Reinhard Keil-Slawik (Ed.)

Position Papers for Dagstuhl Seminar 9348 on

Interdisciplinary Foundations of System Design and Evaluation

September 19 - 23, 1994

organized by

Liam Bannon, Reinhard Keil-Slawik and Ina Wagner

Preface

In systems development, interdisciplinary cooperation among scientists coming from various disciplines has been proposed for a long time now, but still seems to be rarely accomplished or succesfully exercised in real world projects. There are, of course, outstanding examples, but so far interdisciplinarity has not become a matter of everyday practice. Sometimes, quite the opposite seems to be true, especially if we look at the curricula at our universities or if we examine the everyday situation of software developers.

On the other hand, a wealth of knowledge and experience has been gathered within diverse scientific communities, knowledge that is largely tacit, but nevertheless provides a rich background for mutual understanding. Besides the pleasant atmosphere and the excellent organisational framework provided by the people from Dagstuhl Castle this tacit knowledge and common experience contributed largely to the success of this seminar.

However, there is no way to capture the spirit and condense atmoshere of the discussions or cover the many constructive ideas and insights by assembling a number of abstracts, especially since many of the sessions were re-organized according to the interest of the participants and the situation at hand. Thus, what is normally regarded as a hindrance to interdisciplinary collaboration, namely, the variety and diversity of individual perspectives and professional backgrounds, became a source for an enlightening and stimulating discourse. Consequently, it was not the exchange of ideas, results, or position statements which made this seminar an outstanding accomplishment in the view of most participants, but the fact that mutual understanding and cooperative learning took place.

This learning process can hardly be conveyed to others by means of written documentation. Hence, the participants were asked to provide any material for this booklet which would help others to understand the individual work accomplished and its background, and which might provide a piece to the overall mosaic of interdisciplinary foundations of systems design and evaluation in general. As a result the nature, length and style of the individual contributions reflect the diversity we have also experienced during the seminar.

The report starts with documenting the original invitation that was sent to the participants to highlight our starting point. All other contributions then follow in alphabetical order.

Finally, I would like to thank Nina Graf for her enormous effort to help me with the technical side of the editing process. She did a wonderful job in administering the contributions, converting different formats on different technical platforms and putting it all together in a readable form.

Paderborn, September 1995

Reinhard Keil-Slawik

Contents

Invitation to the Dagstuhl Seminar on: "Interdisciplinary Foundations of Systems Design & Evaluation"

Liam Bannon
Reinhard Keil-Slawik
Ina Wagner

At the end of the 1960's, a new discipline began to emerge within the computer science community, namely software engineering, which had as its focus the conceptual and empirical study of software and thesoftware design process and how it could be supported by methods andtools. Much of the work in this field has tended to focus on thestructural and technical properties of programs and associated documents rather than on process aspects.

However, from the very beginning, notions such as "computing as a social activity", "the psychology of programming", "programming as theory building", "collective resource approach" or "process-oriented approach to software development" gave notice of research attempts that took the embedding of tools, methods and techniques into human activities as their starting point. Today, we are confronted with a wealth of theoretical reflections on the design process and there is a growing number of empirical studies and observations trying to unveil the idiosyncrasies underlying the processes of development and use of computer-related artifacts such as software systems, programming languages, development tools and methodologies.

Since software embodies a variety of claims and assumptions about the context and the nature of the problems to be solved by intro duction of the system at the workplace, the properties describing the relations between software and the usage context cannot be simply expressed in terms of formalisms. Too many mutually influential factors have to be taken into account. The nature of the problem as it is perceived by the designers changes with every new insight, and very often incompatible requirements lead to "design conflicts" that have to be resolved.

Generally, it can be said that in any practical design process the designers encounter design conflicts, i.e. they have to decide to what degree should/can certain requirements and demands be fulfilled at the expense of others? Such conflicts can be expressed in a variety of forms, such as: rigidity vs. flexibility, completeness vs. openness, controllability vs. individual autonomy, formality vs. understandability, etc.

Design conflicts stem from the fact that universality always competes with the `appropriateness of the specific'. The crucial point is that design conflicts are empirical problems. They cannot be resolved with reference to the mathematical or technical properties of the artifact to be designed. And as such, they may require an interdisciplinary approach.

Surprisingly often, research within computer science is still conducted in a non-interdisciplinary fashion and interdisciplinary work is not very well integrated into computer science curricula. Thus, scientific relations across disciplinary boundaries as well as across the various scientific communities of a discipline still need to be improved and intensified.

This issue of interdisciplinarity becomes more acute as the range of disciplines seen to be relevant to an understanding of the software engineering process becomes larger. Recently

disciplines such as work psychology and ethnomethodology, for instance, have entered the field with different research strategies and theoretical frameworks. In addition, different scientific schools have emerged over a period of years within specific research traditions. Finally, with the creation of new scientific communities such as HCI (Human-Computer Interaction) or CSCW (Computer Supported Cooperative Work), which take an inherently interdisciplinary approach towards the design of interactive systems, the need for a broader interdisciplinary understanding of the design process became widely acknowledged.

The aim of the seminar is to bring together outstanding researchers and young scholars to

- assess the state of the art, esp. the degree of interdisciplinarity accomplished,
- reflect on design practice with special emphasis on what has been learnt in the past,
- discuss theoretical foundations, i.e., whether or to what extent specific frameworks may provide an adequate platform to integrate knowledge about design,
- evaluate methods, tools, and guidelines,
- propose future research strategies and curricular changes.

Since the above mentioned topics comprise a vast variety of different aspects ranging from the social organisation of system development processes to the effective utilisation of tools and techniques for software development it may be advisable to identify certain topics or problems on which to focus. A tentative list may include:

- Where are the boundaries between different disciplines to be drawn (limits of competencies, interdisciplinary division of labour)?
- Do we need a special educational programme or what should be changed in the traditional computer science curricula?
- How can the division between engineers (constructing) and social scientists (evaluating) be bridged on a methodological level?
- Are there theoretical frameworks which can serve as a common frame of reference for productive interdisciplinary cooperation?
- What are the key issues for interdisciplinary research for the next decade?

The long term goal of the seminar is to further interdisciplinary cooperation. Interdisciplinary work provides a key stimulus for new and innovative approaches to systems development and it may help to create the multitude of perspectives and approaches which is needed to cope with the complexity of the problems at hand.

Some Remarks about the Validation of Information Systems Development

Urs Andelfinger

Introduction

Validation of information systems development can be understood as a communicative process of assessing an information system with respect to the needs and requirements of the real-world situation it is supposed to interact with. In my opinion, validation therefore plays a central role in legitimatizing information systems development. In the following I will draw in outline some remarks about conceptual and methodological foundations of validating information systems. In my opinion, it is important to keep in mind that validation requires an interdisciplinary approach which could well be one reason why concepts of validation still are not equally well developed in computer science as for example formal approaches of program verification.

Validation and Verification

As starting point, I will use validation and verification in the almost classical sense of Boehm's definition that he gave in his paper on the verification and validation of software requirements and design specifications (Boehm:84). Boehm defines and distinguishes these two activities informally as follows:

- Verification is concerned with the question: Am I building theproduct right? This means, that verification essentially determines whether or not the product under construction meets some already known or otherwise given requirements. On the other hand, the relevance of these requirements for the underlying problem or situation is not the main issue of verification. So, verification is essentially a formal activity.
- Validation is concerned with the question: Am I building the right product? This means, that validation essentially determines whether or not the product under construction meets the requirements of the underlying problem or situation. The main issue of validation is therefore to determine and to evaluate the links between the situation at hand and the product that is supposed to offer a solution to some problematic aspects of this situation. So, validation is essentially a human activity that involves assessment with respect to some concrete situation.

In this sense, validation of information systems is concerned with their evaluation with respect to the given (real-world) situation. Validation therefore implies human assessment that cannot completely be formalized or made explicit, but instead always relies on some implicit constituent and some practical real-world background.

A Classification Scheme of Information Systems and their Evaluation

In response to the increasing need for effective software development in the sense of being adequate, reliable and valid for the problems to be tackled, Lehman proposes a classification scheme for computer programs, that increasingly takes into account the embedding of programs into the surrounding real-world situation (Lehman:80). From his classification scheme Lehman

then derives some important consequences for the verification and validation of computer programs. In my opinion this approach seems useful to further clarify the scope and the claim of the validation of information systems development.

Lehman calls the first category of computer programs S-Programs. Their function is formally defined by and derivable from a specification. The relationship of both the specification and the program to the external world "is a casual, noncausal relationship" (Lehman:80, p. 1061). Correctness is the appropriate criterion to judge S-Programs, the proof of the correctness is called verification. Verification is fundamentally limited to the proof, that there is a mathematically sound correspondence between the specification and the program.

The second category of computer programs is called P-Programs. The problem statement underlying P-Programs must be understood as "an abstraction of a real-world situation, containing uncertainties, unknown, arbitrary criteria, continuous variables. ... Both the problem statement and its solution approximate the real-world situation" (Lehman:80, p. 1062). The problem statements of P-Programs are thus definable only partly with respect to the real-world situation they are supposed to interact with.

The third category of computer programs is called E-Programs. E-Programs are supposed to "mechanize a human or societal activity ... The program has become a part of the world it models, it is embedded in it" (Lehman:80, p. 1062). E-Programs are thus continually subject to evolution and change: "The pressure for change is built in" (Lehman:80, p. 1063).

To judge P- and E-Progams as a whole, it is not appropriate to demonstrate the mathematically sound equivalence between problem statement or specification and the program:

1. The problem statement itself is an abstraction and hence a reduction of the real-world situation at hand. The basis for any attempt to verify P- and E-Programs as a whole therefore has to be considered as being inherently fuzzy, which fundamentally restricts the practical relevance of the result of such an attempt: "a program may be formally correct but useless" (Lehman:80, p. 1064).
2. The relationship between the program and the real-world problemd is or should be at least a systematic or causal one. Therefore, the problem statement itself is subject to continous change or evolution because of the intrinsic dynamic of the underlying real-world problem as opposed to the static nature of formal specification and verification.

Absolute correctness as a whole is thus not the issue with P- and E-Programs, even if parts of them can and should be proved to be correct. Instead, the appropriate criteria to judge those programs is their "value and validity of the solution obtained in its real-world context" (Lehman:80, p. 1062). Therefore, "validity depends on human assessment of their effectiveness in the intended application" (Lehman:80, p. 1062). For example, Lehman categorizes operating systems as a whole as E-Programs, even if they are still on a level quite close to formal definability. This does not exclude, that some parts of operating systems can and have to be verified formally in order to get them reliable and safe. But the evaluation of an operating system as a whole necessarily involves the validation of its usability and its relevance with respect to the situation at hand. In my opinion, Lehman's categorization amounts to the consequence, that for most information systems developments as a whole the appropriate means of evaluation is validation instead of verification (see also e.g. Colburn:93).

Validation as Process and Product of Human Assessment

Information systems development can be seen at least under two perspectives, as suggested for example by Floyd and Andersen et al. (Floyd:87, Andersen:90). In the product-oriented perspective, which is predominant in classical software engineering, information systems development is mainly seen as a production process with the information system "as a product standing on its own, consisting of a set of programs" (Floyd:87, p. 194). The context of usage is considered as given and in a certain sense as static throughout the Development Process. This attitude is e.g. reflected in the widespread belief in software engineering that software requirements can be fixed in a consistent way in the very beginning of program development.

In the process-oriented perspective, information systems development is mainly seen "in connection with human learning, work and communication, taking place in an evolving world with changing needs" (Floyd:87, p. 194, ref. also Nygaard:86). Floyd and Anderesen et al. therefore suggest, that information systems development has to be seen simultaneously as an ongoing process of social interaction and negotiation and as a product-oriented activity. The two perspectives are mutually complementary in order to "find ways to a more human-oriented technology" (Floyd:87, p. 207).

Validation of information systems development can take advantage of this dualistic perspective on program development. On the one hand, validation has to evaluate an Information System as a product and will itself produce such a finding as its result. But these results can only be understood adquately if the process that led to the results is also included in the conception of validation. Thus, validation of information systems development is best conceptualized as being both, a product and a process of human judgement and assessment.

The Political Content of Validation

Validation as sketched in the paragraphs above can be seen essentially as a process of social interaction, negotiation and assessment. This means, that validation has intrinsically a political content in the sense that a.) it has to cope with individual perspectives on real-world situations, b.) it has to identify and to consider potentially diverging social interests and c.) it raises ethical values. Validation hence implies "dealing with conflicts and contradictions" (Floyd:87, p. 208) because it deals with real-world situations.

Politically relevant are for example the choices and decisions defining

a) the kind and range of topics that are taken into consideration by the validation, i.e. the definition of the relevant frame of reference and

b) the individuals or social (interest) groups who will be participating in the validation.

As an important consequence of the choices made in a.) and b.), these decisions also determine what aspects and whose interests of the real-world situation will be neglected or excluded by the validation.

Another political aspect is the choice of the means or procedures that are admissible in the validation process. For example, this can be a discourse among selected representatives of the information systems development process or it can be the use of some prototyping techniques. In my opinion, validation is necessarily dependent on some discursive elements, because the task to evaluate an information system always requires some reflection on e.g. how well are the requirements and social interests met, which and whose are not or to what degree and why. This does not mean, that validation can abstract completely from practice, it is rather meant as a

mutually complementary relation. So, discursive elements are in my opinion a necessary constituent for validation, even if they solely can never be a sufficient condition of validation.

Conclusion

Validation of information systems development inevitably involves political aspects. In my opinion, this amounts to the central challenge for the validation of information systems development: On the one hand we know that reality and the development of Information Systems can be perceived and assessed in many mutually conflicting or competing ways. On the other hand, information systems development and its subsequent use in organizations require that we have to come to judgements, to decisions and to action, at least in a preliminary sense. There are propositions, approaches and techniques in computer science that attack specific aspects of this situation:

Some approaches explicitly take their starting point in the inherently ambiguous real-world setting. Instead of seeking to come to an artificially purified uniform world-view they acknowledge the existence of multiple perspectives and interests and try to exploit this situation. Bratteteig et al. for example propose a dialectical approach in Systems Development, that tries to turn the underlying competing socio-political interests and conflicts which are inherent in information systems development into a driving force in the development process (Bratteteig:94).

Other approaches try to come to a settlement of conflicting social interests and social groups by explicit means of discourse. Lyytinen and Klein for example propose the application of the critical theory of Juergen Habermas to arrive at a mutually shared consensus on the planned information systems development (Lyytinen:85). In my opinion, this approach is especially important with respect to the claim of validation because it offers four criteria of validity that have to be observed if legitimate information systems development shall be arrived at. The four claims of validity postulated by Habermas are: "Comprehensibility (clarity), truthfulness (veracity), sincerity (correctness vis-a-vis a speaker's intentions), and rightness with respect to norms. ... If an IS scores well on all four criteria, it would be considered legitimate by the potential users" (Lyytinen:88, pp. 22, 28) The problem with this approach is, that Habermas "provides no systematic procedure to assist this reflection" (Lyytinen:85, p. 230).

In my opinion, validation of information systems development therefore is best conceived of as a combination of dialectical and of consensus-oriented approaches. The combination would consist on the one hand of enriching Habermas' notion of Communicative Action with a sort of allowing some dissens among the participants, in order to allow for the practical need to act, even if there are still disputes to be settled. On the other hand the dialectical attitude would allow for explicitly keeping in mind these conflicting assumptions and interests. Finally, in order to really exploit the dynamic potential of dialectics and the legitimatizing potential of discursive validation, validation as a whole should be understood in an evolutionary and iterative way that remains equally susceptible to Habermas' four claims of validity, to the unsettled disputes of above and to new emerging practical experiences or requirements.

