JUS
JOURNAL OF USABILITY STUDIES

Vol. 6, Issue 4, August 2011, pp. 198-203

# The Unfulfilled Promise of Usability Engineering

**Dennis Wixon**
Senior User Researcher
Microsoft Corporation
One Microsoft Way
Redmond, WA, 98952
USA
denniswi@microsoft.com

Usability practitioners have always shared at least one common goal—create the best possible product or tool with the time and resources provided. How to most effectively achieve that goal has been and will continue to be an essential question for us. Today there are a number of promising frameworks and effective heuristics. There is also a body of experience that we can draw on. This was less true 30 years ago when some of us started. In those days, many of us came from a background in academic research. While that background provided the foundation for much of the work that followed, it was critical to adapt that foundation to the context of hardware and software development and interface design. In adapting that foundation some assumptions and practices could be built upon and others needed to be abandoned. In this essay, I examine the history of a framework that drew from our background in research, but that we adapted in crucial ways. I'll argue that some of the promise of that framework was "lost in translation" as the field matured. Finally, I contend that some of those lost elements of that framework could serve us well in our work today. That framework is usability engineering.

Now I suspect that some of you think that usability engineering is an outmoded approach. We have moved on to user experience. However, I see user experience and usability engineering as complementary frameworks. I believe the elements of usability engineering that we have lost could be helpful in achieving great user experiences.

A goal of this editorial is to provoke reflection. I'd welcome examples that enrich, qualify, or even contradict some of the assertions I make here. In provoking discussion it's important to be both clear and direct. As a result, I will ignore nuances and overstate some truths. So be it. Finally, if I offend anyone by minimizing your contribution, I apologize. It was not intentional. Please take the opportunity to reply and set the record straight.

## History and Assumptions

At Digital Equipment Corporation (DEC) in the early 1980s, the framework of usability engineering grew out of a need to influence product teams. We were dissatisfied with the tools, techniques, and frameworks of thinking that were at our disposal at that time. We came into industry with the framework and mindset of scientists. We believed that we could collect data by running experiments and use the results to influence product design. We had some success. For example, we were able to convince the engineering teams to put an overstrike feature in a new editor by showing that it could be more efficient in some contexts than traditional insert editing. It was, however, a difficult struggle. Despite rigorous methods (i.e., controlled experiments and statistically significant results), engineers were often resistant. They complained that "you didn't test enough people"; they brushed aside statistical tests with "you can prove anything with statistics." Another approach made use of standards, guidelines, or well accepted human factors principles to review and improve products. Despite the fact well-established principles existed (e.g., Smith & Mosier, 1986), they were dismissed as based on opinion. We concluded the root of the problem was cultural.

Engineering is more than a discipline: It is a culture with a set of beliefs and practices both implicit and explicit. We could have set about changing the culture, but that would have been likely to fail and would not have played to our strengths (we were not, by nature, evangelists). Instead we stepped back and tried to understand the nature of the culture and considered how we could adapt our thinking, leverage our strengths, and exploit them. The first step in this process was to examine the definition of engineering.

While there were many definitions, this is the one that seemed most germane: Engineering is a process of applying limited resources to achieve a defined result. This definition provided us with a good starting point for defining usability engineering. First, resources (time, people, money, and intellectual horsepower) are always limited. In fact the limitation of time and resources was one of the biggest cultural changes that, as former academics, we faced moving into industry. Those limitations shaped our work in important ways, and we needed to embrace them (or at least accept them). That assumption changed how we thought and worked. The second and useful part of the definition is a "defined result." A team can't achieve anything without having a shared understanding of what they are trying to achieve. The question for the team is how will we define this result? The definition is a critical inflection point that the design and research team need to influence. There are important pitfalls that product development teams need to avoid in creating a product or tool definition. The first pitfall to avoid is a confusion of means and ends. This seems simple but all too commonly we fall into this trap. For example, in early specifications, we saw statements like "this product will be easy to use because it has a help system." Similarly, I heard senior managers say "it's really easy; all you need to do is point and click." Lest you think that these statements are relics of the past, consider a modern version: "This system will be easy because it uses touch" (or speech or gesture). The separation of "the result" from "the techniques" is an essential element of engineering. It differentiates engineering from craftsmanship in which using traditional methods is key. It also is critical to preserving the creativity of designers, engineers, and researchers. The goal is to successfully meet a customer need, but the means of doing that are open to innovative designs. It also grounds that creativity in the customer need—the goal or the defined result.

This principle of a defined result may seem difficult to apply in a situation where the team is applying an agile approach. However, there are elements of the process of agile development that lend themselves to such a framework. First and foremost, agile development stresses small sprints that produce meaningful results for the customer or the client. In essence the performance of each of these small incremental steps needs to be evaluated against the needs of the customer. In practice this means that research teams working in an agile development

context need to fit their specification, testing, reporting, and fixing within that framework. My suggestion would be that they use the Rapid Iterative Testing and Evaluation (RITE) method (Douglass & Hylton, 2010).

