Computer Science Education
Vol. 17, No. 1, March 2007, pp. 21–57

Routledge
Taylor & Francis Group

Implementations of the CC′01 Human–Computer Interaction Guidelines using Bloom's Taxonomy

Bill Manaris[a]*, Michael Wainer[b], Arthur E. Kirkpatrick[c], RoxAnn H. Stalvey[a], Christine Shannon[d], Laura Leventhal[e], Julie Barnes[e], John Wright[f], J. Ben Schafer[g] and Dean Sanders[h]
[a]*College of Charleston, South Carolina, USA;* [b]*Southern Illinois University, USA;* [c]*Simon Fraser University, British Columbia, Canada;* [d]*Centre College, Kentucky, USA;* [e]*Bowling Green State University, Ohio, USA;* [f]*Juniata College, Pennsylvania, USA;* [g]*University of Northern Iowa, USA;* [h]*Northwest Missouri State University, USA*

In today's technology-laden society human–computer interaction (HCI) is an important knowledge area for computer scientists and software engineers. This paper surveys existing approaches to incorporate HCI into computer science (CS) and such related issues as the perceived gap between the interests of the HCI community and the needs of CS educators. It presents several implementations of the HCI subset of the CC′01 curricular guidelines, targeting CS educators with varying degrees of HCI expertise. These implementations include course/module outlines from freshman to graduate levels, suggested texts, and project ideas and issues, such as programming languages and environments. Most importantly, each outline incorporates Bloom's taxonomy to identify the depth of knowledge to be mastered by students. This paper condenses collaborative contributions of 26 HCI/CS educators aiming to improve HCI coverage in mainstream CS curricula.

1. Introduction

Interacting with computers has become an integral part of today's technology-laden society. We are required to interact with various types of software-driven technology on a daily basis, including desktops, PDAs, cell phones, automobiles, and grocery store self-service checkout devices. Usability breakdowns and resulting failures/accidents are usually (and conveniently) blamed on the users. In reality, many such breakdowns are caused by bad design, and thus can be traced to the original developers (Neumann, 1995). The frequency of such breakdowns has undoubtedly

*Corresponding author. College of Charleston, Computer Science Department, Charleston, SC 29424, USA. E-mail: manaris@cs.cofc.edu

ISSN 0899-3408 (print)/ISSN 1744-5175 (online)/07/010021-37

DOI: 10.1080/08993400601069820

increased within the last decade, due to the increasing numbers of end users with minimal computer expertise. At the same time, user interfaces have increasingly complex functionality, usability requirements, and requirements for ease of learning. Even as early as 1991, approximately 48% of the source code and 44% of the total development time was devoted to the user interface (Myers & Rosson, 1992). There is every reason to believe that these numbers are representative of current projects.

The field of human–computer interaction (HCI) is concerned with the art and science of developing usable, useful systems. Although the computer science (CS) education mainstream recognizes the need for deriving a good conceptual model prior to software implementation, few CS educators know how to develop such a model for the user interface, and even fewer know how to teach others to do so. This is supported by a recent survey among information technology (IT) employers, identifying the 20 most important topics for which software professionals have insufficient training: HCI and user interfaces was listed as the second most important topic, after Negotiation (Lethbridge, 2000). Not surprisingly, this lack of HCI skill is reflected in many software products.

Since the publication of the ACM SIGCHI Curricula for Human–Computer Interaction (Hewett, 1992) many CS educators have incorporated HCI into undergraduate CS curricula. These implementations have mainly been offered as elective courses or modules within other courses, such as software engineering (SE), graphics and multimedia, or even the introductory sequence (CS1 and CS2). There are many success stories, however, it is interesting to note that as of 2001 only 3% of CAC-accredited degree programs required an HCI course at the upper level (McCauley & Manaris, 2002).

We believe that the inadequate coverage of HCI in undergraduate CS curricula is due to several reasons. First, there is a gap between the perspective of the HCI community and the perspective of CS educators. HCI focuses on psychology, guidelines, and user-centered design and evaluation—to HCI practitioners, the user interface is the system. On the other hand, traditional CS focuses on mathematical problem solving, algorithms, and engineering of software—to CS practitioners, the code is the system. Another reason is the lack of expertise among mainstream CS educators in teaching HCI; many of them have never had an HCI class as students. We believe that, given some assistance in establishing HCI courses in their curriculum CS educators will be in the unique position to teach a cohesive intersection of the design, evaluation, and implementation of both the software architecture and the user interface.

This paper reports results from an NSF funded project aiming to improve HCI coverage in mainstream CS curricula (Manaris & McCauley, 2004). It surveys existing approaches and issues and presents several course/module outlines. Its target audience is CS educators with varying degrees of HCI expertise. These implementations include course/module outlines from freshman to graduate levels, suggested texts, and project ideas and issues, such as programming languages and environments. The paper uses Bloom's taxonomy to identify the depth of knowledge to be

mastered by students for each topic presented, therefore making the implementations easier to understand and apply in CS curricula.

2. Background

There have been four significant efforts at HCI curricular guidelines.

- The 1989 Software Engineering Institute (SEI) curriculum module and support materials on user interface development (Perlman, 1989).
- The 1992 ACM SIGCHI Curricula for Human–Computer Interaction (Hewett, 1992).
- The 1994 NSF/ARPA recommendations for HCI education (Strong, 1994).
- The 2001 ACM/IEEE Computing Curricula (CC′01) (Engel & Roberts, 2001).

2.1. HCI Knowledge Units in CC'01

The latest of the curricular guidelines, CC′01, identified eight HCI knowledge units.

- HC1. (Core) Foundations of human–computer interaction (min. 6 hours).
- HC2. (Core) Building a simple graphical user interface (min. 2 hours).
- HC3. (Elective) Human-centered software evaluation.
- HC4. (Elective) Human-centered software development.
- HC5. (Elective) Graphical user interface design.
- HC6. (Elective) Graphical user interface programming.
- HC7. (Elective) HCI aspects of multimedia systems.
- HC8. (Elective) HCI aspects of collaboration and communication.

2.2. Recent Approaches to HCI Courses in CS

Several recent papers have described approaches to incorporating HCI into the undergraduate curriculum. Reimer and Douglas (2003) described a studio-based approach in teaching user interface design. Studio-based courses have been used for many years in other traditional design fields, such as architecture, product design, and studio art. This approach incorporates weekly design problems, collaboration between students and faculty, production of realistic artifacts, and weekly design critique sessions. Instead of lecturing about design guidelines, instructors surround their students with design artifacts and immerse them in a realistic design process. Schafer (2005) also blended the studio approach into an interface design course where students were expected to present and defend various aspects of their interface design in weekly design critique sessions.

van der Veer and van Vliet (2003) suggested greater integration of HCI methods throughout software development. They argued that the user interface is the system and that usability is the decisive factor for software quality. They described a minimal

yet essential HCI component for CS and SE curricula. They also provide examples of common user interface problems and illustrate how these problems could be easily eliminated if a more integrated approach were followed.

Miller (2003) proposed an approach for moving HCI from the periphery to the center of most CS programs. Traditional CS focuses on mathematical problem solving, well-specified problems, axioms, and verifiable algorithms. On the other hand, HCI focuses on psychology, ill-defined problems, guidelines, and user-centered design and evaluation. Miller proposed bridging this gap through HCI modules that apply a theory, derive detailed predictions, and verify those predictions against empirical data. He presents a sample module based on the Keystroke Level Model (KLM).

Leventhal and Barnes (2003) presented another way to bridge the perceived gap between traditional CS and HCI, by integrating HCI in a project-based software development course. This is especially appropriate for departments without a strong HCI orientation. CS graduates will most likely be expected to apply their HCI knowledge to software development, so it is valuable for students to learn HCI in that context. Course topics include introduction to usability, user interface lifecycle, tools for user interface development, design and interaction styles, evaluation, cognitive phenomena, and assistive technologies.

3. Bloom's Taxonomy in Computer Science

One problem with most course descriptions in CS (and other disciplines) is that they simply provide a list of topics to be covered. As a result, the educator has no way of knowing to what extent a given topic should be discussed in class. Bloom's (1956) taxonomy provides a way to organize topics and identify their depth of coverage within the curriculum. Given a concept to be learned by students, Bloom specifies six levels of mastery (or competence) at the cognitive domain. These are recall, comprehension, application, analysis, synthesis, and evaluation. Mastery at a particular level for a particular concept implies mastery at all prior levels.

For example, let's consider the HCI concept of the user interface. The following learning objectives illustrate the six levels of mastering this concept.

1. Recall. "Define user interface." The student is expected to recall memorized information about the concept.
2. Comprehension. "Explain what a user interface is." The student is expected to explain the concept in his or her own words.
3. Application. "Identify the user interface of your car." The student is expected to apply the concept to a particular situation.
4. Analysis. "Analyze the user interface of your car." The student is expected to separate materials or concepts into component parts so that their organizational structure may be understood. For the car interface the student would describe the input and output elements of the user interface, user tasks, and so forth.

