1

Reflective Systems Development

Lars Mathiassen

Abstract. The ways in which we approach systems development practice and research play a major role in shaping professional development within our field. This paper investigates a particular approach, Reflective Systems Development, which has been developed over the past twenty years by a small group of mainly Danish researchers in collaboration with practitioners and students. In this approach, researchers focus on how computer-based information systems are developed in practice; they emphasize the important role played by the local organizational environment; and they combine interpretive understandings of practice with normative propositions to support professional development.

The purpose of the paper is to present and evaluate the underlying assumptions and practices of Reflective Systems Development focusing on the following questions: How should we understand, support, and improve practice? How should we organize and conduct research? How should we relate practice and research? The argument draws on the literature on systems development and on the author's experience as an active participant in developing the approach.

Keywords: information technology, systems development, action research, dialectics, reflection-in-action, communities-of-practice.

1. Introduction

In 1975, when I agreed to help Kristen Nygaard organize and teach new courses in systems development as part of the computer science curriculum at Aarhus University, Denmark, I had no idea of the consequences this would have on my working life. I had just finished my masters degree and was hired as an academic assistant to teach and pursue my interest in pedagogical and didactic aspects of computing (Mathiassen 1976, 1980). I wanted to teach and was happy to get an opportunity to develop concepts and approaches to support this through research. But two experiences changed the course of my professional development.

The first experience was of a negative nature. I was not able to find support for a research career within the pedagogical aspects of computing. I was considered sufficiently skillful and my applications were spoken of in quite positive terms. But other applications were always considered more central to computing and given higher priority than mine. My conclusion from this experience was simple: If I wanted to pursue a research career within computer science I had to change to a new subject area.

The other experience was positive. I got involved in building up the masters program in systems development, first as an assistant to Kristen Nygaard, and later in close collaboration with my colleagues at Aarhus University. I came to like the subject which had not been part of my own computer science education. Many interesting questions and puzzles emerged that challenged me to combine my interest for learning with the technical issues of computing.

For the past twenty years, I have pursued this challenge as teacher, researcher, and industrial consultant in systems development, collaborating closely with practitioners, students, and a small group of mainly Danish researchers. The resulting approach to systems development practice and research has been presented by ourselves as "professional systems development" (Andersen *et al.* 1986, 1990) and discussed by others as "the ordinary work practices approach" (Hirschheim *et al.* 1992) or "the professional work practices approach" (Hirschheim *et al.* 1995; Iivari & Lyytinen 1997).

I use the name "Reflective Systems Development" to acknowledge the relation to Schön's ideas on how professionals think in action (1983, 1987) and to emphasize one of the key assumptions of the approach: Systems development methods and tools play important roles in shaping and improving practices, but their impact is

limited. Systems developers must, in addition to mastering a repertoire of general methods and tools, know how to cope with the specific environment in which they work. Many situations are experienced as unstructured. They involve uncertainty, instability, uniqueness, and value-conflict and they require an ability to go beyond the relatively safe territory of general professional knowledge. Systems developers must open their minds and engage in reflections and dialogues to generate the necessary insights into the situation at hand.

The purpose of this paper is to present and evaluate Reflective Systems Development as an approach to practice and to investigate the underlying assumptions of the research that have lead to this particular framework for systems development. Table 1 provides a summary of Reflective Systems Development based on Checkland's ideas on how intellectual frameworks are used in relation to specific application areas (Checkland *et al.*, p. 283). Table 1 explicates the different, but related purposes of research and practice. It presents the underlying intellectual frameworks, the type of process in which they are applied, and the shared arena to which they are applied. Reflective Systems Development support, in this way, two different modes of inquiry, a research and a practice mode, into the same arena of systems development practice.

I will investigate the position summarized in table 1 by drawing on my personal experiences, relating to the extensive literature on the subject, and focusing on three fundamental questions related

	Research	**Practice**
Purpose	to develop knowledge to understand, support, and improve practice as part of the ongoing professional development	to develop computer-based information systems as part of the ongoing transformation of organizations and society
Framework	dialectics reference disciplines	systems development theory systems development methods
Process	action research	reflection-in-action
Arena	systems development practice	systems development practice

Table 1. Reflective Systems Development as an approach to research and practice.

3

to systems development practice and research: How should we understand, support, and improve practice? How should we organize and conduct research? How should we relate practice and research?

My argument is organized in three parts. First, section 2 provides further introduction through a presentation of the practical and scientific context of this research. Then, the subsequent four sections present Reflective Systems Development in greater detail: section 3 describes its history; section 4 presents the research approach; section 5 presents the related view on practice; and section 6 provides a survey of the resulting contributions to systems development theory and methods. Finally, the last part of the paper discusses the approach: section 7 explores its relationships to other contributions to systems development; and section 8 presents lessons that I have learned, a discussion of strengths and weaknesses, and plans for future development of the approach.

2.   Context

Many researchers and practitioners have struggled to understand and mature the systems development profession. Hirschheim *et al.* (1995, 1996) offer broad paradigmatic analyses and surveys of systems development approaches. Other surveys can be found in Floyd (1987), Nurminen (1988), Hirschheim *et al.* (1989, 1991, 1992), Iivari (1991), and Dahlbom *et al.* (1993). Surveys of Scandinavian approaches are offered by Bansler (1987, 1989) and Iivari *et al.* (1997). These surveys show that reflective systems development is not the only nor the most widely known approach. However, it does have some unique features and an identity of its own. I will explore these in the subsequent sections after having looked at the practical and scientific context.

2.1. Systems development and systems development methods

Systems development is an important practice and research area related to information technology (IT) (Ives *et al.* 1980; Lyytinen 1987b). Systems development is a specific kind of human activity which aims at changing organizations through the use of IT. It includes activities such as analysis, design, programming, implementation, and maintenance. It also includes activities such as project management, quality assurance, and software process improve-

ment. More formally I use the following definition (Welke 1981; Lyytinen 1987a):

> *Systems development is a change process taken with respect to object systems in a set of environments by a development group to achieve or maintain some objectives.*

Systems development is an intentional change process which is driven by certain more or less clear objectives. It is performed by a group of actors that operates in a set of social and technical environments. The resulting changes relate to a number of object systems or perspectives on computer-based information systems. These include the technical perspective focusing on the technical platform, the symbolic perspective focusing on the information contents, and the organizational perspective focusing on the use of the information system (Lyytinen 1987a). The actual change process is shaped by several factors, including the experience and competence of the development group, the considered object systems, the dynamics of the objectives, and last but not least the social and technological environments in which the change process takes place. One particular element in the environments plays an important role within the field—the applied systems development methods (Lyytinen 1987a):

> *A systems development method is an organized collection of concepts, beliefs, values, and normative principles supported by material resources.*

More specifically (Mathiassen 1981; chapter 5 (in this book)), methods provide concepts and principles with related techniques and are often supported by computer-based tools. Their stated purpose is to help the development group carry out the change process. Each method has a limited application domain and it is based on specific values and beliefs which imply a certain perspective.

2.2. Systems development practice

The above definitions describe the research and practice area in a general and invariant manner. The challenges faced in practice have, however, changed as new technologies have emerged and new types of applications have been requested. A survey of some important changes in systems development practices over the past decades is contained in table 2 (Avison *et al.* 1988; Lyytinen 1989; Applegate *et al.* 1996). This survey focuses on changes in: the purpose or justification of using IT, the types of applications that are devel-

	Era I	Era II	Era III
Purpose	Productivity and efficiency	Individual and group effectiveness	Strategic and collaborative
Appli-cations	Automation Separate systems	Support Integrated systems Embedded systems	Strategic systems Process integration Collaboration
Tech-nology	Mainframes Batch processing Simple terminals Databases	Distribution PC's Local networks Graphics Expert systems	Global networks Multimedia Mobile computing Standard software
Skills	Programming Management	Analysis Design Collaboration	Domain Architectural
Im-prove-ment	Methods and tools Project management	Quality assurance CASE	Process improvement

Table 2. Systems development challenges in three eras.

oped, the technologies that are used, the skills that are required by systems developers, and the applied strategies for improving systems development quality and performance. The eras cover: (I) the early sixties to the mid seventies, (II) the mid seventies to the late eighties, and (III) the late eighties until today. Each era introduces new trends which in most cases lead to additional challenges as most of the previous trends are sustainable.

During era I, systems development was mainly practiced as a technical discipline. IT was used to automate existing manual processes, each application was treated as a separate entity, and the overall purpose of using IT was to increase productivity and efficiency. Developers were basically programmers and the dominating strategies for improving practices were based on introduction of new tools for programming and new methods that enforced basic project management disciplines.

The transformation into era II was driven by at least two important trends. Distributed, more user friendly, and more readily available technologies created the basis for new and more extensive uses of IT and the dominating position of technical skills was, as a consequence, complemented with end-user computing and collaboration between different professional groups. At the same time there

was a shift away from technology towards its use. Analysis and design were emphasized as key activities and quality assurance came to play an important role in improving practices. Applications were now being developed to support professional work (including CASE tools), many systems became highly integrated, and there was a considerable expansion of the use of IT in other industrial artifacts.

The most dominant change introduced in era III has been global networks which make it possible to transcend conventional boundaries for using IT. Applications have now become more integral parts of business strategies and they have created new opportunities to establish alliances and collaboration across organizational and national boundaries. At the same time, still more types of standard application software are used challenging the established tradition of developing unique applications tailored to specific organizational needs. During this era there has been an increased focus on skills related to specific business domains and on architectural skills needed to utilize networks and adapt and integrate applications. The new tradition for improving practices is based on a concern for systems development processes and for the technical and cultural environments that shape them.

This picture of the evolution of systems development practice is general and abstracted from the idiosyncrasies of specific organizational settings and from important cultural, political, and legal variances across nations. It does, however, point at some fundamental dynamics of the practice context as it has evolved over the past twenty years: the variety of technologies and the social dependence on IT applications have increased; the complexity of the task has grown; an increasing multiplicity of skills are involved; and a variety of complementary approaches to improving practice have emerged together with a still broader concern for increased professional maturity.

2.3. Systems development research

As the profession has grown in volume and importance, so has the related research. Here, I review different approaches to systems development research as they have emerged over the past decades, and later I will relate to some of the key findings of practice-related research on systems development.

Starting with the broader area of IT-related research we find a multiplicity of approaches together with extensive discussions of

their strengths and weaknesses (McFarlan 1984; Mumford *et al.* 1986; Boland *et al.* 1987; Galliers *et al.* 1987; Cash *et al.* 1989; Nissen *et al.* 1990; Lee *et al.* 1997). Within the specific area of systems development we find a more specialized discussion (Basili *et al.* 1984, 1986; Nunamaker *et al.* 1991; Cotterman *et al.* 1992; Wynekoop *et al.* 1993). Focusing on practice-related research in systems development I distinguish between three types of approaches as illustrated in figure 1 (Munk-Madsen 1986; Nunamaker *et al.* 1991; Wynekoop *et al.* 1993). All of these approaches, often in combination, contribute to the building of research-based knowledge in the form of theories and methods.
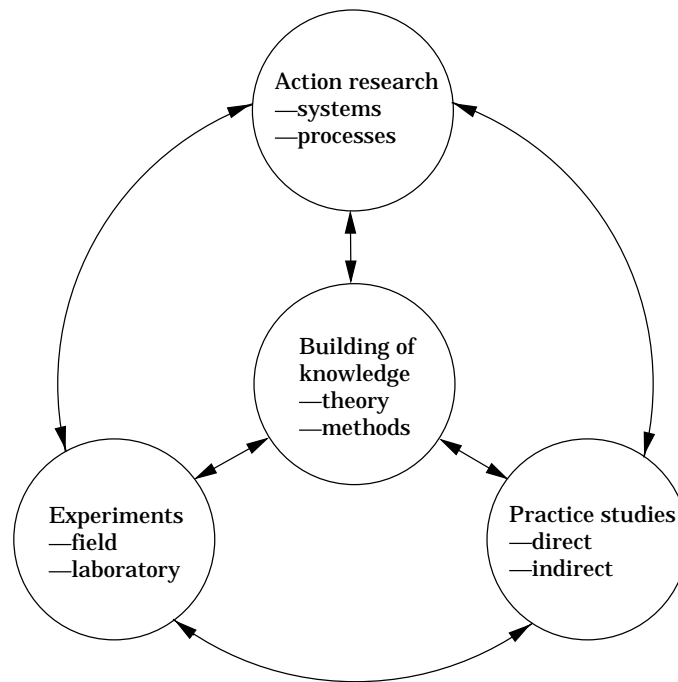

Figure 1. Approaches to systems development research (adapted from Nunamaker et al. 1991).

Action research involves the researchers in practice situations in close collaboration with practitioners. The research activity focuses on the system being developed or on the development process. The strength of this approach is the strong integration of research and

8

practice: practitioners are involved in the research process and researchers gain first-hand experiences. The most important weakness is the limited support provided to help structure the research process and findings. Recent surveys of the use of action research in information systems research are found in Baskerville *et al.* (1996a) and in Lau (1997). The application of action research to systems development is documented in relatively few sources (e.g. Kaiser *et al.* 1982; Mumford 1983; Parnas *et al.* 1986; Salaway 1987; Knuth 1989; Avison *et al.* 1991; Bjerknes 1991; chapters 2, 10, 14, 18). Action research, as we have used it, is further developed in section 4.1.

Experiments involve practices that are controlled by the researchers. Such experiments can either take place in realistic settings or in laboratory environments. A key advantage with this type of research is that the research process is designed to focus on specific questions and issues. The disadvantage, as compared to action research, is the weaker relation to practice. Experiments are more commonly used in systems development research and a number of contributions have been reported based on this approach (e.g. Boland 1978; Vitalari 1983, 1985; Boehm *et al.* 1984; Floyd 1986; Selby *et al.* 1987; Guindon *et al.* 1987; Baskerville *et al.* 1996b; chapters 3, 7).

