Invited paper
Being Critical in, on or around Computing?

Christiane Floyd
Software Engineering Group
Computer Science Department, University of Hamburg
Vogt-Kölln-Straße 30, D - 22527 Hamburg, Germany

INTRODUCTION

When I was invited as an opening speaker for this conference, I was not asked to embed it in what some call the Aarhus tradition that is now giving rise to the fourth decennial international and interdisciplinary scientific event. However, I found that I could not do otherwise. Partly, because I am full of admiration for what has been achieved, and partly because I wonder where the movement is now going.

While I was not aware of the first conference "Working in Systems Development" [6] at the time, I find it quite amazing in retrospect that scientists and professionals in the computing field here saw the need to view their technical work *in its social context* already in 1975. In other countries (including Germany and the US), these questions were either not raised or they were debated under headings such as "Computers and Society" or "Informatics and Society", with quite different implications. In fact, the latter titles are no longer considered to be relevant any more, while the issues of interest are now generally discussed as "Contextual Informatics". Congratulations.

The universe of discourse evolving from the original concern was shaped by several factors: political alliance with representatives of computer users, social theories permitting to view computing in context, and participatory methods for system development. Scandinavia has dominated the international discussion in participatory design (PD) ever since, but there were important variations that reflected other cultural contexts in Europe and North America.

The second conference "Computers and Democracy" [7] showed how the ambition, the self-confidence and the international standing of the Aarhus community had increased. It was an important event for many and had a great impact worldwide. It was also important for me. It gave me an opportunity to consolidate my critical position in software engineering, and my contribution "Outline of a Paradigm Change in Software Engineering" [2] , which focussed on the complementarity of the process-oriented and the product-oriented perspectives, was regarded as a theoretical underpinning for the work going on in this community at the time. While the conference was interdisciplinary, it addressed primarily an audience in computing and provided a platform, where computer professionals could find inspiration and guidance for orienting themselves in the social dimensions implied by the systems they constructed.

These concerns of the 1980s reflected the tacit assumptions of societal makeup, information technologies and applications at that time. In the 1990s they were superseded by an increasing movement to personal and cooperative computing, creative design, innovative technologies, and aesthetical concerns. The emergent social theories portrayed networks of people interacting with IT artefacts in multicultural contexts.

The title of the third conference "Joining Forces in Design" [8] referred to enlarging the horizon in interdisciplinary work. It aimed at studying the nature of design, bringing together scientists from the computer field with social scientists and artists. I was not an active participant then, but the conference had been timed so as to coincide with TAPSOFT'95 (Theory and Practice of Software Development) where I was an invited speaker. I was very positively impressed by this timing and took the opportunity to give a presentation [3], which aimed at creating a bridge between the formal community that I was supposed to address and the design community that met next door. Regrettably, I was quite disillusioned: On one hand, the formal community did not find it worth while to attend my talk. My friends from the design community, on the other hand, gave me indulgent smiles for my futile attempts at bridging a gap.

The *study of computer use* had become a discipline on its own, drawing on approaches from the humanities and the social sciences, developed for understanding human work, learning and communication as well as the place of artefacts in networks of human activities. There are some

AARHUS'05 8/21-8/25/05 Århus, Denmark

outstanding examples of social scientists cooperating with computer professionals so as to inform their technical work. There have been remarkable pilot projects. Moreover, there have also been institutional attempts at promoting such collaborations: research laboratories accommodating social scientists alongside with researchers working in the technical and formal fields, university departments combining the two, curricula in the computing disciplines aiming at multidisciplinary reflection. Thus, the world has changed, and this, again, is a success that the Aarhus community – at least in part – can take credit for. However, the actual connection of these context-oriented reflections on computing with constructive approaches in computing is becoming more and more fragile. I could name several attempts at collaboration that have been given up or that have degenerated into working side by side in adjacent rooms.