5. Synthesis. ''Design a new user interface for your car.'' The student is expected to put parts together to form a whole, with the emphasis on creating a new meaning or structure.
6. Evaluation. ''Evaluate the user interface of your car.'' The student is expected to make judgements about the value of ideas or materials.

A common misconception is to treat the taxonomy as a simple numeric scale (i.e. 1–6), without regard to the semantics of the levels. (This has been an initial tendency, at least in the department of Manaris and Stalvey, where Bloom's taxonomy is being applied across the CS curriculum to specify and refine assessable learning objectives.) Therefore, here we refer to the varying levels of the taxonomy using the two letter abbreviations of their names (Re, Co, Ap, An, Sy, and Ev), to emphasize the semantics of the levels presented in Bloom's original work.

We ask the reader to carefully study the above example in order to internalize the semantics of each level. Otherwise, one may miss the essence of the learning objectives presented in this paper.

Interestingly, the reader may notice that, on the meta level, the learning objective of this section is to ''teach'' how to apply the learning objectives presented in this paper. The authors synthesized the learning objectives presented here in the hope that the reader may easily apply them on their own HCI course.

Although Bloom's taxonomy has been effectively used for many years in many educational domains it has only recently been applied in CS. Lister and Leaney (2003) have discussed using it to assess student performance in an introductory programming course. Scott (2003) has proposed how to apply the taxonomy to testing across the CS curriculum. To the best of our knowledge the taxonomy has never been specifically applied to HCI.

Finally, Bloom's taxonomy is traditionally applied to deriving test questions. (Bloom's official job title was ''University Examiner.'') If one's goal is to derive course requirements (e.g. topics to be covered), this traditional application of Bloom's taxonomy resembles the test-first approach in programming: first, one derives the assessment instruments (e.g. quizzes, assignments, test questions); then, one develops instruction to meet the students' needs (Ardis & Dugas, 2004). In this paper we apply the taxonomy directly to the task of specifying and refining learning objectives. This resembles the design-first approach in programming: first, one designs the curriculum (e.g. identifies the topics and the depth of coverage); then, one develops test questions according to the depth of coverage expected. We believe this approach is better suited to communicating course requirements. It adds a second dimension (depth) to the list of topics to be covered. Also, test questions follow more naturally from such a list.

For each course implementation we present a course outline containing learning objectives and when to first discuss related topics. Each learning objective is combined with a Bloom's level that expresses the student's expected level of mastery upon course completion. Hopefully, this approach facilitates application of the HCI outlines provided into the undergraduate curriculum much more than simply reciting a (flat) list of topics to be covered.

4. Overview of Course Implementations

Obviously, given the general nature of CC′01, there are many possible ways to incorporate HCI into the undergraduate CS curriculum. There are several factors to consider, including student preparation, learning objectives, and various curriculum constraints. This section summarizes the eight course implementations presented in full later in the paper. These implementations have emerged from the collaborative effort of 26 CS educators. Other possible implementations are in preparation by Grissom (2006), Horton (2006), Miller (2006), and Welty (2005).

A brief description of each course implementation is presented in Table 1, along with a list of CC′01 HCI knowledge units covered and the number of lecture hours spent on each unit. Table 1 also lists suggested texts for each course. This snapshot serves as a comparison tool for the courses, allowing the reader to choose an implementation that may effectively be added to the curriculum.

Table 2 provides a quantitative summary of CC′01 knowledge unit coverage per course. For consistency, unit coverage is given as a percentage of course duration.

Table 3 provides the highest Bloom's level reached in each course implementation for each of the eight HCI topics presented in CC′01. It is important to note that these levels are based on the students' expected mastery of a particular topic after completion of the course. Again, topics not covered are marked –. It should be emphasized that the depths of coverage in this table are not absolute. They are relative to: (a) the interpretation of an HCI knowledge unit by each author; (b) the specific learning objectives chosen by each author. For instance, the deepest HC2 objective in Course I is ''apply user and task analysis [Ap],'' whereas in Course II it is ''develop a simple GUI [Sy].'' Table 3 is provided for summary purposes only.

5. Course Implementations

This section provides detailed information on the course implementations summarized in the previous section. The implementations provided are listed from the most elementary to the most advanced. They identify the course level, course title, CC′01 HC knowledge unit coverage, pedagogical considerations, implementation environments (if any), textbooks, scheduling considerations (if any), and assessment of student learning.

5.1. A Freshman Course

This is a web design course entitled ''Designing websites as though users mattered.'' It was developed by Christine Shannon (Centre College, Danville, KY) and has been offered once. This course may be taught in a short intensive term. It covers HC1 (6 hours), HC2 (1.5 hours), HC3 (3 hours), HC4 (4.5 hours), HC5 (7.5 hours), and HC7 (1.5 hours). Because it is aimed at first year students, there are no prerequisites in terms of computer experience beyond basic computer literacy. Students may work

Table 1. Summary of courses presented herein

Course	Description	CC′01 (hours)	Suggested textbooks
I	Level: Freshman course	HC1 (12)	McCracken & Wolfe (2004)
	Title: ''Designing websites as though users matter''	HC2 (3)	Lengel (2004)
	Duration: Short intensive term (appropriate for full semester)	HC3 (4.5)	
	Type: Project-based	HC4 (4.5)	
	Prerequisite: Basic computer literacy	HC5 (10.5)	
	Focus: Web design	HC7 (1.5)	
	Times offered: 1		
II	Level: Sophomore course	HC1 (12)	Preece et al. (2002)
	Title: ''Human–computer interaction''	HC2 (2)	Norman (1998)
	Duration: Semester	HC3 (9)	
	Type: Project-based	HC4 (12)	
	Prerequisite: Programming and GUI building	HC5 (3)	
	Focus: Multidisciplinary human-centered development	HC8 (3)	
	Times offered: 3		
III	Level: Junior course	HC1 (6)	Hix & Hartson (1993)
	Title: ''Software engineering and human–computer interaction''	HC2 (2)	Myers (1994)
	Duration: Semester	HC3 (3)	Myers (1998)
	Type: Project-based	HC4 (6)	Horton (1995)
	Prerequisite: Two semesters of OOP	HC5 (6)	Hayes (2004)
	Focus: Software engineering	HC6 (1)	
	Times offered: >10	HC7 (1)	
IV	Level: Junior/senior course	HC1 (7.5)	Rosson & Carroll (2002)
	Title: ''User interface design''	HC3 (11)	
	Duration: Semester	HC4 (4)	
	Type: Project-based	HC5 (4.5)	
	Prerequisite: GUI programming course	HC8 (3)	
	Focus: Usability engineering		
	Times offered: 5		
V	Level: Junior/senior module	HC1 (1)	Lethbridge & Langaniere (2005)
	Title: ''Module on HCI''	HC3 (2)	Pressman (2001)
	Duration: Two weeks (embedded in a two semester SE sequence)	HC4 (2)	
	Type: Theory-based	HC5 (1)	
	Prerequisite: HC1 and HC2; two semesters of OOP		
	Focus: Software engineering		
	Times offered: 4		

(*continued*)

Table 1. (*Continued*)

Course	Description	CC′01 (hours)	Suggested textbooks
VI	Level: Senior/graduate course	HC1 (6)	Culwin, (1998)
	Title: ''User interface development''	HC2 (2)	Dix et al. (1998)
	Duration: Semester	HC3 (6)	Preece et al. (1994)
	Type: Project-based	HC4 (12)	Preece et al. (2002)
	Prerequisite: Three semesters of programming	HC5 (5)	Norman (1998)
	Focus: User-centered design	HC6 (8)	Spolsky (2001)
	Times offered: >10	HC7 (2)	
VII	Level: Senior/graduate course	HC1 (11)	Norman (1998)
	Title: ''User interface design, implementation and evaluation''	HC3 (13)	Lewis & Rieman (1994)
	Duration: Semester	HC4 (6)	Spolsky (2001)
	Type: Project-based	HC5 (5)	
	Prerequisite: Three semesters of programming, or psychology, or graphic design	HC6 (1)	
	Focus: Task-centered design	HC8 (3)	
	Times offered: 3		
VIII	Level: Senior/graduate course	HC1 (6)	Preece et al. (2002)
	Title: ''User interface design and development''	HC2 (3)	Constantine & Lockwood (1999)
	Duration: Semester	HC3 (10)	Dix et al. (1998)
	Type: Project-based	HC4 (11)	
	Prereq: Three semesters of programming	HC5 (6)	
	Focus: User-centered design	HC6 (4)	
	Times offered: >10		

This table includes a brief description of the course, the number of hours spent on specific CC′01 HCI knowledge units and suggested textbooks. Course numerals (I–VIII) serve as indices to courses within the section *Course Implementations*.

in groups of two or three to design and implement web sites for other offices or departments on campus. The emphasis is on design.

Implementation may be done using existing tools, such as Macromedia *Dreamweaver*, MS *FrontPage*, and Adobe *Photoshop*. Possible textbooks include McCracken and Wolfe (2004) and Lengel (2004). Class sessions last 90 minutes, with nine sessions per week over a three week period. Meetings include both lecture and laboratory activities. At the end of the course students present project designs and implementations. This course can be easily taught in a quarter/semester term by dividing the material and activities appropriately. Table 4 presents an outline of the course (unless stated otherwise, activities are in class).