Practice studies cover a wide variety of approaches to study systems development without active involvement of the researchers. Some approaches study practice directly, e.g. field studies and case studies, whereas others are indirect based on people's opinions and beliefs, e.g. using surveys or interviews. The strengths of this approach are that it focuses on practice and that it provides the researchers with a vast repertoire of techniques to structure the process and the findings. The weakness is that it separates research from practice. The researchers observe and interpret the actions and beliefs of practitioners and the practitioners do not take active part in the research process. Most of the empirical literature on systems development is based on practice studies (e.g. Benbasat *et al.* 1980; Boland 1982; Markus 1983; McKeen 1983; White 1984; Gould *et al.* 1985; White *et al.* 1986; Necco *et al.* 1987; Krasner *et al.* 1987; Elam *et al.* 1987; Curtis *et al.* 1988; Boehm *et al.* 1988; Mathiassen 1988; Aaen *et al.* 1992; Stolterman 1992; Bansler *et al.* 1993; Kozar 1993; Madabushi *et al.* 1993; Waltz *et al.* 1993; Ciborra *et al.* 1994; Tan 1994; chapter 18).

Practice-related research on systems development is based on one, or a combination, of these three approaches. In addition, each research effort can be characterized along a number of other dimensions including: How is practice viewed (structured vs. unstructured; quantitative vs. qualitative)? Which reference disciplines are applied (e.g. mathematics, design theory, organization science, management science, ethnography, psychology, and philosophy)? What kinds of knowledge are developed (interpretive vs. normative; theory vs. method)? What quality criteria are dominating (relevance vs. rigor)? And last but not least: Which problems and issues are studied?

The diversity in practice-related systems development research is considerable. In principle, this is very useful as the complexity and dynamics of the practice under study requires that multiple approaches are applied (Nunamaker *et al.* 1991; Cotterman *et al.* 1992; Wynekoop *et al.* 1993). But the diversity of approaches is, unfortunately, combined with a number of institutionalized specializations with rather little exchange between them. First, systems development research is divided into two research communities: software engineering which is mainly practiced within computer science and engineering departments, and information systems development which is practiced as part of information and management science (Mathiassen 1996). Second, two different research traditions have emerged: a positivist which is primarily occupied with structured, quantitative approaches based on observations and an interpretivist which uses less structured, more qualitative approaches based on interpretations and intervention (Galliers *et al.* 1987; Markus 1997). Finally, there are considerable differences in what is considered the primary type of results and the perceived audience: one group is focusing primarily on concepts, methods, and tools to support practice, and another is striving to develop theories and frameworks to improve our understanding of practice. Each of the related communities-of-practice have developed strong and relevant positions, and one of the key challenges in advancing research is to improve dialogue between them (Robey 1996; Markus 1997).

3. History

Reflective Systems Development has evolved as a result of research, teaching, and consulting. It has been influenced by specific incidents

and institutional settings in Scandinavia. It has also been inspired by the work of an international group of research colleagues. A historic account of its development will help us better understand the characteristic features, strengths, and limitations of the approach. The description is divided into three periods, the seventies, the eighties, and the nineties. The seventies (section 3.1) bridge the first and second era of systems development (see table 2) where the use of IT beyond automation of existing processes accelerated and where distributed and more user-friendly technologies were taken into use. The eighties (section 3.2) relate to the second era where analysis, design, and collaboration between users and systems developers were emphasized and where quality assurance and CASE tools were used to improve practice. Finally, the nineties (section 3.3) correspond to the third era in which IT has become a key factor in shaping both the internal processes and the external relations of most organizations and where the pressures for IT departments to deliver quality services at acceptable expenses have increased.

3.1. The seventies: critical positions

Reflective Systems Development has grown out of computer science as a critical response to its technical and mathematical orientation and as an attempt to orient computer science towards the future practices of its students. Influential students at the Computer Science Department at Aarhus University organized courses on computers and society during the early seventies, and in 1974 they had Kristen Nygaard employed as visiting professor. Kristen Nygaard was well-known and widely respected for his work on the Simula language (Dahl *et al.* 1971; Birtwistle *et al.* 1973) and had just finished his controversial research collaboration with Norwegian trade unions (Nygaard *et al.* 1975).

Kristen Nygaard's initial approach to teaching was simple and based on the fundamental phenomenology of systems development. Two one-semester courses were designed. One, a case course, focused on systems development practices as they could be experienced and observed in organizations struggling to make use of IT. Another, a method course, focused on systems development methods as they could be studied in books and scientific papers describing how computer-based information systems could be developed.

The case course was organized as a collaboration with systems developers, managers, and users. One or more projects were studied

in great detail, the students visited the organizations involved, and the practitioners visited the university. Presentations were made by the practitioners, interviews were prepared and conducted by the students, and a selection of material from the projects was distributed and analyzed. Each group of students had to write a report on specific aspects of the projects and by the end of each case it had to present its key findings at a joint seminar with the practitioners.

The method course was taken after the case course. It was organized as a workshop in which groups of students studied systems development methods, explicating their underlying assumptions, and evaluating their strengths and weaknesses. Each group of students had to present their method to the class and they had to write a report documenting their findings.

In this way, the distinction between what we can experience and observe when systems are being developed in organizational contexts, and what can be read in books and scientific papers on how to develop systems, was built into our approach from the very start (Mathiassen 1981). During this period we and our students essentially tried to learn *about* systems development by using and developing different perspectives to understand and criticize methods and practices. We came to understand that methods are quite different from practices and we came to appreciate the crucial role played by perspectives in evaluating practices and methods (Sandberg 1975; Kling 1980; Mathiassen 1981; Nygaard *et al.* 1985).

During the same period, Kristen Nygaard helped us initiate a number of research initiatives in collaboration with Danish trade unions. These projects built on the experiences from the Norwegian Iron and Metal Workers Project (Nygaard *et al.* 1975), the most influential being the DUE project (Danish acronym for democracy, development and EDP) (Kyng *et al.* 1982). The projects were based on action research in close collaboration with local unions in specific organizations. The objective was to build up knowledge and resources that workers could use to defend and develop their interests as IT became an increasingly important part of the infrastructure at the workplace. This research led to critical views on the use of IT based on our practical experiences from workplaces and studies of Marxist literature (e.g. Braverman 1974), and it became part of one of the dominating traditions in early Scandinavian information systems research.

Bansler (1987, 1989) identifies three early Scandinavian schools of research: the systems theoretical school (Langefors 1966, 1968; Lundeberg *et al.* 1974; Bubenko 1980, 1986); the socio-technical tradition (Høyer 1971, 1974; Bjørn-Andersen *et al.* 1977; Bjørn-Andersen 1980; Olerup 1989); and the critical tradition (Nygaard *et al.* 1975; Carlson *et al.* 1978; Kyng *et al.* 1982; Bjerknes *et al.* 1987a; 1987b). All three schools played a dominant role in the emerging masters program. As *teachers* we spent a great deal of effort together with our students to learn about the methods of these schools and to explicate and evaluate the underlying assumptions and perspectives. During the same period, we spent most of our efforts as *researchers* developing the critical tradition in close collaboration with Swedish researchers involved in trade union research (Carlson *et al.* 1978) and with Kristen Nygaard's research group at Oslo University.

We also took some initiatives to formulate concepts and frameworks for systems development research. These initiatives took us beyond the traditional views of computer science to include social perspectives and issues related to organizational change. Several master theses were written on systems development building directly on insights from the trade union projects (e.g. Munk-Madsen 1983; Mathiassen *et al.* 1983). My own Ph.D. thesis used the critical position from our action research with trade unions to establish a framework for systems development research. This framework evolved during our efforts to educate systems developers and it was later used as a starting point for our empirical research. The framework focused on methods and practices and it emphasized the relationship between the structures and the processes involved in systems development (Mathiassen 1981; chapter 4). The key to formulating this framework was the dialectical epistemology developed by Joachim Israel (1979) who together with Kristen Nygaard supervised my Ph.D.

3.2. The eighties: practice orientation

A number of incidents implied a radical change in our thinking on systems development by the end of the seventies. As we repeated the method course we started to experience the discussions of systems development methods as academic exercises rather distant from the practices discussed in the case course. Most students found it difficult to understand methods that they mainly knew from their

readings. As a consequence, their evaluations of the methods were abstract and at times quite simplistic. At the same time, many students had left the university to work as professionals in software houses and IT departments, and we started collaboration with some of them through informal networks. Finally, the DUE project terminated in 1979 and we needed to develop new research initiatives.

One group of researchers involved in the DUE project maintained the trade union perspective, but gave the research approach a more constructive orientation. Together with Swedish colleagues they initiated a new generation of trade union projects focusing on the development of computer-based tools to support and enhance the quality of working life (Bjerknes *et al.* 1995). The most influential of these were the Utopia project (Bødker *et al.* 1987; Ehn 1988; Kyng 1996) and the Florence project (Bjerknes *et al.* 1987b, 1988a, 1988b). This research became an utopian systems development experience, serving as a real-world laboratory to explore and develop new professional tools and techniques and to demonstrate by example that IT can be used to radically enhance working life.

The rest of us made a different turn. We wanted to learn more about professional practices in software houses and IT departments. We changed perspective from that of organized workers to that of professionals working with IT. But we maintained the action research approach from the first generation of trade union projects forming research groups in specific organizations as means for collaboration between practitioners and researchers. Our goal was to understand, support, and improve the professional practices of systems developers in real-life organizations.

In the MARS project (MARS is a Danish acronym for methodological working practices in systems development) we formed action research groups in four software organizations (MARS Project 1984a). Each group consisted of practitioners, researchers, and students. During the first year we analyzed and diagnosed selected projects to learn about the kinds of problems and challenges that emerged in systems development, the practical use of methods, tools, and techniques, and the kinds of interventions that took place to overcome problems and manage uncertainties and conflicts (MARS Project 1984b). During the second year we experimented with new ways of developing systems, trying to improve the professional practices in specific projects (MARS Project 1985). Our experiences were synthesized in a book jointly designed and written by

14

practitioners and researchers (Andersen *et al.* 1986, 1990). Our most important insights were:

- Systems development methods were seldom, or only partially, followed by experienced practitioners.
- Each environment had distinct characteristics that played an important role for the success and failure of projects.
- The most urgent areas for improving practices were project management and more supportive organizational environments.
- It was possible, but extremely difficult to change working practices.

This action research inspired us to develop our intellectual framework further. We needed to learn more about ways to study and understand professional practices. Lanzara's work on design processes (Lanzara 1983) inspired us to initiate collaboration (chapter 2), and this lead to further studies of Argyris and Schön's work on organizational behavior and learning (Argyris *et al.* 1978, 1996; Schön 1983, 1987). It was also necessary for us to know more about organizations and the ways in which they condition systems development practices. In collaboration with Ciborra and inspired by his efforts to apply transaction cost theory to information systems (1981, 1983) we started to include general organization theory into our systems development curriculum (e.g. Williamson 1975; March *et al.* 1976; Mintzberg 1983; Schein 1985). We also participated in an extensive Nordic researcher-practitioner collaboration called the SYDPOL program (SYstems Development environments and Profession Oriented Languages) and started to study the relations between organizational perspectives and systems development approaches (Aaen 1989; Bjerknes *et al.* 1990). Finally, we engaged ourselves more actively in collaborations with software engineering researchers, most notably Christiane Floyd (1984, 1987; Floyd *et al.* 1992) and Heinz Züllighoven (Budde *et al.* 1984; Züllighoven 1992; Budde *et al.* 1992).

Increasing our focus on practice made us change the way in which we taught systems development. From learning about systems development, we wanted our students to learn systems development with a reflective attitude. This might seem an obvious turn, but there are some serious difficulties involved. One possibility is to let the students practice systems development in real organizations.

15

This raises the issue of keeping the pressures involved in real-life systems development at a reasonable level. Another possibility is to have the students develop systems within a university setting. This leads to difficulties in creating sufficiently realistic conditions.

My own reflections on education developed (chapter 22) as we started to set up experiments in both industry (Stage 1989; Nielsen 1990a) and at the university (chapter 7). These experiments became important elements in educating reflective systems developers, and they expanded our repertoire of research approaches into systems development practice. We used diaries to help the students reflect during their practices and to create empirical documentation for later reflections on the same practices (Naur 1972, 1983; chapter 3).

The researchers in the MARS project eventually left Aarhus University. My own move in 1987 to Aalborg University gave me a different and inspiring environment for teaching systems development. Aalborg University's educational system is both problem-oriented and discipline-oriented and it is organized in projects and courses. Each semester the students spend half of their time in groups working on a project which is formulated by themselves within the theme of that semester. The other half of their time is spent on traditional courses elaborating on disciplines related to the theme.

One year earlier, Andreas Munk-Madsen and I had formed a consulting and training company, Metodica. Our intention was to disseminate and further develop what we had learned in the MARS project. Over the past ten years, Metodica has served as an alternative forum where researchers, consultants, and practitioners engage in training programs and dialogues to learn about, test, and develop new ideas on systems development.

3.3. The nineties: professional development

Our commitment to systems development practice moved the values and issues of the trade union research to the background of our attention. When we started the MARS project, we found that the most urgent problems for professional systems developers were not related to trade unions nor to users. Most practitioners found themselves in environments with static contracts, changing requirements, weak managerial support, and dynamic technical platforms. The kinds of conflicts and uncertainties they were involved in were not primarily rooted in employer-employee relations in the user or-

ganization and the most urgent needs for improvement were not to be found in the area of user participation, but in the area of project management.

