Integrating Human Factors with Software Engineering Practices

William E. Hefley, Elizabeth A. Buie, Gene F. Lynch,
Michael J. Muller, Douglas G. Hoecker,
Jim Carter, J. Thomas Roth

29 July 1994
CMU-CS-94-175

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

To appear in the Proceedings of the 38th Annual Meeting of the
Human Factors and Ergonomics Society, Nashville, TN, October, 1994.

Also appears as Human-Computer Interaction Institute Technical Report
CMU-HCII-94-103

Abstract

Engineering processes and methodologies used in building tomorrow's systems must place a greater emphasis on designing usable systems that meet the needs of the systems' users and their tasks. This paper identifies the need for defining human factors and human-computer interaction (HCI) engineering activities that contribute to the design, development, and evaluation of usable and useful interactive systems, and presents a rationale for integrating these activities with software engineering and incorporating them into the system life cycle.

This work sponsored by the U.S. Department of Defense. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

Keywords: design, software engineering, human factors, software process improvement, user interface design, user interface software, human-computer interaction (HCI)

Integrating Human Factors With Software Engineering Practices

William E. Hefley
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
weh@sei.cmu.edu

Elizabeth A. Buie
Computer Sciences Corporation
Laurel, MD 20707 USA
ebuie@csc.com

Gene F. Lynch
Tektronix, Inc.
Beaverton, OR 97077 USA
gene.lynch@tek.com

Michael J. Muller
U S WEST Technologies
Boulder, CO 80303 USA
michael@uswest.com

Douglas G. Hoecker
Westinghouse Science & Technology Center
Pittsburgh, PA 15235 USA
hoecker@cognac.pgh.wec.com

Jim Carter
University of Saskatchewan
Saskatoon S7N 0W0 CANADA
carter@skdad.usask.ca

J. Thomas Roth
Ergonomics and Safety Technology, Inc.
Pittsburgh, PA 15213 USA
jtr001@delphi.com

1. Introduction

The design and development of human-computer interaction (HCI) has been evolving into a full engineering discipline for achieving system usability— developing systems that support their users in accomplishing their tasks with effectiveness, efficiency, and satisfaction. Advances have been occurring both in user interface engineering (Curtis & Hefley, 1994), focusing on the processes being used to develop artifacts, and in usability engineering (Whiteside, Bennett & Holtzblatt, 1988; Nielsen, 1993), focusing on the products being developed.

Large proportions of these systems are heavily software intensive. System engineering activities must therefore integrate HCI engineering with software engineering to achieve usability in software-intensive systems. This effort can take advantage of successful HCI engineering efforts, which have focused on human factors and HCI methods (Dayton et al., 1993).

Just as software engineering is becoming a discipline with a defined, managed process, HCI engineering is continuing to evolve to a discipline having its own defined interface development processes. These processes will be practiced by people from numerous fields employing a rich collection of analytical, design, development, and evaluation techniques to develop interactive systems that are effective, efficient, and satisfying. Successful HCI engineering efforts often focus on human factors and human-computer interaction methods that can improve the practice of software engineering.

How can HCI enhance current product development practices? As a step in integrating HCI engineering with software engineering, this paper addresses questions of how HCI engineering principles and practice can enhance current software engineering practice.

This paper describes the background and motivation for a seminar to be held as part of the HFES94 Annual Meeting. This working session has two main objectives:

- Identify engineering methods and techniques appropriate to the HCI engineering process model for interactive systems development
- Define the processes for using these techniques in the context of a system life cycle–a process architecture that elaborates on how to specify, design, build, test, and evaluate useful, usable, and satisfying interactive systems.

These efforts aim to propose HCI engineering processes for interactive, software-intensive systems and to extend their understanding of these techniques, methods, and processes to a broader community of researchers and practitioners. We hope that these ideas may mature, spread, and begin to change the software and system engineering practices.

2. State of the world

2.1 People

Who performs these types of tasks today? What skills do they have or need (Dayton, 1993)? Are they being brought in early enough as an integral part of the development process (Whitehurst, 1993; Rousseau, Candy & Edmonds, 1993)? Developing interactive systems requires the timely acquisition and application of new skills to comprehend, apply and improve concepts in development processes, methods, and tools.