Table 2. Percent coverage of CC′01 HCI knowledge units across courses

CC′01 category (percentage of hours of coverage per category per course)	Course implementation (across all courses)								
	I	II	III	IV	V	VI	VII	VIII	Average
HC1 (Foundations of HCI)	33%	29%	24%	25%	17%	15%	28%	15%	23%
HC2 (Simple GUI)	8%	5%	8%	–	–	5%	–	8%	4%
HC3 (Human-centered evaluation)	13%	22%	12%	37%	33%	15%	33%	25%	24%
HC4 (Human-centered Dev)	13%	30%	24%	13%	33%	30%	15%	28%	23%
HC5 (GUI design)	30%	7%	24%	15%	17%	12%	13%	15%	17%
HC6 (GUI programming)	–	–	4%	–	–	20%	3%	10%	5%
HC7 (HCI for multimedia)	4%	–	4%	–	–	5%	–	–	2%
HC8 (HCI for collaboration/ communication)	–	7%	–	10%	–	–	8%	–	3%
Total (approx.)	100%	100%	100%	100%	100%	100%	100%	100%	100%

An en rule (–) means no coverage. Course numerals (I–VIII) serve as indices to courses within the section *Course Implementations*.

Table 3. Depth of coverage of CC′01 HCI units per course using Bloom's Taxonomy

CC′01 category (Highest Bloom's level expected per category)	Course implementation							
	I	II	III	IV	V	VI	VII	VIII
HC1 (Foundations of HCI)	[An]	[An]	[Ev]	[Ev]	[Ap]	[An]	[Ev]	[An]
HC2 (Simple GUI)	[Ap]	[Sy]	[Ap]	–	–	[Ap]	–	[Co]
HC3 (Human-centered evaluation)	[Ev]	[Ev]	[Ev]	[Ev]	[Ev]	[Ap]	[Ev]	[Sy]
HC4 (Human-centered development)	[Ev]	[Ev]	[Ev]	[Ev]	[Sy]	[Ev]	[Sy]	[Ev]
HC5 (GUI design)	[Ev]	[Ap]	[An]	[Sy]	[Ap]	[Ev]	[Ev]	[Ev]
HC6 (GUI programming)	–	–	[Ap]	–	–	[Ev]	[Co]	[Ap]
HC7 (HCI for multimedia)	[Sy]	–	[Co]	–	–	[Sy]	–	–
HC8 (HCI for collaboration/ communication)	–	[Co]	–	[Co]	–	–	[Co]	–

An en rule (–) means no coverage. Bloom's level abbreviations are as follows: [Re], recall; [Co], comprehension; [Ap], application; [An], analysis; [Sy], synthesis; [Ev], evaluation.

Table 4. Outline of the freshman course, "Designing web sites as though users mattered"

Day	CC'01	Assessable learning objectives [Bloom's level]	Activities	Reading
1	HC1	Analyze good/bad designs in everyday life [An] Analyze bad web site designs [An] Discuss importance of good design [Co] Discuss goals of HCI [Co] Discuss capabilities of human beings [Co]	Hw: Find examples of poor design in everyday life Discuss BMW iDrive	{M} Ch 1 – 3
2	HC2	Apply user and task analysis [Ap]	Observe examples of perceived affordance	{L} Ch 1 – 2
	HC4	Apply text formatting in *Dreamweaver* [Ap]	Carry out tasks at a commercial web site Identify tasks involved in getting admitted to college Hw: Prepare draft of user and task analysis for project Hw: Begin log of time spent on project	
3	HC5	Apply a variety of organizational systems [Ap] Use card sorting [Ap]	Do grocery card sorting example Hw: Final draft of project user/task analysis Hw: One page summary of handout article	Handout {M} Ch 4
4	HC5	Apply principles of visual organization [Ap]	Paper prototype example	{M} Ch 5
	HC2	*Dreamweaver*: use of tables and images [Ap]	Hw: Write scenarios for 4 – 5 user tasks	{L} Ch 3
5	HC5	Analyze navigation strategies [An] *Dreamweaver*: Design navigation bars and links [Sy] Design context [Sy] Evaluate designs using "Guidelines for homepage usability" [Ev]	Practice with *Dreamweaver* to make navigation bar for a personal homepage Practice with Photoshop to make buttons, etc. Hw: Complete paper prototype and scenarios for project	{M} Ch 6 {L} Ch 7
6	HC5	Study color models [Re] Devise color schemes [Ap]	*Test 1* User testing of paper prototypes	{M} Ch 9 – 10

(*continued*)

Table 4. (*Continued*)

Day	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
		Choose apropriate colors for text/background [Ap]	Hw: Design Navigation system for project	
7	HC5	Study terminology related to type [Re] Choose appropriate type [Ap]	Hw: Write response to handout—"How does this paper relate to our project?"	Handout
8	HC7	Apply multimedia technology on the Web [Ap]	Hw: Prepare images for project and complete navigation system	{M} Ch 11
		Dreamweaver: design content with multimedia [Sy] Photoshop: design images [Sy]		{L} Ch 4
9	HC4	Prototyping again [Ev]	Paper prototyping of note-taking system	{M} Ch 7–8
10	HC3	Perform human centered evaluation [Ev]	Discussion of case study	Handout
11			*Test 2* Testing of preliminary version of project by classmates	Handout
12	HC1	Discuss accessibility issues [Co]	Hw: Report on user testing (classmates) Hw: Accessibility presentations	{M} Ch 12
13	HC3	Design user testing sessions for participants	Student presentations on accessibility outside the course [Sy]	ID: Ch 10
14	HC1	Analyze globalization issues [An]	Evaluation of college website Hw: prepare scenarios for outside user testing	{M} Ch 13–14
15	HC1	Analyze societal issues: privacy and trust [An]	User testing with outside participants Hw: Complete project, final portfolio, and log	
16			Final presentation to class/ clients	

{M}, McCracken and Wolfe (2004); {L}, Lengel (2004). Bloom's level abbreviations are as follows: [Re], recall; [Co], comprehension; [Ap], application; [An], analysis; [Sy], synthesis; [Ev], evaluation.

5.2. A Sophomore Course

This is a project-based course entitled "Human–computer interaction." It was developed by John Wright (Juniata College, Huntingdon, PA) and has been offered three times. It covers HC1 (12 hours) and HC2 (2 hours), HC3 (9 hours), HC4 (12 hours), HC5 (3 hours), and HC8 (3 hours). It is placed at the sophomore level to expose students to human-centered development early on in the CS/IT sequence. In addition, this makes it accessible to non-majors. Since HCI is inherently multidisciplinary, one may attract psychology, sociology, education, communications, and anthropology students, along with IT and CS majors. The inclusion of non-majors is important, particularly in discussions about multidisciplinary teams.

The project requires some programming, so student teams should include at least one CS major. It is assumed that CS/IT students have some exposure to building GUIs, so the project is presented during the second half of the semester. It includes design, evaluation through paper prototyping, and a final implementation of the GUI (HC2, HC4, and HC5). For the GUI implementations students can choose their preferred environment. Recommendations include Visual Basic, C#, and Java (GUI builder in Borland's JBuilder™, Visual JavaBeans Designer). The first half of the semester focuses primarily on the study of humans, cognition and abilities, communication and collaboration, and how these affect and are affected by technology (HC1 and HC8).

Textbooks used include Preece, Rogers, and Sharpe (2002) and Norman (1998). These two texts work well together, as noted in the end of semester course evaluations.

One other important aspect of this course is an open, collaborative classroom atmosphere. Students are encouraged to speak up in class, make observations, and ask questions. Each session begins with an informal discussion session (5–10 minutes) about the topics being studied (e.g. examples of bad and good UIs). Students are encouraged to attend lectures on cognitive psychology for an extra credit and initiate class discussions. Table 5 presents an outline of the course (unless stated otherwise, activities are in class).

5.3. A Junior Course

This is a course integrating HCI and SE entitled "Software engineering and human–computer interaction." It was developed by Laura Leventhal and Julie Barnes (Bowling Green State University, Bowling Green, OH). This course has been taught at the undergraduate level approximately once per year since 1988. It is a required core course for CS majors. Students may take it as early as the sophomore year. The course covers HC1 (6 hours), HC2 (2 hours), HC3 (3 hours), HC4 (6 hours), HC5 (6 hours), HC6 (1 hour), and HC7 (1 hour). It also introduces several SE units, namely SE1 (2 hours), SE3 (1 hour), SE4 (1 hour), SE5 (1 hour), and SE7–SE9 (1 hour). Students are expected to have completed two semesters of object-oriented programming. There is no GUI building prerequisite.