We had changed our perspective from that of users to that of professional practitioners because we wanted to understand better the situations in which our students would find themselves when leaving the university, and because we wanted to engage in dialogues with practitioners to jointly develop concepts and methods that could improve systems development practices. At the same time, we wanted our discipline to be an integral and important part of the computer science curriculum. We had hoped to work with user-developer collaboration, systems analysis and design, experimental systems development, and to explore further the object oriented heritage from our long-lasting collaboration with Kristen Nygaard (Dahl *et al.* 1971; Birtwistle *et al.* 1973; Holbæk-Hanssen *et al.* 1975; Madsen *et al.* 1993). Instead, our action research efforts led us to write a book on development processes, project management, and change of work practices (Andersen *et al.* 1986, 1990). A natural next step for us was, therefore, to focus as much on the reflections involved in designing and engineering computer-based information systems as on the reflections involved in designing and managing development processes. Such a step was catalyzed by the changes in our educational approach away from learning about systems development to learning systems development with a reflective attitude.

Throughout the late eighties and the nineties this led to a number of projects focusing on professional development. We started to use and further develop Soft Systems Methodology (SSM) (Checkland 1981; Wilson 1984; Checkland *et al.* 1990; Nielsen 1990a; chapters 10, 11, 12, 13). We continued to develop concepts and techniques to support systems development management (chapters 6, 7, 8, 9; Munk-Madsen 1996). We began developing our own analysis and design methodologies (Stage 1989; Mathiassen *et al.* 1993, 1995, 1997; chapters 12, 14). We engaged ourselves in the use and adaptation of software engineering tools (Aaen *et al.* 1991; Aaen *et al.* 1992; Sørensen 1993; chapters 15, 16, 17). Finally, I started a close collaboration with Bo Dahlbom to merge his philosophical background with our interest in Reflective Systems Development (Dahlbom *et al.* 1993; chapters 19, 20, 21). During this period we expanded and strengthened our collaborations with a number of international research groups including Checkland's group in

Lancaster, Klein's group in Binghamton, Lyytinen's group in Jyväskylä, Floyd and Züllighoven's group in Hamburg, and last but not least Dahlbom's group in Göteborg.

4.   Research

Reflective Systems Development has, in this way, emerged as an approach that intertwines both research and practice. It is based on inspiration from a number of sources from which a variety of elements have been picked up along the way to be used, modified, and combined. Eventually, they have been shaped into what is here presented as a coherent approach with features that distinguish it from other approaches to research and practice (see table 1). In this section, I will elaborate on the approach to research. In section 5, I will focus on the other part of the equation, the approach to practice.

4.1. Action research

Reflective Systems Development uses action research as the basic practice form in research. This is illustrated in figure 2 based on Checkland's notion of the experience-action cycle (Checkland *et al.* 1990). The problems, challenges, and opportunities involved in systems development practice are considered the starting point for sys-
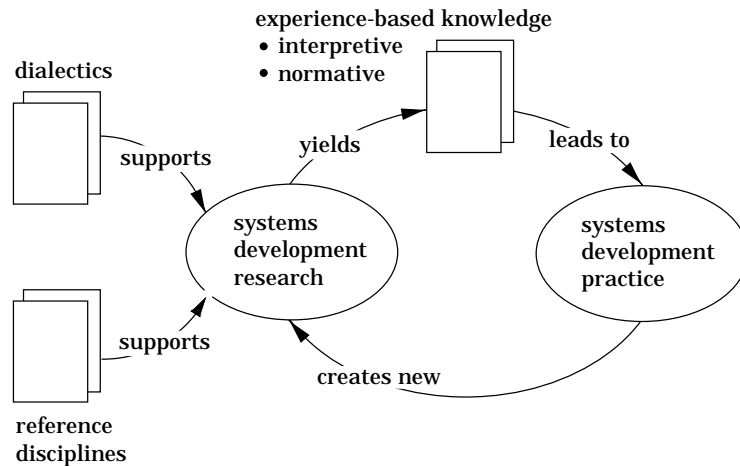

*Figure 2. Action research in Reflective Systems Development
(adapted from Checkland et al. 1990).*

18

tems development research. Research activities yield experience-based knowledge that leads to new, and hopefully improved, practices. The knowledge that is developed is both interpretive, helping us to understand and make sense of practice, and normative, providing support for performing systems development or improving present practices. Also, this knowledge takes different forms. Part of it remains local, individual, and even tacit, while other parts are explicated and made publicly available as systems development guidelines, professional books, and research contributions. The research activity is primarily informed by systems development practice, and it is supported by various reference disciplines (e.g. design theory, organization science, management science, and philosophy), and, as we shall explore further below, by dialectic reflections.

Action research assigns, in this way, primary importance to practice and it emphasizes the intrinsic relations between practice and research (Baskerville *et al.* 1996a). Research is an activity in which practitioners participate and collaborate with researchers. "The practitioner does not function as a mere user of the researcher's product. He reveals to the reflective researcher the ways of thinking that he brings to his practice, and draws on reflective research as an aid to his own reflection-in-action. Moreover, the reflective researcher cannot maintain distance from, much less superiority to, the experience of practice . . . he must somehow gain an inside view of the experience of practice" (Schön 1983, p. 323).

Following Checkland, we can distinguish a continuum of action research forms. In the one extreme, research is an intervention process to inquire into practice. In the other extreme, it becomes an ongoing learning process, or an interaction process, integrated into the stream of ideas and events as they unfold in practice (Checkland *et al.* 1990). This view of action research, implies and builds on a complementary view of practice in which reflection and learning are key elements. This point will be further elaborated in section 5.

Action research is not considered to be a panacea to systems development research. We use it as the basic form of practice in research and we supplement it, whenever feasible and useful, with experiments and practice studies. A combined strategy supports the variety of research goals and compensates for the greatest weakness of action research: the limited support that it provides for structuring the research process and findings. The importance assigned to practice implies that the primary target group for research is practi-

tioners and that relevance is considered the primary criterion on which to evaluate research. But to qualify as research, as opposed to consulting, the results and findings must, at the same time, have sufficient rigor (Munk-Madsen 1986; Baskerville *et al.* 1996a).

Building on the concepts of Vidgen *et al.* (1997) we can relate the different purposes of our research to different types of activities as illustrated in figure 3. The arrows inside the triangle represent distinct, and in some respects divergent, research activities through which the goals are supported. First, our understanding is based on interpretations of practice. Second, to support practice we simplify and generalize these interpretations and engage in design of normative propositions or artifacts, e.g. guidelines, standards, methods, techniques, and tools. Third, we change and improve practices through different forms of social and technical intervention. The triangle represents the unity of the three goals and it illustrates that each of the activities can benefit from the other activities. We reach a deeper understanding of practice as we attempt to change it. We need to understand practice to design useful propositions. Finally, the propositions and our interpretations of practice are used as part of intervention strategies to improve practice.

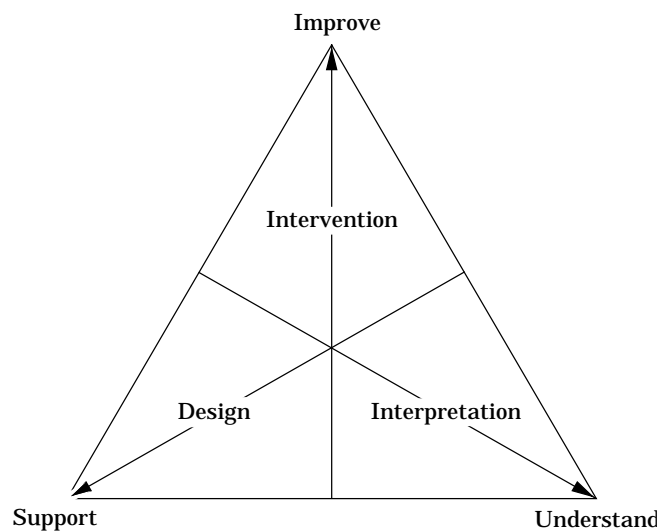Any action research approach embodies particular interests or

Figure 3. Research goals and activities involved in Reflective Systems Development (adapted from Vidgen et al. 1997).

20

viewpoints on what is considered feasible changes of practice (Baskerville *et al.* 1996a). Several stakeholders are involved in systems development including: those who are to use the resulting system (users), those who make decisions on which system to develop (clients), those who develop the system (developers), those who improve the development process (change agents), and those who make decisions on how to develop the system (managers). Each individual involved in systems development can have several interests and act in more than one of these roles.

Reflective Systems Development has a professional focus, taking as its starting point the position of those practitioners (developers, change agents, and managers) who spend the majority of their time in relation to systems development. Users, clients, and managers that are primarily occupied with other issues are, of course, equally important stakeholders, and it hardly makes sense to discuss systems development without including their interests and perspectives. Reflective Systems Development takes, however, a professional standpoint to understand the other stakeholders. Our approach shares this choice with major parts of the literature on software engineering and information systems development. However, this choice also distinguishes Reflective Systems Development from approaches with an application domain perspective (e.g. research on computer supported cooperative work), a user perspective (e.g. participatory design and end-user computing), a business perspective (e.g. management of information systems), or a general research perspective (e.g. research on systems development research).

4.2. Dialectics

Systems development is intrinsically related to change. It is itself a change process aimed at transforming or transcending the contexts in which IT is applied. At the same time, it is a profession in constant change (see table 2) and present trends suggest that this dynamic nature of practice will persist (Avison *et al.* 1988; Lyytinen 1989). Any research approach to systems development needs to include this fundamental aspect of practice into its intellectual framework. In Reflective Systems Development this has been achieved through the use of dialectics.

My own computer science background offered machines, programs, algorithms, and data structures as key concepts, and processes were conceived as sequential and structures as mainly hierar-

chical. These concepts defined a technical universe in which the majority of the world is seen as "external aspects" of computing that only can be addressed using concepts and insights from other disciplines. Through the collaboration with Kristen Nygaard we learned to think in terms of systems (Dahl *et al.* 1971; Birtwistle *et al.* 1973; Holbæk-Hanssen *et al.* 1975) and as part of that to apply different perspectives to understand the same phenomenon (Mathiassen 1981). Systems thinking and multi-perspective reflection helped us, as a first step, to develop an intellectual framework that expanded the horizons of traditional computer science in a way that was acceptable to most of our colleagues.

Our involvement in action research with trade unions confronted us, however, with problems and issues related to organizational change, human interaction, conflict, and power, which could not in a satisfactory way be understood as systems. Munk-Madsen's socially based process concept (1983) and our studies of sociology and, in particular, Marxist theory helped us reflect on these experiences in a more profound way. That led to the use of dialectics (originally based on Israel (1979)) as a framework for studying systems development practices and methods (Mathiassen 1981; chapter 4) which, in turn, led to a new understanding of computer systems and systems development in a social context (Dahlbom *et al.* 1993; chapter 21).

Dialectics is well suited as an intellectual framework for practice-related research in systems development. First, it is based on the understanding that knowledge is intrinsically related to practice. "Man, as an acting and language-using subject, in his concrete daily activities, must be taken as the basis in all analysis of the process of knowledge" (Israel 1979, p. 63). Second, the key assumption is that practice is constantly changing. To understand practice and to act intelligently as part of it, we must understand what changes are and why they take place. The approach offered by dialectics is to focus on contradictions, the relations between them, and the opposites and relations that constitute them.

Marx based his work on dialectics. This has led to the widely held misconception that dialectical approaches are limited to or identical with a Marxist position. This is far from the case. Marx became known for his economic and political theories, whereas dialectics is a general approach to understand and study social phenomena. Any person who reflects on social phenomena faces a number of

contradictions and will more or less explicitly—through the applied intellectual framework and the form of inquiry—take a stance on each of them. Some of the key contradictions are totality versus parts, process versus structure, relations versus entities, and intrinsic versus extrinsic relations (Israel 1979, p. 55–151). When we use dialectics, we engage ourselves in both aspects of each of these contradictions, but we start out from a particular one of the opposites.

First, dialectics emphasizes totality rather than parts. Systems development practice is a totality with many defining characteristics. Each particular concept, e.g. system, development, method, process, structure, data, and information define particular views. They are abstractions that we use to understand or communicate about practice. Dialectics views experience and concrete understanding as more fundamental than abstract concepts. But we need abstractions to make sense of our experiences, and we test them through practice.

Second, in a dialectical approach we emphasize process rather than structure. Processes refer to movement in time and space and to transformation of the given; they help us to focus on transformation, change, and transcendence. Structures are complementary abstractions that refer to the stable and given; they help us to focus on temporary order or patterns. Dialectics sees structures as a property pertaining to process rather than the other way round. This position is elaborated in (Mathiassen 1981) and in chapter 4. As a consequence, we consider systems development processes as more fundamental than methods, project models, plans, and other structural views on development processes.

Third, dialectics emphasizes relations rather than entities. Entities make us focus on persons, projects, computer systems, artifacts, and organizations; we understand each entity concentrating on its own, general features. If we alternatively reflect on relations we understand social phenomena as they relate to each other in a specific context. In the first case, we describe the task in a systems development project in general terms abstracted from the specific use environment. From a dialectic perspective, we might share the same ambition, but based on the understanding that any such abstract formulation represents complex and dynamic relations between the project and its use environment. Depending on which perspective you have, requirements management has different meanings and practical implications.

