Teaching Human Computer Interaction to Programmers

Saul Greenberg

Technical Report 96/582/02
Department of Computer Science,
University of Calgary,
Calgary, Alberta Canada T2N 1N4

March, 1996

Teaching Human Computer Interaction to Programmers

Saul Greenberg

Department of Computer Science
University of Calgary
Calgary, Alberta
Canada T2N 1N4
phone: +1 403 220 6087
email: saul @cpsc.ucalgary.ca

Abstract

Many computer science graduates will likely find themselves developing interfaces in a work culture that has only naïve notions of usability engineering. This article describes a course I developed that prepares students for this eventuality by providing them with practical and applicable HCI skills. All course materials are available on the world wide web, and pointers are provided.

The Challenge

Over the last decade, the interface has become a critical component of commercial software. From a system perspective, the interface dialog standard has shifted away from text-based command line systems and form-filling dialogs, to highly-interactive graphical user interfaces (GUIs). From a user perspective, people are now familiar with the relatively high interface design standard found in shrink-wrapped software, and they are less tolerant of difficult to use systems. From a marketing perspective, the customer base has moved from trained computer specialists towards a "lay" computer literate audience.

As a result, software companies producing high-volume shrink-wrapped products now include interface design teams and professionals as part of their product development groups. More recently, companies with modest audiences for their software, such as those producing in-house software or niche products, now expect that everyday programmers will design good interfaces as well as good code. Unfortunately most programmers are sadly unprepared for this job. Their traditional computer science training rarely included HCI, either because courses were unavailable in their educational program, or because such a course was considered esoteric and for specialists.

This situation is changing. Because of job demands, many computer science students now consider HCI a core skill as marketable as (say) databases and networking, and HCI courses are becoming well attended. For example, the Department of Computer Science at the University of Calgary has offered an undergraduate HCI course since 1981, but only recently has it grown from a 'specialist' course with 30-40 students, to a heavily attended mainstream course with 100 students (i.e., about three-quarters of our computer science majors).

The question that I face as an educator is how to shape students to become programmers with the background and skills required to apply HCI practices to their every day job demands. Because Alberta has a large oil and gas industry with fairly traditional data processing departments, I expect most students will work in groups where the term 'HCI' is unknown, or at best that their managers have fairly naive notions of what "good" interface design is all about e.g., that interface design is merely knowing how to program GUIs using Visual Basic. I not only have to teach students fundamental HCI principles and foundations, but have to give them skills that they could use in a work environment unfamiliar with the idea of usability engineering (Nielsen, 1993).

I created a course to teach HCI to computer science students who see it as just another skill to add to their repertoire and resume. (An abridged commercial version run over two days is also taught to industry professionals.) After taking the course, many of my students do seem to become reasonably adept at applying their learning to practical situations. Since I believe many educators are in a position similar to mine, this article detail the course and offers it to others as a useful starting point.

The following sections provided an overview of the course, the major topics covered and the rational behind them, and the practicum. All course materials are available through the World Wide Web, and include details going well beyond this article (see Sidebar 1).

Pages formatted as html	*Pages formatted as postscript*
• descriptions of each topic • all overheads • associated readings from the text • relevant videos that I show in class • in-class teaching tips • major sources I use to prepare lecture material	• all overheads • copies of all handouts • details on all assignments • notes for teaching assistants.

Sidebar 1. All course materials, structured as html and postscript pages, are available through the World Wide Web: *http://www.cpsc.ucalgary.ca/projects/grouplab/481/481.html*

A Brief Description of the Course

Purpose of the course. Human computer interaction stresses the importance of good interfaces and the relationship of interface design to effective human interaction with computers. On completion of the course, students will have theoretical knowledge and practical experiences in the fundamental aspects of designing, implementing and evaluating interfaces. Students will know what is meant good design, and will have experiences designing systems that are usable by people. Students will know contemporary techniques for implementing interfaces, and will have built applications through prototyping tools, window-based systems, and toolkits. Students will know and have practiced a variety of low-cost methods for evaluating the quality of an interface. The bottom line is that students should have sufficient skills to design, implement, and evaluate reasonable interfaces in real life work environments, even when they may not have a budget or time allowance or managerial support to do so.

Structure of the course. The course unfolds by examining design, implementation, and evaluation as a continual, integrated, and iterative process (Figure 1). Theoretical class lectures are augmented by case studies of interface successes and failures. Students apply the theoretical knowledge learnt in a series of assignments that bring them through an entire design, implementation, and evaluation cycle. The course also introduces students to novel interfaces (illustrated on video) that go far beyond today's standard graphical user interfaces.