Table 5. Outline of the sophomore course, "Human–computer interaction"

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
1	HC1	Discuss HCI and ID definitions [Co]	Hw: Identify good and bad interfaces	{ID} Ch 1
		Discuss rationale for learning HCI [Co]	Recognize UIs as abstraction barriers (e.g. candy machines, doors, etc.)	{tDoET} Ch 1
		Analyze UIs as abstraction barriers [An]		
		Analyze UI components of everyday things [An]		
2	HC1	Discuss UI history and evolution [Co]	Hw: Identify and define terms from {tDoET} Ch 1	{ID} Ch 1
		Discuss disciplines contributing to HCI [Co]	Analyze 1996 Eagle Vision TSi AutoStick® UI and subsequent refinement	{tDoET} Ch 2
		Discuss HCI relevance to business [Co]		
		Discuss usability terminology [Co]	Hw: Assign project teams	
		Apply Norman's design analysis concepts [Ap]		
3	HC3	Discuss cognition and usability (e.g. mental models, standards vs. guidelines, HCI guidelines) [Co]	GUI blunders	{ID} Ch 3
	HC2	Discuss cognition processes (attention, perception, listening, problem solving, planning, reasoning, and decision-making) [Co]	Hw: Identify usability issues	
4	HC5	Analyze Norman's concept of errors, user helplessness, gulfs of evaluation and execution [An]	Hw: Gulf of evaluation and execution	{ID} Ch 3
	HC1	Discuss metaphors (e.g. simple vwesus composite, design guidelines, etc.) [Co]		
		Discuss conceptual frameworks of cognition [Co]		

(*continued*)

Table 5. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
5	HC8	Discuss communication and collaboration (social mechanisms of communication, support of conversation, coordination, and awareness) [Co]	Hw: Analysis of Disney web site	{ID} Ch 4
	HC2		*Quiz 1*	{tDoET} Ch 4
	HC1	Cognitive psychology lecture—extra credit		
		Discuss ethnography principles [Co]		
		Discuss Norman's conceptual models and memory [Co]		
6	HC1	Discuss rationale for learning (reinforcement) [Co]	User or designer blame? (UIs for voting)	{ID} Ch 5
		Analyze importance of aesthetics and repercussions of frustration [An]	Cognitive psychology lecture—extra credit	Handouts
7	HC4	Apply interaction design process, UI lifecycle	Hw: Handling errors (slips and mistakes)	{ID} Ch 6
	HC5	Mayhew usability lifecycle [Ap]	Mid-term exam	{tDoET} Ch 5
		Discuss Norman's dealing with errors (slips and mistakes) [Co]	Cognitive psychology lecture—extra credit	
8	HC4	Identify user needs and requirements [Sy]	Prepare for project—develop a Zen alarm clock	{ID} Ch 7
		Interpret and analyze user data [An]		{tDoET} Ch 6
9	HC4	Analyze scenarios, use cases, essential use cases, and user tasks (task analysis, HTA) [An]	Project: needs and requirements	{ID} Ch 7
	HC1	Discuss Norman's design and evolution [Co]		
		Apply accessibility principles [Ap]		
		Discuss the foibles of computer systems [Co]		

(*continued*)

Table 5. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
10	HC4	Design and evaluate paper prototypes [Ev]	Why evaluate: BMW iDrive, drive-by-wire	{ID} Ch 8
			Project: perform task analysis; refine paper prototype	{tDoET} Ch 7
				Handouts
11	HC4	Analyze user-centered approaches to design [An]	Project: design paper prototype	Ch 9
			Quiz 2	
12	HC3	Perform usability testing with paper prototypes [Ev]	Project: usability testing with paper prototypes	{ID} Ch 10
		Apply ethnography principles [Ap]	Evaluation case studies	
		Apply human-centered evaluation [Ap]		
13	HC3	Apply human-centered evaluation [Ap]	Project: prototype implementation	{ID} Ch 10
	HC2	Develop a simple GUI [Sy]		
14	HC3	Analyze evaluation frameworks [An]	Predictive evaluation: HCI class helps software engineering class evaluate projects from a user-centered perspective	{ID} Ch 11
		Discuss HCI labs and design war rooms [Co]		{ID} Ch 14.5
		Apply predictive modeling techniques (e.g. GOMS, Keystroke level model, Fitt's law) [Ap]		
15	Putting it all together	Project: usability testing of prototypes	Review of Final exam	{tDoET} Ch 7

{ID}, Preece et al. (2002); {tDoET}, Norman (1998). Bloom's level abbreviations are as follows: [Re], recall; [Co], comprehension; [Ap], application; [An], analysis; [Sy], synthesis; [Ev], evaluation.

At the beginning of the semester an effort is made to disrupt the equivalency of CS and coding by emphasizing the usability of non-computing systems. Norman's concept of "The psychology of everyday things" is introduced and students investigate several models of usability. Several models of software and user interface

development are introduced. For the term project students work in teams of two. They begin with task analysis for the project; this is due at mid-term. While the students are working on the task analysis lectures cover the basics of the prototyping tool to be used. After mid-term topics include interaction styles, prototyping, and usability testing. The second phase of the project, which includes a UI prototype and a usability assessment of the prototype, is due at the end of the semester.

For several years the primary text for the course has been Hix and Hartson (1993). Recently the instructors have been developing their own textbook. It is supplemented with a number of articles. Table 6 presents an outline of this course (unless stated otherwise, activities are in class).

5.4. A Junior/Senior Course

This is a design-oriented, project-based HCI course entitled "User interface design." It was developed by Arthur Kirkpatrick (Simon Fraser University, Burnaby, Canada) and has been offered five times. It covers HC1 (7.5 hours), HC3 (11 hours), HC4 (4 hours), HC5 (4.5 hours), and HC8 (3 hours). This course assumes that students will have a class in user interface programming elsewhere in the curriculum (HC2 and HC6). Therefore, in contrast to most other implementations presented in this paper, it focuses explicitly on the process of user-centered design. This is driven by the belief that design, both in the abstract and those aspects specific to user interfaces, is the most challenging part of user interfaces for typical CS students. Thus, if curriculum constraints allow it, one should give students as much practice and theory of user-centered design as possible in an HCI course and leave the programming aspects to other courses.

The course topics follow the sequence of design and evaluation steps for a software product. Students work in groups of three or four (preferred) on a single project for the duration of the semester. In the first phase students create a requirements document. They interview potential users in their place of use, developing scenarios of use and persona descriptions. In the second phase students design an interface. They turn in screen shots of the interface, together with more detailed scenarios of use. In the final phase they revise their interface based upon instructor feedback, construct a paper prototype, and run several users through usability tests with that prototype. The deliverable for this phase is a usability test report. Additionally, there are weekly homework exercises, a mid-term, and a final. The homework typically consists of short analyses of detail points of an interface and gives students practice in specific mechanics of design. The textbook used in this course is Rosson and Carroll (2002), due to its project orientation, but other common texts could be selected. Table 7 presents an outline of the course (unless stated otherwise, activities are in class).

5.5. A Junior/Senior Software Engineering HCI Module

This is an HCI module, which may be incorporated into an upper level SE or related course. It was developed by Dean Sanders (Northwest Missouri State University,

Table 6. Outline of junior course, ''Software engineering and human–computer interaction''

Week	CC'01	Assessable learning objectives [Bloom's level]	Activities	Reading
1	HC1	Discuss HCI and ID definitions [Co]	Students identify the interface and functional components of everyday objects provided by the instructor	Preface, Ch 1–2 of class notes
		Discuss rationale for learning HCI [Co]		
		Discuss UI history and evolution [Co]	Hw: Students select an everyday object and identify the user interface, constraints, affordances, mappings	{M94}
		Analyze UIs as abstraction barriers [An]		{M98}
		Discuss abstraction components in UIs of everyday things [An]		{H04}
2	HC1	Discuss history of HCI [Co]	Hw: Given common household appliance, develop measures as per Eason's usability model.	Ch 3–4 of class notes
		Discuss Shackel's and Nielsen's models of usability [Co]		
		Evaluate UIs using Eason's model [Ev]		
		Measure usability of UIs [Ev]		
		Apply scientific method [Ap]		
3	SE4	Apply software engineering life cycle [Ap]	Movie and discussion of accidents and human variables	
	HC4	Discuss waterfall model [Co]	*Quiz 1*	
		Apply usability engineering life cycle [Ap]		

(*continued*)

Table 6. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
		Apply usability engineering model [Ap]		
4	SE5	Discuss requirements analysis and specification [Co]		Ch 5 – 6 of class notes
	SE8	Discuss team issues [Co]	Notes	
	HC4	Discuss participatory design [Co]		
5	HC4	Evaluate task analysis and specification [Ev]	Task analysis is assigned	Handout on prototyping environment
		Create use cases [Sy]	*Quiz 2*	
		Discuss user profile, needs analysis [Co]		
6	HC2	Discuss prototyping tools [Co]	*Exam 1*	
	SE3	Use Visual Basic environment [Ap]		
7	HC2	Use Visual Basic environment [Ap]	Lab day	Ch 7 of class notes
	HC6	Visual Basic Lab 1		
8	HC5	Apply general design guidelines [Ap]	View and discuss Tandy Trower video: "Creating a well-designed user interface"	Ch 8 of class notes
	SE1		Work day for task analysis	
			Task analysis is due	
9	HC5	Analyze interaction styles: menus, forms, windows [An]	Visual Basic Lab 2	{H95}
10	HC4	Create prototypes [Sy]	*Quiz 3*	Ch 10 & 12 of class notes
	HC6	Learn about UIMS [Re]	Paper prototyping exercise	
11	HC3	Perform usability testing [Ev]	Project: design paper prototype	Ch 13 of class notes
		Evaluate using qualitative/	*Quiz 2*	