Finally, dialectical approaches emphasize intrinsic rather than extrinsic relations. Relating phenomena to each other in an extrinsic way does not change the way we understand them, neither in what they are nor in their function. In contrast, when relating them in an intrinsic way, we understand related phenomena emphasizing what they are in relation to each other. Focusing on extrinsic relations, a project consists of individuals with specific experiences and skills and with a certain division of labor between them. If we focus on intrinsic relations, we see the same project as a dynamic network of collaborations and interactions in which personal relations and emotions play important roles and evolve during the course of the project. Each of these perspectives have different implications for what it means and takes to establish and maintain a well-functioning team.

Dialectics helps us to understand and deal with, in a very explicit way, the dynamic and contradictory nature of our discipline. As we use dialectics, it is not an alternative to hard or soft systems thinking. Nor do we intent to develop it into a meta-theoretical framework which combines various reference disciplines. Instead, we use dialectics as a practical and quite informative way to understand practice and the relation between the knowledge we build and its practical use. It helps us to organize our thinking and to appreciate the strengths and limits of using hard systems approaches, soft systems approaches, and various reference disciplines. This position is developed in (Dahlbom *et al.* 1993; chapter 21) as a dialogue between our mechanistic heritage as materialized in the computer—emphasizing representation, formalization, program, and order—and the romantic challenge inherently involved in making use of computers—emphasizing interpretation, contradiction, and change.

5. Practice

From the very start, our case-based teaching and our action-oriented research brought us into dialogue with professionals and involved us in the problems and challenges in systems development practice. In the following, I take a closer look at practice. First, I focus on the individual level and discuss how practitioners think and learn and how they use methods in practice. Subsequently, I complement this view by discussing learning and innovation on an or-

ganizational level. This discussion explicates important factors that facilitate and restrict improvement of practices.

5.1. Reflection-in-action

Schön's studies of how professionals think in action (1983, 1987) have helped us reach a deeper appreciation of the role of knowledge and of the relation between research and practice in systems development. Schön argues that our traditional conception of knowledge is insufficient to explain how professionals like architects, engineers, therapists, managers, and city planners deal with problematic situations and think as part of their practice. Our traditional conception of knowledge is based on *technical rationality*. According to this model of knowledge, professional activity consists of instrumental problem solving based on rigorous application of specialized scientific theories and techniques. This implies a hierarchical view of professional knowledge in which general principles and theories occupy the highest level and concrete problem solving the lowest (Schön 1983, p. 24).

This model of technical rationality is embedded in the institutional context of professions. Research is most often separate from practice and only connected through exchanges that are carefully controlled through scientific review processes and career patterns. Researchers provide knowledge and practitioners formulate problems and test the usefulness of research results. The model is also reflected in the curricula of professional schools, where the students are first taught the relevant basic and applied science, and only afterwards the skills of applying these to real-world problems of practice.

Technical rationality is an efficient way to organize research and practice within a profession, but it has its limits. It depends on agreement about means and ends. When there is no obvious solution and professional paradigms are conflicting, or when ends are not fixed and clear, but confusing and conflicting, there is no simple way in which professionals can select methods to solve problems. In order to convert most problematic situations to a well-defined problem a practitioner must do a certain kind of work. "When we set the problem, we select what we will treat as the things of the situation, we set the boundaries of our attention to it, and we impose on it a coherence which allows us to say what is wrong and in what directions the situation needs to be changed. Problem setting is a process

in which, interactively, we name the things to which we will attend and frame the context in which we will attend to them" (Schön 1983, p. 40). When practitioners fail to engage in problem setting, they run the danger of misreading situations, not grasping the unique characteristics of the situations they are involved in. Technical rationality leads them to manipulate the situation and this in turn reinforces their confidence in their existing repertoire of concepts and techniques.

Schön proposes a different conception of professional knowledge, *reflection-in-action*, which explicitly addresses how professionals deal with problematic situations involving complexity, uncertainty, instability, uniqueness, and value-conflict. This model starts from the assumption that we in our everyday life are knowledgeable in a way that is often tacit and implicit in our behavior (Polanyi 1967). This form of knowing-in-action consist of actions, recognitions, and judgments that we know how to do; we do not think about them before doing them; and we are often unaware of how we learned to do them (Schön 1983, p. 54). Our spontaneous knowing-in-action usually gets us through the day. But sometimes we are stuck, caught by surprise, or our expectations are not met. We then start thinking about what we are doing, not only in a detached manner, but also while doing it. We reflect on our actions and, in some cases, we reflect-in-action.

Applied science and research-based techniques do, of course, play important roles in professional practice, but the territory they occupy is bounded on several sides by artistry. To mediate the use of applied science and techniques, practitioners engage in the arts of problem framing, implementation, and improvisation (Schön 1987, p. 13). When professionals find themselves in unique or unstable situations they might criticize their initial understanding of the phenomenon and construct and test a new description of it. When they are stuck, they might find completely new ways to frame the situation and impose these on the situation to see different problems and opportunities. In this way, professionals engage in what Schön calls conversations with the situation. In doing so, they use metaphors and on-the-spot experiments to support reflection-in-action.

Situations involving complexity, uncertainty, instability, uniqueness, and value-conflict are quite common in systems development, and problem framing, implementation, and improvisation

are not only useful, but necessary types of competence for systems developers. Our discipline is loaded with methods and tools, but each of these rely on specific assumptions and concepts, frame our understanding in specific ways, and have limited application areas. To understand, support, and improve systems development, we must go beyond theories and methods and understand the role of knowledge and the relation between knowledge and action in systems development practice.

The relation between reflection and action in systems development unfolds differently depending on whether the view of the process is mainly construction, evolution, or intervention (Dahlbom *et al.* 1993; chapter 21). In a construction perspective, the challenge is to develop a technical solution in response to a given data-processing problem; reflection comes mainly before action; and the process is organized as instrumental problem-solving. In an evolution perspective, the challenge is similar, except that the problem is not given and stable; reflection-in-action is needed to frame the situation and set the problem; and the process is organized as an iterative learning process. Finally, in an intervention perspective, the challenge is to change an organization through development of new or modification of existing computer-based information systems; reflection-in-action is needed to support problem framing, implementation of procedures and systems, and improvisations in critical situations; and the process is organized as interventions into a series of problematic situations.

Systems developers are advised to mix construction, evolution, and intervention perspectives depending on the characteristics of their efforts (Dahlbom *et al.* 1993). All projects need to emphasize intervention to deal effectively with uncertainty, instability, uniqueness, and value-conflict—especially in the early and late phases of a project. All projects are also advised to establish and maintain a focus on a data processing problem that makes it possible to combine construction and evolution approaches in order to effectively design and construct application software (chapters 6, 7).

There are some differences between Schön's general conception of the reflective practitioner and my specific understanding of systems development. First, systems development is a particular professional practice different from other disciplines. Systems developers definitely engage in problem framing, implementation, and improvisation activities in which they need to go beyond the model of

technical rationality. But the use of IT requires them not only to know about, but to be experts in, formalization, programming, and technical construction. They therefore need to master a considerable repertoire of methods and techniques. Second, Schön's focus is on individual behavior. The underlying cases typically involve one professional in dialogue with a supervisor or coach. Individuals are, of course, the basic unit in all professional practices. But systems development is a highly collaborative activity which takes place in complex, dynamic arrangements with many actors and organizations involved.

5.2. Communities-of-practice

Brown & Duguid's notion of communities-of-practice provides useful insights into how learning and innovation take place in organizational contexts involving the use of technology (1991). Organizational guidelines, standards, and systems development methods contain abstract, simplified knowledge detached from practice. This kind of knowledge is used to train practitioners, as a basis for design and management of projects, and, quite often, as the established way to represent and talk about practice. Brown & Duguid claim that reliance on such espoused practices, which they refer to as canonical, can blind an organization's understanding and appreciation of the actual practices which include non-canonical practices such as work arounds.

Communities-of-practice emerge and evolve in organizations as practitioners collaborate and share work experiences through extensive use of narration about issues and problems involved in doing the job. Groups of practitioners develop their own, particular understanding of technologies, how they function, and what it takes to make them useful through a collective process of social construction. Membership is informal and based on participation in diagnosing situations and telling stories about them. The resulting communities are fluid rather than bounded, evolve rather than being designed, and typically cross the formal boundaries of an organization.

A conventional view sees systems development methods as prescriptions of practice to be selected and used in the situations to which they apply. Learning a method is in this view seen as transmission of explicit, abstract knowledge from "the head of someone who knows to the head of someone who does not in surroundings that specifically exclude the complexities of practice and the com-

munities of practitioners. The setting for learning is simply assumed not to matter" (Brown *et al.* 1991, p. 47). In contrast, communities-of-practice foster a kind of learning-in-practice in which the specific context and the shared experiences and understandings of its members shape how new methods are adapted, modified, and maybe rejected to transform present practices.

Communities-of-practice help us to understand important factors that support and foster learning and others that restrict or limit learning. To foster learning, an organization must conceive of itself as a community of communities, acknowledge the many non-canonical communities in its midst, go beyond its canonical practices, and allow some latitude in each community-of-practice in its attempts to go beyond the conventional wisdom. Brown & Duguid suggest more generally that enacting organizations are the most successful innovators. These organizations are proactive, highly interpretive, and create many of the conditions to which they must respond. They differ in this respect from discovering organizations that respond to their environments as they learn about them.

I see reflection-in-action and communities-of-practice as two complementary perspectives on professional practice. The first focuses on individuals while the second addresses groups and organizations. A key challenge in most contemporary organizations is to bridge this gab between individual and organizational learning (Argyris *et al.* 1978, 1990; Nonaka 1994). Based on extensive studies of individuals and groups Argyris & Schön have identified two different models of organizational learning. According to Model I, individual behavior is governed by the values of achieving goals, maximizing wins and minimizing losses, minimizing expression of negative feelings, and being rational. In contrast, Model II behavior is governed by valid information, free and informed choice, and internal commitment. Model I and Model II describe two different types of practices, where Model I is likely to be more efficient in dealing with structured situations and routine problems, while Model II supports problem solving behavior, learning, and innovation. Nonaka describes how organizational knowledge is created through a continuous dialogue between tacit and explicit knowledge. Knowledge is created in different patterns: socialization (from tacit to tacit), externalization (from tacit to explicit), internalization (from explicit to tacit), and combination (from explicit to explicit). While knowledge is created by individuals, organizations play an impor-

tant role in articulating and amplifying that knowledge. Nonaka identifies a number of key factors that induce commitment to learning: the intentions of the individuals involved, the autonomy of individuals and groups, and the fluctuations in the interaction between individuals and the external world.

Reflection-in-action and communities-of-practice are useful frameworks to make sense of, design support for, and improve environments and skills related to systems development. But they have a wider and equally important use because systems development itself is a change process that aims at understanding, supporting, and eventually improving practices of other professionals, i.e. users of computer-based information systems. In the early phases of the history of systems development (cf. table 2), when the purpose of using IT was to increase productivity and efficiency mainly by automating existing manual procedures, there were important differences between the practices of users and developers. User practices were typically structured and routinized and computers played a dominant role in shaping them. Development practices were less structured, including both problem solving and intervention, and they were, at that time, only modestly supported by computer usage. Today, these differences have vanished. Computer systems are used to support play, communication, work, and collaboration in a variety of human activities and systems development is just one special case amongst a multitude of professions that are supported. The presented concepts of practice, learning, and innovation are therefore important, not only to understand and improve systems development practices. They are equally important in systems development practice to understand and support the practices of users.

6.   Contributions

The Scandinavian trade union research was based on a knowledge building and dissemination approach, rather than on a traditional, academic publication approach. The primary users of the research were trade unions, rather than academics (Nygaard *et al.* 1975). In the DUE project, we wrote a number of books and reports starting in 1977, but our first scientific publication appeared as late as 1982 (Kyng *et al.* 1982). This tradition conditioned our approach to systems development research, in which the publication of practical knowledge in academic books came to play a dominant role. We see

practitioners as the primary users of our research, even though we have put increasing emphasis on traditional academic publications. From the early nineties, we decided to develop this strategy further by aiming at more publications in recognized journals.

Two other factors have influenced the contributions that have been developed through our research. We have tried to cover the central themes and questions related to systems development practice and we have given priority to those research questions that emerged as part of our involvement in practice (cf. section 3). The themes and contributions from Reflective Systems Development can be organized as shown in table 3, which, at the same time, explicate the underlying rationale and structure of this book.

The themes cover the key elements in systems development as defined in section 2.1: Systems development is a change process

Theme	Question	Contributions
Part I Understanding practice (totality)	How can we support reflection in and help shape understandings of practice?	Chapters 2, 3, 4, 5. Mathiassen (1981) Aaen (1989) Dahlbom *et al.* (1993)
Part II Managing processes (project level)	How can we support analysis, design, and management of development processes?	Chapters 6, 7, 8, 9. Andersen *et al.* (1986, 1990) Munk-Madsen (1987, 1996)
Part III Developing systems (project level)	How can we support analysis, design, and implementation of computer-based information systems?	Chapters 10, 11, 12, 13. Mathiassen *et al.* (1993, 1995, 1997) Stage (1989)
Part IV Improving environments (organization level)	How can we improve the environments that govern systems development practice?	Chapters 14, 15, 16, 17. Nielsen (1990a) Bang *et al.* (1991) Sørensen (1993)
Part V Professional development (individual level)	How can we mature the profession and support individual development of skills and attitudes?	Chapters 18, 19, 20, 21, 22.

Table 3. Research themes, questions, and selected contributions.

taken with respect to object systems in a set of environments by a development group to achieve or maintain some objectives. Our research has basically attempted to understand and appreciate practice as a complex and dynamic totality of human activities related to certain more or less explicit objectives. Based on such understandings, we have addressed the two complementary activities involved in practical systems development: the design and management of change processes, and the design and implementation of computer-based systems involving different object systems. Both of these activities focus on systems development on a project level. We have also studied the environments that govern and shape systems development practices and addressed problems and challenges related to improvements on an organizational level. Finally, we have concentrated on the individual level focusing on professional development and on the skills and attitudes needed in development groups.

