Peer Teaching Extends HCI Learning

Beryl Plimmer and Robert Amor
Department of Computer Science
University of Auckland
New Zealand
+64 9 3737599
{beryl|trebor}@cs.auckland.ac.nz

ABSTRACT
Crafting a good user experience requires skills in several disciplines. Few people have this breadth of knowledge, and undergraduate computer science students are no exception. Encouraging computer science students to appreciate the ways that other disciplines contribute to Human Computer Interaction is important, yet difficult. Our students learn about this disciplinary interdependence through peer teaching as part of a group project. Each group contains students with complementary skills and we expect a transfer of knowledge. Here we discuss the educational theory behind the project, the project's essential elements and an evaluation of how it aids learning. The model we have developed could be easily adapted for other courses which draw on diverse skills.

Categories and Subject Descriptors
K.3.2 **Computer Science Education**: HCI Education

General Terms
Human Factors

Keywords
HCI Education, Peer Teaching, Group Projects.

1. INTRODUCTION
A good user experience is the result of careful planning and evaluation. Too often we allow students to graduate who are technically brilliant, but completely unaware of the frustrating interfaces they inflict on users. To build usable systems students need to know more than technology. They need to understand users' goals, capabilities and what humans find aesthetically appealing. This is what Human Computer Interaction (HCI) is all about.

HCI is undeniably multidisciplinary: the three major contributors are computer science, psychology and graphic design. We teach in a traditional computer science department in New Zealand's largest university. Our Computer Science degree includes one HCI course at third year; however, the structure of the degree encourages students to include courses from related disciplines. The goal of this HCI course is to extend students' understanding of what it means to develop an excellent system – a system that not only executes correctly but is useable.

This ambitious goal is met by combining traditional lectures with a carefully crafted group project. Below we describe the educational theory that underpins the course; the parameters we use to create the project problems; examples of successful projects; an evaluation of the course; and a discussion on how this course model may be useful to others.

2. BACKGROUND
In a single course there is not time for in-depth coverage of the: psychology, graphic design and computer science of HCI. As this is a third year computer science course, we can presume that all students have a solid grounding in computer science, yet they also have a wide range of other skills and knowledge which may be helpful. Constructivist learning theories [3] state that people assemble an understanding of new knowledge by combining new information and experiences with existing knowledge. We can therefore assume that students with knowledge of psychology and graphic design will bring this to bear when they are designing a user experience. It is also clear that students learn as much from their peers as their teachers [10]. This course leverages these two factors: existing knowledge and peer teaching/learning to increase the amount of actual learning that takes place.

To optimize learning a structure is required: First, Kolb's [5] experimental learning cycle suggests that learning is most successful when it is an iterative process that includes; acquiring new knowledge, practicing the application of that knowledge and reflecting on the practice. This suggests a pattern of presenting new knowledge to the students in a formal context (lecture) and then requiring them to practice (project) and reflect (review). Second, activity theory [7] proposes that outcomes are achieved by people using knowledge and tools to transpose inputs (data, goals and constraints) into the desired outputs; there is often a complex interplay between the different components of the system (people, knowledge and tools) to achieve the desired outcome. The parallels between activity theory and HCI best-practice have seen this theory gain support in the HCI and HCI education communities [7]. We structure the project requirements and hand-in dates to foster these types of exchanges. Finally, group work encourages the transfer of knowledge between individuals, and it has a wide range of additional benefits including preparing students for the real-world work environment [1, 6].

ITiCSE'06, June 26–28, 2006, Bologna, Italy.

3. PROJECT MODEL

From an HCI perspective we want the students to think deeply about the user's interaction with the software while considering the user's mental and physical capabilities, and the aesthetics of the interface. We also want the students to utilize user centered software design methodologies. In order to meet these requirements within the semester we tightly constrain the problem and deliverables, while, at the same time, leaving some user details and technology decisions quite open. In this section we discuss the project in terms of the target user, problem domain and course.