Comprehensive concepts of validation are still beyond widespread practical use. Currently, we are investigating conceptual and methodological foundations as sketched above. There still remain some complicated questions to be solved, e.g. how far do we have to go towards a shared understanding before taking action, or e.g. how strong must be the violation of one of the four claims before the system has to be modified. Finally, what is the relationship between reflection-orientied constituents of validation such as discursive elements and its

complementary practical constituents and experiences. We hope that our research eventually contributes to a better understanding of how valid and legitimate information systems development can be best assisted.

References

Andersen:90: Niels E. Andersen, Finn Kensing et al.: *Professional Systems Development - Experience, Ideas and Action.* Prentice Hall, New York, London, Toronto, 1990.

Boehm:84: Barry W. Boehm: Verifying and Validating Software Requirements and Design Specifications. *IEEE-Software*, Vol. 1, No. 1, January 1984, pp. 75-88.

Bratteteig:94: Tone Bratteteig, Leikny Agrim: Dialectics-Structured Handling of Problem Situations in System Development. In: W. R. J. Baets (eds.): *Proc. of the Second European Conference on Information Systems*, Nijenrode University, Breukelen, 1994.

Colburn:93: Timothy R. Colburn, James H. Fetzer, Terry L. Rankin (eds.): *Program Verification - Fundamental Issues in Computer Science*. Kluwer Academic Publishers, Dordrecht, Boston, London, 1993.

Floyd:87: Christiane Floyd: Outline of a Paradigm Change in Software Engineering. In: Gro Bjerknes, Pelle Ehn, Morten Kyng (eds.): *Computers and Democracy - A Scandinavian Challenge.* Avebury, Aldershot, 1987, pp. 191-212.

Lehman:80: Meir M. Lehman: Programs, Life Cycles, and Laws of Software Evolution. *Proc. IEEE*, Vol. 68, No. 9, September 1980, pp. 1060-1076.

Lyytinen:85: Kalle Lyytinen, Heinz Klein: The Critical Theory of Juergen Habermas as a Basis for a Theory of Information Systems. In: Enid Mumford et al. (eds.): *Research Methods in Information Systems, Proc. of the IFIP-WG 8.2. Colloquium*, 1-3 September 1984, North Holland, Amsterdam, 1985, pp. 219-236.

Lyytinen:88: Kalle Lyytinen, Rudy Hirschheim: Information Systems as Rational Discourse: An Application of Habermas' Theory of Communicative Action. *Scand. J. Mgmt.*, Vol. 4, No. 1/2, 1988, pp. 19-30.

Nygaard:86: Kristen Nygaard: Program Development as a Social Activity. In: Hans-Juergen Kugler (ed.): *Information Processing '86. Proceedings of the IFIP 10th World Computer Congress*, North-Holland, Amsterdam, 1986, pp. 189-198.

Problems of Interdisciplinarity

Liam J. Bannon

For a number of years, I have been concerned with the relation between theory and practice in computer systems design, and with the adequacy of the conceptual frameworks that we bring to bear on issues in this field. Starting from a base in cognitive science-inspired HCI work, I have been gradually expanding my focus of concern, going beyond the human-computer dyad to larger ensembles of people working together, and investigating their needs for support in the area of production and communication. What artifacts, tools, methods and media are needed to support people in their work activities and in their communications? How do we ”scientifically” investigate the work? It is apparent that the traditional disciplinary division of labour evident in research investigations does not necessarily provide the the most useful set of questions, from the point of view of practitioners. The complexity of the actual work processes is often ”tuned out” as particular research approaches narrow their focus in order to define what is an appropriate unit of analysis e.g. ”task”, ”act”, ”activity”, etc.

Likewise there is some confusion over what kinds of research methods are appropriate for this domain. The classical formal experimental manipulation has been shown to be of limited utility, as what matters in the real world is often not whether some factor makes a difference or not, but what size of difference does it make? Also, the attempt to control conditions often results in an experimental situation that bears little resemblance to real work settings. This has lead to a rush towards various kinds of field studies, in an attempt to hold in the richness of the work setting, but this also has lead to many studies where the observations are presented as a fait accompli, with little or no conceptual underpinning.

In the area of requirements engineering and other aspects of systems analysis and design, we have recently seen an increased interest in ”alternative” approaches to gathering information from users, such as various forms of participatory practices, and the use of ethnographers to describe the work practices of people. The number of research papers addressing how sociology might contribute to systems design is at this stage, legion, yet the effect of this work on actual practice seems limited. Indeed, perhaps all that is happening is the replacement of one research group (cognitive psychologists) with another (sociologists), neither of whom seem to have much to say to system designers!

Critiquing traditional narrow disciplinary approaches to the study of design does not however address what should be done. The call for ”interdisciplinarity” may not achieve much, unless the implications are spelled out in some detail. Undoubtedly design is a complex human activity, often involving a number of people, and extending over long periods. The need for multiple skills and frameworks for analysing and supporting the design process should be obvious. To what extent true interdisciplinarity can be achieved is an open question, i.e., the attempt to "wed" different disciplines together - for instance, by constructing common dictionaries of terms and concepts which the different disciplines are supposed to utilise, or making mappings across conceptual frameworks, thus attempting to ensure some "shared understanding" among researchers. I have yet to see any exemplary projects that have both argued for this approach and - more importantly - have produced worthwhile results, when viewed from each of the participating disciplines.

At times, it appears that discussions of the need for interdisciplinarity revolves around the utility of a variety of different instruments or methods for gathering data, but it is important to

note that a disciplinary perspective is not defined simply by the methods used. The recent interest in ethnographic field studies in CSCW is a case in point. While anyone can rather quickly become proficient at interviewing or taking notes, the essence of this approach lies in the perspective one adopts in framing the research, the way one chooses to interpret the findings, the authorities one invokes to support a particular interpretation, etc. - in sum, the whole conceptual framework within which one's approach to the world is framed. Note that the argument here is not that different disciplines are unable to contribute to a joint project, but that the aim of building some form of theoretical base to subsume a variety of conceptual frameworks from different disciplines implies a fundamental misunderstanding of different disciplinary perspectives. These different positions emerge out of different backgrounds, research traditions, perspectives, etc. which are not commensurable. For example, attempting to build some form of hybrid unified framework to encompass an empirical functionalist systems approach with an interpretivist constructivist approach seems doomed to failure, as the issue is not simply over different meanings to terms, but relates to fundamentally different World views. Certainly, theories can be broadened to include factors or circumstances previously omitted but such extensions do not thereby subsume other theoretical frameworks. That the quest for an interdisciplinary theory may be misguided does not therefore mean however that various forms of interdisciplinarity cannot work. Certainly, the possibility of a loose form of triangulation of research results through a variety of methods and techniques can be a very promising approach. This does not entail any strict one-to-one mapping between terms, but can allow for substantial overlap between ideas and frames of reference.

In the context of systems design and evaluation, it should be clear that there is plenty of scope for interdisciplinary studies, and indeed over the past few years such studies are emerging. It is my hope that at the Workshop we will discuss the problems and possibilities of such approaches.

Multi-Disciplinarity and Inter-Disciplinarity in System Design

Tone Bratteteig

System development includes continuous and planned development of computer based systems within an organisational setting, involving both technical and social processes. System development points to both construction and understanding: in order to construct a computer system we need to understand the technology and its application area, and in order to make the system useful we also need to construct parts of its social environment (eg, work tasks, user training). The knowledge and skills—the disciplines—involved in system development should be chosen with reference to what we construct and understand. Understanding technical matters is different from understanding somebody's work or organisational culture. Constructing a program is different from constructing a work procedure or a particular division of labour and responsibilities—a power structure. A number of different disciplines are needed during all stages of a system development process. The process can not be fully understood or evaluated without relating the different stages, activities, and disciplines to each other.

System development involves several disciplines, disciplines that have their own histories, theories, values and perspectives, methods, and languages. In discussions about multi-disciplinarity several levels of disciplinary integration can be identified[1]:

- uni-disciplinarity: representatives of individual disciplines meet, but the meeting does not affect their disciplinary identity at all.
- multi-disciplinarity: connects to team work in which cooperation with other team members (disciplines) may result in insight that transcends the traditional disciplinary border. This insight does not necessarily affect the original discipline.
- inter-disciplinarity: holds the perspective that the result can only be accomplished by a truly interactive effort that includes contributions from all disciplines involved. Inter-disciplinarity characterises the team and how the team develops a common foundation by applying multiple disciplines to a common problem area over some time.
- trans-disciplinarity: assumes a team composed by representatives from several disciplines. The process must be carried out as a joint effort, and roles and responsibilities are shared by the team members. The expertise of individual team members is recognised and used to train the other team members. Some view trans-disciplinarity as an objective for inter-disciplinarity in science.

I see system development as a common terrain that can be (should be) interpreted and handled by many disciplines, in different ways. This inter-disciplinary view emphasises the interconnectedness of activities in system development, still appreciating the differences from uni-disciplinary perspectives. Inter-disciplinarity presupposes the existence of individual disciplines that have their own kernel of knowledge, values, perspectives, methods etc. preserving their own foundation. In order to benefit from inter-disciplinarity, some degree of difference between the disciplines is needed. This is particularly true for creative processes like design[2]. Differences are the basis for creativity and action.

Trans-disciplinarity emphasises the conformity at the expense of the differences, and presupposes equal rights and resources for all members of a team—which is not always the case in system development. The complexity of system development suggests that both detailed, specialised knowledge and a general overview are needed. Roles and responsibilities need to be connected with the knowledge needed in order to understand the process. Responsibility connects with power, and I think that several bases for power is needed in system development[3]. Participants in system development do not only represent different knowledge bases, they also represent particular sets of interests in the process and its outcome: the computer scientists interests being different from the interests of management or any user group[4].

The inter-disciplinarity of system development does not imply that the "discipline" of system development should be inter-disciplinary. Inter-disciplinarity characterises the terrain of system development rather than some fixed set of knowledge needed to move in the terrain. The terrain

---

1. Cf. eg, Bailey, D.B. jr. (1984): A Triaxial Model of Interdisciplinary Team and Group Process. *Exceptional Children*, 51(1), pp. 17.25; Jantsch, E. (1980): Interdisciplinarity: dreams and reality. *Prospects*, 10(3), pp. 304-312; Lauvås, K. & Lauvås, P. (1994): *Interdisciplinary cooperation* (in Norwegian), TANO, Oslo
2. Buchanan, R. (1992): Wicked problems in design thinking, *Design Issues*, VIII(2), pp. 5-21
3. Cf. eg, Bjerknes, G. & Bratteteig, T. (1988): The Memoirs of Two Survivors: or the Evaluationof a Computer System for Cooperative Work, *Proceedings of the CSCW'88*, ACM, pp. 167-177; Hales, M. and O'Hara, P. (1993): Strengths and Weaknesses of Participation: Learning by Doing in Local Government, in Green, E. et al, (eds): *Gendered by Design?* Taylor & Francis, London, pp. 153-172

may change—every system development process is unique in some ways. I would argue for a multi-disciplinary approach to system development as an inter-disciplinary area of concern.

The need for a multi-disciplinary approach includes having roots in one discipline when professionally addressing complex phenomena transcending the borders of any discipline (ie, system development) in addition to having an openness to other perspectives grounded in other disciplines. The uni-disciplinary root gives a basis for evaluation of the phenomenon (standards, qualities, values etc), and for action (methods, tools). The multi-disciplinary approach includes an openness to other ways of acting and evaluating, eg, when computer scientists meet their users' professional standards for evaluating a particular computer system, as supplements to the technical standards of the discipline of computer science. Multi-disciplinarity therefore involves appreciation of other disciplines as a basis for mutual respect and for taking and giving responsibility in a concrete collaborative effort.

On the Importance of Everyday Design

Andrew Clement

I propose exploring the notion that systems design and evaluation are endemic to everyday (computerized) work. If we relax our preoccupation with design being principally concerned with readily demonstrable artifacts produced by specialized designers or design processes, then we may better see that design is going on all around us in many subtle and vital ways. The stable, but shifting patterns of work life often arise from reflective and engaged activity of practitioners. In overcoming obstacles and exploiting opportunities they creatively maintain and change the world. This can be observed in the development of rhythms, sequences and social connections within work practices, the rearrangement of tools, work objects and working environments, and the customization of tools that are malleable or open to manipulation. However, full achievement is hampered as much by the lack of legitimacy as by the unsophistication of technique. Reclaiming the authority for everyone, individually and collectively, to adapt the world to their needs, is a worthy interdisciplinary challenge.

4. Cf. eg, Bjerknes, G., Ehn, P., & Kyng, M. (eds) (1987): *Computers and Democracy—a Scandinavian Challenge* Avebury, Aldershot; Schuler, D. & Namioka, A. (eds) (1993): *Participatory Design. Principles and Practices* Lawrence Erlbaum Ass., Hillsdale, New Jersey

Computers & Society:
An Emerging New Discipline

Reinhard Keil-Slawik

Most scientists and computer professionals would probably view computer science as an engineering discipline. This characterization or classification can also be found in many computer science textbooks. But if we examine the material computer professionals basically have to deal with, i.e. software, we quickly come to recognize that this material has a dual nature. On the one hand, software is written text, i.e. the physical embodiment of operation sequences to control a machine. In this respect, designing a computing system, implementing and installing it such that it works appropriately, can be viewed as the typical task of an engineer who has to build technical artefacts which perform specific functions.

On the other hand, the ability to create, modify and exchange physical symbols is a prerequisite for human thinking, communication and the social organization of coordinated activities. The written media is also the means to develop a cultural identity between a large number of people dispersed in time and space, and it allows for the cooperative design of large or complex systems which could not be accomplished otherwise.[1] In this respect dealing with texts is much closer to the social sciences and the humanities. However, my point is not to ask whether or not computer science is an engineering discipline, but to ponder over this dual nature of software and the consequences for computer science as a discipline.

There are at least three problem domains where the dual nature of software plays a crucial role:

- On the *epistemological level*, the processes of using symbols to create meaning and foster understanding are confused with the respective results of these processes, such as formulas, textbooks and computer programs. The transformation of physical symbols is an indispensable part of all mental work. Computers allow us to perform such symbolic operations more effectively. But they can only do so as long as the transformation processes need not be revised due to new demands or insights of the system developers. We may say that computers embody the actual knowledge of the designers, but they do not create new knowledge on their own nor do they process meaning. The attempt to built machines smarter than their creators is an attempt to built cognitive perpetual motion machines.[2]
- On the *practical level*, we have to distinguish between programming as discourse, which is a meaning-creating activity, and programs as text. As designers of interactive systems we have learnt that users do not understand systems by reading bulky user manuals. It is the combination of reading manuals, using the system, and talking to fellow colleagues in which the meaning is created. The same holds for the development process. Documents have to be complemented by prototypes (executable programs) and the development of both of them requires extensive communication. More and more studies and observations of software development processes reveal that the meaning created in the communicative processes among programmers, designers, users, and managers,

1.Cf. Alexander (1964)
2.Cf. Keil-Slawik (1992 and 1994)

cannot be captured in documents.[1] Programming languages and tools have to reflect the control structures of the machine. But to a large extent they have to be designed to foster mutual understanding among developers.[2]

- On the *societal level*, we must acknowledge that the processing, exchange and interpretation of person–related data is an essential part of our social life. Especially when we deal with computers, almost any activity yields personal data which can be recorded, evaluated and processed further on. The automatic processing of personal data gives rise to a number of social conflicts, because it allows people to a certain extent to exert power and control over other people‘s life. Another problem is, of course, that the interpretation of such data is problematic insofar as it can only be done on the background of some socially or culturally stable pattern of behaviour. If this pattern changes the interpretation of the respective data becomes invalid, but this may not be noticed immediately.
  Due to the enormous potential of automatically processing such data the Federal Constitutional Court of Germany has defined the civil right of "informational self–determination". Its scope goes well beyond traditional ideas of privacy, and it has a lot of impact on the use of computers.

