

Undergraduate Software Engineering Curriculum Enhancement via Human-Computer Interaction

Stephanie Ludi

*Software Engineering Department, Rochester Institute of Technology
sludi@mail.rit.edu*

Abstract

More needs to be done to train students to deliver usable software. The current Software Engineering curriculum includes Human-Computer Interaction (HCI) topics in terms of a lecture. This paper presents how an undergraduate Software Engineering curriculum can be enhanced with HCI principles and techniques. The intent is to produce software engineers who value usable software and who can produce usable software.

Creating an atmosphere where HCI is not only presented, but expected in the student projects will provide ample opportunity for students to apply these newly acquired skills. The continued application of these skills will improve software.

1. Introduction

The usability of software, or rather the lack thereof, is at best an office nuisance or at worst the root cause of a tragic accident [2]. Some users groan and trudge through tasks while others become visibly frustrated when goals seem unattainable. The conduit to the system is the user interface, which is often a graphical user interface nowadays. Poorly designed user interfaces persist in a variety of domains.

Prospective software engineers hone their craft through coursework, including group projects. All too often the user interface design the result of one person's effort rather than an effort by the team. The success of the user interface is attributed to the efforts of one person's skill and vision. Such effort is reminiscent of the Capability Maturity Model's Initial level (CMM Level 1). CMM level 1 behavior is not acceptable in software development, and it should not be acceptable to the process of ensuring that users can interact with systems effectively and efficiently.

Many software engineers are simply not trained in HCI concepts and how to merge them with their technical training. Some students receive a lecture on user interface

design principles, while others take a Human Factors course from the Industrial Engineering department. As Software Engineering programs are increasing in number, the time has come for these programs to take a serious look at merging aspects of Human-Computer Interaction into the curriculum.

2. General Curricular Revision

Revision is needed in the undergraduate Software Engineering curriculum. Two different perspectives are needed when shaping the curriculum.

2.1. The HCI Course Within the Department

In order for the Software Engineering to exert the most control over the content and objectives of the course, the offering of the HCI course within the Software Engineering department would be ideal. The ACM SIGCHI Curricula for Human-Computer Interaction Curricula can be used as a starting point [1]. Assuming that one course will be required, introductory HCI topics need to be put into the context of software development.

For a course being proposed at RIT, the course description is:

The course stresses the importance of good interfaces and the how to apply the theories of HCI to user interface design. The course surveys the techniques available in the discipline, demonstrates where and when they are applicable and proceeds to demonstrate via a combination of scientific theory understanding and engineering modeling, to the solution of design problems. Other topics include: interface quality and methods of evaluation; interface design examples; dimensions of interface variability; dialogue tools and techniques; user-centered design and task analysis; prototyping and the iterative design cycle; and prototyping tools.

The proposed course will initially be offered as a seminar. After such a course becomes a permanent part of the curriculum, it will be required for all Software

Engineering students. A sample course could consist of the topics in Table 1.

Table 1. Course topics for an HCI course in a software engineering department

Course Topics	
1)	Introduction to the course a) The specific disciplines that comprise HCI, and how each discipline contributes to HCI b) Examples illustrating the importance of user interface design c) The relationship of the discipline of user interface design to the science of human-computer interaction
2)	Interface quality and evaluation a) Measures of user interface quality b) Methods for observation and evaluation
3)	Dimensions of interface variability a) Languages, communication and interaction b) Dialogue genre; the role of metaphor c) Dialogue techniques (including menus, icons, etc.) d) User support and assistance, documentation
4)	Social organization and work a) Small group dynamics b) Organizational information flow c) Models of work, workflow and cooperative activity Impact on design
5)	Methodology a) Methods for capturing, analyzing and applying data at the organizational and social level of human behavior b) Problems of validity c) Questionnaire design d) Conducting surveys e) Observation
6)	User-centered design and task analysis a) Relationship to software engineering design models; user-centered design, participatory design b) Socio-technical issues c) Task analysis d) Prototyping and the iterative design cycle; the evolution of designs e) The role of principles and guidelines f) Examples of designs
7)	The Human Information Processor a) Description of human architecture and performance of critical subunits (e.g., memory, perception, motor skills, etc.) b) Models of human activity (e.g., GOMS models, Keystroke Level model, etc.) c) Formal specification d) Applications of model human information processor to example problems