To limit the scope there is a single user for the project. However, to encourage the students to learn more about human capabilities that user is representative of a target user group with specific interaction needs. They are, for example, a child, elderly, or disabled in some way: definitely not a healthy 20 year old male, the archetypal computer science student. There is always something in their profile to indicate that 'normal' keyboard/mouse interaction would be unsuitable. Unfortunately, because of ethical requirements we can not use a real person. However, this has some benefits. The students must explore the research literature on the target user group; we provide one or two references and expect them to locate other articles. Also, not having access to real users is the reality of much commercial development [4].

The problem domain is quite specific, but outside the realms of usual computer science assignment scenarios. We provide students with one or two references to the domain and one or two about software written for the domain. In order to explicitly incorporate graphic design the project has a strong visual element.

To summarize, the project has an atypical user requiring a non-standard interface, a strong visual component and addresses an uncommon goal. To satisfy these aims the students, inevitably, are challenged to extend their technical knowledge and employ high-level computer science skills. The deliverables and timetable effectively guide them through a user-centered design process.

The course consists of three lectures a week for 12 weeks (with a two week mid-semester break) and each student attends one tutorial a week. There are 100 students in the course; tutorials are about 33 students each. The lectures are a broad introduction to human computer interaction, based on Dix et al. [2]. The material covered in the lectures is much wider than is required for the project. Thus we are not taking a standard problem-based learning approach where the theory is directly related to the problem [9], but rather expect students to select appropriate basic theory to apply to the project problem and to extend the introductory material with their own research into the specific requirements of the project. However, the theoretical material is delivered in an order that is likely to be useful to them for the project.

4. IMPLEMENTATION

In the first lecture the students are given the project requirements that include: a very brief persona and scenario and references to material on the target user group and domain literature (both the domain itself and software for the domain). The students form into groups of four in the tutorials. We suggest (but do not insist) that each group includes at least one person who knows about psychology, one person who knows about graphic design and one 'A' programmer. To aid group formation each individual writes him/herself a name tag and self-identifies his/her skills with colored dots. We warn them against forming a group with friends, unless the group has the requisite range of skills. We have found that this hands-off approach works remarkably well. By the end of the first week the groups are finalized and the project work started.

In the first three weeks of lectures we cover the user-centered software development methodology, early design techniques (personas, scenarios and low-fidelity prototyping) and the basics of human processing systems (senses, memory and cognition). In week 4 each group hands-in: a fleshed-out persona and scenario (they may invent more detail about the user and his/her goals); two short summaries of their research, one on the target user group and one on the domain; and a low-fidelity prototype of the system (see figure 1). They also present their low-fidelity prototypes in tutorials; the benefit of this is that the students see a wide range of ideas that have been generated to solve the same problem. After the presentations they may make any changes they deem appropriate to their design.

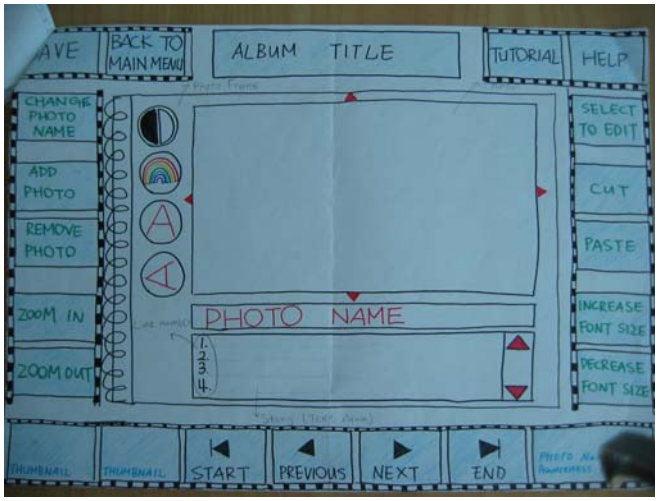

Figure 1. A paper prototype of a grandparent's photo-story album

Over the next three weeks the lectures cover basic models of human computer interaction, design paradigms and implementation issues and support. During this time the students are working on the prototype implementation. We do not specify the programming language, only that the prototype must run in the department's laboratories. In week 7 (after the mid semester break) the prototype implementations are handed-in and there is a 'show and tell' session in one of the laboratories. Each group must have their software running on a machine and the students critique other groups' work.