Publishing contributions in the form of academic books have helped us engage in fruitful dialogue with practitioners and students. We have tried to cover central themes and questions in a number of such books. The theme of understanding practice is addressed in one book. "Computers in Context. The Philosophy and Practice of Systems Design" (Dahlbom *et al.* 1993) offers a framework for reflecting on systems, development, quality, and practices. The book addresses management and improvement of processes as well as design and implementation of systems. It relates contemporary issues within the computer profession to classical disputes in the western history of ideas. The objective is to help and encourage students and practitioners to reflect as part of their practice.

We have written two books on managing processes. "Professional Systems Development. Experiences, Ideas, and Action" (Andersen *et al.* 1986, 1990) offers concepts for understanding systems development practices, provides concepts and techniques to organize and manage systems development projects, and discusses how to organize and manage improvement efforts in a systems development organization. "Strategic Project Management" (Munk-Madsen 1996) contains an approach to project management based on ten years of experience as a consultant and trainer in industry practicing and further developing the ideas in (Andersen *et al.* 1986, 1990).

Developing systems has been addressed with three recent books. "Object Oriented Analysis" (Mathiassen *et al.* 1993), "Object Oriented Design" (Mathiassen *et al.* 1995), and "Object Oriented Analysis and Design" (Mathiassen *et al.* 1997) offer a conceptual framework with principles and guidelines for the design and modeling of computer systems based on object oriented concepts. The methodology is adaptable. It is designed to help practitioners integrate elements from other methodologies and tailor their specific approach to the characteristics of the design situation. The main emphasis is on understanding and developing computer systems.

Finally, we have addressed the theme of improving environments with a book. "Quality Management in Systems Development" (Bang *et al.* 1991) offers an organizational and practical framework for quality management. The book discusses technical, organizational, and cultural issues related to the design and management of quality assurance programs in IT organizations, and it offers specific concepts and techniques to help design quality assurance activities for systems development projects. The main emphasis is on improving practice.

In addition to these books, table 3 contains some of the scientific contributions to Reflective Systems Development. (Mathiassen 1981), (Munk-Madsen 1987), (Aaen 1989), (Stage 1989), (Nielsen 1990a), and (Sørensen 1993) are Ph.D. theses written by the participants. The chapters in table 3 refer to the selection of scientific papers contained in the remainder of this book. These papers were written in close collaboration between myself and my colleagues. Together with the books mentioned above—except (Munk-Madsen 1996)—they document my own research contributions to Reflective Systems Development. As an introduction to each subsequent part of this book, I provide a summary of the contained contributions and of related contributions made by my Danish colleagues Ivan Aaen, Niels Erik Andersen, Finn Kensing, Andreas Munk-Madsen, Peter Axel Nielsen, Jan Stage, and Carsten Sørensen. These contributions are all contained in the list of references.

7.   Relations

The previous sections present a personal interpretation of our experiences. In the remaining part of the paper, I contrast and discuss the resulting approach to research and practice by relating it to its

context (cf. section 2). In this section, I focus on related research by first pointing out the primary sources that have helped to shape our approach, then by contrasting our approach with other similar, but different approaches to research and practice, and finally by discussing the underlying paradigm of our approach.

7.1. Sources of inspiration

We have discussed in detail how Schön's ideas on reflection-in-action and Israel's conception of dialectics have been used as a basis for our research. In addition, a number of other key inspirations need to be mentioned. Naur's notion of programming as a human activity (1992) can be seen as an attempt to establish reflection-in-action as an approach to the core activity of computing. His work has in numerous ways helped us understand key issues in computing, e.g. programming and formalization (Naur 1982, 1985), and inspired us to explore computing practices (Naur 1972, 1983). Many of the contributions of Boehm, Parnas, and other software engineering researchers can in a similar way be seen as attempts to support reflection-in-action in software development. Boehm has mainly inspired us to better understand and manage complex development processes (1981, 1984, 1988), whereas we have used Parnas' ideas to support reflections on software systems (1972) and to understand how these relate to the development process (1986). From software engineering we have also used Weinberg's organic problem solving approach (1986) to understand what it takes to manage and organize technical processes (e.g. Weinberg *et al.* 1982) and to appreciate the limits to applying mechanistic approaches to social phenomena (Weinberg 1982).

Reflective Systems Development involves learning and organizational problem solving. Checkland's work on SSM (1981, 1990) has played an important role in developing our view on systems and systems thinking, and it has served as a useful methodology for action research and reflection-in-action. Further insights stem from Argyris and Schön's work on organizational learning (1978, 1996) and from March and Olson's concepts related to ambiguity and choice in organizations (1976). March and Olson's model of decision processes has helped us to understand decision making in systems development, and it has shown us how to develop practically useful interpretive (rather than normative) models of organizational activities. We have also used Simon's notion of problem solving as satis-

ficing behavior (1955, 1956, 1957, 1982) even though it "is a radically incomplete description of what engineers, agronomists, and physicians actually do" (Schön 1983, p. 170; see also Lanzara 1983). We have used Simon as a basis for understanding the dialectics between construction and evolution approaches—at the same time acknowledging the need to go beyond construction and evolution to practice intervention (Dahlbom *et al.* 1993; chapters 6, 21).

Finally, we have interacted closely with and learned from researchers whose contributions can be seen as expressions of a reflective approach to systems development. Notable examples are Bjerknes' work on dialectics as a practical framework for management of systems development (1989, 1991), Madsen's work on the use of metaphors in systems design (1989, 1994), and Stolterman's empirical study of the ways in which systems designers think in practice (1991, 1992).

7.2. Related positions

Our approach has grown out of the Scandinavian trade union research (Carlson *et al.* 1978; Bødker *et al.* 1987; Bjerknes *et al.* 1987b, 1995; Ehn 1988; Greenbaum *et al.* 1991; Kyng 1996) and we have throughout our work been in close interaction with and inspired by the work of Floyd, Züllighoven, and their colleagues (Budde *et al.* 1984; Floyd 1984, 1987; Floyd *et al.* 1992; Budde *et al.* 1992). In relation to each of these two approaches, there are important similarities and differences.

Greenbaum and Kyng describe the shifts in focus in systems development going from traditional approaches to an approach based on cooperation with users (see table 4; Greenbaum *et al.* 1991, p. 16; Kyng 1996, p. 9). Like the cooperative approach—which originated in and to a large extent is based on the Scandinavian trade union projects—Reflective Systems Development regards the categories in the right column as more fundamental to understanding practice than those to the left. This similarity is expressed in the intellectual frameworks of the two approaches. One of the often quoted sources in the cooperative approach is, for example, Suchman's work on plans and situated actions (1987), an anthropological study of the problem of human machine communication. Even though Suchman makes no reference to Schön's ideas on how professionals think in action (1983) her perspective is very much that of reflection-in-action: "planned, purposeful actions are inevitably

Traditional approach	**Cooperative approach**
• *focus is on*	• *focus is on*
problems	situations and breakdowns
automation	support
information flow	social relationships
formal procedures	situated work
describable skills	tacit skills
expert rules	human expertise
individuals communicating	group interaction
rule-based procedures	experience-based work

Table 4. Differences in focus between traditional and cooperative approaches (Greenbaum et al. 1991).

situated actions . . . taken in the context of particular, concrete circumstances . . . our actions, while systematic, are never planned in the strong sense that cognitive science would have it. Rather, plans are best viewed as a weak resource for what is primarily *ad hoc* activity. It is only when we are pressed to account for the rationality of our actions, given the biases of European culture, that we invoke the guidance of a plan." (Suchman 1987, p. viii).

Reflective Systems Development and the cooperative approach are, in this sense, quite similar. There are, however, important differences. The cooperative approach is based on action research with *users*, and the categories in table 4 are primarily used as alternative perspectives on the use environment. Reflective Systems Development is based on action research with *IT professionals*. It shares perspectives with the cooperative approach, but the perspectives are primarily applied to the development environment. The two approaches have, in other words, quite similar ways to look at computer systems, knowledge, and work. But their primary focus is on two different types of work: that of users of computer systems as opposed to that of professionals developing computer systems. This difference in primary focus is also expressed in the intellectual frameworks. Suchman is primarily concerned with the problem of human machine communication whereas Schön's focus is on how professionals think in action.

These considerations suggest that Reflective Systems Development and the cooperative approach are complementary and pos-

sible to combine. The difference in context has, however, implications that need to be taken into account. The cooperative approach is influenced by well-established work settings, e.g. office work, graphical work, production work, and nursing, that have been shaped by technological and organizational interventions throughout the last century. In contrast, Reflective Systems Development has been situated in relatively young and emerging development organizations with fewer traditions, employees with higher education, and a less developed tradition for bureaucratic arrangements. The underlying intention of the cooperative approach have been to use the increasing application of computer technology to reestablish working environments in which qualifications and experience, rather than procedures and systems, are seen as primary factors and where interaction and communication between employees are viewed as preconditions for developing quality services and products. From this perspective, the primary role of systems developers are those of emancipators and facilitators (Dahlbom *et al.* 1993; chapter 21), taking more or less for granted their ability to develop software and manage projects. The underlying intention of Reflective Systems Development has been to develop the systems development profession from a rather ad-hoc and at times chaotic discipline into a discipline that is respected by users, clients, and society for its commitment and ability to deliver quality services. As a consequence, software engineer and project manager are included as important roles.

The assumptions underlying the approach of Floyd, Züllighoven, and others are outlined by Floyd in her discussion of paradigm changes in software engineering (1987). Floyd contrasts a product-oriented perspective with a process-oriented perspective. She refers to these perspectives as paradigms, in the sense of Kuhn (1962), "since they provide a frame of reference suggesting questions we ask, quality considerations we aim for, and guidelines for interpreting results in our scientific and technical work" (Floyd 1987, p. 195). Some of the distinctive features of the two paradigms are outlined in table 5. Floyd stresses that her intention is to "contrast two perspectives and not two communities of people holding these perspectives. We all argue and act from both of these perspectives. The criticism implied in my argumentation refers to the fact that existing methods and scientific approaches in software engineering embody the product-oriented view almost exclusively. Working towards

	Product-Oriented	**Process-Oriented**
Programs	• formal objects • stepwise refinement • correctness	• tools • learning • adequacy
Systems	• implementation • closed • static	• integration • open • evolving
Quality	• product-related • tests and proofs	• use-related • experiments and use
Software development	• one system • software system • maintenance separate	• sequence of versions • information system • includes maintenance
Programs, errors, and user competence	• program given • operating programs • errors are technical • due to incompetence	• adaptable system • supporting work • errors are social • initiates learning
Understanding programs and the role of documents	• document driven • formal semantics • one formalism • context free	• experience driven • social semantics • natural language • examples
Methods	• uniform results • limit interaction • generally applicable • context-independent	• unique processes • support interaction • application area • tailorable

Table 5. Two perspectives on software engineering based on (Floyd 1987).

overcoming this is both highly imperative, considering the role of IT in the living human world, and also inspiring." (1987, p. 208).

Reflective Systems Development and Floyd and Züllighoven's approach share a strong influence from software engineering research. Floyd's process-oriented perspective expresses many of the assumptions underlying Reflective Systems Development (see for example chapters 6, 7, 11, 12, 19, 21; Dahlbom *et al.* 1993). Our dialectic position—which includes strong mechanistic elements but is basically founded on a romantic world view (Dahlbom *et al.* 1993; chapter 21)—is also similar to Floyd's synthesis of the product- and process-oriented perspectives. The major difference between the two approaches is that Floyd and Züllighoven have developed their approach based primarily on insights from program development and

prototyping—both in laboratory and industrial settings—whereas Reflective Systems Development have been driven by empirical studies of systems development processes and organizations. The primary preoccupation of Floyd and Züllighoven has been inventive: to develop new and useful concepts and tools to support software development. Our primary preoccupation has been empirical: to understand and improve practices.

7.3. Underlying paradigm

Reflective Systems Development, or partial views on the approach, are discussed in a number of paradigmatic surveys (Hirschheim *et al.* 1989, 1991, 1992, 1995, 1996; Iivari & Lyytinen 1997). The most elaborate discussion is found in (Hirschheim *et al.* 1995). They understand paradigms as the "fundamental set of assumptions adopted by a professional community which allow them to share similar perceptions and engage in commonly shared practices" (p. 46). A paradigm consists of two kinds of assumptions, those associated with the way in which systems developers acquire knowledge (epistemological assumptions), and those which relate to their view of the social and technical world (ontological assumptions). Following Burrell & Morgan (1979) and building on earlier discussions (Hirshheim *et al.* 1989) they classify these two types of assumptions along the subjectivist-objectivist dimension and the order-conflict dimension yielding four paradigms:

- Functionalism (objective, order).
- Social relativism (subjective, order).
- Radical structuralism (objective, conflict).
- Neohumanism (subjective, conflict).

The objectivist-subjectivist dimension addresses the views of knowledge in approaches to systems development research and practice. The objectivist position minimizes the role of the observer by applying models and methods derived from the natural sciences to study human activities. In contrast, the subjectivist position uses subjective experiences as the primary source for learning about social phenomena. The order-conflict dimension addresses the built-in assumptions that are made about the real world. The order position emphasizes a social world characterized by order, stability, integration, consensus, and functional coordination, whereas the conflict position stresses change, conflict, disintegration, and power.