2.2 Processes

An important deficiency in the current state of practice is that many HCI design methods are poorly defined. A common criticism of software engineering is also that its processes have not yet reached the level of discipline and proceduralization that are evident in other engineering disciplines. The processes used in developing large, complex systems are often ad hoc, and not often defined and articulated in a manner that encourages repetitive use and further refinement.

However, many of today's state-of-the-practice software engineering organizations are assessing the maturity of the processes they use and are putting into place various forms of continuous process improvement (also called Total Quality Management [TQM]) activities to plan and carry out improvements to their existing software development processes (Herbsleb & Zubrow, 1994). Such organizations are striving to make their software processes understood, repeatable, defined, measured and subject to continual improvement. They are addressing a vital need for process architectures that are usable by large numbers of practitioners to produce high quality software systems.

Unfortunately not many of these efforts include state of the practice in HCI design in the processes being improved. Nor do they use the evaluative methods for determining usability as metrics for the success of their processes. They are improving process (software and to a lesser extent product process), but it is a critically impaired or at least structurally limited process due to its lack of consideration for HCI design and usability issues. Developing high-quality interactive systems requires the appropriate integration of HCI engineering with software engineering during the entire system life cycle.

3. Designing Tomorrow's Interactive Systems

3.1 Desired Goal State

Not only must future systems support operability and learnability goals, they must also be developed with an eye to concerns, such as affordability. Do our engineering processes result in a system that people and organizations can afford to procure and operate? Can a usable system be produced within the schedule and cost constraints that face us as developers?

What goal should HCI engineering and software engineering adopt in this context? They must aim to apply a coordinated engineering process for effectively, efficiently, consistently, and humanely producing high-quality, defect-free products that fully satisfy its users' needs.

3.2 Software Process Improvement as a Model

Unfortunately, because of recurring problems and the immaturity of many organizations, the major process emphasis in these organizations is typically on planning, managing and controlling the progression of software development activities (Humphrey, Kitson & Kasse, 1989; Kitson & Masters, 1992). Recently, however, the software community have begun to pay widespread attention to ways of understanding and improving software processes (Humphrey, 1989; Paulk, 1993a).

The Software Engineering Institute's (SEI) Capability Maturity Model (CMM) (Paulk, 1993a; Paulk, 1993b) defines five maturity levels for software process and describes the processes that typically are in place in organizations at each process maturity level. The CMM provides specific guidance on the staging of activities in software process improvement. This structuring of specific key process focus areas within maturity levels helps organizations prioritize their improvement activities.

The notion that an organization is improving its software process maturity implies that the organization is moving towards defining, applying and improving rigorous development processes. Once a rigorous development process architecture is defined, its can be used to guide development, provide a common basis for communication among engineers and managers, and (perhaps most importantly) provide a basis for further process improvement (Curtis & Hefley, 1992). The software process improvement approach can provide a valuable model for organizations seeking to improve their HCI engineering process.

3.3 Integrating HCI Engineering Processes with Defined Software Engineering Processes

Until recently, the HCI community has placed primary emphasis on design. This emphasis isn't bad–in fact, it's to be expected if HCI practice is at an ad hoc process maturity level (CMM Level 1). Curtis and Hefley have argued (1992, 1994) that rigorous HCI processes, methods, and techniques do exist.

Why try to define an engineering process, if we're really only at a Level 1 or ad hoc stage of building interfaces? As discussed in greater detail below, there is a growing importance of HCI concerns in today's applications. This growing importance, coupled with the maturation of the discipline, present a timely opportunity to make improvements in the system life cycle process and its outcomes by focusing on human factors and HCI concerns.

Thus, if Level 3 is characterized by defined processes, we need to start making the HCI engineering processes repeatable (CMM Level 2) and defined (CMM Level 3). We also need to consider how to integrate these processes into a coherent engineering process. At the same time, we need to start thinking about, and developing plans for, integrating those HCI engineering processes with our defined software engineering processes.

For many HCI development efforts, necessary HCI engineering processes are often not integrated with software engineering processes (Curtis and Hefley, 1992). However, HCI development is beginning to adopt new approaches to system development such as user centered system design (Norman & Draper, 1986), participatory design (Schuler & Namioka, 1993; Muller & Kuhn, 1993; Greenbaum & Kyng, 1991), participatory ergonomics (Noro & Imada, 1991), Scandinavian design (Bødker, 1990; Floyd, 1989), and cognitive modeling (Olson & Olson, 1990; Bösser & Melchior, 1992; Tauber, 1990), which make the end user rather than the technology the focus of the design process. These approaches, together with an increasing awareness of the need for process architectures that integrate HCI and software development processes (Curtis & Hefley, 1994; Long, et. al, 1994; Lim, Long & Silcock, 1990; Browne, 1994; Hix & Hartson, 1993), signal the emergence of process architectures that will be useful in developing usable interactive systems.

