Paper Presented at the 17th Conference on Software Engineering Education and Training (CSEE&T 2004), Norfolk, Virginia (USA), March 1-3, 2004

Software Engineers and HCI Practitioners Learning to Work Together: A Preliminary Look at Expectations

Allen E. Milewski
Monmouth University
amilewsk@monmouth.edu

Abstract

This survey studied the expectations of Software Engineering graduate students who took an HCI course, regarding the interaction of engineers and HCI practitioners in their future workplace. The data are suggestive that students with HCI training, compared both with non-HCI students and with current Industry practices, expect to keep abreast of the HCI field more actively, expect design decisions and usability testing to be more collaborative and expect to see a greater degree of interaction between engineers and HCI practitioners in the future.

1. Introduction

The disciplines of Software Engineering (SE) and Human Computer Interaction (HCI) have each evolved over the past ten years to meet the needs of their customers and the responsibilities of their work assignments. In the course of evolving, each has seen the need to move toward the other. Software Engineering has developed practices for understanding the needs of users and other stakeholders in order to obtain reliable requirements and has developed evolutionary process models to iterate with users during the design phase [7]. Human-Computer Interaction Engineers, for their part, have begun to include analyses of technical platform capabilities and constraints early in their user requirements analysis designs, and now regularly develop software prototypes to test these technical capabilities in addition to their traditional usability testing [6, 8].

This evolved situation has many problems. The most significant one is that the two disciplines don't know enough about each other to realize that the have evolved similarly [1,2]. There is often a striking lack of communication between these two disciplines despite the fact that they often work side by side on a daily basis on software projects. They use different terminology for similar activities and artifacts and even have different views of how much interaction is taking place [3]. In most cases, there is a loss of efficiency since these two fields are performing highly-overlapping functions, at twice the cost, when in some situations, a single person could do it. And, worst of all, there is the increased chance of confusing customers and users alike when two organizations schedule interviews and two organizations handover overlapping requirements documents for validation and sign-off.

Despite arguable improvements in SE-HCI interactions, problems of communication and efficiency certainly persist, so that it is useful to consider how they can be reduced further. Some measures have been proposed, but it can be argued [5] that they are unlikely to have major effects on current SE/HCI interactions, at least when taken in isolation. For example, it is unlikely that combining Software Engineers and Human Computer Interaction into a single discipline or a single role in the lifecycle of a project will solve the problem. First, while user interface issues generally account for nearly half of the lines of code, there are still many application functions that have nothing directly to do with the user. Second, experience has shown the utility of having a "user advocate", semi-separated from the schedule and budget demands of the rest of the project. Finally, there is simply too much to do to combine positions and the interfaces to others in the project (systems engineers, developers, designers, marketers) are typically too numerous for one person to handle.

It is also unlikely that creating a common, integrated process model will, by itself make the current situation more manageable. In practice, process models are chosen and adapted to fit the specifics of the environment and, often, the needs of the specific project. As such, process models are more descriptive of what happens than prescriptive of what must happen. The history of process models is one of trial and error, where processes that are developed in an ad hoc fashion in real-life projects are retained and formalized if they turn out to be effective.

Finally, it is similarly unlikely that initiating a formal effort to create a common terminology for the two disciplines will solve anything. Like process models, the development of a set of useful working terms has a strong grass-roots element. Of course, many of our professions' terms have been introduced by leading thinkers in the fields, but only those that are meaningful in practice remain and become popular. Besides, what set of terms would be chosen to describe overlapping concepts? Existing terms are all laden with baggage for one discipline or the other. And, the strategy of creating brand new, neutral terms inevitably complicates the situation. Since the older terms are seldom dropped, the net effect is to magnify confusion by increasing the number of synonymous terms (or worse- not quite synonymous).

2. The role of education on attitudes

Milewski [5] has proposed that the most significant and lasting solution lies in the education and training programs for both Software Engineering and HCI. If the academic disciplines begin to acknowledge the overlap and specifically explore it, students will enter the workforce in a better position to collaborate with the entire interdisciplinary team required for a successful software project. From this interdisciplinary education will come the abilities and, most important, the attitudes required to continue the evolution of the two fields.

To begin gaining insights into the expectations and attitudes associated with interdisciplinary education and training, survey data were collected from Software Engineering graduate students at the end of their first HCI course experience. The survey questions were based on a survey reported on by Kazman, et al. [3] that was designed to reflect the current situation as Software Engineers and HCI practitioners work together on projects in Industry. In general, when given to workers in Industry, this survey has revealed a marked schism between Software Engineers and HCI practitioners. Despite working on significantly overlapping problems, they often differ in their perceptions of collaboration amount, they often feel they work separately, and they sometimes feel non-collaboration to be an acceptable situation.