The three problem domains mentioned here have some characteristic in common, namely the inseparable intertwinement of technical, cognitive and social aspects.[3] This leads to a typical problem in computer science. Normally, a problem statement in a certain scientific or engineering context embodies the criteria which allow us to assess whether a proposed solution actually meets the problem statement, i.e. to decide whether it is an acceptable solution of the problem or not. Positioning eight queens on a chess board such that no queen can hit the other, is such an example. Most problems in practical and applied computer science, however, are of a different nature, because the specification of the problem is refined and revised as part of the overall design process.[4] In such a situation we may speak of an *open problem specification*. Open problem specification implies that the system design process has be generally characterized as a collaborative learning process, rather than a technical construction process.

The notion of software *versioning* is a strong indicator for the crucial role of open problem specifications in computer science. By talking about version *m.n* of a program we indicate that we view the different versions not as different programs but as one and the same program which is a solution to one and the same problem. The identity of these different artefacts is maintained by the fact that we view them as being the outcome of one and the same design or learning process.

Of course, design is a cooperative learning process in any discipline. But, due to the dual nature of software, the extent to which computer science has to deal with open problem specifications is significantly greater than in traditional engineering disciplines.

This is not only true for single development projects but also for the evolution of the discipline in general. The impact of computers on society as well as the strong demands to develop innovative technologies put forward by the society are strongly intertwined. Again, the dual nature of software plays a crucial role: Today we talk about the *information society* to indicate that the extensive use of information technology may lead to dramatic societal changes

---

1.Cf. Naur (1992) and Floyd (1995)

2.A wonderful example is the WEB system; see Knuth (1984)

3.Cf. also Parnas (1985), Nygaard (1986 and 1992), and Keil-Slawik (1989)

4.Cf. the contribution of Urs Andelfinger in this report, and the references given there.

in the long run. Although the notion of an information society is to a certain extent questionable it shows the strong relation between computer science and society.

So far, computer science has largely ignored the dual nature of software by either concentrating on the structural properties of the material (mathematics) or by assuming that there is essentially no difference between humans and machines. Consequently, building ever smarter machines would be the ultimate technological fix to all problems. However, the need to learn more about the particular relation of technology and its respective development and usage context has become more and more appparent not only to the society in general, but also to the scientific community. More than eight research groups on computers and society have been established in Germany and Austria during the last couple of years.

The task is to develop an epistemological and methodical framework that allows us to study the relation between computers and society in general, and man and machine in particular, such that the specific consequences for computer science become apparent. Wherever there are design options computer professionals need to know the consequences associated with either choice. Conversely, they need to know the specific societal demands and how they affect the design process. Finally, the shorter innovative cycles are and the more society changes due to changes in the technological infrastructure, the more experts are needed who are able to separate technical problems from political, aesthetical and psychological ones in order to clarify their relation and mutual dependance.

In this respect it can be said that computers and society as a discipline is the organizational means for computer science to better understand how computer artefacts have do be designed as to suit the needs for tomorrow‘s society.

References

Alexander, C.: Notes on the Synthesis of Form. Cambridge: Harvard University Press, 1964

Floyd, C., Züllighoven, H., Budde, R., Keil–Slawik, R. (eds.): Software Development and Reality Construction. Berlin: Springer 1992

Keil-Slawik, R.: An Ecological Approach to Resposnible Systems Development. In: Jacky, J.P., Schuler, D. (eds): Directions and Implications of Advanced Computing (DIAC 87). Norwood NJ: Ablex Publishing, 1989

Keil–Slawik, R.: Artefacts in Software Design. In: Floyd et al (1992)

Keil-Slawik, R.: Cognitive Imperialism. In: Güzeldere, G., Franchi, S.: Bridging the Gap. Stanford Humanities Review, Supplement to Vol. 1, No. 1, 1994.

Knuth, D.E.: Literate Programming. The Computer Journal, 27 (2), 1984

Naur, P.: Computing: A Human Activity. ACM Press, Reading (MA): Addison-Wesley, 1992

Nygaard, K.: Program development as social activity. In: Kugler, H.G. (ed.): Information Processing ‘86 – Proceedings of the IFIP 10th World Computer Congress. Amsterdam: North-Holland, 1986

Nygaard, K.: How many choices do we make? How many are difficult? In: Floyd et al (1992)

Parnas, D.: Software aspects of strategic defense systems. American Scientist, 73, 1985

An Emerging Design Approach

Finn Kensing

Introduction

With Keld Bødker and Jesper Simonsen I am working on an approach for designing CSCW systems. This paper is based on an earlier draft by the three of us. We advocate the importance of generalizing from own work practice as designers and from studies of designers working under industrial conditions. We use the term approach as something in between commodified methods and isolated techniques supporting one or a few activities.

Our main interest lies in designing for a specific organization's needs rather than generic products for a larger market. We use the term design in the same way as architects do - focusing on the analysis of needs and opportunities, and the preliminary design of functionality and form, ending up with representations for (others') construction and implementation. Therefor we see results of a design project to include a conceptual design in terms of a written document, sketches, mock ups and/or prototypes. We consider an evaluation of individual and organizational consequences of implementing the design as well as a plan for the implementation to be part of the result too. Based upon a design proposal it should be possible for the organization to say "go", "no go", or "more design is needed". Eventually the project may proceed to construction and implementation, but we consider this latter part of systems development to be outside the scope of our emerging approach which focus on the initial part of systems development.

What is reported on here is part of an research program, the purpose of which is to develop *theories of and approaches to* systems design. The research program comprises design projects carried out by us, as well as by others using our approach, and studies of designers working under industrial conditions.

A design approach

In order to make our own design approach explicit we have reflected upon nine projects we have been engaged in during the last six years [4, 5, 13, 17]. Starting from our own experience as designers, we present a first attempt to generalize in terms of an approach, specifying the *what*, the *how* and the *why*, as well as the *who* and the *where* when in projects we strive to get from an understanding of cooperative work to designing computer support. The point is not to promote what we have done as *the* CSCW approach, nor to start an inquiry to find such an approach. Rather the point is to facilitate one type of learning among practitioners, researchers, and students: learning from guidelines.

Application Area

We developed our approach, and hence our experience in projects, the aim of which has been to investigate opportunities for computer support for a specific organization. In all but one we were brought in because somebody, employees or managers, thought that computers might be part of solutions to problems they had encountered. The initial problem definitions have been quite open. We have carried out detailed studies of the organization's needs and opportunities

and designed tailored applications in combination with (modified) standard products found feasible.

Most of the people we have worked with saw the main part of their jobs as problem solving and problem definition rather than routine work, and cooperation was considered a substantial part of the jobs. The list of jobs comprises: radio journalists**;** university secretaries**;** operations people in an airport**;** managers**,** consultants, and secretaries in a multinational medical company**;** managers**,** editors, secretaries, and store-clerks in a film board; scientists in a R/D lab; and senior managers within the administration at a university.

A common objective of the projects has been to support the existing work force, which was considered overworked. Another has been that the existing work force or management wanted to automate some of the routine tasks. In some projects there was a request for computer support of activities which had really never been done before in the organization. Sometimes the purpose was stated explicitly to improve quality of working life and the product and service delivered by the organization. None had the (explicitly stated) purpose of head count reduction, down-sizing or "right-sizing" (sic!).

Seven projects we have been action-research, one has been a case study and one has been a pilot study. Our experience relates to more "office-like" settings compared to the many detailed "control room studies" within the CSCW community [1, 8, 9]. We find that the same level of attention is critical for design in these settings, and so do others [2].

In the action-research projects we worked closely with those to be affected by the design. The aim of two of these projects [5] was to clarify the employees own needs in terms of computer support in order to prepare for management plans for implementing standard systems. In the other five [17, 18] management and employees had agreed upon the need for a design project. In the pilot study [13] the aim was to develop and test forms of representataion for design. In the case study the aim was to study designers in action. The projects lasted from 3 to 12 calender months and comprised from 2 to 6 person months.

Perspective

We agree with Suchman [19] that categories do have politics. Guidelines may be used in rather different ways according to how you perceive what you are doing, and who is doing it to/for whom. Therefor we need to specify explicitly our basic assumptions and principles and how we as designers, working in participatory projects, perceive organizations; their members and their role in design, as well as our own role; and how we perceive design processes and their products.

Organizations do of course have structural properties, however organizations are not *there* to be studied, rather we perceive them as constantly being enacted through members' interaction and activities. Stable structures - understood as enacted social order - as well as procedural aspects need to be understood as part of a design project. Since organizations are constantly changing, a design might need a review if say it has "simmered" for eighteen months, as one of our designs did in the Film Board.

We see organizations as frameworks for cooperation as well as for conflicts. Therefore groups and individuals participating in design should be expected to have common, as well as conflicting goals. The role of designers is neither to cover up nor to solve political conflicts in design. Rather they should help the parties to formulate their visions, and leave it to them to solve conflicts in relevant fora.

We expect users, given the right opportunities, to be able to make their own decisions concerning what kind of computer support and work re-organization they might need and what

kind they might want to get rid of, cf. [14]. As addressed above "they" however, is seldom experienced to be a homogenous entity. In the Film Board we ran into a conflict between the production manager and the editors [17]. When we realized the conflict we arranged a meeting and explained the consequences as to each of the parties of various design decisions. The production manager then gave in, but subsequently tried - unsuccessfully - to persuade the president to make an end to the project in the department. Whether and how designers might approach conflicts that evolve in a project depends on how the conflict is related to the design project [17].

Working with users and from ethnographic studies of organizational life we have learned that often there is quite a difference between what people say they do and what they observably do. This is not necessarily because people play games (though they do), sometimes they are truly surprised when confronted with the difference. With the journalists in the Radio project our observations told us that there was a contradiction between their initial request for technology to support cooperation and their enacted values showing a desire for working solo. Our detailed study of their work practice aimed at making it discussible in which parts of their work they wanted to cooperate and in which they preferred to work alone with. The final design reflected a joint decision of this aspect [5].

We are in favour of participatory design as a democratic ideal. Also we are in favour of having users at all levels from the organisation participating in managing the project: it as a human right to be able to influence one's own working situation. Also we have pragmatic reasons: as designers we need direct interaction with users' knowledge in order to propose feasible designs, and there is a need for anchoring a design vision with those who are going to create the change.

Though we advocate a participatory approach we have not always succeeded in establishing a *real* working group consisting of users and designers taking joint responsibility for the process as well as the product of the design project. Sometimes we have had to accept that users would just show up at meetings arranged by us, being willing to be observed, test a prototype, decide upon what to do next, or what ever kind of activities we ask them to participate in. This is however not the ideal form of cooperation, either in terms of democratic principles or in terms of anchoring design visions in the organization.

A good product of a design process most often is a mix of tradition and transcendence [16]. One reason for bringing in designers is to transcend the tradition. At least someone in the organization has considered some of the old ways of doing things have lost their rationale, or found that new technological opportunities are worthwhile investigating. We have experienced managers as well as employees in that role. However, designers need to respect traditions in an organization, both as a way of maintaining (or establishing!) credibility but also because there often is a rationale behind phenomena perceived odd by a newcomer. Designers thus have to be careful in reading the meaning attached to mundane activities, modes of cooperation, or artefacts used in the work processes.

Overall approach

We apply a combination of intervention and ethnographic techniques in our overall iterative approach to design. In earlier work [26] we advocate that it is the responsibility of designers to set up activities applying tools and techniques that will allow themselves and users to develop knowledge at two levels, abstract and concrete, within three areas: users' present work, new

systems[1], and technological options[2]. A combination of intervention and ethnographic techniques in an iterative approach has turned out to be a good learning strategy for this purpose.

	Users' present work	New system	Technological options
Abstract knowledge	Relevant (2) structures on users' present work	Visions (5) and design proposals	Overview of (4) technological options
Concrete experience	Concrete (1) experience with users' present work	Concrete (6) experience with the new system	Concrete (3) experience with technological options

Figure 1. Six areas of knowledge in user-designer communication.
(Kensing & Munk-Madsen, 1993).

During a project we use the model in figure 1 as a point of reference. We are responsible for using tools and techniques that support communication with and among users within the areas indicated in the model.

It is crucial for designers to develop a thorough understanding of users' present work (work practice, organization of work, products/services, relations to customers, clients, suppliers, history of recent major changes, management strategies and style, etc.) This in order for the design to reflect - in a realistic way - the traditions of the organization. Realistic in the sense that the design reflects an appreciation of the rationale given by members of the organization, and in the sense that the organization is geared to meet the challenge of the envisioned design. Thus, by detailed studies of the present situation we try to "measure" the organizations needs and readiness for change. What we are trying to avoid is a too futuristic design or a design, the greater proportion of which will never be used. We have found that ethnographic techniques are helpful in accomplishing this.

Ethnographic techniques vs. intervention

Ethnographic techniques come out of a tradition where the basic idea was to develop and present to other scholars an understanding of a foreign culture. In its original form this implied that ethnographers tried not to change what they were studying. Current ethnographers however, reconceptualize this practice and try to establish an encounter between different cultures, for the purpose of informing those involved [1,10]. Also Blomberg, Suchman and Trigg report from a project "linking ethnography with design" in an organizational setting: "We orient to the details of people's practices, recognizing the importance of members' own articulation of what they do [....] we are accountable to the people who are or may become users of our technology" [2].

1.By new systems we mean new (or changed) computer systems and changes in the content and the organization of the users' work.

2.Here technology incorporates not only hardware and software, but also work organization. This may seem strange but in this context we find it useful and acceptable to group these matters. Various organizational options, as well as several hardware and software options, should be considered and coordinated in order to fit together as well as possible.

Interventionists deliberately set up activities designed to change the organization or the work settings of some of its members. The presumption is that it is through change that key factors of organizations and their members perception become observable. Our interventions address each of the three areas of discourse in figure 1. The intentions are to facilitate reflections upon current practice, to generate ideas, and to further develop the "technological fantasy" of users and designers.

We strive to select carefully the area and the mode of intervention based upon what we have learned by the ethnographic techniques. This is in contrast to some consultants bringing with them from site to site a design concept, claiming that what people in the organization know is irrelevant for their re-engineering project. Using ethnographic techniques - as they were originally developed - one spends years to develop and present an understanding of the culture studied. The interventionist is more impatient. Taking into account the time constraints put on most designers in the context we are talking about, we have found that interventions help make short cuts feasible. Also we find that ethnographic techniques provide a significantly deeper understanding than traditional computer science/software engineering techniques. This holds even when the former are used in "a quick and dirty way" compared to what they were originally developed for.

When we first tried to become quasi-ethnographers, colleagues and students claimed that such an approach would take far too much time, so why not start prototyping right away? We found that spending time on analysis, without going to the extreme of systems analysis of the 70-ties and 80-ties, paid back in relation to single out areas of the work relevant for prototyping and in relation to generating realistic design proposals. Also we found that detailed knowledge of users' current work allowed us to discard by 'mental testing' design ideas that turned out not to be worth prototyping [5, 17].