Meanwhile the world has undergone a profound change in terms of politics, economy, theories, technology and social involvement. IT pervades our world and affects it in countless basic and important ways. In fact, it has given rise to a world wide transformation that many people like to call the information society.

Now we meet for "Critical Computing". My basic concern is: Can we live up to this title? What is Critical Computing about? A Google search yields many entries for "safety critical", "business critical" and "mission critical" computing along with the title of the present conference. What do we mean by computing here? What do we mean by being critical? And is critical computing, as understood here, about Critical Computing, as understood elsewhere, i.e. about *computing, where it matters*? Or does the community assembled here withdraw to the safe niches of designing nice IT artefacts, while leaving critical computing to others?

LIVING CRITICAL COMPUTING

There are several reasons why I would like to use my privilege of opening this conference as an opportunity to honour the memory of Kristen Nygaard. One is that he may well be considered the founding father of the Aarhus tradition. Another is that since he died in 2002, I have become increasingly aware of how much I miss him as a friend and colleague. But mostly, I would like to portray him as an outstanding example for what this conference is all about. In our conference on "Software Development and Reality Construction" we embedded a section on *living computer science*, featuring diverse personalities. Kristen Nygaard reflected on how he *lived* Critical Computing [10]. Of course, he was not perfect, his contribution is now history, we cannot aim to act as he did, the world has changed. But we can see through his example what Critical Computing can be.

There is no doubt that Kristen Nygaard worked *in computing*. He started as a *practitioner* in operations research, he got to know computers as tools used to facilitate logistics in technical and commercial fields, and to simulate systems comprising human beings in interaction with technical equipment of various kinds.

The trace that he left in *computer science* reflects this origin: the very name of the language SIMULA that he co-invented with Ole Johan Dahl – no doubt the fruit of controversial discussions between the two partners – points to his concern with modelling and simulation. Around 1965, when the development of SIMULA began, this concern was quite advanced and way ahead of its time. I would like to highlight some aspects of how Kristen Nygaard worked as a computer scientist. One is that he sought the cooperation with Ole Johan Dahl – a renowned computer scientist with a formal background. The other, that the language SIMULA was based on the best programming language available at the time – Algol 60. This shows an awareness and appreciation of state of the art technology and the willingness to build on the conceptual and technical work of others (as seen from now, the choice may seem regrettable). And lastly, the modelling concepts that SIMULA pioneered and that became the start of object-oriented programming were not invented ad hoc, but carefully adapted from philosophy. In fact, this is the distinguishing feature of the Norwegian approach to object-oriented programming as compared with others (mostly from the US).

Kristen Nygaard, therefore, gives an outstanding example of scientific cooperation both within his field and across disciplinary boundaries – a cooperation that was later enhanced when he turned his attention to application systems used in human work and organizations.

Of course, most of the people attending this conference, have known Kristen Nygaard as a *political man in computing*. An attitude, which was controversial, and which he adopted consciously, knowing well that it would get him in conflict with the objectivist scientific establishment at his time [10]. His political work, as well as his scientific contribution is marked by the willingness to find and cooperate with allies, to build on existing structures, to work towards effective societal changes and to make the best possible use of technology in order to bring these changes about.

Alongside with his steady political involvement, Kristen Nygaard also took a profound interest in the *nature of the discipline of computing* [9]. In Norway, the name informatics was used early on, and he gave a beautiful definition of *informatics as a scientific discipline*. Also he supplied an epistemological definition of the concept of "system", making clear that people consider something as a system for a purpose. He also made explicit a profound shift

by introducing the notion of *perspectivity* as a basis for constructive technical work – a radical change from the positivist, objectivist tradition he was brought up in. In laying these foundations, he made important contributions to understanding design in the computing field.

