Integrating software engineering and HCI

David Benyon, Alistair Kilgour and Bengt Sandblad

Napier University,
Dept. Computer Studies
219, Colinton road,
Edinburgh, EH9 2JU
d.benyon@dcs.napier.ac.uk

Heriot-Watt University,
Dept. Computer Science,
Riccarton,
Edinburgh
ack@cee.hw.ac.uk

Center for Human-Computer
Studies,
Uppsala University,
S-752 37 Uppsala
bengt.sandblad@cmd.uu.se

Abstract: There can be little doubt that human-computer interaction has become a fundamental consideration in computer science. All but the most theoretical of computer science and computer studies courses in higher education now include HCI as a core subject and commercial methods for interactive systems development are increasingly paying attention to the needs of users. The purpose of this workshop is to bring together researchers and practitioners in order to explore further the issues of integrating HCI and SE.

Key Words: Software Engineering methods, HCI, SE/HCI

1. INTRODUCTION

There can be little doubt that human-computer interaction has become a fundamental consideration in computer science. All but the most theoretical of computer science and computer studies courses in higher education now include HCI as a core subject and commercial methods for interactive systems development are increasingly paying attention to the needs of users. Methods such as STUDIO (Browne, 1993) have recently been published aimed at making HCI approaches available to systems developers and conceptualisations of the development process such as MUSE (Lim and Long, 1994) aim to show where and how software engineering and HCI should come together and share knowledge of the design process. The popularity of scenario-based design (Carroll, 1995) and its relationship with object-oriented methods of systems development such as Objectory (Jacobson, *et al.*, 1995) represents another point of contact.

However, there remain many difficulties in producing a truly united approach to interactive systems development which deals adequately with both software engineering and human-computer interaction issues (e.g. Benyon and Palanque,

1996). Some of these are primarily practical, e.g. can we find a common language for software engineers and human factors experts to use? Others are more theoretical, e.g. are the concepts employed by the two disciplines compatible?

2. THE ROLE OF TASK ANALYSIS IN SYSTEMS DESIGN

One area of debate concerns the importance of the concept of a 'task' in HCI. This concept has dominated the ontology of HCI since the earliest days as the wealth of literature on task analysis techniques will demonstrate (e.g. Diaper, 1989). However, others (e.g. Fischer, 1989) argue for a communication model of HCI. This model recognises the existence of shared knowledge in a particular problem domain. Humans and computers possess different but complementary knowledge and skills and should cooperate in the problem-solving activity.

The debate about the role of task in systems development is unresolved (Benyon, 1992; Diaper and Addison, 1992). To what extent can the concept of task be introduced into software engineering methods and to what extent is the

notion of task a useful and important concept in interactive systems design?

3. MODELS IN HCI/SE

A recent approach to user interface systems engineering is the 'model-based' approach (e.g. Sukaviriya, *et al.*, 1993). In this approach a domain, or application model is constructed which forms the basis for automated, semi-automated or manual systems design. The model-based approach promises to have many benefits, but one fundamental question which remains begging concerns the nature of the domain model; what should it be like, what should it focus upon?

A domain model for the purpose of software re-use may look very different from a domain model which has the purpose of automatically generating the human-computer interface. There are a number of candidate methods and formalisms which address both the conception and modelling of human-computer systems which focus on different aspects of HCI and SE (Sutcliffe, *et al.*, 1996). How can more recent approaches to HCI such as Activity Theory (Bødker, 1990) or Distributed Cognition (Hutchins, 1995) be reconciled with the needs and concepts of SE?

4. DESIGN APPROACHES

Most attempts at providing an integrated framework for HCI/SE (e.g. Lim and Long, 1994) present the concerns of SE and HCI running in parallel rather than being cooperative and integrated. Participative approaches to systems design and cooperative design (e.g. Greenbaum and Kyng, 1991) appear to conflict with the need to fix requirements as demanded by an engineering approach. Alternative life cycles such as the Star or the spiral model need to be reconciled. How can designers include efficient interface design and development methods and techniques into an Object Oriented modelling and development environment?

5. CONCLUSIONS

The need to bring together and to integrate software engineering and HCI can only be addressed by discussion between the software engineering and human-computer interaction communities. This is as true from the perspective of systems developers as it is from the perspective of computer science educators. In the future do we

expect our software engineers to be experts in HCI, or do we expect our HCI experts to be software engineers? This workshop will take an important step towards focusing the important issues and setting the agenda for future work in the area.

REFERENCES

- Benyon, D.R. (1992) The Role of Task Analysis in Systems design in *Interacting with Computers* 4(1)
- Benyon, D. R. and Palanque, P. (1996) (eds.) *Critical Issues in User Interface Systems Engineering*. Springer-Verlag; London
- Bødker, S. (1990) *Through the Interface*. Lawrence Erlbaum; Hillsdale, NJ
- Browne, D. (1993) *STUDIO* Prentice Hall
- Carroll, J. (1995) *Scenario-based design* Wiley, New York
- Diaper, D. (1989) (ed.) *Task Analysis for Human-Computer Interaction*. Ellis-Horwood,
- Diaper, D. and Addison, M. (1992) Task Analysis and Systems Analysis for Software Development. In: *Interacting with Computers*. vol. 3(3) pp 124 - 139,
- Fischer, G. (1989) Human-Computer Interaction Software: Lessons Learned, Challenges Ahead *IEEE Software*, (January) pp 44-52
- Greenbaum, J. and Kyng, M. (eds.) (1991) *Design at Work: Cooperative Design of Computer Systems*. Lawrence Erlbaum Associates: Hillsdale, NJ
- Hutchins, E. (1995) *Cognition in the Wild*. MIT press
- Jacobson, J., Chistensen, M., Johnson, P. and Overgaard, G. (1993) *Object-Oriented Software Engineering*. Addison-Wesley
- Lim, K. Y. and Long, J. (1994) *The MUSE method for usability engineering* Cambridge University Press
- Sukaviriya, N., Foley, J. and Griffith (1993) A second generation user interface design environment. *Proceedings of interchi 93*, ACM publications, 375-382
- Sutcliffe, A. G., Benyon, D. R. and Bubenko, J. (1996) (eds.) *Domain Knowledge for Interactive Systems Design*. Chapman and Hall; London