Ethnography and intervention are contradictory in terms of basic approach and intended results. However to us at a practical level, the two approaches in combination have been an effective way to learn about the organization and also a main resource for generating realistic visions of future use of technology.

We have one main concern though, which is part of the reason we think it is necessary to reveal and discuss approaches in the CSCW community, part of which develops technologies with a wide range of impacts on organizations, groups, and individuals. Getting to know people in an organization as closely as you do when carrying out in-depth analysis for the purpose of design, you easily get into political/ethical dilemmas [2, 17]. Since organizations are (also) political battle fields - people are fighting for their jobs, for preserving/getting an interesting job, for preserving/increasing their power base etc. And since the introduction of new technologies often affect such issues, designers cannot avoid playing a role and sometimes taking a stand in these battles. This is true whatever approach designers use, but some approaches allow you to keep a higher distance from those affected by your designs than others. Choosing an approach that might get you into close relations with users, you had better be prepared to defend your observations and design ideas - not all designers may be ready for that, nor may their employers give them the opportunity.

Iteration

The overall approach is iterative in two ways. First we iterate between analysis of the present and generating and eventually prototyping design ideas. This is not at all a new idea. What might be new is our hesitation to start prototyping before developing a thorough understanding of the organization in question.

Second we iterate between the two levels of knowledge indicated in figure 1. This is in contrast with most methods currently used in systems design, where an understanding is achieved only by acquiring abstract knowledge documented by formal tools and techniques. Kensing and Munk-Madsen [12] argue from a theoretical standpoint that designers have to put themselves in situations where they experience users while they are performing their every day activities. As illustrated by examples in [5, 17, 18], part of which will be shown below, the consequences this kind of experience had as to the proposed design.

Techniques

In our design projects we have applied a number of techniques to support the investigation os users' present work, technological options and the new system, as well as iterations back and forth between the various areas. Some of the techniques are well-known, such as observation, interviewing, and prototyping, while others are more specifically developed within various design traditions, e.g. design workshops from the Participatory Design-tradition [7, 15, 16]. Each technique provides information which might identify a need for further investigation, either in terms of opening up the search space - when it turns out that the problems are not properly understood, or not agreed upon - or narrowing down the search space - when it turns out that it is necessary to understand the problem in greater detail by e.g. using another technique.

Some techniques rely on users as informants through interviews in situations detached from their ongoing work, others rely on the designers' ability to observe users while performing their daily work, yet others establish a situation in between (e.g. interview in-situ). What ever the situation, we have found it important - even if we do focus on specific issues - to constantly remind ourselves to be open to what ever might come up. Field notes or audio/video recording are helpful tools for documenting and for shifting focus [5, 17, 20].

Ethnographers have developed elaborate techniques for analyzing recordings [11, 21]. We have no experience in this type of analysis, though for some tapes we have done content logging. Running through the tapes several times may generate hypotheses and design ideas, or identify issues to look for in greater detail. An example of this was found in an observed and taped meeting where the production manager and an editor from the Film Board were negotiating a contract with a producer and a director. Running through the tapes several times made us aware that in addition to the negotiation with the producer and the director, negotiations were taking place between the two people from the Film Board. This observation, which we did not note at the meeting or during the first couple of runs of the tape, turned out to have direct consequences for our design proposal later on [17].

Representations

In our design projects we have applied a number of representations to document knowledge on users present work, technological options or the new system. Designers might apply more formal representations for their internal communication; e.g. in order to develop a prototype a consistent data model (e.g. an E/R model) has been used. However, we tend to postpone formal tools and techniques introducing concepts and symbols not common to the users until detailed analysis and implementation of the visions. At that time designers cannot do without them, but still when users are involved in this part of design, some kind of translation might be appropriate. In general, representations are used to develop and represent knowledge with a particular emphasis, so it is a medium for developing knowledge as well as a medium for later

referencing. We judge the relevance of a description on how it facilitate discussions among us as designers and among us and users and their managers.

Questions

I have presented an emerging approach for designing CSCW-systems. We apply a combination of ethnographic techniques and intervention in an overall iterative approach. This is in line with what has been labelled "ethnographically informed design", as represented in e.g. [1, 10]. Their approach implies that the ethnographic study, performed by social scientists, inform the systems design, performed by designers, and further that the system design is evaluated and tested jointly. We take a slightly modified approach. Being computer scientists we have used ethnographic techniques in design processes. Our research goal is to develop theories and approaches for design oriented towards practitioners working under industrial conditions. Here we have not found sociologists available. The crucial question as to whether it is possible for designers in general - as lay persons - to apply concepts and methods from social science and the humanities is a very relevant question which we are investigating by studies of this kind. Another question for us is how to present our approach - what kind of guidelines and at what level of detail - designers from industry might find useful. These are the question I would like to discuss at the workshop.

Acknowledgements

Acknowledgements to Liam Bannon, David Bell, Jeanette Blomberg, Steve Harrison, Scott Minneman, Susan Newman, Lucy Suchman, and Randy Trigg.

References

[1] Blomberg, Jeanette, Lucy Suchman and Randall Trigg: Reflections on Work-Oriented Design in Three Voices. In: S.L. Star: *"Paris workshop"*, 1994

[2] Boehm, B.: *Software Engineering Economics*. Prentice-Hall, 1981

[3] Bentley, R., T. Rodden et al.: Ethnographically-Informed Design for Air Traffic Control. In: Turner, J. & R. Kraut (Eds): *CSCW '92 Sharing Perspectives. Proceedings of the Conference on Computer Supported Cooperative Work*, ACM, pp. 123-129

[4] Bødker, Keld, and Jørgen Bansler: A Reappraisal of Structured Analysis: Design in an Organizational Context. In: *ACM Transactions on Information Systems*, spring 1993

[5] Bødker, Keld, and Finn Kensing: Designing Computer Support for Editorial and Administrative Work in an Editorial Section in a Danish Radio Station. In: *Proceedings of the 16th IRIS (Information Systems Research Seminar in Scandinavia),*, Department of Computer Science, University of Copenhagen, Report no. 93/16, 1993, pp. 376-394.

[6] Ehn, Pelle. *Work-Oriented Design of ComputerArtifacts*. Arbetslivcentrum, Stockholm, 1988.

[7] Greenbaum, Joan, and Morten Kyng (eds).: *Design at Work: Cooperative Design of Computer Systems.* Lawrence Erlbaum Associates, Chichester, UK, 1991.

[8] Heath, C., and P. Luff: Collaboration and Control. Crisis Management and Multimedia Technology in London Underground Line Control Rooms. In: *Computer Supported Cooperative Work*, Vol. 1, Nos 1-2, 1992, pp. 69-94

[9] Heath, C, M. Jirotka, P. Luff, and J.Hindmarsh: Unpacking Collaboration: the Interactional Organisation of Trading in a City Dealing Room. In: G. De Michelis, C. Simone, K. Schmidt (Eds): *Proceedings of the Third European Conference on Computer Supported Cooperative Work*. Kluwer, Dordrecht, Holland, 1993

[10] Hughes, J., D. Randall & Dan Shapiro (1992): Faltering from Ethnography to Design. In: Turner, J. & R. Kraut (Eds): *CSCW '92 Sharing Perspectives. Proceedings of the Conference on Computer Supported Cooperative Work,* ACM, pp. 115-122

[11] Jordan, Brigitte, and Austin Henderson: Interaction Analysis: Foundations and Practice. To appear in *Journal of the Learning Sciences.*

[12] Kensing, Finn, and Andreas Munk-Madsen: Participatory Design; Structure in the Toolbox. In: *Communications of the ACM,* No. 36, Vol. 4, 1993, pp. 78-85.

[13] Kensing, Finn, and Terry Winograd. Operationalizing the Language/Action Approach to Design of Computer-Support for Cooperative Work. In: *Collaborative Work, Social Communications and Information Systems*, R. K. Stamper et al. Eds. North-Holland, 1991, pp. 311-331.

[14] Kyng, Morten: Designing for a dollar a day. *Office, Technology and People*, 4 (1989), pp. 157-170.

[15] Muller, M.J., S. Kuhn, and J.A. Meskill (Eds): *PDC'92 Proceedings of the Participatory Design Conference,* Cambridge MA US, 6-7 November 1992. Computer Professional for Social Responsability.

[16] Schuler and Namioka (Eds.): *Participatory Design: Perspectives on System Design.* Lawrence Erlbaum, Hillsdale NJ, 1993.

[17] Simonsen, Jesper, and Finn Kensing: Take Users Serious, But Take a Deeper Look: - Organizational and Technical Effects from Designing with an Intervention and Ethnographically Inspired Approach. *Proceedings from PDC'94*, North Carolina, October 1994.

[18] Simonsen, Jesper: *Technological Design in an Organizational Context*. Ph.D. dissertation, Roskilde University, Computer Science Department, 1994 (Forthcoming).

[19] Suchman, Lucy A.: Do Categories Have Politics? The Language/Action Perspective Reconsidered. In: *Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13-17 September 1993,* Milan, Italy, pp.1-14.

[20] Suchman, Lucy A., and Randall H. Trigg: Understanding Practice: Video as a Medium for Reflection and Design". In: [18].

[21 Trigg, Randal, Susanne Bødker and Kaj Grønbæk: Open-Ended Interaction in Cooperative Prototyping. In: *Scandinavian Journal of Information Systems,* vol. 3, pp. 63-86, 1991.

Interdisciplinarity and System Design for Post-Fordist Work

Kari Kuutti

Introduction

Information system design as a discipline has a relatively long history – about a quarter of a century. Already very early during this history it was recognized that system design influences and is influenced by issues outside of the realm of technical systems. Correpondingly, the question of the interdisciplinary nature of system design was raised already early in the 1970s .

Despite this early start the practical manifestations of interdisciplinarity have been more than scarce, however: although we have had forceful and visible, programmatic attempts to influence system design towards more broad disciplinary view, it is difficult to find a case where frameworks or methods imported from some other disciplines (than that of logico-technical system design) have had any impact beyond lip service on practical design. Let's look at a couple of examples, the cases of HCI and "social opposition" in information systems research.

HCI and impact of cognitive psychology

Because of the high visibility of HCI issues the importance of the 'interface' in the design of new systems and applications seems now been accepted. However, when we look at current IS design practice, it is still very difficult to see signs of pressure towards any kind of methodical or even conceptual influence coming from cognitive psychology, for example. The development of information system design methodologies (ISDMs) has been a programmatic attempt to collect practical and theoretical knowledge concerning information system design, they contain elaborate transformation procedures between the design domains, (for example: how to analyze reality -> produce an information model -> create an actual database, or: how to analyze information processing needs -> produce a process model -> proceed towards a structured code). One would expect to find methods for interface design here, too. A closer examination leads to disappointment, however. J. Iivari has analyzed the content of four prominent ISDMs (CIAM, ISAC, JSD and NIAM) and found that none of them has any methodical or conceptual tools for handling interface issues. "Still more strikingly, the total neglect of the area of user interface is really surprising" (Iivari 1989, p. 346). Apparently there has not been need enough to incorporate any interdiciplinarity for interface issues into the ISDMs.

Information Systems research and the question of "social"

Fifteen years ago the Information Systems research field looked rather well-organized: the necessity to deal in design with the user needs in a systematic fashion was clearly identified, at least in Scandinavia, and the first methods addressing the issue were already in practical use, e.g. (Lundeberg, Goldkuhl, & Nilsson, 1978). The number of system design methods was proliferating, and there was a certain sense of optimism that by comparing the strengths and weaknesses of different methods a convercenge towards a "standard" type would eventually occur. But that did not happen: instead, the whole 1980s in IS research has been characterized by a growing discussion and criticism against the prevailing way to understand the system to be

designed and the "normal" ways to study the use of computer applications, and the discussion is still going on. The discussion is characterized by Klein & Hirschheim in the following way:

> "It is possible, therefore, to speak of an IS orthodoxy, one where fundamental tenets are shared and form a general conception of how information systems can and should be developed. Recently, however, it is possible to note the emergence of some radically different approaches to ISD, ones which do not share the same paradigm, which possess an underlying philosophy that is quite different from the orthodoxy, and which challenge the basic assumptions, values and beliefs of the past." (Klein & Hirschheim, 1987), p. 275-276).

The major question, emphasised by several authors (Banville, 1991; Boland & Hirschheim, 1987; Lyytinen, 1986) is that the object of research (and finally design) should be understood as a social instead of a technical system. The meaning of "social system" is often explicated to be such as that used in sociological analysis, the main emphasis being thus in the social relations between people (Klein & Hirschheim, 1987; Lyytinen, 1986). The primacy of the "social system" aspect means that it should be the determining factor in information system design and, correspondingly, social concepts and methods should have a visible role in system design. However, despite the heat of the debate over ten years, practically no concepts or methods of social science in system design during the 1980s.

Why interdisciplinarity now?

We can now state the question of this paper. The interdisciplinary nature of system design has been recognized in principle very early, and there has been strong and visible research programmes in introducing interdisciplinarity in system design. Despite the longlasting effort – 15 years in the case of HCI and more than 10 years in the case of the "social turn" IS research – the headway made until the end of 1980s has not been significant.

How it is then explicable that interdisciplinarity has during the last few years suddenly gained much more serious consideration and respect also at practical level , especially in the form of applying ethnographical methods in design? Because ethnographic methods are not by any means new inventions, one is tempted to assume that the situation now possess some essentially different characteristics than ten years ago – something that makes interdisciplinarity much more necessary than earlier.

This paper studies a hypothesis that one reason for the change in attitude may lie in the way how the organization of work is changing. It assumes, that although interdisciplinary approach might have been beneficial also in designing systems for Tayloristically-bureaucratically oriented work, it was not crucial – but designing systems for emerging post-Tayloristic work from a technical viewpoint only will lead to serious problems. The paper suggests this a one reason for the groving popularity of interdisciplinary issues.

Change in the work

Systems for Tayloristic work

The core of the object of design and research in the Information Systems tradition has always been a system representing and delivering prespecified information about clearly-defined "Universe of Discourse" and automating well-defined information processing routines. Routine automation can be seen as a direct continuation of Tayloristic work organization, where the

worker has no need to understand the nature of the work but just follow given directions. The efficient use of a system in that case is simply the error-free execution of predetermined action sequences. No matter how clumsy the system or how bad the interface may be, the user sooner or later memorizes the actions needed and the error rate will eventually fall to an acceptable level.

Post-Fordist work organisation

There exists an extensive discussion on emerging new work organization forms in the work sociology, management and organizational literature. Organizational forms where workers possess more actorship and take the initiative and responsibility in cooperative settings, are presented as the major alternative to recent dominating practices.

It still seems to be difficult to evaluate on which conditions the new forms can be successful and it is perhaps too early to make any final judgement if there is a fundamental change going on - like some researchers suggest - or if the new work organization forms are but one more choice among several alternatives.

In any way it seems clear that there are situations where the new ways of organising work offer promising potential and we can expect that they will increase - either in the form of a radical redesign or a more subtle partial change in the existing way of work. In both cases the need of computer support also for the new features in work will be increasing:

> "Moreover, in a post-industrial society, adaptability is the characteristic of the mordern enterprise (private or public). Mass production with economy of scale is giving way to flexible adaptation to the market or community's needs, as the guiding principle of organizational strategy. Organizations are becoming flatter, project groups tending to replace permanent departemental structures; information systems must keep pace with these and other new organizational demends" (Stamper, Kerola, Lee, & Lyytinen, 1991), Introduction p. xi).

