computer's ability to store or process it. In very complex systems, if information is poorly presented it can overwhelm users, with problems or even disasters as a result. Thus, the ultimate test of computing is whether it delivers real, important information to people with real, practical problems, in a form that they find useful, effective, understandable, and significantly faster and cheaper than the alternatives.

2. An Approach: The Task/Human/Computer Triad

To summarize the previous discussion in the form of a central tenet of computer science: Ultimately, *people* use *computers* to accomplish *tasks*. This implies that an understanding of human capabilities and tasks is as important to the design of effective computer systems as an understanding of computer technologies.

2.1. The Task/Human/Computer Triad

The pursuit of understanding this triad of task/human/computer is the purview of the relatively new, but active and growing field of HCI (The field has been actively researched for at least fifteen years, and it has had its own conferences and journals for about ten years).

Our understanding of the triad of task/human/computer is depicted in the following figure:

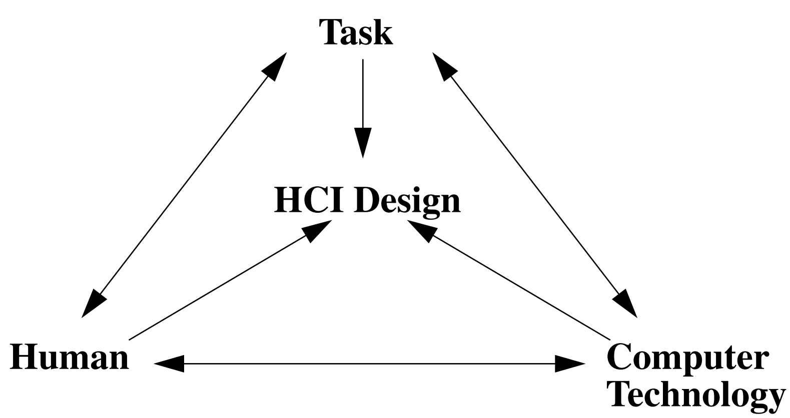

Figure 1. The Task/Human/Computer Triad

According to this figure, a fundamental element in the design of effective systems is to ascertain what tasks users need to carry out. Frequent tasks may be relatively easy to identify, but occasional tasks, exceptional tasks for emergency conditions, and repair tasks to cope with errors in using a system that may not have been built yet are typically more difficult to discover. Methods for doing task analyses more effectively, for representing tasks, and for mapping tasks to user interface functionality are active research areas in HCI. Research in HCI seeks to understand tasks with a view to designing computer technologies to enhance their performance.

A second fundamental element in the design of effective systems is an understanding of the people using them. Depending on the task, this understanding may need to include their perceptual capabilities (capacity to identify an object in context, color vision and deficiencies, peripheral vision, contrast sensitivity, motion sensitivity, etc.), cognitive characteristics (short-term and long-term memory, learning, decision making, attention and set, search and scanning, etc.) and their social/political and organizational contexts (power relationships, organizational structure, communication patterns, goal conflicts, etc.). To enhance this understanding, HCI research produces methods for measuring, modeling, and evaluating human performance with computer systems.

The third element, computer technology, includes visual displays, manual input devices, and, more recently, sound and gesture recognition; interaction styles and techniques (e.g., command languages, menus, natural language interfaces, direct manipulation, demonstrational interfaces, etc.); and user interface development tools and systems. These areas are the traditional realm of computer science and will continue to play an important role in any task/human/computer system.

The elements of this triad interact extensively, hence, double-pointed arrows connect the corners of the triangle. For instance, technologies influence people (e.g., different computer response times cause people to choose different strategies for accomplishing a task, Teal & Rudnicky, 1992). Conversely, human capabilities influence technologies (e.g., the limitations of human memory have influenced the design of menu-driven interaction styles). Tasks influence human capabilities (e.g., people change their knowledge about the task as they attempt to perform the task, i.e., they learn). Human capabilities influence the tasks people need to perform (e.g., people simply cannot remember all the information in a database, so a search task arises). Finally, different tasks call for different technology (e.g., text-editing can be done naturally with a keyboard, but graphic editing requires a drawing device like a mouse or pen) and technology can enable or limit tasks (e.g., spreadsheets have unleashed hundreds of new tasks).

2.2. HCI Research Scenarios

Research around the periphery of the task/human/computer triad can inform the design of computer systems. The knowledge and techniques resulting from HCI research can help meet the challenges that arise when designing systems of people and artifacts to accomplish complex tasks. Design, in turn, provides the arena within which research problems in HCI can be identified and investigated. To illustrate the potential contribution of HCI in concert with other areas of computer science, the remainder of this section describes the challenges faced in three hypothetical research areas: computer-aided laparoscopic surgery, crisis action planning, and software development.

2.2.1. Computer-Aided Laparoscopic Surgery

Laparoscopic surgery is a technique in which narrow instruments and fiber-optic cameras are inserted into the patient’s body through narrow tubes (5-10mm diameter). The images are presented on a video screen and the surgeons look to that screen to view their working area. Currently, laparoscopic surgery is an attractive alternative to traditional surgery because it needs a much smaller incision and, therefore, requires substantially shorter recovery times. However, the procedure is considered difficult and is practiced by relatively few, highly-trained, surgeons.