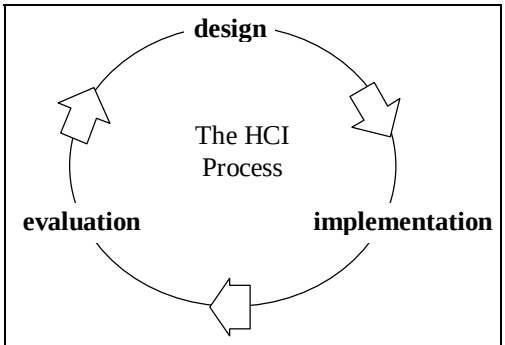


Figure 1. The course rests upon a scaffold of considering HCI as the integrated process of design, implementation and evaluation.

Course text. The course text is the recent book *Readings in Human Computer Interaction: Towards the Year 2000* (2nd Edition), by Baecker, Grudin, Buxton, and Greenberg (1995). Aside from being one of its authors, I chose this book because it contains a huge amount of material related to HCI, structured into 14 chapters. Each chapter introduces and briefly surveys a fundamental topic in HCI, includes important papers written by original authors, and has many pointers to other literature and technical video sources. Because of its richness, I can design the course around the book, rather than have the book force me down a particular curriculum path. I also feel that its breadth and depth make it an excellent resource for students to

continue reading on particular topics.

The student. Students are typically undergraduates pursuing a computer science major at the University of Calgary, and are usually in the third or last year of their degree program. They already have basic computer science skills (programming, data structures, software engineering), but only a few have taken an introductory psychology or a statistics course as one of their options. Most take the course because they see it as a marketable skill, and very few would describe themselves as interested in pursuing a specialist path in HCI.

An abridged version of this course has also been taught as a two day intensive workshop to industry. Students here are typically software system practitioners (not necessarily programmers), who are responsible in one way or another for the interface component of a major project. They take the course because they feel they do not have the skills to tackle their project in anything but an *ad hoc* manner.

The Topics

The topics taught, summarized in Sidebar 2, are structured on several major sections. "Understanding design" applies the design of everyday things to GUIs, and introduces the historical roots of HCI. "Designing with the user" includes methodologies for both designing and evaluating interfaces with direct user involvement. "Designing without the user" considers strategies for design when users are not available. "Implementing GUIs" gives students the programming foundations for building systems. Finally, "the future" guesses at the shape of things to come.

Introduction to the course.

I present students with an overview of HCI, based on the taxonomy found in the ACM SIGCHI Curriculum (Hewett *et al*, 1992), and indicate what this course will cover and what it will leave out. Videos are used in the first few classes to show futuristic and visionary interfaces. They not only inspire and motivate students, but also illustrate how many major problems in Computer Science (outside of HCI) must be solved before these visions can be realized. This is important for drawing the attention of "hard core" students who believe that HCI is not a central part of Computer Science. For example, I show the Apple 2020 video (Apple Computers Inc., 1992), and afterwards students offer and list on the board all the

Section	Topic
Overview	Introduction to the Course and to HCI
Understanding Design	Psychology of everyday things History of Human Computer Interaction
Designing with the User	Evaluating Interfaces with Users quantitative evaluation methods† qualitative evaluation methods Involving the User in the Design Process
Designing Without the User	Task-centered system design High Level Models of Human Behaviour† Design Principles and Usability Heuristics Creativity and Metaphors in Interface Design Graphical Screen Design
Implementing GUIs	Windowing Systems and Toolkits† The Tcl/Tk language†
The Future	Visions of the future

† not covered in the two day industrial workshop

Sidebar 2. Major topics covered in the course.

innovations displayed and relate them to computer science problems. They tend to be surprised at how many hard problems must be solved before these visions can be realized e.g., AI, natural language processing, hardware design, fuzzy data base queries, gesture recognition, and so on.

Understanding Design.

This section of the course gives students a fundamental appreciation of good design, and an understanding of how contemporary interfaces have evolved from ideas presented over thirty years ago.

