HUMAN COMPUTER INTERACTION CLOSES THE SOFTWARE ENGINEERING GAP

John D. Fernandez, Ph.D.
Texas A&M University – Corpus Christi

Abstract

With the pervasiveness of computers throughout our environment, there is a growing demand for diligent Human Computer Interaction (HCI) education of graduate and undergraduate students to close the gap left by software engineering education. This paper describes one approach to teaching HCI while requiring students to develop systems for various city, school, and university organizations. The benefits derived by the students and the clients receiving their services are many. There is a wide range of opportunities for community-based HCI education. The various client applications provide a plethora of learning scenarios that don't fit into the software engineering paradigm. The process, experience, and results of HCI education that closes the gap are presented in this paper.

Introduction

There is growing interest in exploring the benefits of Human Computer Interaction (HCI) design in computer science education programs. The prominence of the Internet and associated Web applications has propelled HCI issues to the forefront of the discussion on software development. Since software engineering is geared more towards the development of large-scale systems, there is an education gap in many computer science programs that don't include a design and implementation course for small to medium size systems that emphasize user interaction. The best course to close the gap is a course in HCI that includes hands-on development efforts with real users.

At Texas A&M University – Corpus Christi, the author experimented with a paradigm that is community-based HCI (CB-HCI) to close the gap in the education of students. This initial experiment was with a graduate course in HCI, but the same paradigm is to be applied to the undergraduate HCI course. Before relating the details of the process and paradigm, it seems appropriate to discuss some fundamental concepts of HCI for those who may not be as familiar with the topic.

HCI Fundamentals and Software Engineering

There are few good textbooks that address HCI with the rigor and span that is appropriate. The three most commonly cited by authors are the books by Preece et al.[7], Shneidereman[10] and Rosson & Carroll[9]. McCracken and Wolfe[6] provide a definition for HCI that includes its major components: Human-computer interaction is a discipline concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. Preece et al.[7] include a definition of HCI design: *designing interactive products to support people in their everyday and working lives*.

Software engineering textbooks are readily available at any level of depth and complexity. The most popular books appear to be the ones by Pressman[8], Sommerville[11], and Humphrey[3]. Some projects are well suited for the traditional software engineering approach, but most are not. McBreen[5] states that software engineering was invented to tackle the problems of really large NATO systems projects.

> These projects pushed the state of the art in both computer hardware and software development for new hardware in the late 1960s and early 1970s. In 1968, a NATO conference identified a software crisis and suggested that for large, high-quality software applications, software engineering was the best way out of that crisis. Since that time, the needs of the U.S. Department of Defense have dominated the conversation about software engineering.[5]

By reviewing the last two paragraphs, it is easy to see that there is a large gap between the two foundational frameworks for developing software. HCI is concerned with products that support people in their everyday and working lives. Software engineering is concerned with developing large systems for the military industrial complex or other large enterprises. It seems difficult to image how these two disciplines could be further apart.

The ISO 13407 standard addresses user-centered development activities and proposes the development model shown in Figure 1 below[4]. One can see some overlap with a simplified spiral life cycle model within software engineering. However, HCI is user-centered development rather than data-centered. HCI involves users in the process of development as much as possible with the goal of creating an interactive system that meets individual users' expectations.[6]

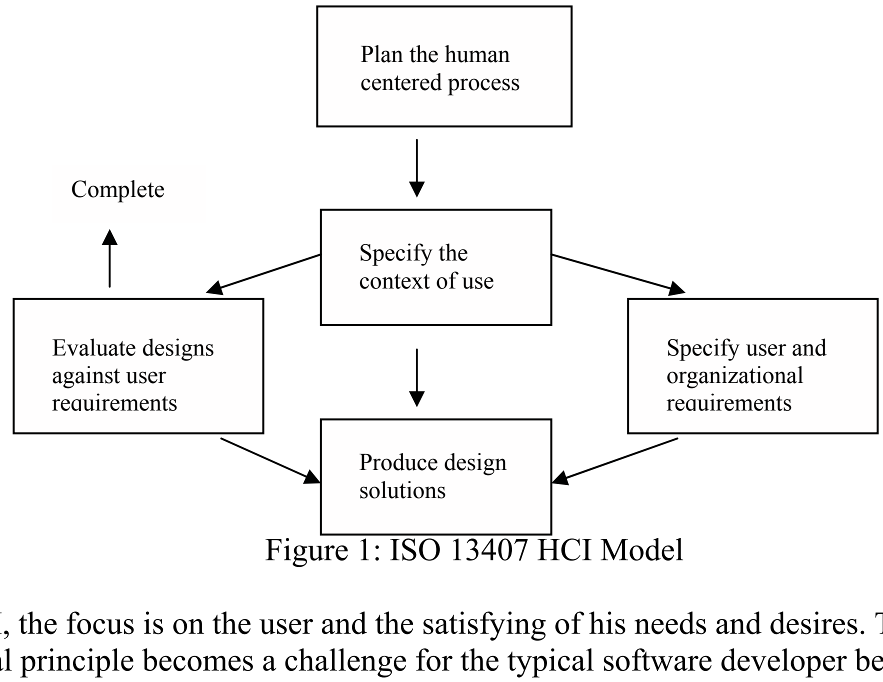
Plan the human centered process
Complete
Specify the context of use
Evaluate designs against user requirements
Specify user and organizational requirements
Produce design solutions

