Workshop position paper accepted at *ICSE'03 - International Conference on Software Engineering, May 3-11, 2003.*
Workshop on Bridging the Gaps Between Software Engineering and Human-Computer Interaction

Software Engineering Overlaps with Human-Computer Interaction: A Natural Evolution

Allen E. Milewski
Monmouth University
amilewski@monmouth.edu

Abstract

It is argued that overlap between the Software Engineering and Human-Computer Interaction disciplines is part of a natural evolution that has been developing throughout the history of both fields. It is further proposed that education and training is the most effective and long-lasting solution to some of the problems of communication and efficiency that have developed. Finally, it is argued that the education curricula are already evolving to reduce these problems.

1. Introduction

The disciplines of Software Engineering and Human Computer Interaction 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 evolve 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. Human-Computer Interaction Engineers, for their part, have begun to include analyses of technical platform capabilities and constraints early in their designs, and now regularly develop software prototypes for user evaluation [1, 4].

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. 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. In most cases, there is a loss of efficiency since two engineers are performing highly-overlapping functions, at twice the cost, when in some situations, one 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.

But, in spite of these problems, it is argued, the current situation is much improved over the historic relationship, and is a natural evolution of the two fields that can be, and already is, being facilitated by fairly simple changes in the training process.

It is important to remember that the introduction of users into the computing environment is not a recent development. Historically, the relationship between software "builders" and Human-Computer Interaction Professionals was almost always entirely antagonistic. There are many stories where HCI professionals needed to plead with a developer to try to get even a small usability change incorporated. The response was often, "the software can't do that", or "the impact of that change is just too large". Conversely, there are also many stories where HCI engineers attempted power-plays to force developers to make seemingly arbitrary changes that were needlessly difficult and complex.

These organizational atrocities are relatively rare in the current, evolved situation. It is more difficult now to tell an HCI engineer what the software can or cannot do because s/he is more knowledgeable in platform characteristics. And, it is more difficult to tell a Requirements Engineer that they don't know what the user needs because they, too, have likely utilized an assortment of techniques for finding out.

This would suggest that the overlap between fields, contrary to being a problem, has actually been a good thing and a natural evolution for disciplines with the mutual goal of producing effective systems. The increased breadth of knowledge in these disciplines has

increased empathy and made the negotiation processes far more realistic and efficient.

2. Some Prescribed Measures may have Small Effects

But, 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 is argued that they are unlikely to have major effects on current SE/HCI problems, at least when taken in isolation.

1) It will not solve the problem to combine Software Engineers and Human Computer Interaction into a single discipline or a single role in the lifecycle of a project. First, while user interface issues generally account for half (or even more) 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.

2) It is 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. – same with tools --may be the only resort for people already in industry

3) It is similarly unlikely that anything will be solved by initiating a formal effort to create a common terminology for the two disciplines. 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).

3. Evolution Through Education and Training

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 attitudes required to continue the evolution of the two fields.

Several aspects of training are especially important in this context:

- First, students in each discipline need to be exposed to and encouraged to explore the terminology of the other and decide for themselves the mapping between them and the most useful commonalities. In general, it is more useful to be aware of terminology differences than to be shoehorned into a single one.
- Second, students need to be steeped in the problem-solving approaches of the other discipline. The pace of software projects in Industry is very fast, and most of the communication required has to be implicit. Successful collaboration requires knowing how other team-members think and approach their tasks. Courses on topics such as the Psychology of Programming and Empirical Behavioral Methods are useful in this respect.
- Third, the issues of overlap between Software Engineering and Human-Computer Interaction need to be covered in detail from a Management standpoint. I recently cited for a friend the potential problem of separate SE and HCI requirements being delivered to the customer for validation. He, a longtime manager, asked: "what manager in his right mind would ever let that happen?". His reminder that inefficiencies come

from management shortcomings was a sound one. Management Training courses need to deal with these issues.

The theme here has been that the Software Engineering and Human-Computer Interaction fields have both been evolving in a positive way and that overlap is part of that continual evolution. Similarly, the educational curricula for these fields are evolving to solve current issues of communication and efficiency. There is an increasing number of programs that acknowledge the overlap and giving their students the opportunity to explore it in detail. Interdisciplinary faculty are becoming more common. Major Software Engineering textbooks have increased the sophistication of their HCI coverage [3]. Finally, nearly every survey textbook on Usability Engineering covers Process at least in part from a traditional Software Engineering standpoint [2, 4].

Many educators in both Software Engineering and Human Computer Interaction have examples of students discovering the similarities between the fields, and the differences as well. It is gratifying that the differences they see are not the artificial differences of the historic relationship, but the differences the really exist, and ought to exist for software projects to be efficient and successful.

10. References

[1] Mayhew, D., *The Usability Engineering Lifecycle*, Morgan Kaufmann, San Francisco, 1999.

[2] Preece, J., Rogers, Y. and Sharp, H.., *Interaction Design, Wiley, New York*, 2002.

[3] Pressman, R.S.., *Software Engineering: A practitioner's approach.*, Fifth Edition, McGraw Hill, New York, 2001.

[4] Rosson, M.B.. and Carroll, J.M., *Usability Engineering: Scenario-Based Development of Human-Computer Interaction.*, Academic Press, London, 2002.