The survey was distributed by email to 22 graduate Software Engineering students. Thirteen had completed their first HCI course three months earlier (HCI's). The remaining nine had no HCI courses (non-HCI's). Results are based on 13 surveys that were completed and returned (eight HCI's and five non-HCI's). While the survey questions are similar in content and format to those of Kazman, et al., [3] one significant difference between the two is that the one given to students asks questions about students' attitudes and expectations about future jobs, while the Kazman, et al. survey asks about actual situations in a current job. As a result of this difference, direct comparisons between students and practicing Software Engineers are difficult. Nonetheless, these preliminary data can give interesting insights about what engineering students with HCI-training expect from their workplace, in the context of both students without such training and of potentially different current industry practices.

2.1 Interdisciplinary knowledge

According to Kazman, et al. [3], practicing Software Engineers and HCI practitioners tend not to have detailed, knowledge of the other's discipline and tend not to keep up with new interdisciplinary information in formal ways. Sixty-nine percent of practicing Software Engineers reported keeping up with HCI advances only by talking informally with HCI practitioners, while 38% reported not keeping up at all. In our sample, Software Engineering students with HCI training expect that they will successfully keep abreast of the HCI field, and will use more formal methods to do so: Seven out of eight HCI respondents included formal methods such as courses, conferences and journals. Only one HCI respondent predicted that they would not keep up with HCI advances in the future. On the other hand, one HCI respondent projected receiving a formal HCI degree. In contrast, two out of five non-HCI respondents predicted that they wouldn't keep up with HCI advances at all. An additional two non-HCI respondents predicted they would keep up informally by talking with HCI practitioners. In sum, HCI-trained students' expectations about interdisciplinary knowledge surpassed the estimation of currently practicing Software Engineers. Students without HCI training were similar to practicing engineers in estimating that they may not keep up with HCI, or would do so only informally.

2.2 Working together and roles

According to Kazman et al., 68% of practicing Software Engineers and 91% of HCI practitioners felt that design decisions that affect the User Interface are currently made unilaterally by Software Engineers without HCI input. None of our current sample of students expected a workplace devoid of unilateral decisions made by Software Engineers- i.e., no one responded "never". Four non-HCI's expected such decisions "fairly often" and one "almost never". In contrast, HCI-trained students were more varied in their expectations: four responded "almost never" and two responded "fairly often". Consistent with Kazman's study, the HCI students predicted unilateral decisions would occur "very often" somewhat more often than did non-HCI-students (two HCI's vs. zero non-HCI's).

Kazman et al, also asked practicing engineers about the responsibility for testing usability. 50% of practicing Software Engineers said that Quality Assurance experts are responsible for user interface testing, 36% said Software Engineers test usability, and 0% explicitly said that HCI practitioners are responsible for testing. Nearly all of our students listed Software Engineers and/or Quality Assurance experts as usability testers. But, seven out of eight HCI respondents also included HCI experts as expected testers. Only one out of five non-HCI respondents included HCI experts among the expected testers.

In addition to showing a difference in expectations between HCI's and non-HCI's, these data reflect an expectation that work critical to a successful user-interface will sometimes be made without HCI input. For both HCI and non-HCI students, these expectations appear to be based on perceived problems associated with integrating HCI into the development process. What is interesting is that there is marked difference between HCI and non-HCI students in the nature of drawbacks expected. HCI-trained students tend to list schedule impact (four of eight) and technical feasibility (five of eight) as the key problems with HCI. In contrast, all of the non-HCI students blame potential problems on shortcomings in HCI method- such as high variability and arbitrary designs. None of the HCI-trained students blame methodological shortcomings.

Finally, students were asked several questions about their expectations of roles in the workplace- specifically as to whether HCI and Software engineering would be separate or combined jobs. These questions did not appear in Kazman et al.. Five out of eight HCI's and three out of five non-HCI's students expected to find dedicated HCI professionals included in their future organizations rather than HCI functions being assumed entirely by Software Engineers. Interestingly, seven out of eight of the HCI-trained students expected themselves, as software engineers, to be directly involved in HCI decisions. Only one out of five of the non-HCI's expected themselves to be involved. These two findings, taken together, suggest that students in general expect separate job positions for HCI and engineering, but HCI-trained students expect a high degree of collaboration on the design and evaluation of usable software systems.