The design of everyday things. The student's first look into design does not even consider computers. I first show them many examples of bad design in everyday things. The goal is to have students realize that human problems and errors when dealing with technology are usually a result of design failure, and that good design accounts for human capabilities. I then introduce Don Norman's principles that help us analyze bad design and create good designs (Norman 1988). I often bring to class a bag full of everyday things, includes staplers, scissors, tape-dispensers, alarm clocks, digital watches, floppy disks, CD-cases, and anything else I find lying around my office. When the discussion turns to design principles of everyday things (e.g., visual affordances, constraints, etc.), we consider how well the items in my bag work. Students bring their own encounters of bad design into the discussion, and often propose fixes to them. The feeling afterwards is that they have acquired a new way of looking at the objects in the world around them. The discussion then moves towards the visual components of GUIs. I use the video "All the Widgets" (Myers 1990) to illustrate how early scrollbars evolved from atrocious widgets with few visual affordances and arcane mappings, into reasonable 'everyday' computer objects that contain features similar to well-designed everyday things.

History of HCI. I make students aware of the intellectual and historical foundations of human computer interaction by presenting a brief history of the early major breakthroughs in HCI. I show many historical videos which the students greatly enjoy e.g., Sketchpad (Sutherland 1983), NLS (Engelbart and English 1994), and the Xerox Star (Smith and Irby 1983). Some are flabbergasted that many so-called "modern" ideas were implemented before they were born!

Designing with the User.

The course moves into the design process by considering how an end-user can be involved in the usability engineering life cycle. It begins by teaching and applying both qualitative and quantitative methods for evaluating interfaces with users, and continues by showing how programmers can involve the user as active members in the design process.

Evaluating interfaces with users. An excellent way of evaluating an interface is by watching users try it out. There are many ways to do this, and I teach a series of qualitative and quantitative methods. Major techniques covered include observational usability methods (e.g., think-aloud, constructive interaction, post-session interviews), and controlled experimentation (experimental design, hypothesis formation, statistical testing, interpretation). This topic has a heavy hands-on component (described later), where students apply both quantitative and qualitative methodologies to analyze selected interfaces.

Because I strongly believe that evaluation should occur continually through the design and implementation process, the various methods are presented as choices that one would select to fit particular problems and stages during the engineering life cycle. I also stress that a good evaluation process means that designers will catch major problems (and successes!) early on, with lesser problems being ironed out as the interface is being refined.

I have found that performing usability studies in class hammers home the relevance of evaluation. For example, I have a student think-aloud as they try to display a slide on an overhead projector rigged with a burnt bulb. It typically takes the student ten minutes to discover the problem and realize that the projector has a spare bulb that they can switch to. I also have them try to change the bulb, and it takes them another ten minutes (if they succeed!) to figuring out how to open the projector to reach the bulb. The class, who are taking notes, then critically analyze the design of the overhead projector, relate them to Norman's design principles of everyday things, and suggest improvements. They notice that most recommendations are simple changes to the plastic overhead case, and that a better projector could probably be built for the

exact same price. The class often wonders why the manufacturer never bothered doing this simple exercise! Other in-class evaluations have included:

- using constructive interaction to reveal conceptual model formation and problems in the controls and labels of a high-end fax machine (its control panel is presented as an overhead);
- using think-aloud to expose difficulties that even technically proficient people have when performing fairly simple tasks in the Windows ’95 file manager;
- an in-class quantitative controlled experiment.

Involving the user in the design process. A fundamental tenant of HCI is that end-users should play an integral role in the design process. After briefly introducing user-centered system design and a (simplified) version of participatory design, I walk students through a variety of methods that involve users in the design of low and high fidelity prototypes. Methods starts as simple verbal exercises, but rapidly go through paper and pen sketches, storyboards, Pictive, scripted simulations and so on, each getting slightly more sophisticated. I stress in class that early versions of prototypes should be low fidelity and low cost (e.g., paper, pencil, and sticky note technology), and that its purpose should be to garner high-level reaction and input from the user. As the design progresses, prototypes will become higher fidelity and more refined, and the user's input should reflect smaller, but still important, design and usability decisions.

Most techniques are demonstrated live. For example, I do a walk-through of a storyboard design. I then introduce an interface as a Pictive design based on sticky notes, and a volunteer interacts with it. The volunteer and class identify problems, and we redesign the system on the fly by having people reconstruct its components on sticky notes. I have also devoted a class to a live Wizard of Oz demonstration (Maulsby, Greenberg and Mander 1993).

Students apply these techniques to their interface design project (discussed later). They use story-boarding and horizontal prototypes to garner user reaction, and a vertical prototype that serves as a proof of concept. Because of class size, some groups use others as “simulated” end users. However, there are always a few groups who find a real user audience, and who go to great lengths to involve them in the on-going design of the application.