4. Importance of HCI Concerns

The development of computer-based systems is changing. The changes, many of which reflect a growing recognition of the importance of HCI concerns, include

- the predominance of HCI-related effort in the life cycle
- the expanding functionality of user interfaces–moving towards intelligent user interfaces and integrated task environments
- the transition of HCI development from an arcane specialty into an established engineering discipline.

4.1 Predominance of HCI

Almost half of the software in systems being developed today and thirty-seven to fifty percent (depending on life-cycle phase) of the efforts throughout the life cycle (Myers & Rosson, 1992) are related to the system's user interface. A significant portion of the resources and efforts in software development are dedicated to that portion of the systems commonly referred to as the "user interface." In fact, the user interface has become a core system engineering issue separate from usability concerns (Curtis & Hefley, 1994).

Moreover, the key constraints on HCI development are evolving. An ever-widening population of potential users continues to make ever-increasing demands for usability. Formerly constrained almost exclusively by technology, HCI development is now driven mainly by usability concerns and increasingly by concerns regarding operability (including learnability).

4.2 Expanding HCI Functionality

The functionality and capabilities of user interfaces are also expanding–moving, for example, toward intelligent user interfaces (IUIs) (Hefley & Murray, 1994) and integrated task environments (ITEs) (Hefley & Romo, 1994). Concepts such as critics (Mastaglio, 1990) and guides (Tuck & Olsen, 1990), rich research topics a few short years ago, are now being incorporated in shrink-wrapped products.

This progression can be expected to continue. For example, addressing integrated task environments (ITE) the United States Department of Defense (DoD) Software Technology Strategy (Department of Defense, 1991) brings out the concept of intelligent adaptive user interfaces (IAUI). This strategy addresses needs for intelligent adaptive user interfaces (IAUI), safe user interfaces, user-tailorable interfaces, task model-based interfaces, adaptive user interfaces, and intelligent user interfaces.

In all categories of products customers demand more for less. They want more functionality for less money. They want more flexibility with less learning or execution effort.

4.3 An Engineering Discipline for HCI

During the 1990s, HCI development will complete the transition from an engineering specialty into an engineering discipline (Curtis & Hefley, 1994), and HCI professionals and “drafted” engineers will find themselves becoming a more important part of the development team. They will also find that this requires greater discipline in their work. This discipline is not just technical, it also involves taking greater responsibility for serious analytical activities that lead to a finished product.

Curtis and Hefley (1992, 1994) have argued that rigorous processes, methods and techniques do exist for HCI development and that they constitute an engineering discipline. Others (e.g., Morrison, 1993) have suggested that HCI cannot be an engineering discipline; however, we believe this argument to be based on an overly narrow definition of engineering. In recent years, other colleagues have also publicly taken a stance of moving towards an engineering discipline (Dowell & Long, 1989; Dumas & Redish, 1993).

The processes used by system designers and developers are aimed at producing a high-quality product—a product that enables the users to accomplish their tasks efficiently, effectively, and comfortably. In a disciplined fashion, designers and developers must address the intended:

- users and their characteristics, such as knowledge and skills
- users' jobs and tasks, including task objectives, performance needs, and interpersonal or group communication needs
- organizational & work environment
- quality of worklife and the quality of users' experience
- technologies to support task performance
- information needed by users and their tasks
- interrelationships between the environment, users, tasks, technologies, and information flows

HCI engineering, as a discipline, is uniquely able to contribute to addressing these issues. In one recent analysis, the interaction times of similar functions, based on detailed cognitive task analyses, were compared for two different proposed workstations. The cost differential for a 0.8 second performance difference in each transaction spread across all operators accounted for a potential savings of $2,400,000 per year (Gray, John, and Atwood, 1992). Some organizations have already begun to define HCI engineering as an important part of their system life cycle (Computer Sciences Corporation, 1990, 1994).