(*continued*)

Table 6. (*Continued*)

Week	CC'01	Assessable learning objectives [Bloom's level]	Activities	Reading
		quantitative measures [Ev] Evaluate using objective/ subjective measures [Ev]		
12	HC3	Apply standards and guidelines [Ap]	Project: usability testing with paper prototypes	Ch 9–11 of class notes
	HC5	Discuss other interaction styles [Co]	Evaluation case studies	
	HC7	Discuss multimedia and speech systems [Co]		
13	SE9	Discuss software quality [Co]	*Exam 2*	Ch 14–15 of class notes
	SE7	Apply coupling & cohesion in coding [Ap] Apply software reuse [Ap]		
14	HC3	Learn about cognition [Re]		
15	HC1	Learn about universal Usability [Re]	Prototype and usability assessment are due	

{DUI}, Hix and Hartson (1993); {M94}, Myers (1994); {M98}, Myers (1998); {H95}, Horton (1995); {H04}, Hayes (2004). Bloom's level abbreviations are as follows: [Re], recall; [Co], comprehension; [Ap], application; [An], analysis; [Sy], synthesis; [Ev], evaluation.

Maryville, MO) and has been offered four times (Sanders 2005). It covers HC1 (1 hour), HC3 (2 hours), HC4 (2 hours), and HC5 (1 hour). This module aims to teach students how to match the interface to the users and their tasks—a more advanced and the most important aspect of UI development. It assumes that the core knowledge units, HC1 and HC2, have already been covered in the introductory programming sequence: students have been taught a particular GUI toolkit and they are required to develop part or all of a GUI in most programming assignments.

The development of this module was guided by the following observation: there is a significant overlap between the steps in designing a user interface and the requirements gathering and systems analysis activities normally associated with software development. In both cases we need to develop profiles of the users and determine the tasks users intend to perform with the software. As the development

Table 7. Outline of junior/senior course, "User interface design"

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
1	HC1	The importance of good UI design [Co]	Paper prototyping and usability testing of transit fare kiosk	{UE} Ch. 1
	HC3	Evaluating a paper prototype using scenario-based design process [Ev]		
	HC4			
2	HC4	User-centered requirements [Ev]	Project: assign teams	{UE} Ch. 2.0 – 2.2
			Project: start requirements analysis	
3	HC1	Discuss example requirements	Analysis of sample work environments and interviews	{UE} Ch. 2.3
	HC4	Work flow, conceptual model, and metaphors for design [An]		{UE} Ch. 3.0 – 3.3
4	HC1	Activity design [Ap]	Locate strengths and weakness of activity design in sample applications	{UE} Ch. 3.4
	HC4	Information design: basic layout [Co]		
	HC5	Project: requirements analysis due		{UE} Ch. 4.0 – 4.3
5	HC1	Supporting sense-making: graphically representing conceptual model, choice of labels [Ev]	Analyze sample applications	{UE} Ch. 4.4 – 4.6
	HC5	Information design: complex layouts [Sy]		
6	HC1	Interaction design: definition of interaction techniques [Ap]	Project: start interface design	{UE} Ch. 5.0 – 5.3
	HC5	Common widgets and interaction techniques [Ap]		
7	HC1	Example interaction design	Analyze sample applications	{UE} Ch. 5.4
	HC5			

(*continued*)

Table 7. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
8	HC3	Mid-term Ethical treatment of human participants [Ev]	Case studies of usability ethics	
9	HC3	Prototyping methods [Sy] Discussion of interface design assignment; return of mid-term		{UE} Ch. 6.0–6.4
10	HC3	When to use each prototyping method [Sy]	Hw: Selecting a prototyping method	{UE} Ch. 6.5
		Evaluation by testing: usability tests and controlled experiments [An]	Project: interface design due	{UE} Ch. 7.3
			Project: start usability test	
11	HC3	Usability inspections: heuristic evaluation [Ev]	Hw: Selecting an inspection method and evaluating an interaction technique	{UE} Ch. 7.4.1
		Inspections: Keystroke level model [Ev]		
12	HC3	Inspections: cognitive walk through [Ev]	Sample walk throughs	{UE} Ch. 7.4.2–7.4.4
	HC8	Emerging interaction paradigms: virtual reality and ApD interfaces [Co]		{UE} Ch. 9.0–9.4
13	HC8	Emerging interactions: collaborative and mobile interfaces [Co]	Project: usability test due	{UE} Ch. 9.5

{UE}, Rosson and Carroll (2002). Bloom's level abbreviations are as follows: [Re], recall; [Co], comprehension; [Ap], application; [An], analysis; [Sy], synthesis; [Ev], evaluation.

proceeds we need to evaluate both the functionality of the software and the design of the user interface. Therefore, these aspects of HCI may be easily incorporated into the context of an existing SE or related course.

Students work in small ad hoc groups, usually different from the project teams, to design, evaluate, and refine a prototype for a user interface. The students are given three to six scenarios of individuals performing tasks on a proposed system. User

characteristics and task descriptions are embedded in the scenarios. Each group is required to use the scenarios as the basis for four major activities. First, they design a small usability study to evaluate the prototype they will produce. Creating the usability study before the prototype forces the students to focus on the users' tasks and goals, rather than artifacts in the interface itself. Second, they develop a paper prototype for a user interface. Third, the groups pair up in class and serve as users for each other's usability study. Finally, each group uses the results of their usability study to revise their prototype. This gives the students practice in designing and evaluating a user interface, as well as experience with a prototyping cycle.

The following outline has been used as a portion of a two semester SE course. Unlike most other outlines in this paper, this one is organized by day (50 minute class period) rather than by week. It has also been used in a reduced version in a one semester course on systems analysis and design. The classroom time can be decreased by eliminating some of the lecture topics, the scope of the prototype can be reduced, and the prototype itself can be developed outside class. However, the evaluation should be a classroom activity.

Only a few SE textbooks have reasonable sections on user interface design. Possibilities include Lethbridge and Langaniere (2005) and Pressman (2001). These may be supplemented with references from the World Wide Web Consortium (W3C) (2005) and course modules of the Network Community for Software Engineering Education (http://www.swenet.org). Table 8 presents an outline of the course (unless stated otherwise, activities are in class).

5.6. A Senior/Graduate Course

This is a junior/senior/graduate course entitled "User interface development" developed by Bill Manaris (College of Charleston, Charleston, SC). It has been offered since 1995 approximately once a year at the University of Louisiana and the College of Charleston. It covers HC1 (6 hours), HC2 (2 hours), HC3 (6 hours), HC4 (12 hours), HC5 (5 hours), HC6 (8 hours), and HC7 (2 hours). It stresses the importance of good interfaces, as well as the relationship of user interface design to human–computer interaction. It aims to expose students to UI design, evaluation, and implementation.

Students are assumed to have no previous experience of building GUIs. Therefore, GUI building should be introduced as soon as possible, so that students have enough time to practice prior to the implementation phase of the project. One possible drawback of this is that it pushes paper prototyping back to later in the semester. However, if students already have some experience of building GUIs prior to this course, paper prototyping could be covered as early as the fifth week. Graduate students are expected to do additional readings and homework. Students are expected to evaluate each other's work through cognitive walk through sessions, prototype evaluation, and demonstrations. Implementation tools may include Python/wxPython and Java/Swing. A good text for this course is Preece et al. (2002), together with Spolsky (2001). Another possibility is Stone, Jarrett,

Table 8. Outline of junior/senior software engineering module

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
1	HC1	Review GUI development topics [Ap]	Hw: Evaluate specified web pages for accessibility and/or internationalization	Class notes
		Discuss universal accessibility and internationalization [Co]		{W3C}
2–3	HC4	Apply guidelines for UI design given user profiles and task analyses [Ap]	Critique examples	Class notes
	HC5			
4	HC3	Apply UI design guidelines and usability principles in UI evaluation [Ap]	Discuss prototyping activity specs	Class Notes
			Hw: Design a small usability study to use with prototyping activity	
5	HC4	Create paper prototype [Sy]	Ad hoc teams begin to develop a paper prototype in class and finish outside class	
6	HC3	Evaluate paper prototype [Ev]	Teams conduct usability studies on one another's paper prototypes	
			Hw: Each team writes report based on feedback from the usability study and refines prototype	

{W3C}, World Wide Web Consortium (2005). Bloom's level abbreviations are as follows: [Re], recall; [Co], comprehension; [Ap], application; [An], analysis; [Sy], synthesis; [Ev], evaluation.

Woodroffe, and Minocha (2005). Supplementary materials may be provided from Culwin (1998), Dix, Finlay, Abowd, and Beale (1998), McCracken and Wolfe (2004), Preece et al. (2002), Preece, Rogers, Sharp, Benyon, Holland, and Cary (1994), and Norman (1998). Table 9 presents an outline of the course (unless stated otherwise, activities are in class).