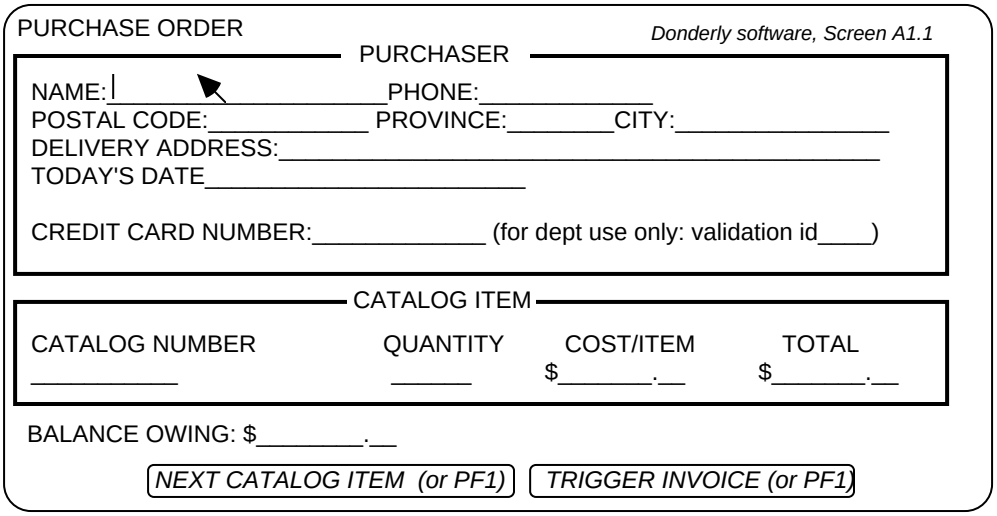
Designing without the User.

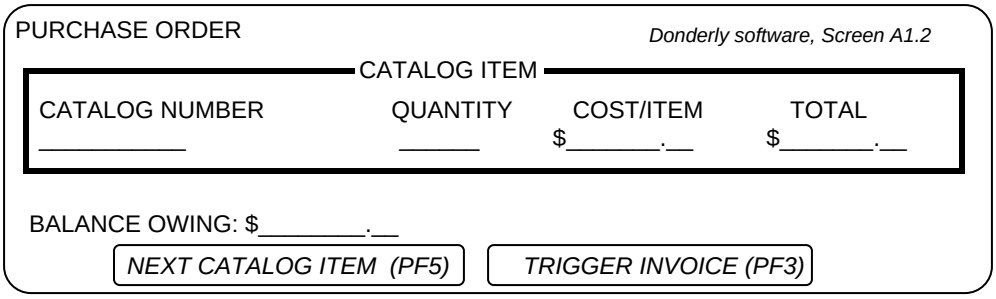
Graduating students may find themselves employed in an environment where they have either no access or irregular access to end-users. Yet design must continue. This section of the course presents several topics on how to design interfaces without the user.

Task-centered system design. Task-centered system design is a technique that allows developers to design and evaluate interfaces based on users' real-world tasks (Lewis and Rieman 1993). It does require some user involvement, at least at the beginning, to solicit good task descriptions. As part of the design, it becomes a requirements analysis, with the requirements being the major tasks that need to be satisfied. As part of evaluation, the evaluator can do a walk-through of the prototype, using the tasks to generate a step by step scenario of what a user would have to do with the system. Each step in the walk-through asks the questions:

- is it believable that a person would do this;
- does the person have the knowledge to do it?

If not, then a bug has been found. The bug is noted and assumed solved, and the process continues.





Specification

1. Screen #1 is the start-up screen displayed when a person approaches the terminal.
2. Shoppers enter all their personal information and their first order on Screen #1 (above) via mouse and keyboard—the mouse is used to go between fields.
3. Shoppers enter further orders by going to successive copies of Screen #2.
4. Shoppers indicate their order is complete by selecting "Trigger Invoice". The system automatically tells shipping and billing about the order, and returns to a blank screen #1.
5. As described in the front of the catalog, shoppers cancel their order by pressing 'Alt-Q', or by walking away. The system automatically returns to an empty main screen after 30 seconds of inactivity.

Example task descriptions for "Cheap Shop"

A man carrying for a demanding toddler buys an umbrella stroller (red is preferred, but blue is acceptable), pays for it in cash, and uses it immediately.

An elderly arthritic woman is price-comparing the costs of a child's bedroom set, consisting of a wooden desk, a chair, a single bed, a mattress, a bedspread, and a pillow. She takes the description and total cost away with her, to check against other stores. Two hours later, she returns and decides to buy everything but the chair.