5. Envisioning A Future Discipline

While HCI engineering is continuing to evolve into a discipline having its own defined interface development processes, promising advances along this evolutionary path are not only found in an evolving discipline for designing usable, effective, and productive systems with concern for usability at multiple levels—at the individual ergonomic level, at the task and job level, at the interpersonal or group communication level, and at the organizational/societal level—but also in defining ways of analyzing, design, and assessing issues directly related to quality of worklife (QWL) (or the quality of life for non-workplace applications) and the users' quality of experience. Floyd (1987) foresaw this paradigm shift in understanding software in its contexts of human learning, work, and communication.

Key to successful system development are an increasing emphasis on defining appropriate human factors and HCI design processes and an integration of these processes with existing system and software development processes. It is easy to see the value of these improvements when a second less in user execution time yields millions of dollars in benefits (Gray, John, and Atwood, 1992); however, many efforts may not have this kind of dramatic outcome. To enjoy the resulting benefits, we will have to take action to develop comprehensive life cycle processes that are repeatable, defined, cost effective, measurable, and traceable. A process that can be evaluated and then improved.

This paper is a call to action and an invitation to all who are responsible for product development and see the need for integration of clearly defined processes that address a range of analysis, design and evaluation functions to assist in developing a vision of future practice that will encompass designing for usability (fit to person), utility (fit to the task and organization), and QWL (fit with respect to social considerations regarding people, groups, organizations, and society).

ACKNOWLEDGEMENTS

This work sponsored by the U.S. Department of Defense.

The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

REFERENCES

Bødker, S. (1990). *Through the interface—A human activity approach to user interface design*. Hillsdale, NJ: Erlbaum.

Bösser, T. and Melchior, E.-M. (1992). The SANE Toolkit for Cognitive Modelling and User-Centred Design. In Galer, M.; Harker, S.; & Ziegler, J. *Methods and Tools in User-Centred Design for Information Technology*, 93-126. Amsterdam, Netherlands; North-Holland.

Browne, D. (1994). *STUDIO: STructured User-interface Design for Interaction Optimisation*. New York: Prentice Hall.

Computer Sciences Corporation (1990). *Digital System Development Methodology (DSDM®)*. Version 3.0, 1990. Falls Church, VA: CSC. [DSDM is a registered trademark of Computer Sciences Corporation.]

Computer Sciences Corporation (1994). *CSC Catalyst* SM. Version 3, 1994. Falls Church, VA: CSC. [Catalyst is a service mark of Computer Sciences Corporation.]

Curtis, B. and Hefley B. (1992). Defining a Place for Interface Engineering. *IEEE Software*, 84-86 (March, 1992).

Curtis, B. and Hefley, B. (1994). A WIMP No More: The Maturing of User Interface Engineering. *ACM interactions* 1(1), 25-42 (January, 1994).

Dayton, T. et al. (1993). Skills Needed by User-Centered Design Practitioners in Real Software Development Environments: Report on the CHI '92 Workshop. *SIGCHI Bulletin 25*(3), 16-31.

Department of Defense. (1991). *Department of Defense Software Technology Strategy (Draft)*. Washington, D.C.: Director of Defense Research and Engineering (DDR&E).

Dowell, J. and Long, J. (1989). Towards a conception for an engineering discipline of human factors. *Ergonomics* 32(11), 1513-1535.

Dumas, J.S. and Redish, J.C. *(1993). A practical guide to usability testing*. Norwood, NJ: Ablex.

Floyd, C. (1987). Outline of a Paradigm Change in Software Engineering. In Bjerknes, G., Ehn, P., & Kyng, M. (eds.) *Computers and democracy: a Scandinavian challenge*, 191-210. Aldershot, Hants, England: Avebury.

Floyd, C. et al. (1989). Out of Scandinavia: Alternative Approaches to Software Design and System Development. *Human-Computer Interaction* 4(4), 253-350.

Gray, W.D., John, B.E., and Atwood, M.E. (1992). The Precis of Project Ernestine or An Overview of a Validation of GOMS. *Proceedings, CHI'92*. New York: ACM, 307-312.

Greenbaum, J. and Kyng, M. (1991). *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, NJ: Erlbaum.

Hefley, W. and Murray, D. (1993). Intelligent User Interfaces. In Gray, W., Hefley, W., & Murray, D. (Eds.), *Proceedings of 1993 ACM/AAAI International Workshop on Intelligent User Interfaces*. New York: ACM.