Weeks 7 to 9 of the lectures cover evaluation techniques. Each group is given another group's project and is required to conduct both expert evaluations and user studies. For the expert evaluation each member of the group completes a heuristic evaluation and then the group merges all of these into a 'master list'. They also evaluate the software against the guidelines for the target user group that they had prepared during the first phase of the project.

Lastly, they prepare a user observation study and trial it with two class members.

We assess the project work in two ways. First, the deliverables are marked. One third of the marks are allocated for each hand-in; research and low-fidelity prototype, implementation, and evaluation. All members of a group are given the same mark for the project work. This contributes 12% to each student's final grade – not commensurate with the effort required. Second, the term test and final examination include questions on the project specifics and process – about 25%. This dual approach allows us to measure individual knowledge without a peer grading system.

Time constraints mean that we can not iterate through the process a second time. However, the low-fidelity prototype presentations in week 4 act as a very effective review and the groups are free to modify their design after the presentation. Also, some of the test and examination questions specifically require them to reflect on the project.

5. EXAMPLES

Defining an appropriate project is essential for the success of this course. We have run this course twice with this structure. The first time the project was to design a paint package for 'Lindsay' a disabled six year old who uses eye-gaze software but wanted to be able to create pictures like his/her friends. Lindsay was the opposite gender to the majority of the group members: mostly Lindsay was a girl – this led to some intriguing comments during the presentations – 'our interface has real colours like blue and red and yucky girl colours like …. pink… '. The software the students produced was innovative and, considering the eight week timeframe, very professional (see Figure 2). More details on this project can be found elsewhere [8].

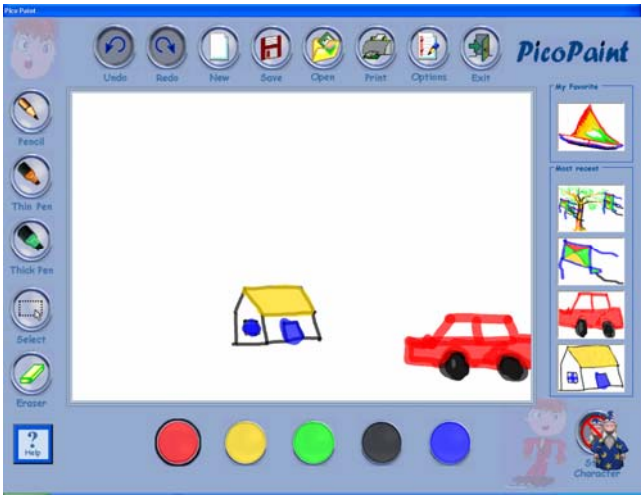

Figure 2. A paint package to use with eye-gaze software

Most recently the project was an electronic photo-story album for a grandparent to record his/her memories. The grandparent has an old shoebox full of photos and the student(s) have been fascinated by the stories associated with the photos. The student(s) decide to write a program for their grandparent to build a photo album and record the stories with the photos. They created paper prototypes (see Figure 1) and implementations (see Figure 3).

The students were provided with references to four papers on interfaces and the elderly, five references to the use of photos in history and four references to photo-based story-telling software. This provided a solid foundation for user group requirements and software development in the field, and a base to locate further information of interest.

The project specification stated that the user was 'a grandparent who wanted to record his/her photos and stories for posterity'. A member of the technical staff made available an online collection of his grandfather's WWI photos and stories which made a realistic and motivating dataset for many of the students. However, the group could choose their own user and many groups based their project around a grandparent of one of the group members (to respect the university's ethical guidelines they could not interview or ask the grandparent to play the role of user in the development process). Regardless of whether they used the dataset provided or a family member's photos, each group wrote a user persona that included relevant (possibly fictional) biographical information.