8)	User interface implementation topics a) Prototyping tools and environments b) Ergonomic issues i) Arrangement of displays and controls ii) Design for disabilities iii) Fatigue and health issues c) The role of graphic and industrial design
9)	Evaluation revisited; learning from HCI research; the role of models a) A deeper look at evaluation b) Learning from HCI research; applying science to interface design c) Conducting and analyzing usability studies
10)	Human-machine fit and adaptation a) Nature and theory of adaptive systems b) Theories of system adoption and methodology used to ascertain and motivate adoption c) User adaptation: ease of learning, training methods d) System adaptation to user types e) Relationship to system design
11.	Beyond the Graphical User Interface

While smaller assignments are provided, a large project will be a significant part of the course. Students will apply concepts and techniques for a course-long project. In the course project, each student team will design and implement a prototype system. The main facets of the project consist of the following:

- First, the students will elicit requirements through the use of questionnaires, interviews and unobtrusive observation (likely not all of these though).
- Second, the students will analyze these requirements and create a paper design, followed by modeling and evaluation of the design with the human information processing model.
- Third, each team will create a working prototype and conduct user testing of the design.
- Fourth the feedback will be used to redesign the prototype, based on the information gathered during the evaluation.
- Lastly, each team will present their prototype to the class.

Such context will allow students to identify with the potential applications in software development. While the course topics appear to be general the discussion, examples, and assignments are presented so as to be more relevant to Software Engineering students. By contrast, the proposed course seeks to improve upon the Human Factors course taken from the Industrial Engineering department. In this course (which several Engineering departments require), HCI is only covered for one week.

While the proposed course is not presented as the ultimate example, it can serve as a starting point.

Another advantage of having an HCI course is to explore special topics with HCI that may not be able to be explored within other courses due to time constraints. Such topics include accessibility and global design issues. Many students (and professional designers) think that the users of the delivered systems are just like themselves. The diversity with the national and international community shows that this is not the case.

2.2. Integrating HCI into Software Engineering Courses

For some, the development of an HCI is not possible or likely in the immediate future. An alternative is weaving HCI topics into existing Software Engineering courses. This arrangement is also a good complement to the presence of an HCI course.

Several courses have the potential to introduce HCI concepts and/or techniques that are companions to Software Engineering concepts/techniques. The depth of the HCI material introduced should be appropriate to the level of the course.

The introductory (generally lower-division) courses provide students with a foundation to build upon during their academic career. Potential pairings of HCI topics and Software Engineering courses are in Table 2.

Table 2. Matching HCI topics with existing lower-division courses

Software Engineering Course	Sample HCI Topic
Freshman Seminar	The relationship of the discipline of user interface design and usability to the science of human-computer interaction
Introduction to Software Engineering	The specific disciplines that comprise HCI, and how each discipline contributes to HCI The psychology of using software User Interface design heuristics. Usability and its role in nonfunctional requirements.
Engineering of Software Subsystems	Relationship to software engineering design models; user-centered design, participatory design Prototyping and the iterative design cycle; the evolution of designs The role of principles and guidelines

The upper-division courses provide students the opportunity to study a phase of software development in more depth or to learn about special topics.

Table 3. Upper-division pairings between HCI topics and courses

Software Engineering Course	Sample HCI Topic
Formal Methods	Modeling human activity
Requirements & Specification	Task Analysis Participatory design Designing and analyzing data from questionnaires
Software Architecture & Design	Design of Graphical User Interfaces (layout, structure, elements) Alternative interfaces
Metrics	Usability metrics used during evaluation
Verification and Validation	User interface verification and validation techniques

3. Expectation Within the Program

While the HCI concepts and techniques are needed in the curriculum, more is still needed. The critical element is expectation.