A "Cheap Shop" clerk, who is the sole salesperson in the store, is given a lengthy list of 10 items to order by a customer who does not want to use the computer. The items are: *<list here>.* After seeing the total, the customer decides to take all but the 4th and 6th item, and adds a new one to the list afterwards. The customer always changes their mind about paying by credit card, and decides to pay cash. The customer wants the items delivered to his home the day after tomorrow. While this is occurring, 6 other customers are waiting for the salesperson.

Figure 2. The first and subsequent screens of the Cheap shop interface, its interface specification, and example task descriptions.

In class, I develop an example of task-centered system design by using an imaginary client called “Cheap Shop”, a catalog-based store. The situation is that Cheap Shop's customers now browse through paper catalogs and then place their orders by filling in a form and giving it to the clerk. Cheap Shop is considering replacing the paper forms by the computer interface proposed in Figure 2. As a home exercise, students try to identify interface problems using their own intuition. In class, specific task examples are then used to develop usage scenarios (three are listed in the figure), and the class evaluates the design by a walking a user through the example tasks step by step. Of course, many deficiencies are discovered that go far beyond those noticed in the home exercise, simply because the task and user situation bring out factors that are not normally considered.

High level models of user behaviour. There are very few theories in HCI, and most tend to deal with low level phenomena such as selection accuracy and speed (Fitts Law), or ways of modeling human goals into low-level actions and predicting performance outcomes (GOMS). Unfortunately, most students do not find these particularly relevant to the jobs they would likely acquire. As an alternative, I provide students with two high-level cognitive models of human behaviour that help them understand how people interact with machines. These are Ben Shneiderman's syntactic / semantic model (Shneiderman 1992), and Don Norman's stages of interaction (Norman 1988). Both are chosen because they profile in general the major steps and bottlenecks in human-computer interaction. These models can be used both to guide design, and as a simple way to identify problems.

Design guidelines and usability heuristics. Guidelines to design have a long tradition in HCI. There are literally thousands of guidelines now available, in many forms and variations. These tend to fall in the categories of: motherhood’s (or general guidelines); specific guidelines that say exactly what should be done in a given situation; style guides that are particular to a look and feel; and widget-level guidelines that are embedded within an actual toolkit.

I concentrate on general design guidelines catalogued by Nielsen (1993), detailing what they mean and how the interface should cater to them. The ones I use are:

- Simple and natural dialogue
- Speak the users' language
- Minimize user memory load
- Be consistent
- Provide feedback
- Provide clearly marked exits
- Provide shortcuts
- Deal with errors in positive and helpful manner
- Provide help and documentation

I also show how these guidelines can be used as a low-cost evaluation technique via *usability heuristics*, where the guidelines become a way to structure their analysis of the interface. Nielson (1993) suggests that several evaluators using these guidelines can capture many of the major usability problems.

Each guideline is in itself a rich topic, and I cover about two of them per class. I also do a heuristic evaluation of several interfaces as guidelines are presented. This includes the Cheap Shop system mentioned in Figure 2, which exposes additional problems not caught by task centered system design.

There is also a hands on component, where the usability heuristics are used to evaluate the student’s final projects. Students get a marking sheet ahead of time containing the guidelines, and they are expected to review their designs for problems through it. I personally do a heuristic evaluation of each of their system, discussing the results with them.

Creativity and metaphors in interface design. Interface design is an art as well as an engineering and science discipline. This makes it worth exploring how creativity can be applied to interface design, and how appropriate metaphors can be chosen. While there is no recipe for creativity, I use Mountford’s (1990) tips for creative design to show how good ideas can be “borrowed” from other fields and how metaphors can be reshaped. Seeing specific examples of innovations is a strong motivator, and I use videos drawn mostly from the SIGCHI Technical Video Proceedings to illustrate novel and creative interface

designs and metaphors. Students tend to be quite impressed by the ideas presented in the videos, particularly those dealing with information visualization.

Graphical screen design. One small but still essential component of graphical user interface design concerns the actual layout of elements on the screen. This is the realm of graphical design, and this topic presents students with some (but by no means all!) rudiments of screen layout. I use many examples of actual screen snapshots to illustrate graphical design principles. Most screens come from the book *Designing visual interfaces*, by Mullet & Sano (1995), but virtually any screen can be analyzed and even re-designed during class time.

Windowing systems and toolkits.

Computer scientists must know how to translate their designs into working systems. As in any craft, the tools available to implementers have a profound effect on the end system, both in the style of the interface and the way original designs are translated into working ones.

To give students the tools of their trade, they first must understand what the tools can offer. They learn about windowing system technology, the offerings of graphical user interface toolkits, and interface builders. They are introduced to features that they should expect to see in the next generation toolkit, such as constraint management and programming by demonstration.