Figure 1: ISO 13407 HCI Model

Within HCI, the focus is on the user and the satisfying of his needs and desires. This foundational principle becomes a challenge for the typical software developer because it calls for skills and abilities that are not normally part of the software engineer's portfolio. HCI is an interdisciplinary methodology that involves skills from disciplines such as psychology/cognitive science, engineering, informatics, computer science/software engineering, ergonomics, human factors, and social sciences[7]. Alan Cooper[2] relates his experience of the broader perspective which he could only see after he extricated himself from the programming grip. He claims that only then did he see that programming is such a difficult and absorbing task that it dominates all other considerations, including the concerns of the user.

After several years of experience developing and managing the development of systems for many organizations, the author can relate to Alan Cooper's comments. The author was privileged to work with many government and contract software developers on a multitude of systems for a variety of client organizations. The dominant attitude in all cases was that the user served at the behest of the development team. The software development machine had to continue to achieve its defined goals of efficient development and product delivery. The user was forced to adjust to the system that resulted from the programmer-centered development effort. Many students in our educational institutions appear to have inherited a technological DNA that predisposes them to think of the user as the factor that limits their creative development efforts. In other words, the user is a problem.

While teaching a software engineering graduate course, the author heard comments from students, such as, "users are idiots." Obviously, these comments are meant as jokes, but it is clear that many students carry a bit of an attitude against the non-computer-oriented community. In the past 25 years, little has changed in the attitudes of students pursuing degrees in computer science or software engineering. In fact, little has changed since the NATO conference which was the genesis of software engineering. The focus of software engineering is still centered on supporting large Department of Defense type systems. This is a much needed discipline, but the majority of computer science students will be developing much smaller systems with an interactive component. Therefore, human-computer interaction design education is a much needed discipline within all computer science programs.

HCI Design Education

One of the goals of HCI design education is to move computer science students away from the keyboard and computer screen and get them focused on the user and his problem domain. All software engineering educators can relate to the tension that exists as students learn planning, analysis, and design techniques. The students want to get to the programming phase right away. After all, they already understood the problem and know exactly what programming logic they need to write to meet their version of the requirements.

HCI design's main goal is to build interactive systems that are easy to learn, effective to use, and enjoyable from the user's perspective[5]. These characteristics are summed up in one word – usability. Usability can only be understood from the user's mind-set. Software developers find it difficult to get into the user's mind to capture the essence of the requirements to be able to build a product that satisfies the user's real needs. So, the initial objectives of any HCI design education must include the enlightening of students to the problems in our software development industry. Cooper's provocative statements relating to this situation are worth reading for their shock affect[2]:

> The high-tech industry has inadvertently put programmers and engineers in charge, so their hard-to-use engineering culture dominates. Despite appearances, business executives are simply not the

> ones in control of the high-tech industry. It is the engineers who are running the show. In our rush to accept the many benefits of the silicon chip, we have abdicated our responsibilities. We have let the inmates run the asylum.

It is good to remind students of the joys of the craft of programming as defined by Frederick Brooks in his classic book published more than 20 years ago[1]. The very essence of the nature of human beings appears to be a desire to create. This is what drives students and all programmers to the keyboard to write source code. There is no craft so inviting as programming to get our creative juices flowing to build something of our own making, a product of our minds. The role of an HCI design educator is to redirect these creative powers to develop products that meet the needs as defined by the user's own mind.

Community-Based HCI Design Education

This redirection of software developers can be started with traditional HCI textbooks, many of which have been cited in the discussion above. However, there is no substitute for real-world experience. The community-based human computer interaction (CB-HCI) education course at Texas A&M University – Corpus Christi provides such an experience.

Before initiating the course, contacts were made with city, school, and university organizations in order to find HCI type requirements that were real and could provide the basis for student projects. The types of projects that seemed to fit best were web-based systems with interactive components. The principals in each of the project offices were visited in order to get a better understanding of the needs and to explain to the users how the students would be approaching them and how they would be completing their projects.

On the first day of class, the students were introduced to the concept of doing a project with a user organization. In order to better understand the students' capabilities to determine the best combination of students for required project teams, a capabilities survey form was prepared and provided to each student for completion. A question that was part of the survey form concerned the availability of transportation. Since some of the work required traveling off-campus to the business district, it was important to know who did not have transportation. Once the forms were analyzed, it was fairly easy to determine how to compose the project teams. Two students were assigned per team. One student was assigned to work alone on a university project. There were a total of 15 students.