Medical and engineering researchers believe that image processing methods and robotic technology could greatly facilitate laparoscopic surgery. For example, technology can provide the ability to look into the patient's body—much like Superman's X-Ray vision—with the surgical team wearing virtual reality goggles and able to operate as if seeing through the patient's skin. Another improvement would be manipulators that could perform tasks such as stitching and tying sutures at a distance, allowing the surgeons' movements to be relatively simple while the instruments perform complex manipulations and supply tactile feedback to the surgeon. While this technology is exciting in its own right, our interest is in the HCI challenges that such an application raises.

What data do surgeons need and how should the surgery team interact with it? Many technical possibilities for the display of data can be imagined. For instance, a system could enhance the video information collected by the fiber-optic cameras, perhaps by superimposing images of solid models or other interpretations added by the computer. These models might have been derived from the image data itself, or from other processes such as a prior CAT scan. The interpretation data may be very direct, such as solid models of organ structures, or it may be very indirect, such as measurements of critical quantities or dimensions, or information from reference texts or other sources. Such abstractions are themselves an important topic for HCI research in data visualization. Additional information might include advice from artificial intelligence agents, such as a circle drawn around anomalous tissue that was detected by the agent, or a presentation of imagery from previous surgery for comparison to analyze change over time. The data could be displayed at any convenient size, allowing the surgeon to zoom in on features of special interest and examine them in more detail; and with 3D modeling in the computer, the surgeon could even "walk around" the structures to see them from all points of view. In such scenarios, systems would need to communicate not only the data itself, but also the limits of the actual knowledge of the system, so that the surgeon could distinguish what is known to be present from what the computer "imagines" to be present as a result of computations and abstractions from the actual data.

On the input side of the system, traditional means of inputting information and commands to computer systems, namely a keyboard and a mouse, are impossible for a doctor to use during surgery. Alternative modalities are required for effective interaction. For instance, speech recognition could be used to call up relevant images from previous operations. Tracking of the surgeon's gaze might take the place of a mouse, allowing the surgeon to point at internal structures without releasing the laparoscopic instruments. The key to effective communication is not to simply provide alternative input modalities, but to carefully analyze how each modality can most naturally be integrated into the surgical procedure.

Beyond the imaginable technical possibilities, however, is the issue of what information the surgical team needs, and how they might interact with it. These questions raise challenges in task analysis of surgeons' activities and cognitive analyses of needs and problems. What cognitive problems do surgeons experience when doing this task and how can they be alleviated? What data will augment surgeons' performance? What is the best way to obtain the answers to these questions?

What is the best way of gathering data about the task and about the surgeons? Unobtrusive collection of information about the task seems crucial. Surgeons, like fighter pilots, are unlikely to provide protocols while in the midst of working. Methods of measuring the surgeons' activities, such as non-intrusive eye tracking or other monitoring methods, are needed to provide information about the surgeon's focus of attention without interfering with the surgical procedure. In addition, we need to develop techniques for obtaining reliable information about knowledge representation, and perceptual and attentional focus, off-line from the actual performance of the task. The play-back and simulation technology described for training (see below) may provide opportunities for gathering such information.

What are the perceptual and psychophysical parameters of the laparoscopic instruments and how can they be better designed? Another research area is the manipulator instruments themselves. Beyond the classic robotics issues related to designing manipulators that accurately perform the task (e.g,. tying a knot in a suture), are the HCI issues of how the instrument interacts with the surgeon. In traditional surgery, the surgeon received tactile feedback from the instruments, but in laparoscopic surgery today, the long, thin, narrow instruments have reduced tactile feedback and the surgeon can only know that the manipulation of the instrument was successful by looking at the video monitor. Computer-enhanced instruments could provide additional feedback, tactile and auditory as well as visual. Research is needed not only in the development of sophisticated control and tactile sensing and input devices, but also in assessing the added value of these modalities. This would include basic studies of each modality, starting with the perceptual and psychophysical parameters of each. When instruments are developed that carry out even more complex actions, they may need to provide initial "what-if" information to the surgeon before actually going to work. If such actions form a sequence, the surgeon would need the ability to visualize and possibly modify each step of the plan before the device begins operation.

How do members of the surgical team coordinate and what problems of coordination are there? Surgery is a complex human coordination problem, and it is unlikely that any artifact, no matter how well conceived, will fit easily into the already finely tuned surgical world without a detailed understanding of the task, the team, and the organizational context of surgery. For example, the coordination of many different data sources (such as the anesthesiologists' data) so that the team could see the whole picture calls for careful design of visual displays and understanding of the cognitive load on various team members. Or perhaps the rapid formation of teams with the best set of skills is a problem. Once the view of the patient's internal organs is in electronic form, and the surgical team is wired to electronic input and viewing, the computer could provide access to off-line information and analysis. For example, a remote expert might guide the operation from a distance, thus leveraging the expertise of a world-class specialist across many locations or projecting expertise into an otherwise inaccessible or dangerous area. Here, developments in computer supported cooperative work could contribute to mechanisms such as "floor control" (coordination of many participants), flexible coupling of views among team members, or the formation of temporary teams. This raises issues in technologies for reliable, distributed, synchronous coordination. Finally, surgery is probably one of the most intense group activities known, and there could be great scientific value in studying a collaborative activity that has already been the subject of intense methodological development by the practitioners themselves. (Contrast this, for example, with the study of group software meetings, where the perceived urgency of the activity and coordination skill of the participants are very low.) Except for the close maneuver of jet aircraft, there is probably no more highly studied and trained-for group task than surgery.