The hands-on component comes in the third assignment (see Section 0). Through a GUI toolkit, students translate their paper prototypes into horizontal and vertical prototypes—often finding out that some of their ideas are not easily implementable—and then turn their prototype into a working system. We use the Tcl/Tk language (Ousterhout 1994) because it is freely available, has a very rapid learning time (compared to other toolkits), and is reasonably robust. It is also available on Linux, which many students have installed on their home machines. Other languages could work just as well as long as it has a modest learning curve (and most do not!), and we have had successful experiences with SUIT (Pausch, Conway and Deline 1992) and Microsoft's Visual Basic.

Visions of the future.

The course closes with video presentations of several visions of the future of human computer interaction. I expect students to realize that today's graphical user interfaces are an artifact of today's technology, and that the affordances of future technology will have a profound effect on the designs they create. T

I also introduce the notion of ethics for programmers, as future systems can have a ruinous effect on society. For example, the video on the Active Badges personal locator system (Hopper, 1993) is a great one for discussing the ethics of privacy, and ask them what they would do if they were hired to implement a system that could act as a surveillance device.

Practicum: The Assignments

In three course assignments, students practice and apply what they have learnt in class. They pursue a controlled experiment and quantitative evaluation in Assignment 1, a usability study and qualitative evaluation in Assignment 2, and a major project on interface design and implementation in Assignment 3.

Assignment 1: Quantitative evaluation. The purpose of this hands-on exercise is to give students experience conducting a controlled experiment, performing a simple statistical analysis, interpreting the results, and considering its implications to design. However, I do not expect students to become behavioral scientists or to run controlled experiments when they get into the work force! Instead, I want to provide students with enough knowledge of the experimental process to help them understand, appreciate, and be critical of the HCI literature that uses this methodology.

The scenario is that a company is designing a portable computer that does not have a keyboard. Most of the interaction will be through the mouse, but occasional text input is needed as well. The company has already ruled out handwriting recognition due to poor recognition rates. Typing will be done by putting a simulated

keyboard on the screen, and by selecting keys with the mouse (called mouse-typing). Because the simulated keyboard can take any shape and key arrangement, the company wishes to consider layouts other than the standard Qwerty.

In the experiment (which changes slightly every year) students compare people's mouse-typing abilities on different keyboard layouts. Some of the keyboard layouts considered over the years are illustrated in Figure 3.

Students, who work in groups of three, run each other as subjects and collect typing times plus comments. The instructor collects and compiles the data from all groups and hands it back to the students. Groups then use an unpaired t-test to check for speed differences between the keyboards. The deliverable is a substantial technical report that presents the experiment, collects and interprets the results, and discusses its implications to keyboard use.