According to the literature, which are the characteristic features of the post-fordist work organization?

One of them is obviously the active, innovative and responsible role of workers: people have to take initiative, detect and solve problems, deal with unforeseen contingencies and work around breakdowns (Drucker, 1991; Seely Brown, 1991; Watson, 1980).

Another distinguishing feature is the strong emphasis on active cooperation in teams, groups, networks and like: (Drucker, 1992; Hughes, Randall, & Shapiro, 1991; Savage, 1990).

Also the changing environment of organizations and their dynamical and emergent features have been emphasised by many writers talking on the new forms of work organization, like (Hedberg, 1991; Schmidt, 1991; Stamper, et al., 1991)[1]

The direction towards "design oriented operation" is shared by many writers, (Drucker, 1992) wants that workers should see themselves as "executives", (Scott Morton, 1991) sees that they are changing from "doers" to "analyzers", (Howard, 1987) talks about "organizational refexivity" as an overall term pointing to learning about an organization and its possibilities and to knowing how to influence them.

The role of information technology in the new work organisation

The emergence of the new organization forms is largely connected in one way or another with information technology. Especially the networking capability, but also the information producing facilities are seen as crucial enablers for the new organization forms – altough

attempts to explicate this aspect further are rare. A picture is outlined where the new work organisation, multiskilled and active workers and advanced information technology become unseparably involved together.

The picture painted this far looks neat - too neat to be accurate. In reality nothing is so straightforward. Many of the issues and relationships are complicated and debated within their home disciplines. It is very plausible that there is no automatical development towards the new forms of work described above. Instead, they is one potential development path among many alternatives in an area where different forces and pressures are present and change dynamically. This is clearly expressed by e.g. (Hughes, et al., 1991) and (Orlikowski & Robey, 1991)

Anyway, information technology has an important role to fulfill and it's connection with the work itsekf is much more intime than before. Routine automation is hopelessly inadequate to deal with this kind of work. Hence the new work organization will need new kinds of applications, where the support for tool-like operation, understanding and communication can be realized, in the direction characterized by Ciborra & Schneider (1992)

“New systems for information creation rather than information management should possess the following properties:

1) They should facilitate the process of reinvention that any complex technological artifact undergoes when put to use.
2) They should not conceal the relations between routines embedded in systems and formative contexts. On the contrary, they should make those relations explicit and available for questioning.
3) They should function as media for enhancing coordination and communication within and across the teams. Problems and solutions shift all the time, and because systems are open and pasted together by nature, they should support loosely coupled forms of organization.
4) They should provide real-time feedback to users on their current organization of work and the emerging coordination and communication patterns
5) They should function as “systems for experts”, but not in the way implied by current conceptions of “expert systems”. That is, in addition to supporting or replacing the

---

1. Taken together, this sounds amazingly similar to the recommedations of the sociotechnical theory, formulated as early as the 1950s and 1960s. Julkunen argues that sociotechnics were in fact ahead of their time and the development of reality has only recently been catching up with their theories. She identifies five sociotechnical arguments which she believes being in fact now more actual and crucial for efficient operation of organizations than they were during the time when they were formulated (Julkunen, 1987), p. 251 : 1) The organisation of work is important and there exists a possibility for alternative organisational forms. 2) Managing "stochastic" technology needs self-steering at the basic level of the production organisation. 3) Coping with a "turbulent" environment has the same demand. 4) Self-steering needs internal integration. 5) The question of workplace democracy is not about power but about the maximisation of the steerability of an organisation.

It looks like the sociotechnical humanisation movement has earlier been more like experiments whose major aim has been to counteract the social prolems of production, like high absenteeism and worker's resistance, and the legitimation of the experiments has been based on the needs and motivation of a worker and on increasing democracy in the workplace. The postfordist work organisation is applied under different conditions and grounded in production economics. The rhetoric on humanisation and democracy has then been stripped away as unnecessary. Although the new work organisation forms approach the "humanisation" ideals, the personal experience of workers is not necessarily that of a humane and interesting work but a more demanding and stressing labouring.

knowledge-based routines of professionals and managers in specific domains of expertise, they should support people's capabilities for reflection and inquiry within the contexts in which they are embedded. They should help people build up, question, and modify practical knowledge according to the emergence and the shift of problematic situations and contexts" (Ciborra, 1993)(p. 286).

Implications in system design

What has to be taken into account in designing this kind of systems?In addition to the technical issues – and heavily influencing the way they can be applied – several important areas can be recognized:

Because of the situational character of the work and the necessity to make sense of them, and because workers belong to particular communities of practice, it is necessary to pay attention to issues of personal interpretation and cultural factors.

Because situations emerge in the course of a work process in an unforeseen way and it is thus impossible to plan and organize everything beforehand, it is necessary to pay attention to articulation aspects.

Because work is done by several active subjects who have to share viewpoints, negotiate, make decisions and coordinate their actions, it is necessary to pay attention to communication aspects.

Because work has to be continuously adapted to the changing environment, it is necessary to pay attention to aspects of learning, construction and reconstruction.

Because in the new work organization forms the scope of worker's actions is broadened, new skills needed and his or her responsibility increased, it is necessary to pay attention to the aspects of power, control, conflicts and emancipation.

These areas fall clearly outside the technical point of view but are nevertheless crucial in design. Hence the risen interest in interdisciplinarity – that seems to be the only way to go...

References

Banville, C. (1991): A Study of Legitimacy as a Social Dimension of Organizational Information Systems. In: H.-E. Nissen, H. K. Klein, & R. Hirschheim (Eds.): *Information System Research: Contemporary Approaches & Emergent Traditions. Proceedings of the IFIP TC8 Conference ISRA'90* (pp. 107-129). Amsterdam: North-Holland.

Boland, R. J., & Hirschheim, R. A. (Ed.). (1987): *Critical Issues in Information Systems Research*. Chichester, John Wiley & Sons.

Ciborra, C. (1993): *Teams, Markets and Systems. Business Innovation and Information Technology*. Cambridge, Cambridge Univ. Press.

Drucker, P. F. (1991): The New Productivity Challenge. *Harvard Business Review, 69*(6), 69-79.

Drucker, P. F. (1992): The New Society of Organizations. *Harvard Business Review, 70*(5), 95-104.

Hedberg, B. (1991): Imaginary Organizations. In: R. Stamper, P. Kerola, R. Lee, & K. Lyytinen (Eds.), *Collaborative Work, Social Communications and Information Systems. Proceedings of the IFIP TC 8 Working Conference COSCIS'91,* Amsterdam: Elsevier/ North-Holland.

Howard, R. (1987): Systems design and social responsibility: the political implications of "computer-supported cooperative work". *Office, Technology and People, 3*(2), 175-187.

Hughes, J., Randall, D., & Shapiro, D. (1991): CSCW: Discipline or Paradigm? A sociological perspective. In: L. J. Bannon, M. Robinson, & K. Schmidt (Eds.), *Proceedings of the 2nd ECSCW.* (pp. 309-323). Amsterdam: Kluwer.

Julkunen, R. (1987): *Työprosessi ja pitkät aallot.* Tampere: Vastapaino.

Klein, H., & Hirschheim, R. (1987): Social Change and the Future of Information Systems Development. In: R. Boland & R. Hirschheim (Eds.), *Critical Issues in Information Systems Research* (pp. 275-305). Chichester: John Wiley & Sons.

Lundeberg, M., Goldkuhl, G., & Nilsson, A. (1978): *Systemering.* Lund: Studentlitteratur.

Lyytinen, K. (1986): *Information Systems Development as Social Action: Framework and Critical Implications. (Dissertation).* Jyväskylä: University of Jyväskylä.

Orlikowski, W., & Robey, D. (1991): Information Technology and the Structuring of Organizations. *Information Systems Research, 2*(2), 143-169.

Savage, C. M. (1990): *Fifth Generation Management. Integrating Enterprises through Human Networking.* Digital Press.

Schmidt, K. (1991): Computer Support for Cooperative Work in Advanced Manufacturing. *Intl. Jrnl. Human Factors in Manufacturing, 1*(4, (October)), 303-320.

Scott Morton, M. (Ed.). (1991): *The Corporation of the 1990s: Information Technology and Organizational Transformation.* New York: Oxford Univ. Press.

Seely Brown, J. (1991): Research that reinvents the Corporation. *Harvard Business Review, 69*(1), 102-111.

Stamper, R., Kerola, P., Lee, R., & Lyytinen, K. (Ed.). (1991): *Collaborative Work, Social Communications and Information Systems. Proceedings of the IFIP TC 8 Working Conference COSCIS'91.* Amsterdam: North-Holland.

Watson, T. J. (1980): *Sociology, work and industry.* London: Routledge & Kegan Paul.

Software-Ergonomics – An Interdisciplinary Subfield of Informatics

Susanne Maass

What will informatics students do in their future jobs? What should we teach them in order to prepare them for their professional lives?

Technical systems are meant to enhance human life. They change our lives in many respects. As computer scientists we should not limit ourselves to the technical aspects of the artifacts we are creating and restrain our responsibilities to those aspects. We must see the whole picture, even if we cannot understand and control all of it.

In my view the education we provide at informatics departments of universities should not be technology only. We also need elements from the social sciences. The field of user interface design clearly shows the need for and the value of interdisciplinary work and education. For the design of usable and useful systems expertise in at least three areas is required:

1. Designers need to know the technical options they have. As their home discipline in most cases is informatics, they are rather well equipped with this kind of knowledge.
2. They must be aware of general human cognitive skills and their limitations. This is the field of cognitive psychology.
3. They must be able to assess people's work in its organisational context with respect to criteria for humane work. Occupational psychology deals with these aspects.

Of course we cannot expect every designer to be an expert in all of these areas, but they should at least have a basic understanding for them. In the best of all worlds, informaticians would work in design teams with experts for psychology and work design. In most cases, however, they will have to introduce these non-technical aspects in the design process themselves. In addition, designers will have to cooperate with prospective users of their systems. In order to be able to successfully communicate and negotiate standpoints with these users and with experts of other fields, designers must possess very good communicative skills: They must ask and listen carefully and they must present their own ideas clearly. (By the way - to me this seems to be the essence of interdisciplinary work in general.)

In Germany we recently developed a curriculum for the interdisciplinary area of human centered system design, called software-ergonomics (Software-Ergonomie). It is meant to be taught to students of informatics. In addition to the relevant technical subjects we propose to teach them the basics of cognitive and occupational psychology.

Students have to get an understanding for software development as a part of work and organisational design. They must learn to recognise work conditions and requirements and to design the functional division of labour between humans and computers accordingly. Task adequacy is one of the main criteria for the design of system functionality as seen at the user interface and system handling. So the users' work must be well understood in the first place.

In our curriculum the process of system development and introduction is explicitly addressed as something that has to be carefully organised in consideration of various interests and technical as well as non-technical requirements.

Of course informatics faculty will need some support from other disciplines in order to provide their students with software-ergonomics education. Joint seminars between computer

scientists and psychologists or social scientists, for example, can have an educational effect on both, students and teachers. On top of that, practical projects with outside partners, preferably with experienced design specialists from industry as mentors, are most motivating and instructive. In such projects students will learn that in real world system design there are many individual or organisational, social and non-technical factors that influence final design decisions, at times even more than technical considerations.

References

Maaß, S. (1993): Software-Ergonomie. Benutzer- und aufgabenorientierte Systemgestaltung. *Informatik-Spektrum 16*, 4, 1993, S. 191-205

Maaß, S., Ackermann, D., Dzida, W., Gorny, P., Oberquelle, H., Rödiger, K.-H., Rupietta, W., Streitz, N.A. (1993): Software-Ergonomie-Ausbildung in Informatik-Studiengängen bundesdeutscher Universitäten. *Informatik-Spektrum, 16*, 1, 1993, S. 25-30

Metaphorical Design

Kim Halskov Madsen

Most of the earlier research on computers and metaphors emphasized ease of learnability and ease of use. But currently there is a growing interest in addressing the role of metaphor in the *design process.*

A perspective on Metaphor

Pragmatic approaches acknowledge that in the context of real-world situations, metaphor inevitably involves incompleteness and mismatches and that the power of metaphor may be attributed to such disparities between the source and the target domain. Rather than using a structural or formal mapping definition of metaphor, a pragmatic approach emphasizes the use of metaphor as a particular kind of *"seeing as"* governed by previous situations and examples rather than by rules and fixed categories

A collection of cases

Case 1: A small command language

Although a metaphor may not be explicitly supported by the computer system, users often understand the system in metaphorical terms. An investigation of the language usage of employees at a Danish library reveals that they understand the structure of their computer system in terms of at least three different metaphors: the physical space metaphor, the conversation partner metaphor, and the organism metaphor.

Investigation of language usage was based on tape recordings of conversations with three employees who were asked to describe how they used the various computer applications at the library.

For instance identification of the physical space metaphor was based on the criterion that a logical or functional part of the system was referred to in terms of concepts normally used about a physical space: "then I could go *in* and make back-up copies"

Case 2: Links between documents

Erickson presents a design task where users can define links between parts of different computer documents so that when changes are made in one part the other parts are automatically changed. Candidate metaphors include the 'TV broadcasting metaphor", the link metaphor, and the pointer metaphor.

Case 3: An Automated Teller Machine

MacLean tell a story about how metaphor played an important role in the design of a bank Automated Teller Machine. In one case".. the designers had personal experience of a bagel store which handled its lengthy queues by having an employee work along the queue, explaining the choices available and helping fill out their order on a form. The customers would hand over their forms when they reached the counter, enabling their requests to be processed more speedily". The familiarity with the bagel store arrangement lead the designers to the innovative idea of having bank cards which the customers could pre-program while waiting in line.

Case 4: Production planning

A major Danish corporation's production planning was discussed in a workshop by using a metaphorical design approach. The participants in the workshop suggested many metaphors. Among other things, it was suggested to see production planning as: house cleaning, cooking, a soccer match, transportation, a power plant, cattle raising,

Case 5: Service at libraries

In a technology assessment project at the Danish research libraries, the impact of the use of computers on the service provided by the libraries was one of the key issues . In order to stimulate discussion about the impact of computers, the staff was challenged by three different metaphorical views of what a library is or could be. The three metaphors were "the warehouse", "the store" and "the meeting place". Each of the three metaphors provide a different account of what a library is, leading to different computer applications and eventually different kinds of service.

Guidelines

Guidelines derived from the cases are organized along the three main activities of metaphorical design.

Generating

Listen to how users understand their computer systems
Build on already existing metaphors.
Use predecessor artifacts as metaphors.
Note metaphors already implicit in the problem description
Look for real world events exhibiting key aspects .

Evaluating

Choose a metaphor with a rich structure
Evaluate the applicability of the structure
Choose a metaphor suitable to the audience
Choose metaphors with well understood literal meanings. .
Choose metaphors with a conceptual distance between the source and metaphorical meaning.
Do not necessarily explicitly incorporate the metaphor in the final design.

Developing
Elaborate the triggering concept..
Look for new meanings for the concept.
Restructure the perception of reality.
Elaborate assumptions.
Tell the metaphor's story.
Identify the unused part of the metaphor.
Generate conflicting accounts..

Characteristics

We can make several theoretical observation about characteristics and the nature of the role of metaphor in design.