Table 9. Outline of senior/graduate course, "User interface development"

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
1	HC1	Discuss HCI and ID definitions [Co]	Recognize UIs as abstraction barriers (e.g. instructor's watch, automobile)	{ID} Ch 1
		Discuss rationale for learning HCI [Co]	Interact with Norman's refrigerator (DoET)	{UIDfP} Ch 1
		Discuss UI history and evolution [Co]	When designers ignore consumers: BMW iDrive, Konica camera/MP3 player	
		Analyze UIs as abstraction barriers [An]		
		Analyze abstraction components in UIs of everyday things [An]	Innovative UIs: Dasher, TextArc	
2	HC1	Discuss usability terminology [Co]	Analyze 1996 Eagle Vision TSi AutoStick® UI and subsequent refinement	{UIDfP} Ch 2–4
		Analyze UIs using Norman's design concepts [An]		{tDoET} Ch 1
		Discuss measurable human factors [Co]	Project: pick a freeware program; identify usability issues	{HCI.a} Ch 3
		Analyze UIs using Nielsen's usability principles [An]		
		Analyze UIs using interaction frameworks [An]		
		Analyze bad UIs [An]		
3	HC3	Apply cognition and usability concepts (e.g. mental guidelines) [Ap]	Project: presentations models, standards versus guidelines, HCI	{ID} Ch 2, 3
		Apply interaction styles and paradigms [Ap]		{DUI} Ch 2
4	HC5	Analyze metaphors (e.g. simple versus composite, design guidelines, etc.) [Ap]	Refine a composite metaphor	{UIDfP} Ch 5, 6

(*continued*)

Table 9. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
	HC7		Design a simple command line interface	{DtUI} Ch 8
		Design a simple command-line interface [Sy]	*Test 1*	
		Discuss natural language interfaces [Co]		
5	HC6	Discuss prototyping languages and IDEs (e.g. Visual Basic, NetBeans, V C++, Gnome, Python, Tcl/tk) [Co]	Hw: Develop a GUI prototype with IDE of choice	Handouts
	HC2	Apply event control model [Ap]		
		Apply basics of a GUI prototyping language and IDE [Ap]		
6	HC2	Apply basics of a GUI prototyping language and IDE [Ap]	Demos of IDE and sample GUIs	Handouts
	HC6	Apply GUI modularization techniques (e.g. presentation–translation–application layers, Model–View–Controller, etc.) [Ap]		
7	HC4	Apply interaction design process, UI lifecycle Mayhew usability lifecycle [Ap]	View Tandy Trower video: ''Creating a well-designed user interface''	{ID} Ch 6
				{UIDfP} Ch 12
8	HC4	Identify user needs and requirements [Sy]	Project: identify user needs for freeware program of choice; develop scenarios for different user profiles	{ID} Ch 7
		Apply scenario-based usability methods [Ap]		{UIDfP} Ch 7, 9–11

(*continued*)

Table 9. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
9	HC4	Task analysis (HTA) [Ev]	Project: perform task analysis; refine paper prototype	{ID} Ch 7
				{HCI.a} Ch 7
10	HC4	Develop paper prototypes [Ev]	View Nielsen-Norman Group video: ''Paper prototyping: a how-to training''	{ID} Ch 8
			Project: develop paper prototype for program of choice	Handouts
11	HC6	Develop formal models of interaction design (e.g. state transition diagrams) [Ev]	Project: evaluate and refine prototype	{aJGPP} Ch 1
			Project: develop STD	{HCI.a} Ch 8
12	HC5	Develop selection spaces (e.g. GUI menus, web interfaces) [Ev]	Project: implement prototype	Handouts
				{UIDfP} Ch 14–17
		Apply design patterns for web interfaces [Ap]		{DtUI} Ch 7
13	HC3	Apply human-centered evaluation [Ap]	*Test 2*	{ID} Ch 10
14	HC3	Apply predictive modeling techniques (e.g. GOMS, Keystroke level model, Fitt's law) [Ap]	Project: evaluate another team's prototype	{ID} Ch 14
				{UIDfP} Ch 13
15		Putting it all together	Project: presentations Partner/course evaluations	

{aJGPP}, Culwin (1998); {HCIa}, Dix et al. (1998); {HCIb}, Preece et al. (1994); {ID}, Preece et al. (2002); {tDoET}, Norman (1998); {UIDfP}, Spolsky (2001). Bloom's level abbreviations are as follows: [Re], recall; [Co], comprehension; [Ap], application; [An], analysis; [Sy], synthesis; [Ev], evaluation.

5.7. A Senior/Graduate Course

This is a junior/senior/graduate course entitled "User interface design, implementation, and evaluation," developed by J. Ben Schafer (University of Northern Iowa, Cedar Falls, IA). It has been offered every spring since 2004. It covers HC1 (11 hours), HC3 (13 hours), HC4 (6 hours), HC5 (5 hours), HC6 (1 hour), and HC8 (3 hours). It assumes that HC2 (GUI implementation) has been covered in prerequisite courses. Students are required to attend two "lecture" sessions, one group "design critique" and anywhere from 2 to 10 hours per week of student-led work meetings. Graduate students are expected to do additional readings and homework. Students may be either CS majors who have completed at least a three semester programming sequence or upper division majors from an HCI-related field, such as psychology and graphical design. Students work in teams of four or five on a semester-long project of their selection. Projects should conform to three key requirements.

1. The problem must be primarily an interface problem.
2. The problem must be one for which the group can find at least two real users who are willing to work with the group. As a minimum these users should be able to meet with the group at three different times: task analysis, paper prototyping, and user testing.
3. The problem must be one that is suitable for the task-centered design approach used in the course.

Recommended textbooks include Norman (1998), Lewis and Rieman (1994), and Spolsky (2001). Table 10 presents an outline of the course (unless stated otherwise, activities are in class).

5.8. A Senior/Graduate Course

This is a project-based HCI course entitled "User interface design and development." It was developed by Michael Wainer (Southern Illinois University, Carbondale, IL) and has been offered approximately once per year since 1994. It covers HC1 (6 hours), HC2 (3 hours), HC3 (10 hours), HC4 (11 hours), HC5 (6 hours), and HC6 (4 hours). It is assumed that students have a strong CS background but no previous experience of building GUIs. Therefore, prototyping and the design process is emphasized, rather than full implementations for project work. Paper prototyping is introduced early as it helps to promote both team building and class interactions. The course aims to show how interaction design and software design can coexist and to sensitize CS students to user-centered design principles and working as a team. It also seeks to dispel the notion that simply knowing how to code a GUI is all that is needed to make usable products.

One possibility for a course project is a memory game with customization features. Roughly two-thirds of the semester focuses on project design and evaluation activities. Most assignments are done in groups of three or four. Project activities and

Table 10. Outline of senior/graduate course, "User interface design, implementation and evaluation"

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
1	HC1	Discuss HCI and ID definitions [Co]	Project: discuss project constraints and guidelines	
		Discuss rationale for learning HCI [Co]		
2	HC1	Analyze UIs as abstraction barriers [An]	Hw: It bugs me!	{tDoET}
	HC5	Analyze abstraction components in UIs of everyday things [An]		
		Discuss usability terminology [Co]		
		Analyze UIs using Norman's design concepts [An]		
		Analyze UIs using interaction frameworks [An]		
		Study of bad UIs [An]		
		Discuss metaphors [Co]		
3	HC3	Apply cognition and usability principles (e.g. mental models, HCI guidelines) [An]	*Quiz 1*	{TCUID} Ch 1, 2
	HC4	Apply user/task-centered design process [Ap]		
		Apply interaction design process, UI lifecycle [Ap]		
4	HC4	Identify user needs and requirements [Sy]	Project: project proposal, user visit plan	
		Analyze scenarios and personas [An]	Project: conduct site visits	
		Apply site visit guidelines [An]		
5	HC4	Develop paper prototypes [Sy]	Project: visit report including user and task analysis	{TCUID} Ch 3
	HC5	Apply interaction styles and paradigms [Ap]		

(*continued*)

Table 10. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
	HC6	Compare prototyping languages and IDEs [Co]	View Nielsen-Norman Group video: ''Paper prototyping: a how-to training''	
6	HC1	Apply UI design guidelines [Ap]	Project: develop paper prototype	
		Apply guidelines for preventing errors [Ap]	*Quiz 2*	
7	HC1	Discuss UI history and evolution [Co]	Project: perform scenario walk throughs	{TCUID} Ch 4
	HC3	Discuss cognitive walk throughs [Co]		
	HC4	Analyze predictive modeling techniques (e.g. GOMS, Keystroke level model, Fitt's law) [An]		
	HC8	Develop task analysis/ scenarios [Ev]		
		Discuss ubiquitous computing [Co]		
8	HC3	Apply cognitive walk throughs [Ap]		
9	HC1	Evaluate UIs using Nielsen's usability principles [Ev]	Project: present cognitive walk through results	
	HC3	Evaluate UIs using cognitive walk throughs [Ev]		
		Apply heuristic evaluations [An]		
10	HC3	Design heuristic evaluations [Sy]	Project: perform heuristic evaluations	{UIDfP} Ch 1–6
11	HC1	Discuss measurable human factors [Co]	Project: implement prototype, design user evaluation plans	{TCUID} Ch 5
	HC3	Apply human-centered evaluation [An]	*Quiz 3*	