How should the system be evaluated? Any new surgical device or procedure must be subjected to extensive evaluation before it is tried in the field. The talents of human factors experts and empiricists will be essential in the early evaluation of these devices; they cannot simply be distributed on computer disks and tried by a vast user community. Instead, one must design tests and experiments that will establish the effectiveness and safety of the new procedures. It may be possible to leverage synthetic virtual reality and simulation models, such as cognitive models of users, to generate new methods of testing effectiveness. This might lead to a reduction, though probably not elimination, of the need for extensive testing on animals.

How do surgeons learn the tasks and can the system itself be leveraged to help in their training? The same systems that record and enhance the visual aspects of surgery, and provide tactile and auditory feedback through the instruments, could be used to "play back" particularly elegant surgical procedures. They could be paused to allow an instructor to discuss a fine point, fast-forwarded over the routine aspects, reversed and shown again for emphasis, etc. The student could see, hear, and feel what the expert experiences. Also, a veridical simulation could provide realistic practice for the student. To make effective use of these techniques, research is also needed in how people acquire skills and knowledge through simulated experience in addition to book-learning or through lectures.

2.2.2. Crisis Action Planning

The previous scenario highlights challenges faced in developing technologies for a relatively small, well-orchestrated group of highly trained experts. This scenario focuses on emergency response to an earthquake or other crisis. It illustrates the HCI challenges for situations requiring time-critical, large-scale, coordinated efforts of diverse user populations. These situations are characterized by:

- the need to collect and analyze large amounts of rapidly changing, unreliable, and incomplete information from numerous, diverse sources;

- the involvement of many organizations that have different perspectives and methods for solving problems and that must provide each other with information and analyses;

- the need to create and iteratively revise effective plans and schedules involving highly constrained and interrelated activities and resources.

In this scenario, we focus on the related tasks of acquiring, visualizing, manipulating, and communicating information, and the support of team coordination, as organizations cooperate to plan and execute emergency relief.

How can data be collected reliably and quickly during a crisis? Effective response to crises depends on the ability of command centers to receive accurate descriptions of damage, medical and fire emergencies, food, water and evacuation needs, and many other conditions. Information is often gathered and reported by people inexperienced at identifying features relevant for response (e.g., minimally-trained volunteers may be able to report a fire, but may not think to report its magnitude, the potential for spread, the building material, etc.). Information collected by different organizations may overlap, leave gaps, or use different vocabulary. Information collected in the field is communicated slowly, either verbally via overburdened radio channels or on paper by courier. People often don't know which command centers to contact for each problem.

From the technology viewpoint, suppose disaster workers could be supplied with a hand-held computer and radio communication device that enables interactive collection of relevant information for a variety of situations. Capabilities might include reporting assistance, hands-free or pen-based input, efficient and secure data-burst transmission, automatic location-sensing, call directing, and integrated voice and data communication. However, even if the technical foundations for such a device were feasible, its success would depend heavily on careful HCI analyses. It is necessary to understand the capabilities and expertise of differing users, the relation between the task of emergency reporting and the other tasks users perform in the field, the demands of environments in which users work (e.g. lighting levels), the kinds of messages that must be encoded and whether they require verbal contact or verification from recipients, the legal, social and historical constraints on who can report different types of information (e.g., can untrained volunteers provide legally acceptable damage reports for federal or commercial financial assistance?), and the relation between this device and others (e.g., is it integrated with ambulance radios?). All of these questions illustrate that an HCI approach requires a broad view of factors influencing total system usability.

How can information from diverse sources be integrated for shared use? Once information reaches a command center, it must be integrated with other information to form the basis for decision-making, a staggering technological task. Relevant information may already exist in many disjoint databases: topographic maps, aerial photographs, utility company records, inventories of supplies, and so forth. Currently, human experts from each of the organizations that have existing databases manipulate their respective databases independently and verbally share information with other individuals, but if experts are unavailable or if individuals do not know which organization's database contains the necessary information, this system breaks down. Computerized intelligent agents could be used to understand the content of databases, maintain updates, manage their diverse interfaces, and integrate the results of complex cross-referenced queries, so that any individual could easily access the necessary information in any database. Again, however, social and organizational aspects of the task must be considered (e.g., some databases may be proprietary under normal circumstances; how might this affect access to information in a national disaster?). In addition to the relatively static databases listed above, dynamic databases will be required to continuously update the data with information flowing in from the field (with all the associated reliability problems) and other, unforeseen databases may need to be incorporated into the system on the spot (e.g., Taco Bell was an unforeseen organization participating in the Hurricane Andrew response; such organizations would bring their own databases of food inventories and mobile kitchen availability).