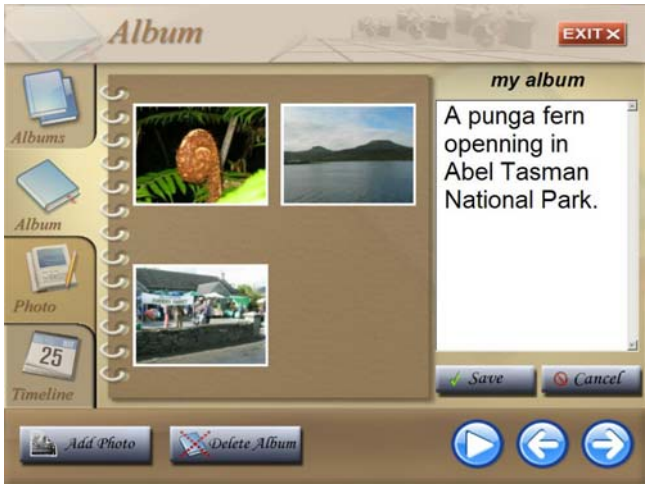

Figure 3. An Implementation of a grandparent's photo-story album

Both of these problems required the students to find out about nonstandard users – children or elderly, with physical constraints – eye-gaze interaction or age-related problems, a domain they were unlikely to know about – kids drawing or the use of photos in family history, and both projects had a strong visual element.

6. EVALUATION

At the end of the most recent offering of the course we surveyed the students to ascertain their thoughts and feelings. We asked them to rate the importance of the contributing disciplines to HCI on a four point Likert scale (Irrelevant, Unimportant, Important, Essential) we coded their responses correspondingly 1 – 4 (Table 1). The analysis shows a similar pattern with computer science and design knowledge: most students scoring them important or essential and no one scored them irrelevant. Psychology had the most variation in responses with 45% scoring it important but 8% saying it was irrelevant.

Table 1. Importance of contributing disciplines

How importance was ……. to your project?	mean	std
Computer science	3.40	0.55
Design knowledge	3.30	0.54
Psychology knowledge	2.74	0.88
Domain knowledge	2.61	0.68

We also asked what contributing disciplines they had skills in prior to the course and whether they taught other members of the team. A huge 72% claimed to have design skills, 29% had knowledge of psychology, while 11% had knowledge of the domain (photo-stories). As for teaching skills to their peers: 34% taught computer science skills, 56% design, 31% psychology, and 12% the domain, with 92% claiming to have taught something to other team members. Certainly from the students' perspective they believed that peer teaching was taking place.

Using the same four categories we asked which parts of the project each student contributed to; 25% claimed all four parts, 20% three parts, 28% two parts and 27% one part. We checked how this corresponded to the answers of the other members of the group. Although there were insufficient samples for statistical measures we noticed that of the 23 groups, in 3 groups all the members contributed to all parts, 5 groups had one member who participated in all parts, while the other groups had varying arrangements of contribution. To ensure that all students considered the different parts of the project they were told to assume that there could be test and examination questions on any aspect of the project.

We asked the students to answer questions on the relationship between the theory and project, and about the group work on a strongly disagree, disagree, agree, strongly agree, scale which we coded correspondingly 1 – 4 (Table 2). Most people thought that the project helped their understanding of the theory and the theory helped with the understanding of the project, those that didn't tended to think there was little relationship either way, scoring both questions disagree or strongly disagree. There was a strong correlation between the two group work questions (r 0.82), 70% agreed that the group work helped their learning (38% strongly agreed, 32% agreed). Similarly 75% agreed that group work made the course more enjoyable (25% strongly agreed, 50% agreed).

Table 2. Practice/Theory link and group work

Statement	mean	std
The project helped my understanding of HCI theory	3.40	0.58
HCI theory helped with the project development	3.40	0.54
Working in a group helped my learning	2.78	0.85
Working in a group made the course more enjoyable	2.66	0.68

Lastly we asked two open questions: what was the most interesting part of the course and what we should consider changing. About half the respondents said the project and group work was the most interesting part of the course, with the show and tell also getting a number of favorable comments. For changes, about 10% thought we should drop the group project.

We noticed with both the Lindsay and grandparent projects that the students put in considerable effort to make the software usable for their target user. In the semester following the Lindsay project we were delighted to see a group of students in the lab applying personas and a scenario to a group project for a non-HCI course.

