Programmers, Software, Engineers and Designers

Design is a creative activity of making artifacts that are usable for some specific task. Software design in particular strives for creating products that enrich the interaction between humans and computer applications. While the software programmer/engineer is concerned with developing reliable, robust, and maintainable software, the software designer is concerned with creating products that fits within the user's overall activities, enhances productivity, and produces a satisfying experience [Winograd, Kapor]. To accomplish this goal the software designer needs to be able to apply knowledge of human goals, capabilities, and limitations with knowledge of computer capabilities and limitations [Preece].

While it is important for a software product to provide the necessary functionality to perform its intended use, it is also important that it presents its functionality in a manner that is consistent with the user's understanding. For example, the DOS operating system provided all the necessary functionality for managing file and folders on a personal computer from a command line. But the graphical user interfaces and the desktop metaphor of Apple's Macintosh and Microsoft's Windows have transformed the personal computer operating system into a product that could be easily used by the most non-technical users because its presentation fit their conceptual understanding of managing files and folders.

The study of Human-Computer Interaction (HCI) has collected an array of anecdotal evidence (e.g., Norman 88, Sachs) and significant empirical evidence (e.g., Landauer) that reveals the ability for computer technology to deliver on its promises, improving our productivity and enhancing our quality of life, rests squarely on how well the application fits our conceptual understanding of how things work. While computer processing power and data storage has increased dramatically, we have yet to attain a proportional increase in the productivity of work and quality life. In part, this is because we have moved into an era where computers are being used not simply for number crunching activities, but more to augment a variety of tasks that humans are better suited for than machines. These are the types of tasks that cannot easily be codified in any kind of quantitative theory, tasks that include the ability to read, understand, negotiate, and administer [Landauer].

Left to their own devices, computer programmers take a "systems-centered point of view", concerned about "how the software works and what parts of it do what" [Landauer, p. 217-218]. The predominant users of the number crunching era were more technically literate, willing to put up with a high threshold of indignation (the highest level of behavioral compromise a user is willing to make to accomplish their goals) [Saffo]. The users of the new era are less so. They do not want to know how the inner mechanisms of the machine work; rathethey want to know how the machine will work for them. This is exactly how we need to design such systems: the application should fit the users conception of the process, "the user-task model", while the inner mechanisms, "the engineer model", should be as transparent as possible [Gentner and Grudin]. Just as most automobile drivers (myself included) know very little of how a car actually works, and

yet find the task of driving natural, similarly we want to design computer applications that are natural to use with little worry or concern about how they are being accomplished.

In the area of HCI research, a number of approaches have evolved to meet this challenge. These include User-Centered Design [Landauer], Human-Centered Systems [Flanagan, Huang, Jones, and Kasif], Participatory Design [Muller and Kuhn], and Contextual Design [Beyer and Holtzblatt]. Though they differ in their techniques, these approaches have a general common vision of seeing "the interplay between human activity and technological systems as inextricably linked and equally important aspects of analysis, design, and evaluation" [Flanangan, Huang, Jones, and Kasif, p. 3]. The different techniques find ways to interject the designer in the user's world and the user in the designer's world in order to develop a shared conceptual model of the task and the context in which they are being done [Muller and Kuhn].

The difference between a software designer and software engineer has been compared analogously to the difference between a building architect who designs a structure and a contractor that builds it [Winograd]. While there are HCI degree-granting programs such as Stanford's Center for HCI study, and larger companies such as IBM which have design and usability labs, it is still currently the computer science programmer doing both design and development, like an architect that both designs and constructs the building. So it is very important that we educate computer science students in the techniques of software design that embrace the human activity as an integral component of the analysis, design, and evaluation.

References Cited

Beyer, H., and Holztblatt, K. (1997) Contextual Design: A customer-centered approach to systems designs, Morgan Kaufman Publishers, San Francisco, CA.

Flanagan, J., Huang, T., Jones, P., and Kasif, S., eds. (1997) NSF Workshop on Human-Centered Systesm: Information, Interactivity, and Intelligence, Final Report. (http://www.ifp.uiuc.edu/nsfhcs).

Gentner, D. and Grudin, J. (1996) "Design Models for Computer-Human Interfaces", IEEE Computer, Vol. 29, No. 6, pp. 28-35.

Landauer, T. K. (1995) The Trouble with Computers: Usefulness, Usability, and Productivity, The MIT Press, Cambridge, Mass.

Muller, J., and Kuhn, S. (1993) "Participatory Design", Communications of the ACM, Vol. 36, No. 6, pp. 24-28.

Norman, D. (1988) "The Psychopathology of Everyday Things", Chapter 1 in The Psychology of Everyday Things, Basic Books.

Preece, J. (1996) Human-Computer Interaction, Addison-Wesley, Menlo Park, CA.

Sachs, P. (1995) "Transforming Work: Collaboration, Learning, and Design", Communications of the ACM, Vol. 38, No. 9, pp. 36-44.

Saffo, P. (1996) "The Consumer Spectrum" Chapter 5 in Bringing Design to Software, T. Winograd ed., Addison-Wesley, Menlo Park, CA.

Winograd, T. (1996) "Introduction" in Bringing Design to Software, ACM Press, New York, NY, pp. xiii-xxxv.