How can people use this information, individually and in groups? Assuming the technological problems of data integration are solved, these data must ultimately be used by humans to aid in their response planning. With so much data, the issue of effective presentation and manipulation of information is of primary importance. Preliminary analysis of the crisis management task reveals that many tasks are map-based, (e.g., implementing mass care requires identifying where the victims are, where the food is, what routes are passable between the location of the food and the location of the victims, where the ambulances are, etc.). Even assuming a map-based display of information, a raft of HCI questions arise. How should different types of information be encoded on the map, using what dimensions (e.g., shape, color, size)? How can uncertainty, or the continuous nature of some information be depicted (e.g., the degree of impassability of roads)? Given space limitations of a map, how do you handle over-crowding of information? How can other modalities, like animation or voice annotation, be effectively used? What interactive techniques are effective for manipulating data (e.g., crossing off totally impassable roads with gesture recognition)? How might different subtasks, or different users (e.g., color-blind volunteers) benefit from different displays of the same information? Can automatic display-generation systems help solve some of these problems?

In addition to questions that pertain to individual use of data, the group-work aspect of crisis action planning introduces even more complications. How can a shared display adapt to the different conventions of information display that people from different organizations might bring with them? How can several people manipulate the same data presentation at the same time? What additional complications arise if some of those people are remotely located? Might serendipitous benefits be realized from public display of this information, that is, providing the map on an entire wall of the command center so that everyone could see it at all times, rather than confining it to the screens of workstations only available on demand?

How can we promote communication and collaboration among groups developing courses of action? The preceding questions concerned the collection, manipulation and display of information directly related to the crisis at hand. In addition, crisis action teams need information about the roles, the make-up, and the status of the teams themselves. Anecdotal evidence suggests that this information is often lacking in crises; people report that they do not know whom to contact to report or ask for information and critical roles are often duplicated or omitted. Again, some of this information can be pre-planned and built into a system prior to the occurrence of a crisis. For example, the Federal Response Plan lays out the anticipated responsibilities of several organizations involved in disaster response (e.g., the Red Cross, the National Guard, etc.). Such information could be pre-programmed into a database with easy access for all participants. As with the physical data (e.g., road conditions), however, even these anticipated roles are dynamic during a crisis and must be kept up to date. As noted previously, unanticipated parties may also respond (e.g., Taco Bell) and it would facilitate planning if their capabilities, availabilities, and activities were tracked in a central facility. In times of crisis, however, these organizational bookkeeping activities often seem less important than actually providing services to the victims; therefore, the overhead of maintaining this information in traditional ways may prove excessive. This well-known problem might be circumvented by clever design of the more task-oriented systems. That is, since all the groups will want data from the central information systems, those systems could be designed so that their use has the side-effect of updating the organizational database.

How should technical solutions be evaluated? Because of the number, complexity, and inter-relatedness of activities that need to be coordinated in large actions like these, it is difficult to assess the effects of individual technical solutions. An empirical, laboratory, hypothesis-testing approach to evaluation, often used to choose between interface possibilities in less complex systems, is inappropriate for testing the entire system. More sophisticated empirical techniques, models of interaction, or other analysis tools will have to be developed to test software and hardware devices before they are put into the field. In addition, more sophisticated measures than the typical performance time or occurrence of errors need to be developed (e.g., the quality of plans and services needs to be made operational). At very least, the software and devices should be built with the capability of automatically recording their use, so post-disaster analysis of what parts of the system were actually used can be performed.

2.2.3. Computer Program Development as a Human Activity

Software development is a pervasive and central task in computer science whose nature is changing rapidly. The traditional practice of coding systems from scratch, of maintaining those systems in terms of that code, and of relying on the virtuosity of individual, skilled programmers has not scaled to the needs of large, complex system development. In its place we can expect future software development to center around component-based software construction, reusable software design frameworks (or architectures), and distributed, cooperative interaction between developers.

In this context traditional tools and their interfaces will no longer suffice. Instead of simple text editors, compilers and debuggers, which are oriented to a coding-from-scratch development process, developers will need sophisticated software design tools. We envision that these tools will allow new software to be interactively and incrementally built from existing components, permit the codification and dissemination of good software engineering practice, aid the process of cooperative software development, and raise the level of abstraction at which software is developed and enhanced. Such tools will require advancements on a number of research fronts: here we are primarily interested in the part that HCI can play in unleashing the full potential of the new approach.

These new developments in software engineering raise the following HCI challenges:

Will tools for building software from the "interface toward the computer" improve designers' performance? Increasingly the developmental bottleneck for many systems is the construction of the user interface software. Based on emerging HCI research we can envision a whole new paradigm for constructing such systems: software is built by designing the user interface first, rather than adding the interface onto separately implemented functions. In this paradigm, users with little special training in programming use graphical displays and tools to indicate what the result should be. These interface construction tools would aid in analyzing the interface, to assess, for instance, its demands on human performance, how much it requires end-users to remember, what cues it gives the users as to what to do next, the efficiency of user actions necessary to accomplish benchmark tasks, and what errors might be likely to occur. Such immediate HCI analysis tools would encourage exploring many interface alternatives and iterative design.