Hirschheim *et al.* use the resulting ideal types to structure, make sense of, and contrast the contents of scientific papers, academic books, and empirical cases within our field. Specific approaches, e.g. Reflective Systems Development or the cooperative approach (Greenbaum & Kyng 1991; Kyng 1996), focus on a particular group of practitioners and researchers, their activities and assumptions, and their accounts of systems development concepts and practice. At the most concrete level we find practice. At a more abstract level we find approaches as abstractions over the efforts of a particular group of actors. And at the most abstract level we find paradigms, in the sense described above, as abstractions explicating particular conceptual and philosophical foundations. Each specific activity in systems development research and practice represents one or more approaches, and each approach represents one or more paradigms. Paradigmatic analyses do not lead to simple classifications. Instead, they use the abstract distinctions to characterize the primary and secondary features of each approach.

Reflective Systems Development is interpreted as a social relativist approach with additional functionalist and neohumanist features (Hirschheim *et al.* 1995, p. 50, 128–138). The practice orientation of the approach and the strong relations to Schön's ideas on reflection-in-action (1983, 1987) contribute to a subjective orientation, which means that the approach is primarily social relativist or neohumanist. A strong focus on the professional team of systems developers—rather than on the organization as a whole or on the interactions between the different groups of actors involved in systems development—has in many ways drawn our attention towards effective teams rather than conflictual relationships. This supports a primarily social relativist interpretation of the approach, in particular for those elements which relates closely to the organization and functioning of smaller groups of professionals (e.g. Andersen *et al.* 1986, 1990; chapters 2, 3).

A number of other features do, however, point in direction of a position which is primarily neohumanist. These include the roots in Scandinavian trade union research, the use of dialectical reflection, the explicit inclusion of conflict in both Schön's and our own notion of the reflective practitioner, and our elaborations of soft systems thinking (chapters 10, 11, 12, 13). Reflection in terms of contradictions and explicit recognition of social conflicts have been part of the approach since its very start (Mathiassen 1981) and they have re-

mained key elements of our world view (chapters 4, 10, 11, 18; Dahlbom *et al.* 1993).

A further discussion of this issue must include emancipation as a key value of neohumanism, both in the sense of realizing the full creative and productive potential of individuals, and in the wider sense of establishing "social conditions, which encourage effectiveness through organizational democracy, specifically overcoming existing forms of authoritarianism and social control if they perpetuate inequities of the status quo in the work place" (Hirshheim *et al.* 1994). Reflective Systems Development shares these concerns on the individual level (Dahlbom *et al.* 1993; chapters 2, 3, 14, 18, 22). In the wider, organizational sense we focus on informal rather than formal aspects of practice (chapters 4, 15, 16), the use of mixed and tailored approaches based on risk management rather than the rigorous use of general methods (Mathiassen *et al.* 1993, 1995, 1997; chapters 6, 7, 8, 9), and on organizational learning and intervention rather than instrumental views of technical implementation (chapters 10, 11, 15, 16). These views are all part of our philosophy for management of systems development which aims at creating and maintaining "environments in which practitioners become empowered" (Weinberg 1986). The extent to which these views and approaches are expressions of an emancipatory position is, however, quite difficult to determine based on theoretical considerations. A similar discussion of the underlying paradigm of SSM led to rather strong disagreements on exactly that issue (Jackson 1982, 1983; Ackoff 1982; Checkland 1982; Churchman 1982; Mingers 1984). Whether Reflective Systems Development is primarily social relativist or neohumanist remains, to a large extent, a question that should be answered based on specific practices.

Reflective Systems Development has, in addition, quite outspoken functionalist features. The professional focus implies that data (as opposed to information), software systems (as opposed to tools to support work), and construction and evolution (as opposed to intervention) all are seen as important, and integral elements of practice. This position contributes to the functionalist features of the approach and—as we have expressed it—to the ambition to become skillful engineers with core competence in the use of IT based on a romantic world view (Dahlbom *et al.* 1993; chapter 21). Finally, it is fair to say, that radical structuralism has played a minimal role in Reflective Systems Development.

8. Discussion

In this final section, I discuss some of the lessons we have learned and relate them to other research results, I address criticisms raised by others and point to possible future developments, and I summarize the argument of the paper.

8.1. Position in context

My experiences can be expressed as a number of lessons about systems development research and practice (see figure 4). These lessons summarize the presented concepts and viewpoints (cf. figures 1, 2, and 3), and they describe key positions of Reflective Systems Development. In the following, I discuss each of these lessons and place them into a wider research context.

Lesson 1: Reflection-in-action provides a useful understanding of systems development practice.

Lesson 2: Methods are primarily frameworks for learning.

Lesson 3: Improving practice requires an experimental attitude and an appreciation of the involved communities-of-practice.

Lesson 4: Sustainable project management traditions are fundamental in successful improvement strategies.

Lesson 5: Action research brings relevance to the research process while supplementary approaches improve the validity and reliability of the research results.

Lesson 6: Dialectics is a useful framework for understanding practice based on different reference disciplines.

Lesson 7: Combining interpretations with interventions and normative propositions increases the relevance of the research.

Lesson 8: Research and teaching are relevant practices for learning about systems development.

Figure 4. Key lessons in Reflective Systems Development.

Lesson 1: Reflection-in-action provides a useful understanding of systems development practice. This lesson is supported by a number of empirical studies. In an early study, Boland compares a traditional, structured design approach in which the designer asks questions, analyzes responses, and makes suggestions, to an open interaction between designer and user with initial sharing of information

and mutual learning (Boland 1978). The second approach, in which the actors reflected on the situation and learned as they went along, produced higher quality designs with important implementation advantages. These findings have been confirmed and further elaborated by Tan (1994) concluding that effective communication and interaction in systems development is the outcome of a complex process that is influenced by the personal and situational characteristics of the participants (see also Newman *et al.* 1990).

Quite a number of studies supplement these findings by pointing at the great variety of skills, attitudes, and approaches needed by both individuals (e.g. Vitalari *et al.* 1983, Vitalari 1985) and teams (e.g. White 1984) to successfully design systems. Vitalari & Dickson suggest that successful practitioners develop a flexible strategic approach in their problem-solving behavior and encourage analysts to keep a diary of strategies they have used in the past and create new strategies as situations develop (Vitalari *et al.* 1983, p. 954; cf. chapter 3). White compares the performance of two teams and concludes that the least successful gave priority to facts and possibilities, impersonal analysis, matter-of-fact approaches, logical thinking, and technical skills and development. In contrast, the approach used by the more successful of the two teams included personal relations, the importance of mutual help and support, and an enthusiasm and willingness to learn that helped to support interaction and increase understanding.

Our first lesson is further supported by the extensive studies carried out by Curtis *et al.* (Elam *et al.* 1987; Guindon *et al.* 1987; Krasner *et al.* 1987; Curtis *et al.* 1988; Walz *et al.* 1993). These researchers found that experienced software designers primarily make use of a repertoire of past experiences and adapt the best fitting of these to the present design situation (Guindon *et al.* 1987). They also found that context-sensitive learning is important and that much of the information that needs to become part of a team's memory is not captured formally in the project documentation. They recommended active promotion of acquisition, sharing, and integration of knowledge between practitioners (Walz *et al.* 1993). In summary, they see the development of large software systems as a learning, communication, and negotiation process, in which tools and methods must accommodate change and support representation of uncertain design decisions, and in which the specific environ-

ments must become a medium for communication to help integrate people, tools, and information (Curtis *et al.* 1988).

Finally, there are a number of Scandinavian studies that support the first lesson. Stolterman's practice studies of how designers think about design and methods reveal that the rationality of the process is different from the abstract and simplistic rationality of methods and that the rationality of the process is something that emerges as a result of the confrontation between the designer and the specific design situation (Stolterman 1991, 1992). Bjerknes suggests more specifically that the tensions and contradictions involved in specific project situations constitute an important source of insight and development that can help form and manage systems development projects (Bjerknes 1989, 1991).

Lesson 2: Methods are primarily frameworks for learning. This second lesson is a corollary of the first. I have chosen to emphasize this experience because we need a practically useful and not overly ambitious understanding of the role of methods within our profession. Quite a number of studies show that there are major differences between the concepts, beliefs, values, and normative principles of methods (cf. section 2.1), and the ways in which experienced practitioners work (e.g. Guindon *et al.* 1987; Stolterman 1991, 1992; Bansler *et al.* 1993). The implication of such findings is that we should not primarily understand methods as prescriptions of practice. When methods are used in practice, they are adapted, used selectively, combined with other methods, and complemented by experience and reflection-in-action.

Based on our own experiences in developing methods (Mathiassen *et al.* 1993, 1995, 1997; chapter 14) we have come to understand methods primarily as frameworks for learning and only secondarily as frameworks that can be adapted to practice. This view is supported by an exploratory study of Kozar in which he found that inexperienced practitioners, who had spent less time in the organization, didn't have alternative methods they would have to give up, and perceived themselves as risk takers, are the ones that are most willing to adopt new methods (Kozar 1993). Seeing methods primarily as frameworks for learning emphasizes the difficulty involved in transforming current practices, and it makes us appreciate both the possible, but also the limited, roles that methods can play in practice (Fitzgerald 1996).

This view of systems development methods is similar to that of Avison & Wood-Harper (1991) in that it stresses the importance of adapting and combining several approaches to support understanding of the different objectives and object systems involved in practice (cf. section 2.1). One of their contributions, Multiview (Avison *et al.* 1990), is formulated as a meta-method, helping practitioners to adapt and combine methods to analyze and design socio-technical aspects, human-computer interaction, and the technical aspects of computer-based information systems. Reflective Systems Development is, in contrast, a general approach to systems development practice and research. It is, compared to Multiview, less normative, not as oriented towards specific activities and perspectives involved in systems development, and its application area is wider including research.

Lesson 3: Improving practice requires an experimental attitude and an appreciation of the involved communities-of-practice. Improvement of practices in systems development is a specific type of organizational change that involves new applications of IT to support management, coordination, and specification. It is, in general, widely accepted that organizations cannot be changed successfully based solely on rational approaches. Successful organizational change involves learning, iteration, and emergence of new ideas and initiatives as the process evolves (March *et al.* 1976; Argyris *et al.* 1978; Schein 1985; Borum 1995). This experience has been expressed in different ways focusing specifically on changes that are driven by applications of IT. McKenney & McFarlan (1990) have, for example, proposed a model in which they combine Gibson & Nolan's stage hypothesis for organizational maturity in using IT (1974) with Schein's conception of organizational change (1985). The resulting model combines experiments with appropriate management and control measures (see chapter 16). Experiments are needed to support learning and adaptation, but too little management can result in stagnation of the change process. Appropriate control measures are used to ensure the cost effectiveness of the change process, but a process that becomes too focused on control can inhibit successful diffusion of the applications in question.

Given such general knowledge about organizational change and IT assimilation, it is not surprising to find that successful efforts to improve practice in systems development require an experimental attitude as well as an appreciation of established and

emerging practices and beliefs. This point is developed by Ciborra & Lanzara based on a field study of improvement efforts in systems development in a large European computer manufacturer (1994). They use the concept of *formative context* to make sense of the observed failures and successes. A formative context corresponds quite closely to the notion of communities-of-practice. It is a set of institutional arrangements and cognitive imageries that inform the actors' practices and reflections in organizations. Formative contexts are rather stable expressions of particular practices, but they are, at the same time, greenhouses in which experiments for organizational change take place and new practices emerge. Ciborra & Lanzara found that a key to explain the failures and successes of the improvement effort was the identification of a preexisting formative context (based on established work routines and hierarchical arrangements) and an emerging formative context (based on new methods and network arrangements). They suggest that limited appreciation of the formative contexts involved in improving practices and limited capability to support experiments to challenge and develop established practices and beliefs are responsible for many of the failures involved in changing practice.

Lesson 4: Sustainable project management traditions are fundamental in successful improvement strategies. In our own studies, we found that even ambitious and intensive improvement efforts often fail to go beyond the espoused level of practice, because of the highly dynamic and complex nature of systems development. Projects are confronted with changing requirements and technologies, fixed budgets, high task uncertainty and complexity, extensive separation of concerns, and the demand for intensive coordination efforts. The resulting turbulence creates an environment with little or no room for experimenting with new methods and tools and for learning new practices. This led us to emphasize establishment of sustainable project management traditions as a fundamental element in successful improvement of practices (Andersen *et al.* 1986, 1990; Munk-Madsen 1987). The established traditions should be sustainable in more than one sense. They should help the actors to focus on the task at hand and on their own continuous learning, and they should do so with both endurance and strength, so as to effectively address the dynamic and complex nature of systems development practice.

A similar position is taken in the Capability Maturity Model (Humphrey 1989). The experiences so far with this improvement strategy show that approximately 80% of all development organizations operate in an ad-hoc fashion without appropriate project management traditions, and that actions addressing this situation help establish a situation in which continuous learning and improvement can take place. This particular model suggests focusing on the following management disciplines: project planning, project oversight, requirements management, configuration management, management of subcontracts, and quality assurance. Adoption of disciplines like these will, in terms of Weinberg's (1986) philosophy for technical management, help establish an environment in which the actors become empowered to focus on development of useful systems and on improvement of their own practices.

Lesson 5: Action research brings relevance to the research process while supplementary approaches improve the validity and reliability of the research results. Action research has been used as a successful strategy to software engineering research (e.g. Parnas *et al.* 1986; Knuth 1989). However, only little attention is paid to issues related to research approaches within software engineering, and many research efforts are based on action research without this being mentioned or explicitly discussed. Within organization and information sciences we find a more articulate debate of research methods. Baskerville & Wood-Harper (1996a) date the origin of action research back to Lewin's studies just after the second world war (1947). They argue that information systems is a very appropriate field for the use of this particular approach which merges research and practice in a way that produces exceedingly relevant results.