2.3 Software process interaction

The Kazman et al. data reveal a general dearth of contact between Engineers and HCI practitioners, especially in the view of Software Engineers. Only 20% of Software Engineers report that contact occurs "very frequently", 30% "occasionally" and 50% "rarely" or "never". In sharp contrast, all of our HCI students expected contact to occur more frequently than "never" or "rarely". Five out of eight students expected "very frequent" contact while another three HCI respondents expected contact to occur "occasionally". Non-HCI's expected much less contact: Two of five non-HCI's expected contact "rarely" or "never", three expected occasional contact. None of the non-HCI students expected "very frequent" contact.

In addition to contact being infrequent, Kazman et al. report that it tends to occur late in the development process rather than early when it has greatest value. For example, 33% of engineers claim contact occurs after development is over, 30% say contact occurs during development and only 5% say contact occurs during requirements or specification. Again, 24% say there is never contact. In addition, 30% of engineers indicated that HCI methods were employed during development, while 20% indicated their use during requirements. Such a difference was not found in expectations between HCI and non-HCI students. The majority of both groups expected contact during requirements and specification, (seven out of eight HCI's and five of five non-HCI's) and also during post-release work (three of eight HCI's and three of five non-HCI's). Similar results were found when asked what phase of development HCI methods would be used.

These data suggest that HCI-trained students expect there to be strong contact between engineers and HCI practitioners and that there be regular usage of HCI methods. In general, while HCI's and non-HCI's expected markedly different amounts of contact, they seemed not to differ appreciably in expectations about when that contact would occur during a project.

3. Summary

These preliminary survey data on students' expectations and attitudes about HCI and SE interaction must be viewed with considerable caution. Students' responses reflected their expectations about the future while Kazman et al's data reflected practicing engineers' view of current practices. It is possible that even the practicing engineers, if asked about their expectations of how software process should work, would respond as did the students. Therefore, differences between students in the present study and the engineers in Kazman et al, may not reflect the causal effects of HCI education. Moreover, because the sample size of completed questionnaires was extremely small, differences observed between HCI's and non-HCI's are suggestive only. We are continuing to collect data from a larger group of students which will both bolster robustness and permit exploration of any trends that may occur in expectations across time.

Nonetheless, it is interesting to view the expectations of Software Engineering students as they finish training and begin to enter working organizations. The expectations of HCI-trained students contrast with current practices and with non-HCI's in striking ways with respect to the relative roles and interaction between HCI practitioners and engineers. First, students with HCI education expect to know more about and to keep more current with HCI findings compared with non-HCI students and compared with what is common in current practice. Second, HCI-trained students expect engineering and HCI to continue as separate job roles, but expect software design decisions and usability testing to be more collaborative than is typical currently. Finally, engineering students without HCI-training are more similar to practicing engineers in their expectations about interaction frequency. In contrast, compared with current practice, HCI-trained students expect to see a greater degree of interaction between Software Engineers and HCI practitioners. It is possible that these expectations about increased collaboration may result in increased collaboration and that the natural merging of Software Engineering and HCI will continue to evolve.

4. References

[1] John, B. E. & Bass, L. (2003) Avoiding "We can't change THAT!" Software Architecture and Usability. Tutorial materials presented at CHI 2003 (Ft. Lauderdale, FL, April 5-10, 2003). http://www-2.cs.cmu.edu/~bej/usa/publications /CHI2003TutorialPacket.pdf
[2] Johns, B.E., Bass, L. and Adams, R.J. Communication across the HCI/SE divide: ISO 13407 and the Rational Unified Process. In J. Jacko and C. Stephanidis (Eds.) Human-Computer Interaction: Theory and Practice, Lawrence Erlbaum, Mahway, NJ, 2003.
[3] Kazman, R., Gunaratne, J. and Jerome, B.. Why Can't Software Engineers and HCI Practitioners Work Together? In J. Jacko and C. Stephanidis (Eds.) Human-Computer Interaction: Theory and Practice, Lawrence Erlbaum, Mahway, NJ, 2003.
[4] Mayhew, D., The Usability Engineering Lifecycle, Morgan Kaufmann, San Francisco, 1999.
[5] Milewski, A.E. Software engineering Overlaps with Human-Computer Interaction: A Natural Evolution. Position Paper for the Workshop: Bridging the Gaps Between Software engineering and Human-Computer Interaction, International Conference on Software engineering, Portland, 2003
[6] Preece, J., Rogers, Y. and Sharp, H.., Interaction Design, Wiley, New York, 2002.
[7] Pressman, R.S.., Software engineering: A practitioner's approach., Fifth Edition, McGraw Hill, New York, 2001.
[8] Rosson, M.B.. and Carroll, J.M., Usability Engineering: Scenario-Based Development of Human-Computer Interaction., Academic Press, London, 2002.