What representations of design information are useful for software designers? Design information is currently represented by voluminous textual documents. The situation must be substantially improved. One plausible approach would use rich graphical representations for characterizing system structures. This would go well beyond the simple graphical design notations that are used today. For example, instead of using simple lines to depict component interactions in a design, it should be possible to use a varied set of visualizations to distinguish between kinds of interaction: dataflow, procedure call, broadcast, and so forth. Relationships between parts of a design could be handled in a similar way. For example, a graphically-depicted design could be turned into executable code by having the system designer indicate visual links between library components and the parts of the design that they implement. The contribution of HCI may not be in providing the insight that leads to the discovery of a new representation for design information. It may be, instead, to determine which candidate representations would be most useful to software designers by understanding the designers and their task through cognitive modelling and empirical study.

What coordination problems do teams of programmers experience and can tools be devised to alleviate or eliminate those problems? We envision an environment in which multi-person software development is supported by tools that help developers coordinate their interactions. Such tools would provide coherent visible access to the large amounts of data that define the current status of a project. They would integrate smoothly the various functions of code re-use, version and configuration management, project tracking, cost and time estimation, bug tracking, and resource sharing. HCI can help us tune such a collection of tools. It can tell us when a tool is functioning well, when a tool must be polished, redesigned, or discarded. Another exciting application of HCI research in cooperative software development is in the management of shared knowledge. We look to an active system that helps direct its multiple users. The system has a model of what its users know, what they know in common, what knowledge is available from software component libraries, and what the users need to know to make progress. This work will not only push ahead software development, but it will also make contributions to HCI and the foundations of cognitive science.

What do good designers know and can that knowledge be represented in a way that would augment designers' performance? Good software developers gradually acquire a body of knowledge about good system design. Currently this knowledge lives in the heads of those designers, however, or is scattered throughout the documentation of particular systems. We can imagine support for software development in which this information is codified and made available to other developers. This could take the form of on-line handbooks. It could also be embodied in semi-automated *design agents* that assist users in applying the knowledge to particular problems. Under this scenario, a user would move from system requirements to an implementation by consulting handbooks or interacting directly with design tools that subsume large parts of the analysis and selection. These handbooks would be tied into the component and design libraries so that design is smoothly integrated with component selection.

What human-computer interaction data is most important to measure when developing a system and how should that data be represented to software designers? We can imagine a world in which large-scale applications are developed by gradual transformation of prototypes into working systems. To make this possible, the development environment will need to provide strong interface support for keeping track of the system parts that are not yet fully instantiated and by allowing developers to test and analyze incomplete systems. An essential aspect of this approach will be the use of non-intrusive monitoring. As a prototype evolves, data will need to be collected and analyzed in preparation for the next iteration. The key HCI concerns here are to provide developers with ways to know what to monitor and with appropriate visualizations and organizations of data that allow the developer to analyze the results of the monitoring activity.

2.2.4. Summary of Research Scenarios

We believe that these three scenarios have illustrated the richness of the HCI approach to designing task/human/computer systems. Task analysis, the consideration of human capabilities and propensities (both cognitive and social), as well as the design of new technology, permeates these examples. They illustrate the necessity for the knowledge and skills of interdisciplinary teams of researchers and developers, collaborating for the common goal of delivering an integrated, usable, effective system for accomplishing the task at hand.

3. HCI Education in SCS

Success in the projects outlined above will require not only awareness of HCI issues by all involved computer professionals, but also the skills of specialists trained explicitly in HCI tools and techniques. We recommend that our educational programs be modified to provide this awareness and training. Some teachers throughout SCS have already included HCI material in their courses, but it is now time to coordinate these efforts and augment them with explicit instructional goals.

We recommend that every student who completes a degree at any level in SCS at CMU have at least a minimum level of understanding of HCI. They should understand that much of CS involves designing systems to aid people in accomplishing tasks. Understanding the tasks, and the capabilities of the people, in relation to the systems will produce better designs and more effective systems. They should further understand that there is a field of Human-Computer Interaction that includes:

- Techniques for understanding tasks and their relationships to the people who perform them and the computers that aid in that performance.
- Knowledge about the capabilities of people and techniques for understanding the role of those capabilities in the context of tasks and computer systems.
- Knowledge about interface styles and I/O devices; when to use them, how to build them effectively into a system, and how to evaluate them.
- Tools for designing and building interfaces.
- Methods of evaluating the effectiveness of human-computer systems.
- The relation of HCI to the rest of computer science (e.g. the relation to programming languages, software engineering, artificial intelligence, verification, hardware, etc.).

3.1. Undergraduate Education

ACM SIGCHI has recently published a report on Curricula for Human-Computer Interaction which outlines HCI-oriented curricula in CS, Psychology, or Information Science departments. That report, tailored to the particular needs of SCS, is a basis for the following recommendations.

We would like to see the fundamental tenet of HCI (that people use computers to accomplish tasks, so an understanding of human capabilities and tasks are as important to the design of effective systems as an understanding of computer technology) be infused into the general CS curriculum. That is, we see many opportunities for HCI issues to be discussed in the context of other more traditional CS courses. To illustrate, the following required courses listed in the "B.S. in Mathematics/Computer Science for students entering in Fall 1991", could include material relevant to HCI:

- 15-127 Intro to Programming & Computer Science. Along with the case studies currently offered, introduce the triangle diagram of HCI systems and design. Assignments might include programming an interactive interface.

- 15-211/212 Fundamental Structures of Computer Science I & II. Introduce interaction styles (command line dialogues, menu systems, etc.) as fundamental structures in the context of programming and formal languages.