The following Web-based development projects were selected for the students to complete:

1. Kiosk Information System for the University Library
2. Commission for Children and Youth
3. Youth Opportunities United
4. Upward Bound Program
5. Blanche Moore Elementary School Library Data Management
6. Computing and Mathematical Sciences Advising and Mentoring
7. Virtual Tour System for the University Library
8. Coastal Bend Alliance for Youth

The Upward Bound Program had its offices on campus and the project was sized so that one student was assigned to work with that user organization. About a month into the semester, one student developed a personal problem and had to leave the University, so the Youth Opportunities United was completed with only one student.

Before releasing the students to have their first client meeting, lecture material was presented on the basics of HCI, scenario-based HCI development, gathering information and conducting interviews. The students were then assigned to conduct their first visit with the client. During this time, students were reviewing a total of 10 web sites each in order to find the positives and negatives that would help them in their design decisions. The most significant sites (positive or negative) were presented by students to their class members.

A key point in analyzing requirements was for the students to determine all the stakeholders of the system they were designing. Each student built a personal profile (persona) for each stakeholder of the system, assigning each persona a name and an educational and family background. After further analysis of scenarios wherein each persona was placed, a primary persona was chosen for designing the system. The primary persona is the principal stakeholder user of the system. Rosson and Carroll's book[9] was used as a guide to cover the basic material concerning scenario-based HCI. Some of Cooper's design concepts[2] were also incorporated.

Low fidelity prototyping was emphasized in the lecture as the best approach for initial discussions with users when preliminary designs were being explored. Students found these to be very useful because the clients did not feel compelled to agree to what appeared to be a fully developed screen demonstration of a prototype. One team decided to show the client a more high-fidelity prototype and the team reported having had great difficulty getting the client to focus on the big picture of the design when the colors in front of them were distracting. These students reported how they now could see the value of using low-fidelity prototypes for the preliminary discussion.

Before the third meeting with the clients, the students presented their first version of the high-fidelity prototype to the class for initial comments from their peers. After the system was in semi-final form, students were asked to prepare questionnaires and evaluation sheets for usability evaluations by class members. Using a large computer lab, four evaluations were conducted on each of two nights. Students evaluated each other's work and provided written feedback to the developers. Some of the students expressed that this had been a valuable experience for them because they could see how what seems obvious to a developer may not be that obvious to a user.

During the final client presentations, a couple of students commented that this was the first time they had done work for a real client. They were so pleased with the experience and the clients were ecstatic with the products received.

The grading policy used for the course was as follows:

• Mid-term exam	20%
• Analysis of commercial Web pages	15%
• Homework and in-class assignments	5%

- Research Paper 15%
- Term Project (presented as Final Exam) 45%

Students were required to do a research paper on a topic of their choice related to HCI or Web development in general. There papers were also presented to the class so all could benefit from their work.

Conclusions

Using a community-based paradigm for teaching Human Computer Interaction design has proven to be quite successful. Students have bridged the gap between classroom theories and the real world of users. Letters of appreciation were received from the client organizations. Community leaders built relationships with local and international students. The benefits continue today as students have obtained employment or moved on to complete their degree and the Web pages they built are serving the community. This is surely one great way to find joy in one's craft.

References

1. Brooks, F.P., Jr., *The Mythical Man-Month: Essays on Software Engineering - Anniversary Edition*, Addison-Wesley, 1995.
2. Cooper, Alan, *The Inmates are Running the Asylum,* Sams Publishing, 1999.
3. Humphrey, Watts S., *A Discipline for Software Engineering,* Addison Wesley, 1995.
4. ISO 13407 Standard, Developer View Index of the EMMUS Web site, www.ucc.ie/hfrg/emmus/methods/iso.html. Accessed September 11, 2003.
5. McBreen, Pete, *Software Craftsmanship: The New Imperative,* Addison Wesley, 2002.
6. McCracken, D. and Wolfe, R., *User-Centered Website Development: A Human-Computer Interaction Approach,* Pearson Education Inc., 2004.
7. Preece, J., Rogers, Y., and Sharp, H., *Interaction Design: Beyond Human-Computer Interaction*, John Wiley & Sons, Inc. 2002.
8. Pressman, Roger S., *Software Engineering: A Practitioners' Approach,* 5th Ed., McGraw-Hill, 2001.
9. Rosson, M.B., and Carroll, J.M., *Usability Engineering: Scenario-Based Development of Human-Computer Interaction,* Morgan Kaufmann Publishers, 2002.
10. Shneiderman, Ben, *Designing the User Interface,* Addison Wesley Longman, Inc., 1998.
11. Sommerville, Ian, *Software Engineering,* 6th Ed., Addison Wesley, 2001.

Bibliographical Information

JOHN D. FERNANDEZ
Dr. Fernandez is Assistant Professor of Computer Science in the Department of Computing and Mathematical Sciences. Having served 20 years in the U.S. Air Force and 10 years in private industry, Dr. Fernandez brings real-world experiences into the classroom for his students. His research interests are in HCI, information assurance, and software engineering.