To my knowledge, Kristen Nygaard did not ever concern himself with developing or even adapting *social theory*, he stayed within computing, though he encouraged and enabled people in his environment to do so. As for himself, the idea of perspectivity, elaborated in [9], allowed him to regard the acting system designer as a responsible professional making choices. Moreover, his emphasis on the "conflict perspective" rather than the harmony perspective made it clear that he viewed society as shaped by clashes of interests and ongoing struggles for compromises between different interest groups.

Notwithstanding his critical action, Kristen Nygaard, throughout his life, retained an *enthusiasm for innovative technology*, exemplified mainly in connection with the development of the Apple / Macintosh line, which he followed closely. Also, having accomplished his political missions in his later years, he returned to a keen interest in object-oriented programming. This – not participatory design – would be his old age field of scientific activity, so was his plan.

Beyond all this, perhaps the most important contribution of Kristen Nygaard was that of networking and creating a community. It is the community that meets here and now, the PD family that to some extent owes him its existence and now stands on its own.

So, these are the levels of critical computing that Kristen Nygaard points to. As intended by the organizers the focus of this conference is to be on *taking critical action*. In the rest of my paper, I will look at the changed conditions for critical action in the twenty first century: What is meant by critical action? Who is involved in critical computing? At what levels can critical computing take place and how? How can we find ways to interact with those who are responsible for shaping the critical technologies of our age?

CRITICAL COMPUTING PRACTICE – OR THE UNBEARABLE LIGHTNESS OF DESIGN

Critical computing rests on having an impact on computing practice. At the centre of critical computing there are IT artefacts, being critical refers to how they are developed and used. Understanding computer use, however, is not enough – it needs to be intertwined with development to have an impact. As compared to the 1980s, computing practice has changed profoundly, which makes it more difficult to assess where critical action is meaningful. Computing can be seen as nexus where different forms of practice meet. While from within computing, the development perspective is dominant, from a social science point of view the concern is with use. To promote a discussion on how these related, I would like to point out some fields of tension.

Experimental design vs. routine development: Judging from the papers submitted for this conference, it seems that the focus here is on the challenge of advanced interfaces to sophisticated IT gadgets rather than on the drudgery of conventional information systems. The community seems to prefer flying in high altitudes to tying itself to the ground of daily practice in organizations. This is a fascinating orientation in design research, well suited to enhance creativity and communication in interdisciplinary settings. How can it lead to critical computing? What will the impact be? Hopefully, model designs will inspire others who produce real-life technology – but the development of real-life technology seems to be largely outside the scope of discussion.

Design vs. construction: If design is a separate sphere of concern, how does it actually relate to constructing useful and reliable systems? There are different scenarios for combining design and construction, through prototyping and, beyond that, through methodological approaches to participatory software development. However, this intertwinement needs to be lived and made fruitful in research and teaching. While people working constructively in computing need to respect the competence of use-specialists, this respect needs to be mutual. High-quality software is the result of good design and good construction – the borderline between the two not being sharp. To enable critical computing, design needs to impact construction.

Design for requirements vs. design for opportunities: The original concern of critical computing was with IT systems supporting working life. Such systems were embedded in organizations with well-defined (though sometimes not articulated) interests. Also, there were well-defined user groups with knowable competencies and skills. Embedding IT systems in individual or collective work gave rise to requirements – which, though difficult to articulate and subject to change, can nevertheless be made explicit. This scenario is by no means outdated, but is no longer addressed directly. Why? Does research focus on innovative technology providing opportunities for unknown users? Does personal computing – from text processing to edutainment – obey to entirely different laws? Is it not necessary to relate the two forms of design?

Localized design settings vs. global distribution of labour: For those preparing themselves to work in the computing professions, it is becoming more difficult to get a clear picture of what they will be asked to do on their jobs. The settings of work in the computing fields are diverse – they influence the scope and meaning of design. There are a few specialized large companies producing most of the software in use. Design, there, refers to working on products that are sold on the open market, with little contact

to users. In many organizations, existing software or legacy systems are enhanced. What is the place and scope for design here? In the context of outsourcing, there is a new form of division of labour, requiring intercultural communication and very strict conventions for work. Again, what is the place for design? Can design flourish only in special reservations or can we impact practice on rough terrains as well?