This defined result usually consists of a specification of what the product or tool will do. Logically the specification consists of two kinds of requirements: binary and continuous. Binary elements are those that are present or not. For example, the system will support proportionally spaced fonts. It either does or it does not. Continuous requirements are more like qualities and imply a measurement scale and a criterion. For example, the mean time between failures will be 100,000 hours, under conditions of continuous operation, with the temperature range of -10 to 50 degrees centigrade, and 15 to 85% relative humidity. Note that this kind quality guides the team in terms of both design decisions (the kind of metal or plastic to be used) and in terms of evaluation (the conditions under which we test prototypes).

The insight of usability engineering is that we can define human performance and evaluation in exactly the same way as these continuous engineering qualities. In the context of user experience, we can define success by both performance and evaluative metrics. Examples of performance metrics are time needed to complete a task, the probability of success, and the number of errors made. Examples of evaluative metrics are user perception of ease of use, confidence that the task was done successfully, and willingness to recommend the product to others. By setting those kinds of metrics, we steer the conversation away from a focus on features (binary) and a premature discussion of implementation methods to a consideration of what the product should do for the user or customer. For the most part, the customer or user is relatively unconcerned about how a product achieves a given result, merely that it does so. In other words, we did not place sufficient emphasis on setting measurable product goals with the development team. Instead we chose to use "traditional metrics" (such as time to complete a task) and as a result missed the opportunity to set more sophisticated goals at the early stages of design and measured performance against them later. In short, all too often our measures were not grounded in relevant product goals.

Nowadays user experience professionals often work with marketing and sales teams. Marketing teams tend to think of a product in terms of value propositions. In effect these value propositions are reasons why people buy and/or use a product. Arguably these value propositions are best expressed in terms of what the user can do and how easily they can do it. Such an approach is well captured by usability engineering metrics and goals. Such metrics and goals extend the conversation beyond a list of features to a more holistic product concept that better reflects value. In addition as the market evolves toward a more evaluative framework, these more complete specifications become an essential part of the value proposition. For example, it's not enough that my cell phone make calls (functionality), accesses the Web (functionality), or takes pictures (functionality). I may expect my cell phone to make me feel like I am a sophisticated trendsetter. Such an evaluative construct can only be measured by user research.

## The Benefits

Some important principles follow from treating usability (or any other customer related attribute) as a measurable quality with pre-defined conditions of success:

- It gives the team a shared goal.
- The goal is objective and external; it's not a matter of the team's opinion or expertise.
- We can design and test for goals.
- When stated as a measurable quality, it leads the team to consider users and their work (or play) in a holistic way and not segment their experience into discrete features.

- It sets up the researcher (and the designer) for success. They can encourage the development of an early prototype and set up an objective test to determine degree of success.
- If the prototype fails to meet the test criteria, a careful review of the test data will likely offer clues with respect to the solution.

This final point deserves some elaboration. The quantitative result of a test provides the team with objective data that can inform next steps. The product passed the test, and we can stop iterating on the UI. Or, the product failed, and we need to fix it. Careful examination of the test results provides information on what to fix. This process was systematized and described by Good, Spine, Whiteside, and George (1986).

At DEC, once we adopted an engineering based approach and embraced an engineering culture, our relationship with development teams changed dramatically. Resistant teams were won over because the approach "made sense," i.e., it fit their culture. We felt more like members of the team and were much more successful in getting problems with the interface addressed.

It's important to note two things about this cultural change. First, it drew on our strengths: We were collecting empirical data in a controlled and careful way. While we were not testing hypotheses, we were doing controlled observation and taking measurements. Second, our goal changed from hypothesis testing to problem detection. That made our work much more relevant to product development. For the most part, product design consists of a large number of interdependent decisions. It's impossible to test each decision and all their interactions experimentally. Instead it works to take a holistic approach and test the entire product (which can be seen as a theory of user work) and detect and prioritize places where the product fails. The most important aspect of usability engineering that has been overlooked is the value of the "up front" part of the process. That up front work includes the following steps:

1. Working with teams to define user relevant goals. For example, people should be able to complete task X in Y time with no more than Z errors, experience with the product should leave the users feeling that the product is of high quality and sophisticated, and after using it 50% of those tested will recommend it.
2. Working with teams to operationalize those goals by co-defining a set of tasks, agreeing on a measuring scale for each goal, and setting target levels.
3. Agreeing on a plan that includes the specifics of the test—the when, where, and what. For example, we will evaluate a prototype interface on tasks A, B, and C.

That kind of process establishes both a framework and a schedule for testing. It helps avoid traditional cul-de-sacs such as "we're not ready to test yet" followed by "we're out of time and can't address any of the issues you bring up," could you "verify" that the interface works, and the relegating of research to low-level and often irrelevant questions like, what font should we use?