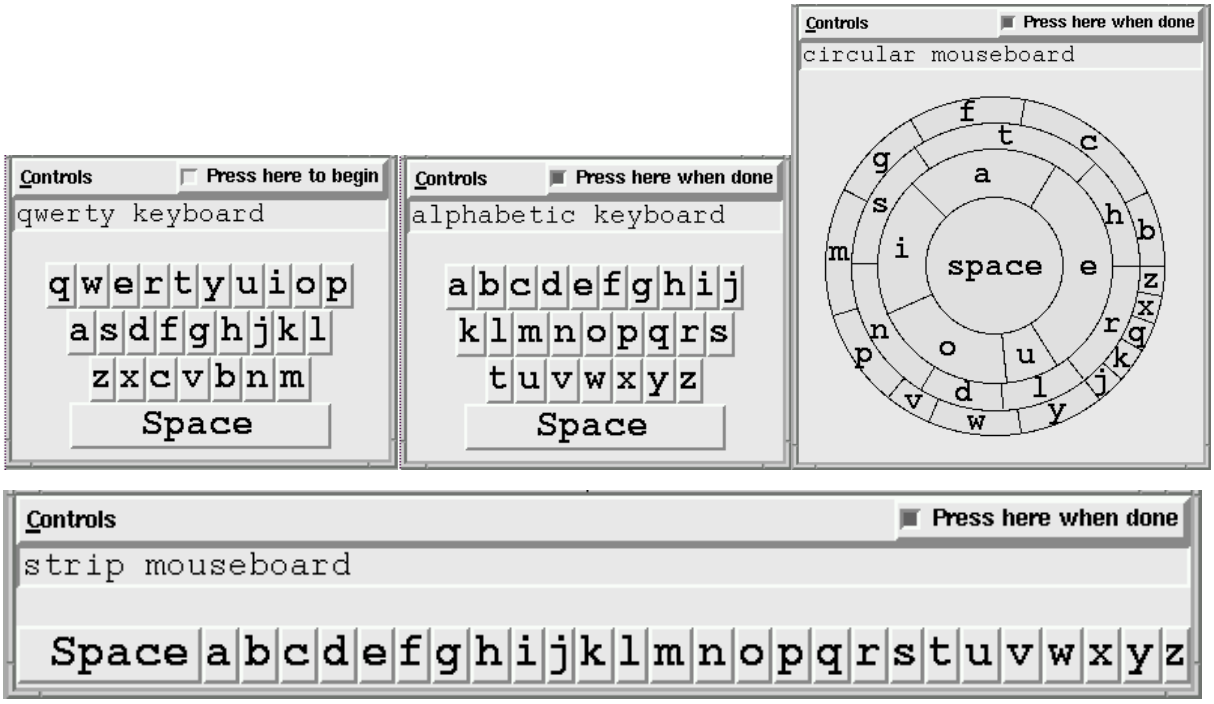


Figure 3. A few of the mouseboards contrasted over the years. The circular mousebard has frequent letter pairs placed as large keys near the center. Other mouseboards include the Dvorak keyboard, and an alphabetic keyboard in column order.

Assignment 2: Qualitative evaluation. This exercise on qualitative evaluation gives students experience conducting usability studies on a real product. Methods used in this study mirror those taught in class, and include conceptual model discovery, strict observation, think-aloud, constructive interaction, questionnaires, and interviews. All are economical methods, and students are encouraged to apply them in actual work practices.

Groups pretend they are working for a company developing the system. Using each other and volunteers as subjects, they examine a system for usability problems. Systems investigated over the years include:

- Dobis, a very large library catalog system used by the University's library;
- a CD-ROM booking system also used by the library; and
- XWais, a front end to the Wais text indexing system that is used locally to access the HCI Bibliography (Perlman 1994).

Groups deliver a substantial technical report oriented towards a vice-president of their company. It includes observations made, the major problems detected, and some design recommendations. The report also contrasts the methods used, recommending which ones should be adapted in future system evaluations.

Assignment 3: Design and implementation project. The term project is a major portion of the course. Its main purpose is to give students hands-on experience applying the design concepts learnt in class. Students are free to define their own project area, as long as it is conducive to creating an interesting interactive application. Students can come up with an application from scratch, or they may decide to remodel an existing application to make it more effective, or they may go to an actual user group and design a system that fits their needs (an optional but preferred path).

The project and its deliverables are incremental. In the first phase, students create an initial paper prototype and design rationale which is presented in lab time and critiqued by other students and by the teaching assistant. In the second phase, they produce screen snapshots of a horizontal prototype (via a GUI toolkit) and a re-design rationale (also critiqued in lab time). They then implement a high fidelity vertical prototype to give a good feel for the system. They concentrate on interface design, simulating back-end functionality when necessary. The final deliverables are a reasonably robust working prototype, a minimalist manual, and a short design critique of the final system. I then meet with each group for a half-hour, see their system in action, and evaluate it immediately via heuristic evaluation techniques (Nielsen 1993).

I am always impressed with what students can do. Although they are only given a month for all design stages and programming, most of the projects are very good, with some being absolutely outstanding. The projects indicate the success of the course, for students really do apply their HCI learning as they iterate through their system designs.

Comparison with other HCI Educational Pedagogy

The ACM SIGCHI Curriculum (Hewett *et al* 1992) is an obvious source and inspiration to HCI educators. The document is at its best when considering how HCI can be introduced throughout the curriculum within various disciplines to produce HCI specialists. It also provides outlines of several courses that could be offered in different departments. The course described in this article contain parts of Computer Science CS1: User Interface Design and Development, and CS2: Phenomena and Theories of Human-Computer Interaction.

From an NSF-funded workshop, Strong and many other HCI professionals produced the report "New Directions in Human-Computer Interaction" (Strong *et al* 1994). The report promotes the importance of HCI in education, research, and practice. In education, Strong suggests that computer science must be transformed to include HCI, perhaps through one or two specialty courses:

- The integration of interface design and development processes into the computer science curriculum should be focused on creating an undergraduate capstone experience (e.g., a senior project); and
- Universities should be encouraged to perceive HCI as a "critical technology" and the accompanying skills and knowledge as fundamental to a student's education and preparation for jobs in the information age. (Strong *et al* 1994, page 5)

The course I offer fit these criteria to some extent. The assignments, especially the final project, do become capstone experiences. Similarly, the course is intended to prepare students for their jobs by concentrating on fundamental skills that should be applicable in environments that do not fully incorporate HCI practices.