Physical structure plays an important role.
Metaphor is an inherent part of everyday language.
Metaphors often originate from everyday experience.
Abstract concepts are understood in terms of concrete things.
Metaphors provides detailed and specific design options.
Metaphors may provide the basis for justifying design decisions.
A metaphor provides the user with a model of the system.
Seeing something as something else.
Provide a novel view of reality
Provide a shift in focus of attention.
Problem setting

The Technical = the Non-Technical

Eric Monteiro

It is fair to say that SD continues to wrestle with a more fruitful way to approach the question of interdisciplinarity in SD (ISD). To express this more bluntly: we are today well aware of the fact THAT IT has a number of non-technical aspects; what we know a lot less about is more precisely HOW this works.

The basic message, to give it straight away, is that the practice of SD seems to be depressingly insensitive to the high-flying theories produced by SD. The sophisticated theories of foundational nature coming out of a large number of disciplines (sociology, anthropology, psychology, organisational theory and philosophy) seem to be quite independent of the practice of SD. Developing such theoretical frameworks keeps a whole industry of researchers in SD in business but does not seem to improve the practice of SD significantly. The major problem, as I see it, is that the issues which these other disciplines focus on is very difficult to relate to the practice of SD. Within the practical world of SD these non-technical issues do not show up -- at least not in the form which sociology, philosophy etc. frame them. Two illustrations how ISD fails in this respect may be summed up by these equations: "Modelling = epistemology" and "SD = organisational development".

There is no lack of proclamations of the social, political, economical or psychological aspects of IT. Quite the contrary. To be unbearable blunt once again, the status of ISD by and large boils down to the following maxim: IT 'enables' certain actions/ patterns of behaviour and

it 'constrains' others. And this result seems to be relatively invariant to whether one uses structuration theory [Orlikowski 92, Walsham 93], hermeneutics [Boland et al. 88] or constructivism [Bijker 93] (or any of your own personal favourite conceptual framework for the non-technical) as a basis for the argument.

It thus seems to me to be a huge gap between, on the one hand, the technical computer system itself, and, on the other hand, this pool of non-technical, aggregated, macro-level concerns. The problem is not that ISD does not have a sufficiently large base of sociological, philosophical etc. theories to draw upon -- it has touched upon huge pile of theories. The problem is rather, as I see it, that there is so few accounts of the MECHANISMS, that is, how these aspects are played out or operate on a micro scale. If you will bear over with the deterministic language, I would, instead of settling for 'IT enables certain actions...', like to ask questions closer to this: WHICH actions were enabled/ constrained by WHAT design issue? Again, this should not be read as a pledge for technological determinism; it simply indicates in which direction, along the continuum ranging from social to technological determinism, I would like to see ISD move in.

These mechanisms, these things lying in the middle between the computer system itself and the aggregated effects, are what I believe should be the true subject matter of ISD. Personally, I see the field of Science and Technology Studies (STS) as one of the most promising approaches to ISD (see for instance, [Bijker et al. 92, Law 91]). Programmatically speaking, these mechanisms are neither technical nor non-technical. They are both. The real challenge to ISD is to develop the ability for systems developers to recognize and track such mechanisms at play, how they in one instance are perfectly technical issues but how they are translated into non-technical ones (and vice versa). There is thus no a priori distinction between the technical and the non-technical.

References

R. J. Boland Jr. and R. H. Greenberg: Metaphorical structuring of organizational ambiguity. In: L. R. Pondy and R.J Boland and H. Thomas (eds): Managing ambiguity and change. John Wiley, 1988

Wanda J. Orlikowski: The duality of technology: rethinking the concept of technology in organizations. Organizational studies, 3(3):398-427, 1992

Geoff Walsham: Interpreting information systems in organizations. John Wiley, 1993

Wiebe E. Bijker: Do not despair: there is life after constructivism. Science, Technology, & Human values, 18(1):113-138, 1993

W. E. Bijker and J. Law (eds): Shaping technology/ building society. MIT Press, 1992

J. Law (ed): A sociology of monsters. Essays on power, technology and domination. Routledge, 1991

Traditions for Information Systems Development –a Coward Simplification for Researchers, a Complication for Systems Developers, and a Hindrance for Interdisciplinary Work?

Torbjørn Naslund

I am an information systems development researcher. I am worried about the fragmentation of my research area.

Much research in information systems development is directed towards normative statements or theories for how to develop good computer artifacts. This is positive - we need good artifacts, and we need to know how to achieve them.

I see a difficult problem in the way we do this, however. There is a clear tendency for several isolated traditions. These traditions function as social paradigms, where researchers INSIDE each of these traditions share basic assumptions and objectives to such a degree that they are taken for granted, seldom are stated explicitly, and are not discussed. The other way round, very little debate is going on BETWEEN the different traditions; the traditions tend to isolate themselves. When discussions really occur, differences in terminology and the fact that assumptions and objectives are not made explicit may lead to misunderstandings and misinterpretations. Hence, researchers from other traditions than where oneself is situated may be seen as naive, ignorant, or difficult to understand.

Each tradition has taken its own foci for research, and claim that the research leads to gradually better recommendations for systems development. For a practitioner in systems development, this knowledge is very difficult to make use of. For a practitioner, researchers may be seen as biased advocates for single ideas which are fragmented and difficult to integrate with each other. Research results are offered systems developers as a "babble of many voices".

For interdisciplinary work, the situation becomes a hindrance. It is not only a question about how to bridge different disciplines, but also a question of which traditions inside information systems development to fraternize with, which are open to cooperation with other disciplines, and how well the unstated assumptions of each tradition fit assumptions and values of the other discipline.

The only viable solution I can see for the long run is to try to break the isolation between traditions, and engage in dialogue also over the borders of the traditions. This does not appear to be an easy task, however. For a researcher, it is much easier to publish and engage in dialogues with equals, than with people who behave as if they were naive, who are difficult to understand, and/or who challenge the researcher's own assumptions. Which different traditions can be found in information systems development, then - and what are their foci and basic assumptions? A very sketchy and tentative outline is the following list (where each tradition is described as an ideal type in order to highlight the differences):

1. The information systems tradition (IS). The main focus is on organisational impact and achievement of organisational objectives with the help of IT. The artifacts (called information systems) are assumed to be large and difficult to overview, but technically simple to construct. As a basis for construction, information flow (data flow), information structure and/or information use (in particular for decision making) must

be determined. Systems development starts from the identification of an organisational problem, which is investigated in a systematic way by the internal systems development organisation of the enterprise (sometimes assisted by external consultants). The IT solution to the problem can be derived from the investigation, if the investigation has been carefully performed. The technical construction of the information system will then be straightforward.

2. The software engineering tradition (SE). SE focuses on the technical construction of the computer artifact (software, software system). The development is performed by software engineers in order to fulfill the needs of a customer. The customer knows what is needed, even if he cannot state it in a way which is useful for the software engineer. Requirements elicitation is thus a difficult but important task, which must be performed before the software construction starts. The requirements should be specified in detail in a requirements specification. Important objectives to achieve in software construction are fulfilment of the stated requirements, as well as high technical quality and maintainability of the software. The construction process is seen as difficult to manage and coordinate - mainly since it is assumed that the project is very large, and requires a large number of programmers/software engineers working in parallel.
3. The participatory design tradition (PD). PD takes a stand against many of the assumptions made both in the IS and SE traditions. There are no definite needs which can be elicited in advance, neither is it possible or desirable to determine the qualities of a future computer artifact through analytical means. Instead, the artifact should be shaped in a gradual process in which prospective users and systems developers cooperate closely. Empirical experiments and users' opinions are taken very seriously, while project management, organisational objectives and the original plan for the artifact are given less attention.
4. The human computer interaction tradition (HCI). HCI stresses the cognitive fit of artifacts to the users' abilities - in particular when the users are discretional rather than regular users of the artifact. The artifact is often assumed to be developed as a product, to be sold on the market. Consequently, the users are seldom identifiable persons, but rather hypothetical, abstract descriptions of people. The interface - which determines the behaviour and appearance of the artifact - should be designed early in development process, by a special interface design team. The rest of the system should be developed by others, who should adapt their work to the result of the interface team. Since it is often assumed that the product is a mass market product, a good interface design is seen as a good investment, even if it is a cumbersome process to achieve this.
5. The knowledge engineering tradition (KE). In KE, there is a strict focus on the statements about the usage domain (called knowledge) which are built into the system. It is seen as a difficult but challenging task to acquire this knowledge trough investigation of written material and elicitation of knowledge from skilled persons (called experts). What role the system should have in its use setting is typically not emphasized.

We can perhaps also talk about an emerging tradition in CSCW, in which communication between people is particularly focused. Currently, it appears rather to be a situation that CSCW is a meeting place for researchers from several different traditions, and from several different disciplines, rather than being a tradition of its own. CSCW may thus function as an important meeting place between several traditions for systems development. The obvious risk, however,

is that CSCW emerge as an arena where different traditions defend their own assumptions rather than try to understand each other and make use of each others contributions.

I claim, finally, that each of these traditions have important knowledge to provide to information systems development practice and research, but that incompatibility and implicitness of assumptions, use of tradition-specific language, and the low degree of exchange between the traditions effectively block much of the positive impact this knowledge could have.

Dialogical Software Design

Jürgen Pasch

Software developement is not merely a mathematical or technological challenge, but a *complex social process*, in which the kind of communication and cooperative, creative interaction of the participants determine the quality of the collaboratively developed product. Qualified design is not primarily tied to given guidelines, but is guided by insights emerging in the design process and by the quest for quality shared by all participants. Practical experiences in system design, as well as my systematic field study reveal that the *quality of the social process* is of primary importance [1], [2].

Design first and foremost requires experience, intuition, imagination, and common sense. The designer orients himself by his tried and tested patterns of architecture.

In the design situation the participants carry on an *argumentative dialogue*. Concepts or models are suggested, brought into question, assessed and evaluated, and counter proposals are made. The Norwegian sociologist Stein Bråten demonstrated the role of models in communication processes [3].

If dialogue partner A possesses a sound model of the domain under consideration he is the *model–strong* actor. The *model–weak* partner B has no elements of this domain that are not a subset of A's model. With his model, A has pre–defined the domain under consideration. Bråten calls this situation *asymmetric dialogue*.

For clarification an example: A work group should design a program system. A group member brings a prepared design — the model — to the project meeting. The other group members have not worked out any definite ideas because they expected to draw up the design collaboratively. The model–weak members of the group find that they must examine the pre–defined model, which they must first understand. This understanding means the adoption of the model. All further argumentation for or against the model can only be expressed in terms of the model with the result that the model becomes more established.

Bråten summarizes this adaption to the model in two theses: If B *wants to maintain the dialogue* he is forced to express himself in terms of A's model, which he must first understand. He is therefore dependent on A. Ironically, the more successfull his attempt to strengthen his weak position through his understanding of A' s model, the more he comes under A's control. A now has ultimate power through his ability to a) *simulate* and declare his model as sound and b) to *simulate B's simulations*, which B needs to understand the model.

There is a further dilemma: If the dialogue partners have *a number of models, which have little or nothing in common* communication and participation are impossible.

When the dialogue partners *intersubjectively cross their perspectives*, this is a *symmetric dialogue*. By perspective I mean a class of related views on relevant aspects of an area on concern from a common view–point. In this type of dialogue statements such as „I understand what you mean“ or „I am not of your opinion“ are possible. Only a division among the actors of the domain of concern in disjunctive subsets make these statements possible. In symmetric dialogues the developers gain a shared understanding, which enables them to act collaboratively or individually as defined by the group. While perspective is always implied in our thinking, we can attempt to make them explicit, allow them to interact and gain deeper insight from their interrelation. Thus, I see in multiperspectivity a prerequisite for dialogical design.

The (technical) quality of program systems depends on the quality of their design and on the accurate realization of this design as an implementation.

A desirable system design is distinguished by straightforward orientation towards the design achieved at any given time, i.e. the design decisions are characterized by coherence, conceptual integrity and — finally — by their completeness with respect to the task assigned.

If this standard of quality is to be met even after a module–oriented division of labour, then all subsequent module–local design decisions and revisions of the system design must satisfy the conceptual integrity. Ideally, this requires *a holistic understanding of the system design* which is best acquired by participation in the design process itself.

This emerges when the program system is dialogically designed through the intersubjective crossing of perspectives and when the group comes to an understanding through a common perspective in respect to the theory of the program system. This common perspective enables each member of the group to change, expand and reconstruct the program — even after its completion according to the theory. *This quality is impossible without the developers*. The overall understanding obtained during dialogical design cannot be disseminated in documents. It is therefore clearly preferable that the same people participate in the design and realization activities.

I collected data on the effect of collaborative work on designing the architecture of software and its implementation from 25 student groups. The two main findings of my field–study with respect to „symmetric–dialogue groups“ were: These groups negotiated a *consensual definition of the situation* of the outset. During the design process they developed *techniques of mutual considerations and contradictions*. This is a group–specific ever–recurring dialogue pattern, which I call *ostinato*. The musical term ostinato means a persistant recurring bass. In dialogical design it means a persistant recurring dialogue pattern.

The definition of the situation entails intersubjectively valid group consensus on working practices, group meetings, roles, thought–, speech–, and participating schemata, acceptable and unacceptable behavior and the ostinato to be developed. I observed *the definition of the situation is of primary importance* for dialogical design.

The rivalry generated by mutual contradiction proved to be a positiv and vital force in the group dynamic. Suggested ideas are exhaustively discussed, further developed or refuted according to their usefulness. It is characteristic to these groups that patterns of behaviour emerged during the process, which ultimetely lead to understanding as long as the definition of the situation is valid. Vehement situations are considered as „normal“ by the students.

A strong identification and a holistic understanding of the designed system of the group members is striking. *These groups embody their programs.*

Groups operating with asymmetric dialogue arrive at *pathological types* of system design such as the following:

Design by Dominance. A person or a group of persons monopolizes the design process. The dominant party views the others' lack of understanding or their passive position as incompetence. To avoid conflicts between the active and the passive party, a few modules emerge, which — in the eyes of the dominant side relate to the implementing abilities „of the rest". The participants' roles, and thus the latent group conflict, are frozen until the end of the project, and they *shape* the product structure. Non–participants have a hard time in grasping the inherent logic of the product structure.

*Design by Patchwork.*The group members or subgroups try to avoid cooperation. They *disperse* the „design" as soon as possible into as many subsystems as there are parties. These subsystems, taken by themselves, may be well designed. But the system as a whole is characterized by redundancy of algorithms and data objects. Moreover the interfaces between the subssytems are awkward, reflecting the lack of cooperation. Good cooperative work is more than the total of the work of each member of the group.

Social processes cannot be formalized or controlled but influenced. Conventional software development methods for design have their place, but need to be tailored to the needs of the communicative processes at hand, so as to show multiperspectivity. Methods do not determine the quality of software products, but people involved in the design processes allow quality to emerge. Fundamental prerequisities for being able to carry out cooperative work are human abilities in dialogue, mutual acceptance and an organizational setting, which enables the autonomy of the design group.

References

[1] Pasch, J.: *Software-Entwicklung im Team. Mehr Qualität durch das dialogische Prinzip bei der Projektarbeit.* Springer-Verlag: Berlin, Heidelberg, New York, 1994.

[2] Pasch, J.: Dialogical Software Design. In: Bullinger, H.J. (Hrsg.): *Proceedings of the 4th International Conference on Human-Computer Interaction.* Stuttgart, September 1991. Elsevier: Amsterdam, London, New York, Tokyo, 1991; S. 556 - 560.