Large scale IT systems vs. light weight IT artefacts: The development of software, depending on the actual application, relies on quite different skills and requires different methods. This is one reason why approaches to software development ranging from the waterfall model to agile methods have been developed, and also why participation, prototyping, and change can be accommodated or are rejected. The variety of IT systems also has a bearing on possibilities of critical action. How does reliability relate to usability? How can real-time systems be developed using participatory design? I would like a discussion on strategies for how critical computing can affect those areas of work where computing is actually critical for modern society.

Designing new vs. adapting and enhancing existing systems: While experimental development often pertains to new systems or artefacts, computing practice more and more consists in using, adapting, maintaining and re-engineering existing ones. What do we mean by design in this context? How can IT systems be introduced, how can their use be organized? What does critical action refer to?

It seems to me that all discussions concerning critical action *in* computing or *on* computing need to involve a careful empirical analysis of what actually goes on, what communities of practice exist in the computing field, and how it is possible to intervene in the network of activities involved.

CRITICAL THEORY ON COMPUTING – WHERE DO WE STAND?

In the 1980s, critical theory essentially referred to Marxist approaches for understanding the labour process on an individual or collective level in a society shaped by class struggle and conflicts of interest. Critical theory then pertained not to technology itself, which was considered neutral, but to how technology was used. It called for social or political change in order to allow for technology use in the interest of the working class.

The profound crisis initiated by the end of the Cold War – in German simply called "Wende" (turn) – in 1989 has not only lead to global political and economic change and a new world order marked by neo-liberalism, but also to a near-breakdown of ideologies and belief systems based on ideas of solidarity, egalitarian society and empowerment.

In the ensuing void, families of theories became important that emphasized the place of individuals in networks of communication, shaped by cultures and the use of artefacts. These families of theories – the constructivist schools of thought, the social analysis of actors and networks, the study of artefacts in cultures, the analysis of power in the manufacture of knowledge, and so on – have been developed for understanding human learning and creativity, individual and cooperative work, the interrelation between technology and organizations, and the interaction of cultures in a globalized world with multiple identities. They also have shed a new and fascinating light on computing and the development and use of IT and provide a starting point for dealing with the problems at stake here. However, these approaches mostly have been developed with no specific concern for computing.

Therefore, we face the task of selecting suitable approaches and tailoring them to the needs of our discipline. As the intertwinement between computer technology and the human world takes place in a variety of contexts, elaborating an adequate understanding for it becomes an extremely challenging task. In order to be fruitful, it seems mandatory to me to focus on the *specifics of computing technology*, and not to use the theories in a general way. There have been many attempts to promote the interaction between social schools of thought and the computing communities, I have been active in "Social Thinking and Software Practice" [1]. If, for example, we "attribute agency to artefacts", as we should, no doubt, following Latour, it seems nevertheless important to study the specific agency of computer systems, as distinguished from – for example – that of maps. "Blackboxing" a computer system comes with specific implications. Likewise, power struggles manifesting themselves in classifications, will become reified when computer-implemented. And so on. I use the term "auto-operational" to denote the specific agency of computer systems [4]. The greatest challenge seems to me, to view IT systems as artefacts in communities of practices, as boundary objects, as nodes in actor networks, and so on, while – at the same time – remaining aware of their formal nature and their technical makeup (I have just attempted with Stefan Ukena to do this for ontologies in [5]).

In my opinion, theories that we owe mainly to feminist discussions, emphasizing enaction, the body, interactivity and performativity, pose the most radical challenge to computing because they question the key role of the computing paradigm itself. Depending on how far back we look in history, we can see ourselves at the end of one century that focussed on formal symbol manipulation, on information processing, on computing in the most general sense as the foundation of cognition, thereby raising the paradigm of computing to the level of a basic paradigm for being human. Or we can see ourselves at the end of 2500 years of history of philosophy starting in Greece that has lead us eventually to ideas like separating the mind from the body and considering intelligent action on its own.