There are, of course, many other courses on HCI, each as unique as the instructor that teaches it. Most include core aspects of HCI, but they vary considerably in their focus and the topics covered. Strong's report contains, as an appendix, outlines of twenty four different HCI courses offered at various universities. The ACM Curriculum also includes a course as a case study. The HCI Education Survey (Perlman and Gasen) contains information about programs, faculty, and courses with an emphasis on Human-Computer Interaction.

Conclusions

Over the next decade, we will see HCI gain prominence as a valid stream both within Computer Science and other disciplines. We will also see all levels of the software industry accept the relevance of good interface design, and embrace the practice of usability engineering. The single course described here is somewhat of a stop-gap that gives programmers enough of a foundation to introduce design and usability engineering in their every day jobs. The tradeoff I chose was to concentrate on simple techniques that are immediately applicable in conventional work environments, over sophisticated and perhaps more accurate techniques that would be difficult or costly to introduce in software shops with little knowledge of HCI.

This article is really only the first part—an overview—of a two part article on teaching HCI to programmers. The second part, available through the World Wide Web (Sidebar 1), contains all its details, including overheads, handouts, and assignments.

References

1. Apple Computers Inc. (1992) 2020. Distributed as part of the video set from the Apple Developer's Conference. Videotape.
2. Baecker, R., Grudin, J., Buxton, W., and Greenberg, S. (1995) *Readings in Human Computer Interaction: Towards the Year 2000* (2nd Edition), 950 pages, Morgan Kaufmann Publishers, California. ISBN 1-55860-246-1.
3. Engelbart, D. and English, W. (1994) A Research Center for Augmenting Human Intellect. SIGGRAPH Video Review, **106,** Videotape**.**
4. Hewett, T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G., and Verplank, W. (1992) *ACM SIGCHI Curricula for Human-Computer Interaction*, Report of the ACM SIGCHI Curriculum Development Group, ACM.
5. Hopper A. (1993). The Active Badge System. In SIGGRAPH Video Review, **89,** Videotape.
6. Lewis, C. and Rieman, J. (1993) *Task-Centered User Interface Design.* Shareware book available via anonymous ftp from ftp.cs.colorado.edu
7. Maulsby, D., Greenberg, S., and Mander, R. (1993). Prototyping an intelligent agent through Wizard of Oz. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 277-284, Amsterdam, The Netherlands, May, ACM Press.
8. Mountford, J. (1990) Tools and techniques for creative design. In Laurel, B. (Ed.), *The Art of Human Computer Interface Design.* Addison-Wesley, 17-30.
9. Muller, M. J. (1991). PICTIVE - An Exploration in Participatory Design. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pp. 225-231.
10. Mullet, K. & Sano, D. (1995) *Designing Visual Interfaces.* Prentice Hall.
11. Myers, B. (1990) *All the Widgets.* SIGGRAPH Video Review, **57,** Videotape**.**
12. Nielsen J. (1993). *Usability Engineering*, Academic Press.
13. Norman, D. (1988) *The Psychology of Everyday Things.* (*The Design of Everyday Things* in paperback). Basic Books.
14. Ousterhout, J. (1994) *An Introduction to Tcl and Tk.* Addison-Wesley.
15. Pausch, R., Conway, M., and Deline, R. (1992) Lessons learned from SUIT, the Simple User Interface Toolkit. *ACM Transactions on Office Information Systems* **10**(4), 320-344.
16. Perlman G. (1994). The HCI Bibliography: Past, Present, and Future. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing System*s, Volume 2, pp. 71-72.
17. Perlman, G. and Gasen, J. HCI Education Survey. Available from http://www.cis.ohio-state.edu/~perlman/educhi.html.

18. Shneiderman B. (1992). *Designing the User Interface: Strategies for Effective Human-Computer Interaction (2nd Edition)*, Addison-Wesley.
19. Sutherland, I. (1983) *Sketchpad.* SIGGRAPH Video Review, **13,** Videotape**.**
20. Smith, D. and Irby, C. (1983) *Xerox Star User Interface*, SIGGRAPH Video Review, **56,** Videotape**.**
21. Strong, G. and many others (1994) New Directions in Human-Computer Interaction Education, Research, and Practice. Drexel University. Available from http://www.sei.cmu.edu/arpa/hci/directions/