Hefley, W. E. and Romo, J. (1994). New Concepts in Engineering Processes for Developing Integrated Task Environments. *Proceedings of the IEEE 1994 National Aerospace and Electronics Conference.—NAECON'94 (Dayton, OH, May 23-27, 1994)*, Vol. 2, pp. 680-687. New York: IEEE.

Herbsleb, J. and Zubrow, D. (1994). Software Process Improvement: An Analysis of Assessment Data and Outcomes. 1994 Software Engineering Symposium. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Hix, D. and Hartson, H. R. (1993). *Developing User Interfaces*. New York: Wiley.

Humphrey, W. S. (1989). *Managing the Software Process*. Reading, MA: Addison-Wesley.

Humphrey, W. S., Kitson, D. H., and Kasse, T. C. (1989). *The State of Software Engineering Practice: A Preliminary Report*. (Tech. Rpt. CMU/SEI-89-TR-1). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon Univ.

Kitson, D.H. and Masters, S. (1992). *An Analysis of SEI Software Process Assessment Results 1987-1991*. (Tech. Rpt. CMU/SEI-92-TR-24). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Lim, K. Y., Long, J. B., and Silcock, N. (1990). Integrating Human Factors with Structured Analysis and Design Methods: An Enhanced Conception of the Extended Jackson System Development Method. In Diaper, D., et al (eds.), *Human-Computer Interaction - INTERACT'90*. Amsterdam: Elsevier, 225-230.

Long, J., Hakiel, S., Hefley, B., Damodoran, L., and Lim, K.Y. (1994). Guilty or Not Guilty? Human Factors Structured Methods on Trial. *Human Factors in Computing Systems-CHI'94 Conference Companion*, 181-182. New York: ACM.

Mastaglio, T. W. (1990). *User modelling in cooperative knowledge-based systems* (CU-CS-486-90). Boulder, CO: University of Colorado, Boulder. Dept. of Computer Science.

Morrison, P. (1993). Is There a Discipline of User Interface Design? *Ergonomics in Design*, 23-28 (October, 1993).

Muller, M.J. and Kuhn, S. (Eds.) (June, 1993). Sepcial issue on Participatory Design. *Communications of the ACM* 36(6).

Myers, B.& Rosson, M. (1992). Survey on User Interface Programming. *Proceedings CHI'92*. New York: ACM, 195-202.

Nielsen, J. (1993). *Usability Engineering*. Boston: Academic Press.

Noro, K., and Imada, A.S. (1991). *Participatory ergonomics*. London: Taylor and Francis.

Norman, D.A., and Draper, S.W. (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Olson, J., and Olson, G.M. (1990).The growth of cognitive modeling in human-computer interaction since GOMS. *Human-Computer Interaction* 5(2-3), 221-65.

Paulk, M. C., et. al. (1993a). *Capability Maturity Model for Software, Version 1.1*. (Tech. Rpt. CMU/SEI-93-TR-24). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Paulk, M. C., et. al., (1993b). *Key Practices of the Capability Maturity Model, Version 1.1*. (Tech. Rpt. CMU/SEI-93-TR-25). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Rousseau, N. P., Candy, L., and Edmonds, E. A. (1993). Influence, discretion, and time available: a case study of HCI practice in software development. *Interacting with Computers* 5(4), 397-411.

Schuler, D., and Namioka, A. (Eds.) (1993). *Participatory design: Principles and practices*. Hillsdale, NJ: Erlbaum.

Tauber, M. (1990). ETAG: Extended Task Action Grammar - A Language for the Description of the User's Task Language. *Human-Computer Interaction - INTERACT'90*. Amsterdam: Elsevier, 163-168.

Tuck, R., and Olsen, D. R. (1990). Help by Guided Tasks: Utilizing UIMS Knowledge. CHI '90 Conference Proceedings, Seattle, WA, 1990, April 1 (pp. 71-8). New York: ACM.

Whitehurst, H.O. (1993). Human Factors Contributions Early in System Development.*Ergonomics in Design*, 17-22 (October, 1993).

Whiteside, J., Bennett, J., and Holtzblatt, K. (1988). Usability Engineering: Our Experience and Evolution. In Helander, M. (ed.), *Handbook of Human-Computer Interaction*, 791-817. Amsterdam: Elsevier.