- 15-312 Programming Languages Design & Processing. Introduce the design of programming languages as an interface between the user (a programmer) and the computer.

The infusion of HCI concerns into several required courses could satisfy the educational goal set above. In addition, HCI material may also be appropriate for presentation in the following elective courses (again, for illustration, selected only from those offered in the fall of 1991).

- 15-384 Artificial Intelligence: Robotic Manipulation. Issues in the human interface to programming robots or tele-manipulation of robots.

- 15-412 Operating systems. Issues in the human use of operating systems (e.g., the command language for shell operations, help systems, etc.).

- 15-413 Software Engineering. This course description already includes "documentation" and the "design of friendly user interfaces." HCI issues also arise in the software development process itself (e.g., collaborative version-control systems).

- 15-682 Engineering of Knowledge-Based System. Issues in the user interface of knowledge-based systems (e.g., what information about the system's reasoning needs to be presented to the user to allow the user to evaluate the system's conclusions).

We understand that the relative newness of HCI as a field, and its interdisciplinary nature, often makes it difficult for instructors to gather materials appropriate for undergraduate coursework. The HCI faculty will be available to instructors wishing to include HCI concerns in their classes, to help develop "modules" of a few lectures and supporting materials that can be incorporated into their courses. We expect that this infusion process will develop over time, as faculty interest and opportunities allow.

We recommend the eventual introduction of two HCI courses as undergraduate CS electives to provide a solid background for our students interested in concentrating in interface development or HCI research. The ACM SIGCHI Curricula report suggests two such courses for a CS curricula. The objectives of their HCI-CS1 are "to provide an adequate basis in software design and implementation for user interfaces...[and]...an appreciation of the importance of further subjects covered in CS2.” The objective of their HCI-CS2 is to “train students in the underlying science and its application to user interface design. It is an engineering course replete with engineering models drawn from psychological theory.” We suggest that for SCS’s program, courses similar to these be developed, but that they be offered in reverse order. The first course presenting the HCI issues and underlying science, and the second course using that science to build systems. These courses would eventually be at the 300-level and have an associated project.

In addition, there could be more specific HCI-related courses, depending on faculty interests. Advanced undergraduate courses (400-level and above) might be cross-listed with the Masters or Ph.D. courses.

3.2. Masters-Level Education

The specialized knowledge and skills that are required of HCI professionals go beyond what can be taught in an undergraduate CS curriculum. Currently at CMU, skills at this level, coming from diverse disciplines, are delivered through interdisciplinary “professional masters” programs, e.g., the degree program given by the Information Networking Institute, a joint venture between the Graduate School of Industrial Administration (GSIA) and SCS, or the Masters of Software Engineering (MSE) program, a joint venture between the Software Engineering Institute (SEI) and SCS.

A professional masters in HCI would teach practical skills useful to the design of task/user/computer systems, and be interdisciplinary with other appropriate organizations at CMU, possibly the Departments of Psychology or Social & Decision Sciences, or GSIA. We expect that a professional masters program in HCI would require between one and two years, with at least 40% of the time occupied by hands-on project work. Courses might include the skills that are taught in tutorials at the ACM-SIGCHI conference. For example, design of visual information presentations, building computer-supported cooperative work interfaces, experimental design in HCI research, GOMS analysis, cognitive walkthroughs, the task-artifact framework for HCI design, icon design, and so forth. Course offerings might be shared with the elective graduate courses in HCI, presented below, or with advanced undergraduate courses.

An evolutionary path to a fully-formed professional masters in HCI might be through an HCI *track* in the MSE program. An MSE-HCI track would consist of 3 to 4 semester courses within the regular MSE program. This track would have at least one pure HCI course and additional material and experiences that are HCI in nature (e.g., Research Methods in Cognitive Psychology). The HCI course may be the same one delivered in the undergraduate program, with an additional, substantive project for the graduate students.

In either Masters-level program, project work would be required to exercise the skills taught. Therefore, the HCI faculty are prepared to supervise Master-level project work, either individually or in studio courses.

3.3. Ph.D. Education

Meeting the previously stated goals of developing an understanding of HCI for Ph.D. students would not require a Core Unit in the Ph.D. curriculum. Instead we propose a series of lectures along the lines of the 3-hour “Introduction to HCI” tutorial at SIGCHI. Students wishing to learn more about HCI would have ample opportunity in the electives discussed below.

The existence of an HCI area in SCS will generally raise the consciousness of all students pursuing work that creates artifacts. We hope that this will cause more attention to be paid to the user interfaces of such artifacts. In addition, we expect that students wishing to claim that their systems allows some person, or team, to be more productive, will want to substantiate their claims by techniques that are well established in the HCI community. The HCI faculty are available to help with teaching and/or overseeing the execution of these techniques, including serving on the committees for theses wishing to make such claims.

Some students will concentrate their study in the area of HCI. It is important that these students demonstrate at least a rudimentary mastery of the skills of the trade. The skills are important for three reasons. First, they will help the students in doing their own research. Second, they will facilitate an understanding of the research of others in HCI. Third, they provide a foundation for working as part of interdisciplinary team on projects. These skills should be demonstrated in course or project units, prior to beginning the thesis. They include:

- Build a system, or modify an existing system, with a UI development tool and/or prototyping tool or UIMS.
- Perform an empirical evaluation of an existing system or system component.
- Use a formal analytic method to evaluate a task/user/computer system prior to its implementation. (Typical formal analytic methods include GOMS, cognitive or programming walkthroughs, and task-artifact claims analysis.)