(*continued*)

Table 10. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
12	HC5	Discuss selection spaces (e.g. GUI menus, web interfaces) [Ev]	Project: conduct user evaluations	{UIDfP} Ch 7 – 12
	HC8	Discuss ubiquitous computing [Co]		
		Discuss computer-supported cooperative work [Co]		
13			Project: user evaluation results	
			Final exam	
14	HC5	Discuss design principles for the Web [Co]	Project: change list, presentation plans	{UIDfP} Ch 13 – 18
			Quiz 4	
15		Project open house	Project: final prototype and report	
			Course evaluations	

{tDoET}, Norman (1998); {TCUID}, Lewis and Rieman (1994); {UIDfP}, Spolsky (2001). Bloom's level abbreviations are as follows: [Re], recall; [Co], comprehension; [Ap], application; [An], analysis; [Sy], synthesis; [Ev], evaluation.

deliverables are sometimes assigned as homework to force due dates and better provide feedback to students throughout the semester. A tutorial on the implementation of interfaces with Java and a modern form editor is given. Students may continue with this environment to create detailed screen shots for their final design submission.

A good text possibility for this course is Preece et al. (2002) as well as Dix et al. (1998). For more emphasis on software design for user interfaces one may consider Constantine and Lockwood (1999) and Horrocks (1999). Additional background material for class readings may be found at various web sites, such as Ambler (2005), Nielsen (2005), and Sun Microsystems (2005). Table 11 presents an outline of the course (unless stated otherwise, activities are in class).

6. Discussion

Given the importance of usability in interactive technology as well as the leading role of CS graduates in developing such technology, there is an increasing need to incorporate HCI in undergraduate CS curricula. As the outlines provided demonstrate, there are different ways to do so. For instance, if the curriculum has

Table 11. Outline of senior/graduate course, "User interface design and development"

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
1	HC1	Discuss HCI and ID definitions [Co]	Discuss examples of bad usability: 2000 election, automated readings customer service, etc.	{ID} Ch 1
		Discuss motivation for learning HCI [Co]		
		Discuss human-centered development, usability testing [Co]	Discuss alternatives for obtaining credit card info	
		Analyze UIs using Norman's design concepts [An]		
2	HC1	Apply interaction design process [Ap]: basic activities, stakeholders, user-centered design, life cycles, big upfront design versus agile methods, interaction design versus software design	Hw: Capture user needs (digital photo printing service application/ kiosk/web service for various types of users)	{ID} Ch 6 readings
	HC4			
3–4	HC4	Develop UIs through iterative prototype refinement [Ev]	Software prototyping (lab tutorial)	{ID} Ch 8.1–8.2
	HC2		Hw: Paper prototyping of memory game, gather feedback	Readings
	HC6	Discuss types of prototypes (low versus high fidelity) [Co]		
		Discuss MVC and software prototypes [Co]		
		Abstract prototypes and interface flow [An]		
5–6	HC4	Develop tasks, scenarios, use cases, essential use cases, HTA [Ev]	Discuss/compare expressions of tasks and task analysis for various examples: ATM, game turn, making tea	{ID} Ch 8.1–8.3
				{ID} Ch 7.3

(*continued*)

Table 11. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
		Analyze functional & non-functional requirements [An]	Hw: Submit final report	{ID}7.6 – 7.7
		Analyze conceptual model, metaphors, task allocation [An]		{ID} Ch 2 Readings
7	HC1	Apply evaluation techniques [Ap]	Discuss Hutchworld case study	{ID} Ch 10
	HC3		*Exam 1*	
8	HC5	Analyze, synthesize and evaluate project UI based upon detailed design guidelines and standards [Ev]	Hw: Detailed design of customizer application	{ID} Ch 8.4
			Concept feedback, prototyping	Readings
9	HC4	Apply structure of event handling software, event loop, callbacks, MVC, widgets, layout managers, resources, internationalization [Ap]	Review code samples various languages and approaches	Readings
	HC6		Why layout managers?	
10	HC3	Apply evaluation framework, paradigms, user observations [Ev]	Perform evaluations on Customizer App design	{ID} Ch 11 – 12
11	HC3	Design interviews, questionnaires [Sy]	Discuss appropriate heuristics for customizer application	{ID} Ch 13
		Compare heuristic evaluation to user testing [Co]	Hw: Write heuristic evaluation questionnaire	Readings
		Discuss evaluation in practice [Co]		
12	HC5	Use statecharts to specify interaction design [Sy]	Horrocks' VB calculated comparison, other state chart examples	Readings
		Discuss notations, screen rules,	Hw: Merge heuristic evaluation	

(*continued*)

Table 11. (*Continued*)

Week	CC′01	Assessable learning objectives [Bloom's level]	Activities	Reading
		heuristics, design checks [Co]	questionnaires to a single questionnaire per group. Evaluate customizer application	
13	HC3	Apply predictive models (GOMS, KLM) [Ap]	Conduct and receive heuristic evaluation on project designs, compare interfaces with KLM	{ID} Ch 14.5 readings
14	HC3	Apply design metrics [Ap]	Work out examples, discuss role of metrics/heuristics in design	Readings
	HC5	Discuss essential efficiency, task concordance, task visibility [Co]		
15	HC3	Apply Fitts' Law [Ap]	Examples	{ID} Ch 14.5
		Apply design patterns in HCI [Ap]	Hw: Final project report with severity ratings and recommendations	{ID} Ch 8.4
		Discuss future trends [Co]		
			Final exam, course evaluations	

already covered HC2 (simple GUI) in an earlier course (e.g. CS1/CS2) then the HCI course may concentrate on a user-centered perspective and UI design, as opposed to programming (e.g. Course IV). An alternative would be to cover HCI design as a module in a SE course (e.g. Course V). However, some CS departments can only incorporate HC2 within an independent HCI course. This is the approach used by most outlines in the paper. Finally, an interesting possibility is to follow an HCI-first approach (e.g. Course I), which establishes a HCI foundation for the rest of the curriculum.

Interestingly, all outlines provided include the following CC′01 knowledge units: HC3 (human-centered software evaluation), HC4 (human-centered software development), and HC5 (graphical user interface design).

In particular, HC3 focuses on students being able to evaluate a user interface by applying human-centered usability principles and guidelines. In terms of phrasing

learning objectives, if the emphasis is placed on the artifact (user interface), students are expected to reach the highest cognitive level on Bloom's taxonomy, namely evaluation. If, on the other hand, the emphasis is placed on the methods, students are expected to reach the application level. In the latter case, higher Bloom levels would involve analysis of the methods (analysis level), the design of new usability methods (synthesis level), and evaluation of the new methods (evaluation level). Clearly, the latter learning objectives involve methodology proper, and as such do not belong in an undergraduate CS curriculum. Regardless of this, the ability to make these distinctions demonstrates the potential of Bloom's taxonomy as a tool for specifying and refining learning objectives.

HC4 focuses on students being able to develop software from a user-centered perspective. This requires prototyping and iterative refinement using feedback from user testing. As with HC3, an undergraduate course will typically be concerned with the analysis, synthesis, and evaluation of specific UI designs.

HC5 focuses on students being able to design usable user interfaces. This is by far the most important topic missing in traditional CS curricula. The incorporation of this knowledge unit is the main motivation for this work.

Although these three categories are listed only as electives in CC′01, they seemed essential in the authors' attempts to incorporate HCI in the undergraduate CS curriculum. On average, HC3 occupies approximately 24% of the course, at the application level or higher, HC4 occupies 23% of the course, at the analysis level or higher, and HC5 occupies 17% of the course, at the synthesis level or higher. These numbers are, of course, approximate, given the overlap of these knowledge units. Still, their total (64%) suggests that user-centered development topics, such as task analysis, GUI design, and usability evaluation, are essential components of an HCI course.

Due to the emphasis on project work, the learning objectives provided are targeting higher learning levels of Bloom's taxonomy. This is observed in both upper and lower level courses. There is a common understanding among the authors that HCI cannot be effectively taught without expecting students to be actively involved in project exercises. It should be emphasized that project work does not always mean coding; it does, however, always involve design and evaluation.

For HCI courses incorporating some programming teaching materials may be structured according to four software development phases: requirements analysis, design, implementation, and evaluation. Because evaluation should ideally follow every phase of user interface development, some evaluation skills should be taught early on. Homework expectations may range from very little programming to a semester-long project.

Projects may range from written reports, to simple GUI applications (e.g. calculators), to GUI wrappers for command line applications, to student-selected innovative multimedia applications. One might also assign a sizable software artifact where the students implement only the UI layer. In such a case having access to a clean API is most effective. Another possibility is to select a very small application and ask students to develop the complete artifact. If the students follow the proper methods

they will learn valuable HCI concepts by developing and evaluating a complete system. Evaluating such a system's usability would be easy due to its small task set.