What now, if we take seriously the idea that the history of bodily enaction of experiences is fundamental for cognition instead. What is the place of computing in this context? Computing comes with the aura of disembodiment, of being abstract and allowing for further abstraction – from values, from physical needs, from being shaped by different cultures, from co-evolution with other living beings on a planet with limited resources. What if we seriously acknowledge that our human reality is not so? Are we heading for being disembodied cyborgs or rather for acknowledging that we are human beings enacting our lives in unique ways in communication with others and that computing is to be a part of it? And, if we settle for the latter, what critical action in computing can we adopt? It is my conviction that the key to critical action is authenticity, making our own values explicit, respecting those of others and reflecting so as to find common steps that we can take.

CONCLUSION

In the twenty first century, the conditions for critical computing have changed profoundly. While there is a danger to be critical around computing without any bearing on what is actually implemented in society, there is also an opportunity to find new ways for being critical in computing and on computing that become effective. In order to do this, a careful analysis of the place of computing in society is needed and theories for understanding IT and computing in a social context must take the specific nature of this technology into account. Going beyond this, we may challenge the universal claims of the computing paradigm itself as an objectivist, neutral instance governing knowledge and decision-making. We might rather emphasize the role of embodied individuals enacting their personal histories, reflecting their personal values and thus making their unique contributions. In the context of Critical Computing, Kristen Nygaard, in his time and in his own way, has given an outstanding example of making such a contribution. Following his example does not mean to attempt to be like him, but to find our own unique ways, reflecting our own values and priorities to do so.

REFERENCES

1. Dittrich, Y., Floyd, C., Klischewski, R. (eds.) (2002). Social Thinking - Software Practice, MIT Press, Cambridge (MA).
2. 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. Dower Publishing Company, Aldershot, Hampshire, pp.192-210.
3. Floyd, C. (1995). Theory and Practice of Software Development - Stages in a Debate. In: Mosses, P. D., Nielsen, M., Schwartzbach, M. I. (eds.): TAPSOFT'95: Theory and Practice of Software Development, LNCS 915, Springer Verlag, Berlin, pp. 25-41.
4. Floyd, C. (2002) Developing and Embedding Autooperational Form. In: Dittrich, Y., Floyd, C., Klischewski, R. (eds.): Social Thinking - Software Practice, MIT Press, Cambridge (MA), pp. 5-28.
5. Floyd, C. Ukena, S. (2005) On Designing Ontologies for Knowledge Sharing in Communities of Practice. In: Proceedings of Philosophical Foundations of Information Systems Engineering 2005 (PHISE'05) Springer Verlag, Berlin, pp. 559-569.
6. Arbejdskollektivet (eds.) (1975) Århus-konferencen januar 1975 – Arbejdsformer I systemudvikling. [Working Approaches in Systems Development] Datalogisk Afdeling, Aarhus Universitet PB-45-46.
7. Bjerknes, G., Ehn, P., Kyng, M. (eds.) (1987). Computers and Democracy - a Scandinavian Challenge. Dower Publishing Company, Aldershot, Hampshire.
8. Kyng, M., Mathiassen, L. (eds.) (1997) Computers and Design in Context. MIT Press, Cambridge (MA).
9. Nygaard, K. (1986). Program Development as social activity. In Kugler, H.G. (ed.): Information Processing 86 – Proceedings of the IFIP 10th World Computer Congress. North-Holland, Amsterdam, pp.189-198.
10. Nygaard, K. (1992). How many choices do we make? How many are difficult? In: Floyd, C., Züllighoven, H., Budde, R., Keil-Slawik, R. (eds.): Software Development and Reality Construction, Springer Verlag, Berlin, pp. 52-59.