A thesis project in HCI will not be confined to a single corner of the task/human/computer triad. Rather, an HCI thesis will investigate the influence of a theory, method, or system on at least one of the other two corners and/or articulate what is learned about designing a computer system.

Elective graduate courses in HCI will serve two purposes: to teach and exercise the rudimentary skills discussed above, and to explore research issues in the field of HCI. The skills courses may overlap with Masters-level coursework in HCI and are listed above. The research issues courses will reflect timely interests of the faculty and students. Such research issues might include:

- Comparative cognitive architectures for HCI modeling.
- User interface software.

- Visual programming techniques and methods for their evaluation.
- Advanced interface techniques and multimedia.
- User-centered design techniques.
- Computer-Supported Cooperative Work: study, design and implementation
- Intelligent interfaces.
- HCI in Robotics.
- Evaluation/measurement techniques.

4. Probability of Successful HCI at CMU

We believe SCS can successfully implement the task/human/computer research and educational approach for both historical and current reasons. Historically, we have had success in many of the component parts of this task, for example, building and deploying real systems (which can demonstrate the effectiveness of HCI techniques). Over the past twenty years, we have trained graduate students who have gone on to be leaders in the field of HCI. We have fostered and maintained excellent working relationships with other departments at CMU, (e.g., Psychology and GSIA) that will contribute to our understanding of HCI issues.

The most compelling reason for optimism is that our faculty (both SCS and elsewhere at CMU) have strong interests in various aspects of HCI, and the requisite skills necessary to do important and influential research in HCI. This section illustrates the breadth and depth of these interests (i.e., the *critical mass* of HCI research already underway) by listing HCI research areas and the faculty at CMU who are already doing research in those areas.

Understanding Human Information processing

— How do people learn and perform computer-based tasks? If we understood better how people operate, then it would be easier to design and evaluate user interfaces. This research is sometimes performed by cognitive psychologists.

- Models of cognitive architecture (can we create a computer program that performs similarly to the way people do?) — John Anderson, Bonnie John, Jill Larkin, F. Javier Lerch (GSIA)

- How do human memory, motor skills, attention, problem solving, and learning impact the design of the computer interface (including the documentation)? — Bonnie John, Jill Larkin, Karen Shriver (English)

- The study of individual differences, including disabilities and skill deficits (e.g. in reading ability), and how these impact user interfaces — Steve Roth

New interactive techniques, devices, and modalities

— How can people and computers communicate?

- Natural language (English, etc.) understanding and generation — Robert Frederking

- Speech understanding and generation, lip reading — Alexander Hauptmann, Xuedong Huang, Raj Reddy, Alex Rudnicky, Alex Waibel

- Gestures (when the path that the input device travels through is interpreted, such as drawing an X over an item to delete it), character recognition, pen-based computing — Roger Dannenberg, Takeo Kanade, Brad Myers, Dean Rubine, Alex Waibel

- Video and other media included in an interface (multi-media) — Roger Dannenberg, Dean Rubine, Scott Stevens (SEI)

- Integration of multiple input and output media, multi-modal interfaces — Roy Maxion, Alex Rudnicky, Alex Waibel
- On-line help systems — Roy Maxion
- New interaction techniques using a mouse and keyboard — Brad Myers, Andrew Witkin
- New hardware input devices, like data gloves, eye tracking, etc. — Roy Maxion, Dean Pomerleau, Alex Waibel
- Designing interaction techniques to deal with human error — F. Javier Lerch (GSIA), Roy Maxion
- Output devices (3D, head-mounted displays, etc.) — Mel Siegel

Analysis and Evaluation

— Given that you want to create a computer-based application, how do you know what the interface should be like? After you have created an interface, how do you know if it is any good?

- Task analysis (analyzing the task the user will need to do) — Bonnie John, Roy Maxion
- Model-based analysis (using a model of human performance to evaluate a specification of an interface; this does not require the interface to be implemented yet) — Bonnie John, Alex Rudnicky
- Applying formal methods of software engineering to interactive system design (using mathematical models for system description to embody claims about usability) — Gregory Abowd
- Methodologies for evaluating implemented or prototyped interfaces using experiments — Bonnie John, Roy Maxion, Alex Rudnicky
- Usability engineering (embodying the analysis of usability into the design and implementation process) — Roy Maxion
- Usable documentation — Roy Maxion, Karen Shriver (English)

Graphic Design

— The look of the interface.

- Typography, icons, graphical layout, presentation, organization — Dan Boyarski (Design Dept.)

User Interface Software

— How do you implement an interface?