Unfortunately many students limit the use of the techniques and concepts to the course that they are acquired. After the course, students often revert back to what technique or process has worked for them in past projects, no matter how unstructured or haphazard the technique or process is. The follow-through to HCI concepts and techniques is no exception.

In order to truly have an impact on usability, academic programs must ask students to continuously practice what is learned and to incorporate the concepts and use the acquired techniques in subsequent projects. There must be the expectation that students will apply HCI best practices along with Software Engineering best practices. The expectation builds as the students progress through their course of study.

Such expectation can be accomplished in the grading of student projects. While the main concept being conveyed by the assignment will continue to be the main focus of grading, a portion of the project grade should be allocated to those concepts that the department also deems important at the stage of the student's academic career.

For example, a student is assigned a project in a design course. The majority of the grade is for the design concepts being applied. In addition, 10% of the overall grade can include the correct application of user interface design. This is akin to the expectation that students organize and cite references in their papers.

An alternative is to not include the user interface design tasks in the grade at all. Instead the omission of

the tasks (and any others deemed important) is cause for a penalty in the overall grade.

In any case, as the students build upon their knowledge base and skill set, they should be expected to apply these in subsequent projects in other courses.

4. The Role of Faculty

The successful grafting of HCI onto a Software Engineering program will not be possible without faculty support. Since many Software Engineering faculty are not trained in HCI, the gap in expertise can be addressed in different ways.

When new faculty needs to be hired, a candidate who has expertise in HCI and Software Engineering can be sought after. This would be the ideal, as such a faculty member can teach HCI-specific courses as well as Software Engineering courses in order to meet the general needs of the department. Such faculty can also play an instrumental role in enhancing the Software Engineering curriculum with HCI.

Existing faculty who do not have expertise in HCI but are interested in the field, can pursue professional development in general HCI or specialized areas within HCI that relate to current expertise such as Requirements, Design, or Process. Such faculty can also play an instrumental role in shaping curriculum, especially since he or she has more experience with the needs of the department than that of new faculty (who need some time to adjust).

Another option for a department to pursue when the desire and resources exist is to create a joint appointment. The joint-appointment can be made with another department that teaches HCI-related courses. For example a professor in Psychology or Computer Science with HCI expertise and technical experience can potentially fulfill most needs of a Software Engineering department when no other alternative exists. Such an arrangement would work best when HCI-specific courses are offered rather than asking a faculty member from another department to teach a design or testing course with integrated HCI objectives. The downside to such an arrangement is the fact that the faculty member is not full-time. The part-time appointment means that the faculty member cannot devote the same amount of time needed to shape curriculum as a full-time faculty member.

5. The Role of Industry

In order for academia to take usability seriously, industry must make it clear that they believe that usability is important.

Many departments have councils consisting of representatives of local software development companies. These councils provide input on curricula in terms of trends in industry and issues that should be addressed. The departments listen to industry since it is the customer for their graduates.

When industry discusses the importance of usability in software, departments will take notice. Departments will be motivated to act more quickly if industry vocalizes the importance of usability and the need for students to be trained in HCI.

6. Summary

In order to produce software engineers who place value in producing usable software, both academia and industry must do their part. Academia needs to prepare software engineers by training them to produce usable software at every phase of development. When the academic program imparts the expectation that students apply the concepts and techniques in all courses, the students will make HCI a part of their repertoire.

Industry can impart its need for software engineers who can deliver usable software to academia. Software Engineering departments listen to industry in order to make their graduates more appealing to industry. While industry does not directly drive curriculum, the influence exists when courses are revised from year-to-year.

The time for change is here. Usable software is attainable, and software engineers are capable of delivering it when adequately trained.

7. References

- [1] ACM SIGCHI Curricula for Human-Computer Interaction Curricula, available at: <http://www.acm.org/sigchi/cdg/cdg3.html>
- [2] N. G. Leveson, and C. S. Turner, "An investigation of the Therac-25 accidents", *Computer*, 26(7), July 1993, pp.18-41.