7. DISCUSSION

There are two key elements of this course: first the nurturing of peer teaching. The group project is clearly a very successful tactic to foster peer teaching. The students' evaluation of the course indicates that there was substantial sharing of knowledge and that they found this both enjoyable and useful. They undoubtedly appreciate the contributions of computer science and design to HCI. We would have liked to see a higher appreciation of psychology too. We are considering a number of ways to achieve this, possibly more emphasis on the contribution of human factors and ergonomics in the lectures along with more specific links to psychology in the project.

Most members of the class thoroughly enjoyed the group project. There is a small minority who find group work difficult, yet employers continually ask tertiary institutes to produce graduates who can work together. In two offerings of this course we have had about 50 groups, only one group had serious problems that required our intervention. We believe putting the responsibility on them to form their own groups is a vital ingredient here.

This course's approach reflects the real-world where a group comes together for a particular project. Group members will be chosen for the specific skills they can bring to the project. Pertinent information must be gathered from research and a plan devised and executed. And just like real projects, different groups operate in different ways.

The second key element is defining a project that requires diverse skills and independent research. Although we present a lot of high-level HCI theory in the lectures, we expect the students to do exactly what is required in a real-world situation, to filter the appropriate information to use in their project. The survey responses indicate that they successfully related the theory and practice. In addition, just as they would in the real-world, they are expected to research independently about the specific needs of the target users and system.

Restricting the human-computer interaction space with a specific user's needs forces the students to think carefully about the interaction their software provides, rather than simply adopting the programming language defaults. Additionally, it makes them aware of accessibility issues.

Setting a non-standard project opens the students' eyes to the information they can gather from resources such as the ACM digital library. This type of project also, inevitably, poses some technical challenges which keeps the 'geeks' engaged in the project.

Having all the groups working on the same project also has benefits. The presentations of both the low-fidelity prototypes and implemented prototypes were commented on as being a highlight of the course by a number of students. Rarely do software designers have the opportunity to see a large number of prototypes for the same problem. The breadth of innovative ideas produced is a delight to both us and the students.

There are many higher level courses that could take a similar approach, particularly when the students have previously completed a range of optional courses. It is essential to check that there are sufficient numbers of students with suitable backgrounds; for example our projects would be less successful if only 5% of students had studied psychology. Also, a carefully thought-out project will draw together the different sub-disciplines and challenge the students to conduct some independent research.

8. CONCLUSIONS

The approach described here draws on sound educational psychology theory to encourage students to share their existing knowledge with their peers thus enriching the learning experience. The students learn the basic theory of HCI and practice user-centered software development. They successfully related the theory to the practice through the use of special-case and research intensive projects. In addition, the generic skills of group work and independent research are fostered. This approach could be adapted to other courses where the students have diverse backgrounds.

9. REFERENCES

[1] Brown, J. and Dobbie, G., Supporting and evaluating team dynamics in group projects in The proceedings of the thirtieth SIGCSE technical symposium on computer science education ACM Press (1999).

[2] Dix, A., Finlay, J., Abowd, G.D., and Beale, R., Human-computer interaction. 3rd ed. Prentice Hall (2004).

[3] Gardiner, L.F., Producing dramatic increases in student learning: Can we do it? National Teaching and Learning Forum, (1997). 6, 2: 8-10.

[4] Greenburg, S., Teaching human computer interaction to programmers. Interactions, (1996). 3, 4: 62-76.

[5] Kolb, D.A., Experiential learning: Experience as the source of learning and development. Prentice-Hall Inc (1984).

[6] Koppelman, H., van Dijk, E.M.A.G., van der Mast, C.P.A.G., and van der Veer, G.C. Team projects in distance education: A case in HCI design. Proc. ItiCSE 2000. ACM, (2000), 97-100.

[7] Nardi, B., Context and consciousness: Activity theory and human-computer interaction. MIT Press (1996).

[8] Plimmer, B. A computer science HCI course. Proc. HCI 2005, (2005), 185-199.

[9] Vat, K.H. Teaching HCI with scenario-based design: The constructivist's synthesis. Proc. ItiCSE. ACM, (2001), 9-12.

[10] Vygostsky, L.S., Mind in society: The development of higher psychological processes, ed. Cole, M. Harvard University Press (1978).