- Tools for prototyping, design, construction, and analysis of user interfaces; toolkits, UIMSs, interface builders — Dario Giuse, Wilfred Hansen, Brad Myers, Mark Perlin, Dean Rubine

- Models for interaction (transition networks, event languages, automata, constraints), architectures, verification of software against the models — Gregory Abowd, Dario Giuse, Michael Horowitz, Brad Myers, Mark Perlin, Steven Shafer

- Real-time issues, synchronization (e.g., multi-media interfaces require the sound be synchronized with the animation; how do you specify and implement this?) — Roger Dannenberg, Dean Rubine

Intelligent Interfaces

— How can AI technology make user interfaces more helpful, easier to use, and easier to create?

- Automatic creation of user interfaces — Brad Myers, Steve Roth

- Demonstrational interfaces (when the user operates by direct manipulation on an example object, but the system generalizes to a whole class of objects) — Brad Myers

- Adaptive interfaces (when the interface changes itself as it learns about the user) — Tom Mitchell, Dean Pomerleau

- Intelligent agents and intelligent help (the system proposes solutions to the user's problems) — Tom Mitchell

Computer-Supported Cooperative Work

— Much human activity involves collaborating with other people to accomplish tasks. How can computers help with this?

- Advanced forms of electronic mail — Jim Morris, Sara Kiesler (SDS)

- Tools for collaborative writing — Jim Morris, Chris Neuwirth, Dave Kaufer (English)

- Impact of cooperative work technologies on groups — Jolene Galegher (English), Sara Kiesler (SDS)

- Integrating user models and system models in groupware — Gregory Abowd

Social Organization and Work

— How does the introduction of a new computerized application affect the people

and organizations that use them?

- Impact of computers on organizations — Sara Kiesler (SDS)
- Trust in machine advice — F. Javier Lerch (GSIA)

Programming Languages as Human-Computer Interfaces

— How do humans create programs? How can programming languages and environments enable more readable, effective, and efficient expression of programmers' intentions?

- How do humans generate and understand programs? What kinds of software structures will be easiest to understand? — Wilfred Hansen, F. Javier Lerch (GSIA), Mark Perlin, Mary Shaw
- What are the best ways to teach programming — Phil Miller, Mark Perlin, Steve Shafer, Bruce Sherwood (EDRC)
- Visual Programming (using graphics and other alternative representations of programs instead of text) — Brad Myers, Mark Perlin, Steve Shafer
- Programming for end users (many people are programming for themselves using spreadsheets. How can we bring this capability to other domains?) — Wilfred Hansen, Brad Myers, Dean Rubine
- User interface aspects of Software Engineering — Gregory Abowd, David Garlan, Mary Shaw

Application Areas

— Many important advances in user interfaces have resulted from studying the interfaces for particular kinds of applications. Just a few are listed below. Section 2 of this document contains other examples.

- Computer aided design and computer aided manufacturing (CAD/CAM) — Robert Coyne, Tom Mitchell, Brad Myers, and most of the Engineering Design Research Center
- Text editing — Brad Myers
- Process control, embedded systems (copiers, appliances, military systems, etc.) — Roy Maxion, much of the SEI
- The interface to the operating system (often called the desktop or visual shell, e.g., the Macintosh Finder) — Brad Myers
- Medical systems — Dario Giuse, Takeo Kanade, Mark Perlin
- Robot control — Ikeuchi Katsushi, Dean Pomerleau, Steve Shafer, Chuck Thorpe

- Computer-Aided-Instruction — John Anderson, Jill Larkin, Phil Miller, Mark Perlin, and most of the Center for Design of Educational Computing

- The user interfaces of AI programs, knowledge acquisition — Dario Giuse, Bonnie John, Roy Maxion, Mark Perlin, Steve Roth

- Computer music — Roger Dannenberg, Dean Rubine

- Visualization of data and programs (using graphics to make them more understandable) — Roy Maxion, Brad Myers, Mark Perlin, Steve Roth, Andy Witkin

5. Recommendations

In conclusion, the Human-Computer Interaction faculty make the following recommendations.

- An area in the Computer Science Department be recognized that focuses on Human-Computer Interaction. This area, like other areas, will:
    - Have regular meetings,
    - Host an official seminar series, and
    - Deal with recommendations for hiring and promotions.
- Courses in HCI, and HCI materials to be used in other CS courses, be developed for undergraduates and graduate students, as outlined in Section 3 of this report. These courses, and the infusion of HCI materials into other courses, should begin as soon as practical and be phased in over several years.
- The department, and the HCI area, promote collaboration so that all research in the department can benefit, when appropriate, from taking the task/human/computer perspective into account.

We believe that since HCI will increasingly be an important area for research and commercial products, that this will improve the quality of the education, research, and status of the School of Computer Science at Carnegie Mellon University.

6. References

Bobrow, Daniel G., Mittal, Sanjay, and Mark J. Stefik. “Expert systems: Perils and promise.” *Communications of the ACM* 29.9 (1986): 880-894.

Markus, M. Lynne. “Power, politics and MIS implementation.” *Communications of the ACM* 26.6 (1983): 430-444.

Myers, Brad A., and Rosson, Mary B. (1992). “Survey on user interface programming.” In proceedings of CHI, 1992 (Monterey, California, May 3- May 7, 1992) ACM, New York, (1992):195-202.

Teal, S. L. & Rudnicky, A. I. (1992). “A performance model of system delay and user strategy selection.” In proceedings of CHI, 1992 (Monterey, California, May 3- May 7, 1992) ACM, New York (1992): 295-305.