[3] Bråten, S.: Asymmetric Discourse and Cognitive Autonomy: Resolving Model Monopoly through boundary shifts. In: Pedretti, A. (Hrsg.): *Problems of Levels and Boundaries;* Princelet Editions: London Zürich, 1983; S. 7-28.

A Technique to Help Overcome Communication Barriers

G. Blain, N. Revault, J-F. Perrot, H. Sahraoui

We feel that one of the research trends that we are pursuing could provide an interesting tool in cooperative work. The idea is to see the computer as a help toward a better understanding between people working on a joint project.

Our basic postulate is that having to formulate one's thoughts through a computer-interpreted medium helps in clarifying them, in removing unexpressed assumptions etc. This is the cornerstone of the so-called "metamodelization" technique we are developing at LAFORIA, which I shall briefly outline here.

Specifically, we are exploring a technique for bridging the gap between the user's and the implementor's points of view in designing a software application. The software tool that implements this technique relies heavily on OO technology and on rule based programming. It borrows part of its inspiration from the AI subfield of Knowledge Acquisition. One of its applications

The basis of our approach is to provide two different models of the application, one on the user's side (call it UM), the other on the implementor's side (IM), together with a transformation mechanism (TM) that maps the first onto the second. Model UM is supposed to be easily understandable by the user - actually, it should be produced by the user herself - whereas model IM ought to be readily executable by the computer, that is, it should be directly translatable into some executable formalism. Transformation TM, of course, should operate in a way that ensures adequacy between the meaning of model UM as intended by the user and that of model IM as evidenced by the execution mechanism.

A typical example is the design of a Relational Database scheme. UM will be some sort of semantic network, whereas IM will be expressed in a SQL-like formalism. TM will embody the expertise of th DB designer.

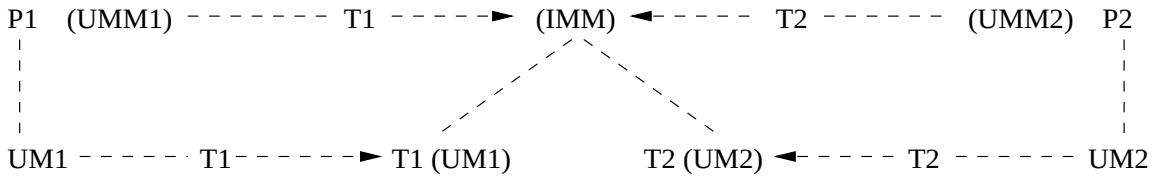
Now the two models are expressed in different formalisms. The key idea behind our method is that those formalisms do not need to be fixed, but that instead each one must be defined specifically for the application at hand. Following standard object-oriented philosophy, such a formalism is reified into a (Smalltalk) object which we call a metamodel. This name is justified by the nature of the link between the model and its metamodel, which parallels the link between an instance and its class, and a class and its meta-class, in OO Programming. Before creating our two models UM and IM, we must build the two metamodels UMM and IMM. In our example, UMM will specify the sort of semantic network that we use and IMM will describe our pre-SQL notation for RDB schemes.

For a UM-to-IM translation between models to be possible, there must be some sort of relationship between the metamodels UMM and IMM. This very relationship is materialized by the set of rules that define transformation TM. If the correspondence between UMM and IMM is close, the rules will be simple, if not they will need the elicitation of a good deal of knowledge, as is the case with the database example. The complexity of the rule base somehow measures the distance between UMM and IMM.

One way to use the technique is to define an interpretable IMM that is close enough to the UMM so as to preserve some of its graphic properties and to endow it with an mechanism that visualizes the interpretation process. By running the process on several examples the user can thus "verify" that the behavior of her model actually conforms to her expectations. This is a

crude but efficient way of testing specifications. Our team designed such a piece of software that is now in use in an insurance company.

We propose here to apply this approach to interhuman communication : if two persons P1 and P2 are to communicate at all, there must be some correspondence between the "languages" they use (i.e. the metamodels UMM1 and UMM2). This correspondence should be exhibited by one common interpretable IMM and two transformation rules T1 : UMM1 -> IMM and T2 : UMM2 -> IMM. Suppose that what P1 has in mind is described by a model UM1 (in formalism UMM1) - resp P2, UM2, UMM2. The transformed models T1(UM1) and T2(UM2) - both in IMM - are certainly different, but some comparison of their meaning can be obtained by concurrently visualizing their interpretation.



References

Revault, Sahraoui, Blain, Perrot : A Metamodelling technique : The MetaGen System. In: *TOOLS-Europe '95*.

Living the Gap

Jörg Pflüger

Those who have no idea that it is possible to err cannot
learn anything else than know-how.
Gregory Bateson

The interdisciplinary problem in computer science represents a modern version of the old dichotomy of explaining and understanding. Any established piece of software (or software system) can be considered as an "explanation" of the problem which it undertakes to solve. It changes and fixes systematically the task in question and there upon people's further experience - it "constructs reality". The social environment of formal systems has to be understood as a transcending difference. A "hermeneutic view" is necessary to reconstruct the parts of reality lost in formal reduction. Therefore, the above mentioned dichotomy addresses essentially a problem of "languages". One has to deal with the incompatibility of the discrete, unambiguous language of formal artifacts with the informal concepts of everyday life, social relations, conflicts, and power structure. That means, interdisciplinarity is not so much a problem inter disciplines than between different views, concepts, attitudes, approaches, and methods.

In my opinion computer science will only survive as a discipline in its own right if it attempts to integrate these different languages; that is, we have to educate people who are able to stand the tension between the irreducible semantic levels of formal representations and their informal counterparts. Mediating heterogeneous ways of thinking seems to me more important than bringing together results and theories from different disciplines. This does not lead to a productive way of "bridging the gap" (in which I don't believe anyway), assigning the social sciences in connection with computer science an essentially "negative", critical role:

Social scientists should teach software engineers to question what they are doing - not to take too much for granted, neither with respect to their own concepts nor what they know about the problem in question. They should be trained to acquire adequate knowledge by participatory observation and by talking to their clients. They have to understand why the users complain and what has been lost after the reduction of complicated processes to formal artifacts.

Referring to possible changes of a curriculum in computer science the integration of social aspects has to concentrate on a confrontation of methods rather than on the additive presentation of empirical facts. We have to teach a "will to know", how to understand ambivalent relations which reflect conflicts and antagonisms, and what it means to err. Stretching the argument a little further I would say: A computer scientist who could interpret poems and judge moral dilemmata would be a better computer scientist.

Some Remarks about Formal and Informal Specification Methods in the Context of Software Development

Matthias Rauterberg

Starting point for our reflections

Analysis of current software development processes brings to light a series of weaknesses and problems, the sources of which lie in the theoretical concepts applied, the traditional procedures followed (especially project management) as well as in the use of inadequate formal design methodologies. This chapter contains an ample store of proposed solutions based on current practice in software development. These point to the significance of a domain specific formalism. Analysis of actual software development processes shows that there are three essential barriers: the specification barrier, the communication barrier and the optimisation barrier (Rauterberg & Strohm 1992).

Speaking quite generally, one of the most important problems lies in coming to a shared understanding by all the affected goups of the component of the worksystem to be automated (Naur 1985) - that is, to find the answers to the questions of "if", "where" and "how" for the planned implementation of technology, to which a shared commitment can be reached. This involves, in particular, determining all the characteristics of the work system that are to be planned anew (Rauterberg 1993). Every work system comprises a social and a technical subsystem. An optimal total system must integrate both simultaneously. In order to arrive at the optimal design for the total working system, it is of paramount importance to regard the social subsystem as a system in its own right, endowed with its own specific characteristics and conditions, and a system to be optimised when coupled with the technical subsytem.

Barriers in the framework of traditional software development

The "specification barrier" is an immediate problem even at a cursory glance. How can the software developer ascertain that the client is able to specify the requirements for the subsystem to be developed in a complete and accurate way which will not be modified while the project is being carried out? The more formal and detailed the medium used by the client to formulate requirements, the easier it is for the software developer to incorporate these into an appropriate software system. But this presumes that the client has command of a certain measure of expertise. However, the client is not prepared to acquire this - or perhaps is in part not in a position to do so - before the beginning of the software development process. It is therefore necessary to find and implement other ways and means, using from informal through semi-formal to formal specification methods.

It would be a grave error with dire consequences to assume that clients - usually people from the middle and upper echelons of management - are able to provide pertinent and adequate information on all requirements for an interactive software system. As a result, the following different perspectives must be taken into consideration in the analysis and specification phases.

The "communications barrier" between applier, user and end-user on the one hand and the software developer on the other is essentially due to the fact that "technical intelligence" is only inadequately imbedded in the social, historical and political contexts of technological development. Communication between those involved in the development process can allow

non-technical facts to slip through the conceptual net of specialised technical language, which therefore restricts the social character of the technology to the functional and instrumental.

The application-oriented jargon of the user flounders on the technical jargon of the developer. This "gap" can only be bridged to a limited extent by purely linguistic means, because the fact that their respective semantics are conceptually bound make the ideas applied insufficiently precise. Overcoming this fuzziness requires creating jointly experienced, perceptually shared contexts. Beyond verbal communication, visual means are the ones best suited to this purpose. The stronger the perceptual experience one has of the semantic context of the other, the easier it is to overcome the communications barrier.

At its best, software development is a procedure for optimally designing a product with interactive properties for supporting the performance of work tasks. Because computer science has accumulated quite a treasure trove of very broadly applicable algorithms, software development is increasingly focussing attention on those facets of application-oriented software which are unamenable to algorithmic treatment. While the purely technical aspects of a software product are best dealt with by optimisation procedures attuned to a technical context, the non-technical context of the application environment aimed at requires the implementation of optimisation procedures of a different nature.

It would be false indeed to expect that at the outset of a larger reorganisation of a work system any single group of persons could have a complete, exact and comprehensive view of the ideal for the work system to be set up. Only during the analysis, evaluation and planning processes are the people involvable to develop an increasingly clear picture of what it is that they are really striving for. This is basically why the requirements of the applier seem to "change" - they do not really change but simply become concrete within the anticipated boundary constraints. This process of crystallisation should be allowed to unfold as completely, as pertinently and - from a global perspective - as inexpensively as possible. Completeness can be reached by ensuring that each affected group is involved at least through representatives. Iterative, interactive progress makes the ideal concept increasingly concrete. There are methods available for supporting the process of communication which ensure efficient progress (Nielsen 1990) (Nielsen 1993).

First steps to implement technology

The analysis phase is frequently the one most neglected. This is essentially due to the fact that methods and techniques need to be used primarily the way occupational and organisational sciences have developed and applied them (e.g., Maculay et al 1990). Inordinately high costs incur from the troubleshooting required because the analysis was less than optimal (Rauterberg & Strohm 1992). The time has come to engage occupational and organisational consultants at the analysis stage who have been especially trained for software development!

Once the analysis of the work system to be optimised has been completed, the next stage is to mould the results obtained into implementable form. Methods of specification with high communicative value are recommended here. Sufficient empirical evidence has accumulated by now to show that task and user oriented procedures in software development not only bring noticeable savings in costs, but also significantly improve the software produced (Karat 1990). How then, can the both barriers mentioned above be overcome?

The first thing is to determine "if" and "where" it makes sense to employ technology. "Although the view is still widely held that it is possible to use technology to eliminate the deficiencies of an organisation without questioning the structures of the organisation as a whole,

the conclusion is nevertheless usually a false one" (Klotz 1991). The intended division of functions between man and machine is decided during the specification of the tool interface. The tasks which remain in human hands must have the following characteristics (Volpert 1987) (Ulich et al 1991):

1. sufficient freedom of action and decision-making; 2. adequate time available; 3. sufficient physical activity; 4. concrete contact with material and social conditions at the workplace activities; 5. actual use of a variety of the senses; 6. opportunities for variety when executing tasks; 7. task related communication and immediate interpersonal contact.

Once those concerned are sufficiently clear about which functions are amenable to automation, the next step which should be taken is to test the screen layout on the end-users with hand-drawn sketches (the extremely inexpensive "pen and paper" method). If the range of templates is very large, then a graphics data bank can be used to manage the templates produced on a graphics editor. The use of prototyping tools is frequently inadvisable, because tool-specific presentation offers a too restrictive range of possibilities. The effect of the structuring measures taken can be assessed with the help of discussion with the end-users, or by means of checklists.

The use of prototypes to illustrate the dynamical and interactive aspects of the tools being developed is indispensable for specifying the dialogue interface. However, prototypes should only be used very purposefully and selectively to clarify special aspects of the specification - not indiscriminately. Otherwise there looms the inescapable danger of investing too much in the production and maintenance of "display goods". A very efficient and inexpensive variation is provided by simulation studies, for example, with the use of hand prepared transparencies, cards, etc. which appear before the user in response to the action taken (Karat 1990).

Simple and fast techniques for involving users include discussion groups with various communication aids (metaplan, layout sketches, "screen-dumps", scenarios, etc.), questionnaires for determining the attitudes, opinions and requirements of the users, the "walk-through" technique for systematically clarifying all possible work steps, as well as targeted interviews aimed at a concrete analysis of the work environment (Grudin, Ehrlich & Shriner 1987) (Macaulay et al 1990) (Nielsen 1993). Very sound simulation methods (e.g. scenarios, "Wizard of Oz" studies) are available for developing completely new systems without requiring any special hardware or software. Nielsen (1993) presents a summary of techniques for the analysis and evaluation of interactive computer systems.

Conclusion

One of the principal problems of traditional software development lies in the fact that those who have been primarily involved in software development to date have not been willing to recognise that software development is, in most cases, mainly a question of occupational and/ or organisational planning. Were software development to be approached from such a perspective, it would be planned from the beginning to engage experts in occupational and organisational planning in the process of software design. This would require interdisciplinary cooperation between occupational and organisational experts on the one hand and software development experts on the other. The extensive qualification required in each of these fields makes it virtually impossible to dispense with such interdisciplinary cooperation. We must start learning to jointly plan technology, organisation and the application of human qualification.

References

Karat C.-M., 1990: Cost-Benefit Analysis of Iterative Usability Testing. In: Diaper, D. et al. (ed.) *Human-Computer Interaction - INTERACT '90*. Amsterdam: Elsevier Science. 351-356DIAPER D

Klotz, U., 1991: Die zweite Ära der Informationstechnik. *Harvard Manager 13*(2): pp. 101-112

Naur, P., 1985: Programming as Theory Building. *Microprocessing and Microprogramming 15*: pp. 253-261

Nielsen, J., 1990: Big paybacks from 'discount' usability engineering. *IEEE Software 7*(3): pp. 107-108

Nielsen, J., 1993: *Usability Engineering*. London: Academic Press.

Rauterberg, M., 1993: Anforderungen an die Prozessgestaltung der Softwareentwicklung. In: W. Coy, P. Gorny, I. Kopp & C. Skarpelis (Hrsg.): *Menschengerechte Software als Wettbewerbsfaktor. (German Chapter of the ACM Berichte, Band 40);* Stuttgart: Teubner. pp. 592-599.