A variety of action research approaches can be identified depending on the underlying process model, the structure imposed on the process, the way in which the researchers are involved in practice, and what are considered to be the primary goals of the process (Baskerville *et al.* 1996a). Our own approach has grown out of the Scandinavian trade union research tradition. It has been further developed and inspired by Argyris and Schön's approach to organizational learning (1978, 1996) and Checkland's merger of systems science and action research (1981; Checkland *et al.* 1990). In our approach, the process is mainly iterative and reflective, the structure is fluid, the researchers are involved in close collaboration with practitioners, and the goals are mixtures of professional training,

organizational development, and building of experience-based knowledge about systems development.

Baskerville & Wood-Harper discuss several problems related to action research, including the dilemma between practical goals and research goals, the possible bias implied by the funding structure of the research, and the increasing association of action research with consulting. There is definitely a risk that action research degenerates into consulting. To prevent this from happening, we have found it useful to use experiments and practice studies as supplementary approaches. Doing so can help the researchers maintain an appropriate level of discipline and obtain sufficient rigor in terms of validity and reliability of the results (Munk-Madsen 1986). The validity of the results expresses the degree to which the results are in accordance with the actual experience. The reliability expresses the degree to which similar results could be obtained by others.

Lesson 6: Dialectics is a useful framework for understanding practice based on different reference disciplines. Systems development is a process which aims at changing organizations through use of IT (see section 2.1) and which itself is subject to continuous changes as part of ongoing technological development (see section 2.2). This highly dynamic and emerging nature of systems development is widely recognized and accepted (Lyytinen 1987a; Avison *et al.* 1988, 1990; Lyytinen 1989). In addition, systems development is in many ways a political process (Markus 1983; Robey *et al.* 1984; Newman *et al.* 1985), systems developers face many conflicts and contradictions (Bjerknes 1989, 1991), contradictory issues and traditions are involved in improving practice (Ciborra *et al.* 1994), systems development methods are formulated in contradictory terms (Beath *et al.* 1994), and research produces contradictory results (Fenton 1993; Robey 1995). These findings strongly suggest that we need intellectual frameworks in systems development that both focus on change and support understanding of contradictions.

Robey examines four theories or reference disciplines—political theory, organizational culture, institutional theory, and organizational learning—that explain contradictions and which are likely to account for the observed contradictions related to the use of IT (1995). Each theory provides a conception of contradiction by including forces both encouraging and restricting organizational change. Robey also suggests that these theories may be related using structuration theory (Giddens 1984), a general social theory fo-

cused on the contradictory and intrinsic relations between action and structure, as a framework. Robey argues that a search for explanations of the observable contradictions related to IT use will likely produce more conceptual and methodological diversity (1995, 1996). This represents a threat to the coherence of the research community, but it contributes, at the same time, to advancing the field towards a better understanding of observed phenomena.

Robey's position expresses well the underlying rationale for our use of dialectics and a number of reference disciplines. Dialectics is practiced in many different forms, and structuration theory is itself one such instance aimed at understanding the contradictory relations between action and structure in social contexts. We have used Israel's dialectic approach to social science (1979) to make sense of the observed contradictions in practice, in particular those between processes and structures (Mathiassen 1981; chapter 4). We have found that dialectics helps create a certain level of coherence in our thinking even though we apply and combine different reference disciplines. In our approach, dialectics plays a role similar to that which systems theory plays in SSM (Checkland 1981; Checkland *et al.* 1990). Dialectics supports systematic thinking about practice and it serves as an intellectual framework that can house both hard and soft systems thinking (Dahlbom *et al.* 1993; chapters 4, 10, 11, 12, 13).

Lesson 7: Combining interpretations with interventions and normative propositions increases the relevance of the research. Studies of organizational culture provide us with a deep understanding of organizational practice. Schein (1985) suggests that we distinguish between three levels of organizational practice: (1) what we can immediately observe, i.e. the formal aspects of practice which are constituted by procedures, structures, buildings, technology, methods, and programs, (2) what the involved actors believe and think, i.e. the level of espoused theories (Argyris *et al.* 1978), and (3) what the actors actually do, i.e. the deeply rooted assumptions that govern the behavior of the involved actors and which correspond to their theories-in-use (Argyris *et al.* 1978). From a research point of view, the first two levels of organizational practice are more readily available than the last, but the last is of primary importance if we want to understand and improve practice (Argyris *et al.* 1978; Schein 1978; Brown *et al.* 1991; Ciborra *et al.* 1994). One of the key ways in which we can gain insight into the basic assump-

tions of an organizational culture is by trying to change it (Schein 1978). There is a fundamental dialectic related to research into systems development practice: if we want to understand practice, we need to change it, but to change it in a sensible direction, we need to understand it.

This insight is reflected in the canonical form of action research which involves diagnosing practice, action planning to improve practice, action taking, evaluation of outcomes, and specification of learning (Baskerville *et al.* 1996a). This canonical form describes the activities involved in each specific case of intervention and it is an important element in our approach to research (cf. figure 3). In our research approach, interpretations play a major role in diagnosing practice, in evaluating outcomes, and in specifying learning. Also, interventions are involved when specific actions to improve practice are planned and taken. Finally, design of normative propositions is involved when actions are planned to improve a specific situation. But such designs can also be part of the specification of learning and thereby bring experiences beyond the specific case.

Interpretation of practice, design of normative propositions, and interventions into practice are, more generally, the basic activities involved in practice-based systems development research. Each case of intervention is tightly related to a specific context, and one of the key challenges is to develop knowledge that is more generally useful. To do so, we must engage in several, different cases over time (Baskerville *et al.* 1996a). We must accumulate experiences as a combination of general interpretations of practice (e.g. concepts and theories) and normative propositions (e.g. principles or methods) that support and improve practice. Design of normative propositions is an excellent opportunity to rethink and evaluate the underlying theories. Subsequent interventions, in which principles and methods are used to support practice, provide valuable feed-back on the relevance of research results.

Lesson 8: Research and teaching are relevant practices for learning about systems development Researchers are themselves practitioners. They all practice research, but most also practice teaching, some practice consulting, and some practice systems development. In each of these different professional roles the researcher is confronted with first hand experiences that in many cases are relevant for systems development research. Some re-

50

searchers have used their own practice as programmers or systems developers as the basis for their research (Naur 1972; Parnas *et al.* 1986; Knuth 1989). Other researchers have used their teaching of systems development as a basis for research (Boehm *et al.* 1984; Selby *et al.* 1987; Iivari 1997; chapters 3, 7). But we can also, in a more fundamental and simple way, use our own practice as inspiration and to learn what works in practice.

Research involves quite a number of practices that relate to systems development. Research is often carried out in projects in which various artifacts (e.g. theories, methods, prototypes, scientific papers, and books) are produced in collaboration with different actors. Practice-based research involves analysis of work (i.e. systems development) and of the use of technology (e.g. CASE tools). The research process is itself supported by technologies that are used to coordinate activities and produce results. Quality and quality assurance are key elements in research on all levels. Finally, improvement of practice is, or should be, one of the important goals of any research unit.

Teaching systems development involves, in a similar way, activities which relate to systems development. Students develop programs and systems. Students are encouraged to reflect on what they learn in such projects. Computer-based artifacts have to be evaluated. Communication, interaction, and learning are key ingredients in establishing appropriate educational environments. Finally, teaching practices are constantly being developed and improved to reflect changes in the use of IT.

In their daily activities, researchers are engaged in situations that provide rich sources for learning about systems development and good opportunities to experiment with systems development knowledge. This lesson suggests that researchers and teachers practice what they preach and include into the research process critical reflections on their own experiences.

8.2. Criticism and future development

Hirschheim *et al.* (1995) suggests that the primary strengths of Reflective Systems Development relate to its practice orientation, its emphasis on learning, its intrinsic relation between reflection and action, its process orientation, and finally its strong focus on experimentation (p. 132–134). Their analysis also suggests a number

of relevant weaknesses (p. 134–137) which I will briefly review in the following

1. Lack of clarity and structure. This is a fair criticism as there is no comprehensive presentation of the approach. However, this book is an attempt to remedy this situation and make available a complete survey of the contributions, related work, and underlying assumptions of Reflective Systems Development. Furthermore, from a more practical point of view, we have put great emphasis on presenting clear and useful concepts, principles, and guidelines that can help practitioners and students reflect upon and improve their practices (Andersen *et al.* 1986, 1990; Bang *et al.* 1991; Dahlbom *et al.* 1993; Mathiassen *et al.* 1993, 1995, 1997; Munk-Madsen 1996).

2. Lack of attention given to the cost and effectiveness of the approach. This criticism points at a worthwhile and challenging task. Reflective Systems Development is not itself a systems development method or, like Multiview (Avison *et al.* 1990), a meta-method. Rather it is a unified framework for studying and practicing systems development. As pointed out by Hirschheim *et al.*, it hardly makes sense to design metrics and measure the efficiency and effectiveness of such a framework. But longitudinal studies of the effects of using specific strategies, methods, or techniques based on Reflective Systems Development would provide valuable insights.

3. Simplistic view of what it takes to improve practices. This criticism addresses an issue that needs to be explored and integrated further into our approach. Part of our work is quite focused on communication and reflection and do not elaborate on institutional, cultural, and political perspectives (e.g. chapters 2, 3), while other parts of our work discuss these issues in some detail (e.g. chapters 15, 16, 17, 18; Bang *et al.* 1991; Dahlbom *et al.* 1993).

4. Limited understanding of the role that bias and prejudice play in human understanding. This criticism relates to the previous point and it raises additional issues that needs to be developed as more integral parts of the approach. In part of our work (e.g. Andersen *et al.* 1986, 1990) we have not provided in-depth treatment of the difficulties involved in transcending traditions, whereas the issue has been dealt with in more detail in other parts (e.g. Dahlbom *et al.* 1993, in particular chapters 6, 8, 9, 11).

5. Lack of an explicit notion of what it means to change professional practices to the better. This criticism raises a relevant and difficult problem. The position we have taken so far is similar to the

pragmatic view of SSM (Checkland 1981; Checkland *et al.* 1990). According to SSM, the value of improvement efforts are highly dependent on the situation and the involved actors. Planned improvement efforts are debated as possibly relevant and implemented efforts lead to new situations, in which we can experience their value. Recent attempts to support software process improvement (Humphrey 1989; Paulk *et al.* 1993) take a different position and suggest generally applicable models of ideal software processes. We need to further explore, both theoretically and practically, how such models and evaluations of improvement initiatives can be used as part of continuous learning processes.

6. Lack of explicit, theoretically well-founded principles of participation and project organization. This is a fair criticism. The strong professional orientation of the approach has lead to a rather narrow perspective, which needs to be elaborated further with notions of participation, collaboration, and mutual learning and with more explicit views of the roles played by users and managers. Important initiatives in that direction have been taken by Andersen *et al.* in their comprehensive framework to understanding and managing the contractual relationships involved in systems development (1996).

The criticisms of Hirschheim *et al.* (1995), together with my own investigation in this chapter, points to a number of possible avenues for further development of the approach. The identity of Reflective Systems Development is strongly related to its professional focus, but there is a need to broaden the *context* covered by the approach. The traditional focus on projects needs to be more strongly supported by organizational perspectives and managerial practices, and the traditional focus on professional teams needs to be further developed with experiences and frameworks emphasizing the interactions between different groups of actors involved in systems development efforts.

The objective of the approach is to understand, support, and improve systems development efforts. Further action research on process *improvement* could be used to develop a deeper and more practical understanding of the complexities and dynamics involved in making development organizations learn and change to the better. Process improvement efforts address the most important challenge faced by the profession, i.e. to increase its credibility. Practical involvement in such efforts would also allow us to experiment with

contemporary systems development techniques and methods as part of specific improvement efforts.

The theoretical foundation of Reflective Systems Development is still informed by relatively few sources. Further elaboration, including additional *theory* on technology, management, and organizations, would help us gain a deeper understanding of systems development practices and the challenges involved in transforming and transcending them. We could investigate the possible use of institutional theory as a reference discipline that could complement the organizational learning and organizational culture perspectives, and we could look closer at the possible use of structuration theory to further elaborate the dialectical basis of our intellectual framework (Robey 1995).

Finally, any practice oriented research effort on systems development has to reflect on the technological conditions under which the profession is practiced (see section 2.2). Contemporary *technology* is, needless to say, quite different from the technologies twenty years ago. Networks, distributed computing, multimedia, mobile computing, increased use of standard software, new tools to support end-user computing, and dramatic changes in the use of IT imply new requirements for IT professionals. By situating our research efforts in environments where such emerging requirements are important we would adapt our thinking to meet new technological challenges.

8.3. Summary

The purpose of this paper has been to provide a comprehensive presentation of one specific approach to systems development research and practice. I have described the context of the approach together with its historic evolution; I have explicated the underlying assumptions about research and practice; I have provided a survey of the contributions of the approach; and I have discussed the relations to other contributions to systems development. The discussion has concentrated on three questions of general interest: How should we understand, support, and improve practice? How should we organize and conduct research? How should we relate practice and research? My answers to these questions, which are summarized as lessons about systems development research and practice (see figure 4), describe key positions of Reflective Systems Development.

Acknowledgments

References

Aaen, I. (1989): *Systems Development. Between Scylla and Charybdis.* Ph.D. thesis, Aalborg University. (In Danish)

Aaen, I. & C. Sørensen (1991): A CASE of Great Expectations. *Scandinavian Journal of Information Systems*, Vol. 3.

Aaen, I. (1992): CASE Tool Bootstrapping. How little stones fell great oaks. In K. Lyytinen *et al.* (Eds.): *Next Generation CASE Tools.* Amsterdam: IOS Press.