## Victims of Success

The assumptions and framework for usability engineering are outlined in several places (Nielsen, 1993; Whiteside, Bennett, & Holtzblatt, 1988; Wixon & Wilson, 1997). No doubt the most popular work is *Usability Engineering* by Nielsen (1993). The book contains several other insights that have been widely adopted and have contributed significantly to the development of our field. One of the most important of those was discount usability testing. The approach of conducting tests quickly and cheaply played an instrumental role in gaining wide adoption of usability testing techniques across industry. In the early 1980s there were a few dozen usability labs, now there are hundreds of labs and thousands of tests conducted every year. There is no doubt that such testing has vastly improved user experience and lead to the commercial

success of many products. This spectacular success is no doubt due to many factors, but the fact that the field had a clear, compelling, and low-cost process was a contributor.

At the same time, the lesson extracted from Nielsen's book has emphasized the discount aspect of testing. Fast and cheap are watch words that appeal to management and are essential elements of fitting into an engineering process. All resources are limited. At the same time, deemphasizing the idea of empirical definition of product qualities early in development has handicapped our field in many ways. First, it has deprived us of influence at the early definition of the product development process. It has also deprived product teams of thinking of users in a more holistic way and allowed teams to continue to fragment the user's experience into features supplemented by fanciful use cases derived from the functions themselves. In contrast, the qualities of experience are supposed to be defined by a realistic task that is meaningful to users, described in terms that would make sense to them. For example, you should be able to create this figure in 15 minutes, with relatively little frustration. The failure to adopt and evangelize such an approach with product teams all too often relegates the usability test to a last minute affair where it can only address small details because the team has already made the major decisions and is out of time. I am arguing that one of the main reasons we are not asked to participate in early development is that we have chosen not to champion the setting of measurable goals. For many people outside the discipline, usability work was defined by testing in a room with one-way glass that produced incremental improvements in interfaces. In a sense we have become victims of our own success. Our tests always find problems and often we and the development team are satisfied with that result.

## The Promise

There are a number of ways to incorporate measurement into the early stages of product work. Many teams now develop scenarios that allow teams to imagine how the product would work and anticipate how the users would interact with it. That approach is valuable for such visioning and for approaching the design in a more holistic way. Teams also develop use cases that anticipate how the product would be used. Those scenarios and use cases can be valuable. Their value, however, can be enhanced if they are rooted in some understanding of the customer and not simply extrapolated from technology. Here is where a fuller application of usability engineering methods adds value: The scenarios and use cases should be supplemented by a testable definition that measures the degree to which the product succeeds and sets a goal with respect to that definition. In that sense, usability engineering complements these popular approaches and, I believe, makes them more acceptable to product teams. There are examples, many of which have not been published in the literature. However, some have (Good et al., 1986). Also suffice it to say that our experience in the early days at DEC convincing product teams to consider test results changed dramatically after we adopted this approach. In the early days using our more "academic" approaches we were lucky to get product teams to adopt single recommendations (like putting overstrike in an editor). After adopting testable definitions we were able to induce product teams to address about 70-90% of the problems we identified and were able to improve user performance by approximately 20-30%.

In summary I think the full promise of usability engineering is yet to be fulfilled. We've developed the tools and methods. We built the infrastructure. We've gained credibility with product teams. We have a number of complementary approaches. Let's take the next step and work with teams to define usability metrics and goals and conduct tests that evaluate prototypes. In the 1980s we found that approach to be highly successful in improving designs and gaining credibility with engineering teams. I believe it's equally applicable now.

## Acknowledgements

Many people contributed to the development of usability engineering. John Whiteside articulated the approach. Michael Good and Tom Spine refined, applied, and developed both the thinking and the methodology. Intellectual antecedents included Jack Carroll (usability definition) and Tom Gilb (impact analysis). The approach also drew from the philosophy of management by objectives and some techniques of the quality literature (e.g., Pareto Diagrams).

## References

Douglass, R., & Hylton, K. (2010). Get it RITE. *User Experience*, 9, 12-13.

Good, M., Spine, T.M., Whiteside, J., & George, P. (1986, April). User-derived impact analysis as a tool for usability engineering. *Proceedings of the CHI'86 Human Factors in Computing Systems Conference*, Boston, ACM.

Nielsen, J. (1993). *Usability engineering*. New York: Academic Press.

Smith, S., & Mosier, J.N. (1986). Guidelines for designing user interface software, *ESD-TR-86-278* . MITRE Corporation.

Whiteside, J., Bennett, J., & Holtzblatt, K. (1988). Usability engineering: Our experience and evolution. In M. Helander (Ed.) *Handbook of Human Computer Interaction*. New York: North Holland.

Wixon, D., & Wilson, C. (1997). The usability engineering framework for product design and evaluation. In M. Helander, T. Landauer, P. Prabhu. *The Handbook Of Human Computer Interaction*. New York. Elsevier.

## About the Author

**Dennis Wixon**

Over the past twelve years Dennis has managed user research for a number of teams at Microsoft. Dennis previously worked at Digital Equipment Corporation. He has authored over 50 publications. He and Dr. Daniel Widgor have just published, *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Wixon holds a PhD in social psychology from Clark University.