Rauterberg, M., & O. Strohm, 1992: Work Organization and Software Development. In: P. Elzer & V. Haase (eds.): *Proceedings of 4th IFAC/IFIP Workshop on "Experience with the Management of Software Projects". Annual Review of Automatic Programming 16*(2): pp. 121-128

Ulich, E., M. Rauterberg, T. Moll, T. Greutmann, & O. Strohm, 1991: Task Orientation and User-Oriented Dialogue Design. *International Journal of Human Computer Interaction 3*(2): pp. 117-144

Volpert, W., 1987: Kontrastive Analyse des Verhältnisses von Mensch und Rechner als Grundlage des System-Designs. *Zeitschrift fr Arbeitswissenschaft 41*: pp. 147-152

Social Dimensions

***Mike Robinson*[1]**

This paper to some extent reflects the dilemmas posed in the Motivation to the Dagstuhl Seminar — namely that software engineering tends "to focus on structural and technical properties of programs rather than on process aspects". There is a ubiquitous tendency for the software tail to wag the social dog. The process is repeated in exaggerated form with the appearance of Virtual Reality (VR) and Cyberspace. The possibility of three dimensional representation was actualised. This was mainly a technical achievement. With helmet and gloves (or even with a simple "spaceball") it became possible to climb in with the images, to push them around like surreal footballs on the moon. The technical achievement was breathtaking. Quite what moonball has to do with CSCW or with work in general still has to be explained— although there is no shortage of pundits projecting the transformation of life, the universe, and everything (Benedikt, 1992). And with a multi- and hyper-media supporting INTERNET, with computer power, accessibility, and connectivity continuing on a unique and impossible exponential, there is an intuitive feeling they may be right. Social control, social specification of the desiderata of cyberspace seem illusory. Social science once again trots along meekly behind the turbulent frontiersmanship, the revolutionary interventions of the financially and politically backed visions of the technical. In general, it is left to find a place rather than specify the space.....

The term social dimensions arose as a complement to "spatial dimensions" in VR within the COMIC Project. Spatial dimensions both describe the nature of the space provided by VR, and the possibilities of movement within it. Space is uniform: i.e. *any* point has a position relative to any other point, and *any* object has position, height, width, breadth, and may have properties such as colour, texture, movement, momentum, spin, etc. Measurement is always in the same units, whatever the nature of the object, wherever it may be. Thus Cyberspace is considered to be "a place", *one* place. It is a place where people can meet. It is a place in the same way that our planet is a place. Any position on it can be reached and calculated from any other place. The notions of "virtual rooms", "virtual cities" etc. re-inforce this notion that cyberspace is a place. It is a natural metaphor.

Social dimensions was first introduced as a term that mirrored the physical dimensionality of cyberspace. The idea was to allow different sorts of manipulation and movement that would be informed by metaphors from work practice rather than from physicalist analogies. For instance, sometimes it is nice to be open for casual meetings with others, sometimes it is not. Sometimes it is important to be "peripherally aware" of what colleagues are doing, sometimes it is not. These possibilities were envisaged in a "see" metaphor on which it was possible to move between settings with a spaceball. In "infra-red" extreme the user would see only "warm" objects (people); in "white light" people and most objects are visible; in "ultra violet" only objects are visible; at the "x-ray" extreme it is possible to "see into" objects (their code, for instance). All gradations between settings are possible, thus supporting people in achieving their own most useful visualisations. This first example of a "social dimension" was very close to a

1.With thanks to the ESPRIT COMIC Basic Research Project for supporting this work.

physical dimension — but included the notion of “tailoring”, or the person using the system configuring it personally, differently from others.

A workshop was held earlier this year in Aarhus to discuss “social dimensions”. (Robinson, 1994b). The main conclusions that emerged were: a shifting but ever present division between structural and environmental features that were/ were not under the control of people using the system; the importance of uneven infrastructures and their social consequences. The best example of the latter is the (Star and Ruhleder, 1994) in their study of a distributed, heterogeneous scientific community. An example directly from VR is provided by (Benford et al., 1994)who discuss problems of interaction between those with two- and three-dimensional access to VR systems.

Another spontaneous distinction made by most people is between local and wider contexts, accompanied by a desire to have the ability to treat them differently in terms of access, privacy, etc. (Robinson, 1994a). As before, this demands that VR systems are tailorable. To some extent this is provided in Cyberspace prototypes by the ability to have “meeting rooms”, “conference tables”, etc. which give a local, private(ish) context.

(Nardi, 1993) advances a more radical thesis on tailorability. She argues that, ultimately, it is neither a possible competence nor a feasible activity for systems designers to try and provide/ structure applications for “domain experts” — people with a deep knowledge of an activity or subject who wish to support themselves with computers. Instead designers should content themselves with providing tools, with accessible technical environments that “domain experts” can utilise to structure their own environments according to their own needs and aesthetics. Nardi’s viewpoint seems admirably suited to VR, and can be seen in the work of (Mariani et al., 1994) on VR-VIBE, MASSIVE, etc. In these systems, users are able to set and manipulate their own mappings between properties of objects and spatial and other dimensions in Cyberspace. In effect, “domain experts” are able to create and inhabit their own worlds.

At this point we find that cyberspace has broken its own boundaries. Instead of a single (even if large) chrysalis of “place”, a universe of many, possibly infinitely many worlds has butterflied. Different mappings of the properties of objects-in-the-world onto spatial and coloured dimensions of virtual reality give different worlds. Worlds that reflect orthogonal comprehensions. Worlds between which movement and communication is extremely problematic. Indexicality, the root of mutual comprehension and conversation, fails between such worlds — which not only have their own syntax and semantics, but their own forms of life, their own physics. Even the taken for granted distinctions between subject and object, between objectivity and subjectivity fail to hold, fail to have any traction between such worlds.

In this multi-worlded Cyberspace we have achieved something very like normality, like everyday experience, like the world as we know it. Unbounded; contradictory; largely unexpected except in small, safe, well-known corners. Comprehending usage is rather like asking what a continent is for — more a category mistake than a sensible question. Crafting the world, and cyberspaces that are part of it, is irreductably an interdisciplinary exercise to say the least! The dominance of the technical is an illusion following from a “waterfall conception” of design and implementation preceding use. All systems are redesigned in use. And from this perspective use precedes design. The problem for interdisciplinarity is not the precedence of the technical, not the absence of some discipline or other, but — like VR, like all other human efforts — how to live with, and make the best of incommensurables.

References

Benedikt, Michael. 1992: *Cyberspace, First Steps*. Cambridge, MA: MIT Press.

Benford, Steve, Chris Greenhalgh, Lennart Fahlén, and John Bowers. 1994: *Embodiment.* COMIC Report. University of Lancaster, 1994.

Mariani, John, Tom Rodden, Andy Colebourne, Steve Benford, Adrian Bullock, and Dave Snowdon. 1994: *Populated Information Terrains*. COMIC Report: Lancaster University, 1994.

Nardi, Bonnie, A. 1993. *A Small Matter of Programming: Perspectives on End User Computing*. Cambridge, MA: MIT Press.

Robinson, Mike. 1994a: *Finger Tips: Summary of an informal survey of uses of UNIX "finger", with implications for new "fingers" and related event distribution mechanisms*. GMD (FIT-CSCW), 1994a.

Robinson, Mike. 1994b: *Supporting Social Dimensions In Large Information Spaces: Report On A Joint Comic/Eurocode Workshop, Ärhus, 10 & 11 May, 1994*. COMIC SF 4.6, 1994b.

Star, Susan, Leigh and Karen Ruhleder. 1994. Steps towards an Ecology of Infrastructure. In *CSCW '94, Chapel Hill, N. Carolina, USA*. ACM.

Empirical Analysis of Software Design Processes: One Approach to Interdisciplinarity

Sabine Sonnentag

At the Dagstuhl seminar various approaches to interdisciplinary design have been discussed. One possibility and necessity for interdisciplinary work between computer science/informatics and psychology is the empirical analysis of work processes in software design and software development. With this research actual work processes in the field can be described. It can be evaluated to what extent 'new' approaches to design, such as user participation, and methods are applied and what factors might inhibit their successful use. This research can reveal major problem areas both at the indicidual and the team level. For a summary of this research cf. Brodbeck & Frese (1994) and Frese & Hesse (1993).

More cognitive-oriented research at the individual level showed that software design is often not performed in the way methods developed in computer science prescribe that software design should be. This can be drawn from the literature on opportunistic vs. top-down behavior (e.g., Guindon, 1990) and from our work on planning processes in software design (Sonnentag & Frese, 1994).

Therefore, the mere recommendation of 'better' design approaches, methods, and tools is relatively useless unless we can not be sure that the prerequisites of their successful application - in the cognitive, motivational, social, and organizational area - are given.

References

Brodbeck, F.C. & Frese, M. (Eds.)(1994): *Produktivität und Qualität in Software-Projekten. Psychologische Analyse und Optimierung von Arbeitsprozessen in der Software-Entwicklung.* Muenchen: Oldenbourg.

Frese, M. & Hesse, W. (1993): The work situation in software development. Results of an empirical study. *Software Engineering Notes, 18* (3), pp. 65-72.

Guindon, R. (1990): Designing the design process: exploiting opportunistic thoughts. *Human-Computer Interaction, 5,* pp. 305-344.

Sonnentag, S. & Frese, M. (1994): Planning Processes in Software Designers. *Paper presented at the Workshop 'Integrating cognitive and organizational approaches to the study of computer-based systems' of the European Association of Cognitive Ergonomics,* 09.09.1994, Bonn.

Evaluation of Computerized Work
Recent Problems in Cognitive Ergonomics

Chris Stary

When it comes to the evaluation of computerized workplaces traditional methodologies of cognitive ergnonomics are not suitable. The main reason for that difficulty is their orientation towards software features instead of the orientation towards tasks and users. However, the EC-Directive for Human-Computer Interaction 1990 requires an evaluation against task and user appropriateness, transparency, flexibility, and control for each interactive workplace in a company. Since there exists no proper instrument for checking these criteria, an interdisciplinary approach for the development of such an instrument involving researchers from computer science, cognitive psychology, work psychology, and organizational theory has been started. The crucial issues in development are the clarification of ambiguous terms and notions, the scientific interpretation of the EC-Directive, and the reliability and validity of the methodology, integrating knowledge from the mentioned disciplines.

Design Methods and their Use – A Question of Rationality Resonance?

Erik Stolterman

The skepticism and reluctance of practitioners to use system design methods is well-known and by some considered as a problem. Why is it that practitioners seems to be so unwilling to use methods which are developed by highly qualified and experienced practitioners and researchers? Is there anything wrong with the methods or maybe with the practitioners? Or is it anything wrong with the basic idea of methods?

I am working with the above mentioned phenomena which in this context will be called the methodological misfit. This misfit is quite well-known and there are some attempts to overcome the problem. My hypothesis is that there is a common basic assumption behind many of these attempts. I will use the concept of rationality resonance to label that assumption. Rationality resonance can be described as the relation between the rationality underlying a certain system design method and the idea of rationality carried by the practitioner. It is often assumed that there has to be a resonance between these two, if the method should be understood and used. I will question the idea that rationality resonance is what designers of methods should strive for. Some general aspects on the concept of rationality and its relation to methods are presented. And also some comments upon the relation and difference between rationality and rational actions.

The idea of rationality resonance can be used to reveal some of our most fundamental assumptions concerning the whole idea of methods as a way of supporting the design process. And it will maybe help us to give some new interpretations and some new understanding of the phenomena of methodological misfit."

Position Statement for the Dagstuhl Seminar on Multidisciplinary Design

Lucy Suchman

I come to this seminar as an anthropologist who's been working for the past 15 years in an industrial research center dedicated to the invention of new computer-based technologies. During the time that I've been involved in this world I've been drawn increasingly into the activities and concerns of system design. The issues that I'd like to propose for our discussions are informed by my background and reflections on my current position and experiences within the terrain of "multidisciplinary design." They include the following considerations and questions:

- A critical consideration of the various interests pressing for the incorporation of more disciplines into design. I'm concerned here with how it is that "multidisciplinary design" might involve on the one hand a kind of voluntary joining together of different

perspectives in the interest of creating useful artifacts, and on the other an enrollment of more and more disciplines into the service of producing and marketing new commercial products. While I'm not suggesting that there is any inherent conflict between these two agendas, I do think we need to be clear about the differences.

- Closely tied to this, the question of relative power and resource distribution among the "disciplines" (or more broadly, knowledges -- see below) engaged in design activities. Are they in fact equally valued, or do some dominate while others are seen as providing peripheral, albeit crucial services? What would it mean to move toward a more genuinely symmetrical multidisciplinarity?
- I take it that a goal of multidisciplinary design is to incorporate a greater range and diversity of knowledges into the creation of new artifacts. On that premise, I'd like to question the notion of "disciplines" and the extent to which that term really expresses the diversity of relevant knowledges. I'm concerned here with extending the scope of our discussions from a focus on academic training to a focus on institutional locations of design and their implications for the definition of relevant knowledges and the structuring of working relations and practices.
- Finally, I think we need to open up the notion of "design" from bounded events associated exclusively with particular sites of professional practice, to an extended course of ongoing activities of technology production and use, taking place across sites and involving diverse (and differentially visible) participants. The sense of "multidisciplinarity" changes on this latter view of design.

In general I believe that the transformation of design into a multidisciplinary activity in the fullest sense (that is, as something made up of multiple, equally valued knowledge and experiences, taking place throughout the production and use of working technical systems) requires forms of social change that extend well beyond the bounds of the disciplinary elaboration of professional design practice. How might those of us positioned as academics/ industrial researchers contribute to bringing about that order of "multidisciplinarity"?

Position Paper for the Dagstuhl Seminar "Interdisciplinary Foundations of Systems Design and Evaluation"

Ina Wagner

My current research is centered around three major issues:

- the ways in which electronic networks contribute to the redefinition and creation of terrains/milieus for individual and collective activity, with distinct spatial, temporal, and cultural properties;
- the political uses made of these emerging potentialities (and the ethical and political issues involved);

- the environment in which systems designers have to manoeuvre and how this shapes and constrains the kinds of technical choices they develop.

Studying these issues requires to look carefully into observable practices of designing and using IT systems. At present we are working on a series of case-studies, mostly in an exploratory mode:

- We look into the work practices of small software teams that develop graphical software under high time pressure for a large media organization; and of an edp-department within an hierarchical organization which simultaneously works on large software development projects and services its diverse clientele (researchers, administrators) on a daily basis. We are interested in understanding the technological regimes these teams use (tools, languages, programming styles, standards, methods, testing); in the involved actor networks; in the design process, its CSCW aspects (shared resources, forms of cooperation, articulation work, technical support); and in time-management practices (the timing of project activities, temporal conflicts).
- We are studying how artists are using the technicalities for generating new work practices, products, and relations to their "public". Based on case-studies of programmable lighting in the theatre, of architectural practice and of film making, we look into how artists combine tools and media; how they use computers for de-stabilising form and content (also in their time-space relatedness) and for re-"writing" and re-contextualising them. We also try to understand, in how far standardisation (reducing the space for individual styles) and formalisation (abstract, rule-bound procedures displace intuition and "bricolage") further the "de-professionalisation" of specific artistic skills and techniques and their integration into other fields of practice.
- Our third "field" are current practices of remote work in two large computer firms. In both cases we are involved in a pilot, involving qualified workers who alternatingly work in their office, at their clients' work site and at home. On one hand we are interested in observing creative uses of emerging new time-space structures. On the other hand our task is to help shape technical and organizational support for this type of alternating work. The political framework - a "high performance" setting in which people are forced to operate in a "profit-center" mode - provides a special challenge for handling labour-related issues such as working time regulations, the definition of access to communal resources, management control.

Interdisciplinary design provides a strong orientation for all three projects. Their aim is to understand and describe work practices and the (emergent) settings and organizational forms in which they are embedded in ways which are useful for systems designers.