In our view Bloom's taxonomy is a useful tool for refining HCI (and other) learning objectives. It helps the instructor in understanding how to teach the material (e.g. lectures and assignments), as well as how to assess student performance (e.g. test questions). If included in the course syllabus it also helps students better appreciate what is expected of them. As previously mentioned, the computer science department of two of the authors (Manaris and Stalvey) is now adopting Bloom's taxonomy in all major CS courses.

In closing, this paper has presented various implementations of the HCI curricular guidelines included in CC'01. These implementations employ Bloom's taxonomy to identify expected levels of student competence for each of the learning objectives. We hope this material may provide a useful starting point for instructors to effectively incorporate HCI into undergraduate CS curricula. This is essential in order to address the lack of HCI skills of CS graduates—the future developers of the tools that surround us.

Acknowledgements

This material is based upon work supported by the National Science Foundation under grant no. 0226080. Additional support was provided by the College of Charleston. Renée McCauley was the co-PI on this project. Sarah Douglas helped shape project objectives. Robert Aiken, Sally Fincher, and Blaise Liffick provided general advice and feedback. Sarah Douglas, Arthur Kirkpatrick, Laura Leventhal, and Craig Miller served as workshop instructors. Christopher Andrews, Douglas Dankel II, and Scott Grissom provided feedback on this paper. Christopher Andrews, Dennis Bouvier, Douglas Dankel II, Charles E. Frank, Robert Franks, Mary Granger, Scott Grissom, Thomas Horton, Hubert Johnson, Myungsook Klassen, Aparna Mahadev, David R. Naugler, Kris Powers, Nan C. Schaller, and Charles Welty contributed through discussions at the workshop and/or subsequent reflection meetings.

References

Ambler, S. W. (2005). *Inclusive modeling: User centered approaches for agile software development.* Retrieved September 13 2005, from www.agilemodeling.com/essays/inclusiveModels.htm

Ardis, M. A., & Dugas, C. A. (2004). Test-first teaching: extreme programming meets instructional design in software engineering courses. *Proceedings of 34th ASEE/IEEE Frontiers in Education Conference* (pp. F1C25–F1C30). Savannah, GA: IEEE.

Bloom, B. S. (Ed.). (1956). *Taxonomy of educational objectives: Handbook 1: Cognitive domain.* New York: Longmans, Green and Co.

Constantine, L. L., & Lockwood, L. A. D. (1999). *Software for use.* New York: ACM Press.

Culwin, F. (1998). *A Java GUI programmer's primer.* Upper Saddle River, NJ: Prentice-Hall.

Dix, A., Finlay, J., Abowd, G., & Beale, R. (1998). *Human computer interaction* (2nd ed.). London: Prentice Hall.

Engel, G., & Roberts, E. (Eds.). (2001). *ACM-IEEE Computing Curricula 2001*. Retrieved December 19, 2006, from www.sigcse.org/cc2001/HC.html

Grissom, S. (2006). *CS368 Usability design* [course materials]. Retrieved December 19, 2006, from http://www.csis.gvsu.edu/~grissom/368/

Hayes, B. (2004). Qwerks of history. *American Scientist*, *92*(1), 12–16.

Hewett, T. T. (Ed.). (1992). *ACM SIGCHI curricula for human-computer interaction*. New York: ACM Press. Retrieved December 19, 2006, from http://sigchi.org/cdg/

Horrocks, I. (1999). *Constructing the user interface with state charts.* Harlow, UK: Addison Wesley.

Horton, T. (2006). *CS305: Usability engineering* [course materials]. Retrieved December 19, 2006, from http://www.cs.virginia.edu/~horton/cs305/

Horton, W. (1995). Top ten blunders by visual designers. *Computer Graphics*, *29*(4), 20–24.

Hix, D., & Hartson, R. (1993). *Developing user interfaces.* New York: John Wiley & Sons.

Lengel, J. G. (2004). *The web wizard's guide to Dreamweaver.* Boston, MA: Pearson Addison Wesley.

Lethbridge, T. C. (2000). What knowledge is important to a software professional? *IEEE Computer*, *33*(5), 44–50.

Lethbridge, T., & Langaniere, R. (2005). *Object-oriented software engineering* (2nd ed.). Boston, MA: McGraw Hill.

Leventhal, L., & Barnes, J. (2003). Two for one: Squeezing human–computer interaction and software engineering into a core computer science course. *Computer Science Education*, *13*(3), 177–190.

Lewis, C., & Rieman, J. (1994). *Task-centered user interface design.* Retrieved December 19, 2006, from http://hcibib.org/tcuid/

Lister, R., & Leaney. J. (2003). Introductory programming, criterion-referencing, and Bloom. *Proceedings of the Thirty-Fourth SIGCSE Technical Symposium on Computer Science Education*, 143–147. New York: ACM Press.

Manaris, B., & McCauley, R. (2004). Incorporating HCI into the undergraduate CS curriculum: Bloom's taxonomy meets the CC′01 curricular guidelines. *Proceedings of 34th ASEE/IEEE Frontiers in Education Conference* (pp. T2H10–T2H15). Savannah, GA: IEEE.

McCauley, R., & Manaris, B. (2002). *Survey of departments offering CAC-accredited programs.* Retrieved December 19, 2006, from www.cs.cofc.edu/~mccauley/survey/

McCracken, D. D., & Wolfe R. J. (2004). *User-centered website development: A human–computer interaction approach.* Upper Saddle River, NJ: Pearson Education.

Miller, C. S. (2003). Relating theory to actual results in computer science and human-computer interaction. *Computer Science Education*, *13*(3), 227–240.

Miller, C. (2006). *HCI 360 evaluation for human–computer interaction* [course materials]. Retrieved December 19, 2006, from http://facweb.cs.depaul.edu/cmiller/hci360/syllabus.html

Myers, B. A. (1994). Challenges of HCI design and implementation. *Interactions*, *1*(1), 73–83.

Myers, B. A. (1998). A brief history of human-computer interaction technology. *Interactions*, *5*(2), 45–54.

Myers, B. A., & Rosson, M. B. (1992). Survey on user interface programming. *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI '92)*, 195–202. New York: ACM Press.

Neumann, P. (1995). *Computer-related risks.* New York: ACM Press.

Nielsen, J. (2005). *Current issues in web usability.* Retrieved December 19, 2006, from http://www.useit.com/alertbox/

Norman, D. A. (1998). *The design of everyday things.* New York: Basic Books.

Perlman, G. (1989). *User interface development,* SEI Curriculum Module SEI-CM-17-1.1. Retrieved December 19, 2006, from http://hcibib.org/sei-module.txt

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction design.* New York: John Wiley.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., & Cary, T. (1994). *Human–computer interaction.* Harlow, UK: Addison Wesley Longman.

Pressman, R. (2001). *Software engineering: A practitioner's approach* (5th ed.). Boston, MA: McGraw Hill.

Reimer, Y. J., & Douglas, S. (2003). Teaching HCI design with the studio approach. *Computer Science Education*, *13*(3), 191–205.

Rosson, M. B., & Carroll, J. M. (2002). *Usability engineering: Scenario-based development of human–computer interaction.* San Francisco: Morgan Kaufman.

Sanders, D., (2005). Incorporating human-computer interaction into an undergraduate curriculum. *Journal of Computing Sciences in Colleges*, *20*(4), 92–97.

Schafer, J. B. (2005). The integration of the software studio approach into the undergraduate computer science curriculum. *Proceedings of the 38th Annual MICS Symposium (MICS2005)*. Retrieved November 5, 2005, from www.cs.uni.edu/~schafer/publications/MICS_2005.pdf

Scott, T. (2003). Bloom's taxonomy applied to testing in computer science classes. *Proceedings of the Twelfth Annual CCSC Rocky Mountain Conference*, 267–274. New York: ACM Press.

Spolsky, J. (2001). *User interface design for programmers.* New York: Apress–Springer Verlag.

Stone, D., Jarrett, C., Woodroffe, M., & Minocha, S. (2005). *User interface design and evaluation.* San Francisco, CA: Morgan Kaufmann.

Strong, G. (Ed.). (1994). *New directions in HCI education, research, and practice,* NSF/ARPA Report. Retrieved December 19, 2006, from www.sei.cmu.edu/community/hci/directions/TitlePage.html

Sun Microsystems. (2005). *Java software human interface.* Retrieved September 13, 2005 from http://java.sun.com/developer/techDocs/hi/index.html

van der Veer, G., & van Vliet, H. (2003). A plea for a poor man's HCI component in software engineering and computer science curricula; after all: The human-computer interface is the system. *Computer Science Education*, *13*(3), 207–225.

Welty, C. (2005). *COS386 graphical user interface design* [course materials]. Retrieved December 19, 2006, from http://www.cs.usm.maine.edu/~welty/cos368/368fall2005/

World Wide Web Consortium. (2005). *Web accessibility initiative.* Retrieved November 5, 2005, from www.w3.org/WAI