Aaen, I., A. Siltanen, C. Sørensen & V.-P. Tahvanainen (1992): A Tale of Two Countries. CASE Experience and Expectations. In K. E. Kendall *et al.* (1992): *The Impact of Computer Technologies on Information Systems Development.* Amsterdam: North-Holland.

Aaen, I. (1994): Problems in CASE Introduction. Experiences from User Organizations. *Information and Software Technology*, Vol. 36, No. 11.

Ackoff, R. L. (1982): On the Hard Headedness and Soft Heartedness of M. C. Jackson. *Journal of Applied Systems Analysis,* Vol. 9.

Andersen, N. E., F. Kensing, M. Lassen, J. Lundin, L. Mathiassen, A. Munk-Madsen and P. Sørgaard (1986): *Professional Systems Development. Experiences, Ideas, and Action.* Copenhagen: Teknisk Forlag. (In Danish)

Andersen, N. E., F. Kensing, M. Lassen, J. Lundin, L. Mathiassen, A. Munk-Madsen and P. Sørgaard (1990): *Professional Systems Development. Experiences, Ideas, and Action.* Englewood Cliffs, New Jersey: Prentice-Hall.

Andersen, N. E., M. Franckson *et al.* (1996): *Euromethod Version 1.* European Commission. Information available at http://www.fast.de/Euromethod/.

Anderson, R. E. *et al.* (1993): Using the New ACM Code of Ethics in Decision Making. *Communications of the ACM*, Vol. 36, No. 2.

Applegate, L. M., F. W. McFarlan & J. L. McKenney (1996): *Corporate Information Systems Management. Text and Cases.* Fourth edition. Chicago: Irwin.

Argyris, C. & D. Schön (1978): *Organizational Learning.* Reading, Massachusetts: Addison-Wesley.

Argyris, C. & D. Schön (1996): *Organizational Learning II.* Reading, Massachusetts: Addison-Wesley.

Avison, D. E. & G. Fitzgerald (1988): Information Systems Development. Current Themes and Future Directions. *Information and Software Technology*, Vol. 30, No. 8.

Avison, D. E. & A. T. Wood-Harper (1990): *Multiview. An Exploration in Information Systems Development.* Oxford: Blackwell Scientific Publications.

Avison, D. E. & A. T. Wood-Harper (1991): Information Systems Development Research. An Exploration of Ideas in Practice. *The Computer Journal*, Vol. 34, No. 2.

Bang, S., S. Efsen, P. Hundborg, H. Janum, L. Mathiassen & C. Schultz (1991): *Quality Management in Systems Development.* Copenhagen: Teknisk Forlag. (In Danish)

Bansler, J. (1987): *Systems Development. Theory and History in a Scandinavian Perspective.* Lund: Studentlitteratur. (In Danish)

Bansler, J. (1989): Systems Development Research in Scandinavia. Three Theoretical Schools. *Scandinavian Journal of Information Systems*, Vol. 1.

Bansler, J. & K. Bødker (1993): A Reappraisal of Structured Analysis. Design in an Organizational Context. *ACM Transactions on Information Systems*, Vol. 11, No. 2.

Basili, V. R. & D. M. Weiss (1984): A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*, Vol. 10.

Basili, V. R., R. W. Selby & D. H. Hutchens (1986): Experimentation in Software Engineering. *IEEE Transactions on Software Engineering*, Vol. 12.

Baskerville, R. & A. T. Wood-Harper (1996a): A Critical Perspective on Action Research as a Method for Information Systems Research. *Journal of Information Technology*, Vol. 11.

Baskerville, R. L. & J. Stage (1996b): Controlling Prototype Development Through Risk Management. *MIS Quarterly*, Vol. 20, No. 4.

Beath, C. M. & W. J. Orlikowski (1994): The Contradictory Structure of Systems Development Methodologies: Deconstructing the IS-User relationship in Information Engineering. *Information Systems Research*, Vol. 5, No. 4.

Benbasat, I., A. S. Dexter & R. W. Mantha (1980): Impact of Organizational Maturity on Information System Skill Needs. *MIS Quarterly*, Vol. 4, No. 1.

Birtwistle, G. M., O.-J. Dahl, B. Myhrhaug & K. Nygaard (1973): *Simula Begin*. Lund: Studentlitteratur.

Bjerknes, G., P. Ehn & M. Kyng (Eds.) (1987a): *Computers and Democracy*. Aldershot: Avebury.

Bjerknes, G. & T. Bratteteig (1987b): Florence in Wonderland. Systems Development with Nurses. In Bjerknes *et al.* (1987a).

Bjerknes, G. & T. Bratteteig (1988a): The Memoirs of two Survivors—or Evaluation of a computer system for Cooperative Work. In: *Proceedings of the Second Conference on Computer Supported Work,* ACM.

Bjerknes, G. & T. Bratteteig (1988b): Computers—Utensils or Epaulets? The Application Perspective Revisited. *AI and Society,* Vol. 2, No. 3.

Bjerknes, G. (1989): *Contradictions—A Tool to Understand Situations in Systems Development.* Ph.D. thesis, Oslo University. (In Norwegian)

Bjerknes, G., B. Dahlbom, L. Mathiassen, M. Nurminen, J. Stage, K. Thoresen, P. Vendelbo & I Aaen (Eds.) (1990): *Organizational Competence in Systems Development.* Lund: Studentlitteratur.

Bjerknes, G. (1991): Dialectical Reflection in Information Systems Development. *Scandinavian Journal of Information Systems*, Vol. 3.

Bjerknes, G. & T. Bratteteig (1995): User Participation and Democracy: A Discussion of Scandinavian Research on Systems Development. *Scandinavian Journal of Information Systems*, Vol. 7, No. 1.

Bjørn-Andersen, N. & B. Hedberg (1977): Designing Information Systems in an Organizational Perspective. *TIMS Studies in Management Sciences*, Vol. 5.

Bjørn-Andersen, N. (Ed.) (1980): *The Human Side of Information Processing.* Amsterdam: North-Holland.

Boland, R. J. (1978): The Process and Product of System Design. *Management Science*, Vol. 24, No. 9.

Boland, R. J. & W. Day (1982): The Process of System Design. A Phenomenological Approach. In M. Ginzberg *et al.* (Eds.): *Proceedings of the 3rd International Conference on Information Systems.* Ann Arbor, Michigan.

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

Boehm, B. W. (1981): *Software Engineering Economics.* Englewood Cliffs, New Jersey: Prentice-Hall.

Boehm, B. W., T. E. Gray & T. Seewaldt (1984): Prototyping versus Specifying. A Multiproject Experiment. *IEEE Transactions on Software Engineering*, Vol. 10, No. 3.

Boehm, B. W. (1988): A Spiral Model of Software Development and Enhancement. *Computer*, Vol. 21, No. 5.

Boehm, B. W. & P. N. Papaccio (1988): Understanding and Controlling Software Costs. *IEEE Transactions on Software Engineering*, Vol. 4, No. 10. Reprinted in DeMarco *et al.* (1990).

Borum, F. (1995): *Organization, Power, and Change.* Copenhagen: Copenhagen Business School Press.

Braverman, H. (1974): *Labor and Monopoly Capital. The Degradation of Work in the Twentieth Century.* New York: Monthly Review Press.

Brooks, F. P. (1987): No Silver Bullet. *Computer*, Vol. 20, No. 4. Reprinted in DeMarco *et al.* (1990).

Brown, J. S. & P. Duguid (1991): Organizational Learning and Communities-of-Practice. Toward a Unified view of Working, Learning, and Innovation. *Organization Science*, Vol. 2, No. 1.

Bubenko, J. A. (1980): Information Modeling in the Context of Systems Development. In S. H. Lavington (Ed.): *Information Processing 80.* Amsterdam: North-Holland.

Budde, R., K. Kuhlenkamp, L. Mathiassen & H. Züllighoven (Eds.) (1984): *Approaches to Prototyping.* Berlin: Springer-Verlag.

Budde, R., K. Kautz, K. Kuhlenkamp & H. Züllighoven (1992): *Prototyping. An Approach to Evolutionary Systems Development.* Berlin: Springer-Verlag.

Burrell, G. & G. Morgan (1979): *Sociological Paradigms in Organizational Analysis.* London: Heineman.

Bødker, S., P. Ehn, J. Kammersgaard, M. Kyng & Y. Sundblad (1987): A Utopian Experience. On Design of Powerful Computer-based Tools for Graphical Workers. In Bjerknes *et al.* (1987).

Carlson, J., P. Ehn, B. Erlander, M-L. Perby & Å. Sandberg (1978): Planning and Control from the Perspective of Labor. A Short Presentation of the DEMOS Project. *Accounting, Organizations and Society*, Vol. 3, No. 3–4.

Cash, J. I., I. Benbasat, K. L. Kraemer & P. R. Lawrence (Eds.) (1989): *The Information Systems Research Challenge*, Vol. 1–3. Boston, Massachusetts: Harvard Business School.

Checkland, P. (1981): *Systems Thinking, Systems Practice.* Chichester: John Wiley.

Checkland, P. B. (1982): Soft Systems Methodology as a Process. A Reply to M. C. Jackson. *Journal of Applied Systems Analysis,* Vol. 9.

Checkland, P. & J. Scholes (1990): *Soft Systems Methodology in Action.* Chichester: John Wiley.

Churchman, C. W. (1982): Reply to M. C. Jackson. *Journal of Applied Systems Analysis,* Vol. 9.

Ciborra, C. (1981): Information Systems and Transaction Architecture. *International Journal of Policy Analysis and Information Systems*, Vol. 5.

Ciborra, C. (1983): Reframing the Role of Computers in Organizations. The Transaction Cost Approach. *Journal of Information Economics and Policy*, Vol. 21.

Ciborra, C. & G. F. Lanzara (1994): Formative Contexts and Information Technology. Understanding the Dynamics of Innovation in Organizations. *Accounting, Management & Information Technology*, Vol. 4, No. 2.

Cotterman, W. W. & J. A. Senn (Eds.) (1992): *Challenges and Strategies for Research in Systems Development.* Chichester: John Wiley.

Curtis, B., H. Krasner & N. Iscoe (1988): A Field Study of the Software Design Process for Large Systems. *Communications of the ACM*, Vol. 31. No. 11.

Dahl, O.-J., B. Myhrhaug & K. Nygaard (1971): *Simula 67 Common Base Language.* Oslo: Norwegian Computing Center.

Dahlbom, B. & L. Mathiassen (1993): *Computers in Context. The Philosophy and Practice of Systems Design.* Cambridge, Massachusetts: Blackwell.

Dahlbom, B. (1996): The New Informatics. *Scandinavian Journal of Information Systems*, Vol. 8, No. 2.

DeMarco, T. & T. Lister (1990): *Software State-of-the-Art. Selected papers.* New York: Dorset House.

Ehn, P. (1988): *Work-Oriented Design of Computer Artifacts.* Stockholm: Arbetslivscentrum.

Elam, J. J., D. B. Waltz, H. Krasner & B. Curtis (1987): A Methodology for studying Software Design Teams. An Investigation of Conflict Behaviors in the Requirements Definition Phase. *Empirical Studies of Programmers. Second Workshop.* Norwood, New Jersey: Ablex Publishing.

Fenton, N. (1993): How Effective are Software Engineering Methods. *Journal of Systems and Software*, Vol. 22.

Fitzgerald, B. (1996): Formalized Systems Development Methodologies. A Critical Perspective. *Information Systems Journal,* Vol. 6.

Floyd, C. (1984): A Systematic look at Prototyping. In R. Budde *et al.* (1984).

Floyd, C. (1986): A Comparative Evaluation of Systems Development Methods. In T. W. Olle *et al.* (Eds.): *Information Systems Design Methodologies. Improving the Practice.* Amsterdam: North-Holland.

Floyd, C. (1987): Outline of a Paradigm Change in Software Engineering. In Bjerknes *et al.* (1987).

Floyd, C., H. Züllighoven, R. Budde & R. Keil-Slawik (Eds.) (1992): *Software Development and Reality Construction.* Berlin: Springer-Verlag.

Galliers, R. D. & F. F. Land (1987): Choosing Appropriate Information Systems Research Methodologies. *Communications of the ACM*, Vol. 30, No. 11.

Gibson, C. F. & R. L. Nolan (1974): Managing the Four Stages of EDP Growth. *Harvard Business Review*, Vol. 52, No. 1.

Gould, J. D. & C. Lewis (1985): Designing for Usability. Key Principles and What Designers Think. *Communications of the ACM*, Vol. 28, No. 3.

Greenbaum, J. & M. Kyng (Eds.) (1991): *Design at Work. Cooperative Design of Computer Systems.* Hilsdale, New Jersey: Lawrence Erlbaum.

Guindon, R., H. Krasner & B. Curtis (1987): Breakdowns and Processes During the Early Activities of Software Design by Professionals. *Empirical Studies of Programmers. Second Workshop.* Norwood, New Jersey: Ablex Publishing. Reprinted in DeMarco *et al.* (1990).

Hirschheim, R. & H. K. Klein (1989): Four Paradigms of Information Systems Development. *Communications of the ACM*, Vol. 32, No. 10.

Hirschheim, R. & H. K. Klein (1991): Rationality Concepts in Information System Development Methodologies. *Accounting, Management & Information Technology*, Vol. 1, No. 2.

Hirschheim, R. & H. K. Klein (1992): Paradigmatic Influences of Information Systems Development Methodologies: Evolution and Conceptual Advances. In M. Yovits (Ed.): *Advances in Computers, 33.* New York: Academic Press.

Hirschheim, R. & H. K. Klein (1994): Realizing Emancipatory Principles in Information Systems Development. The Case for ETHICS. *MIS Quarterly*, Vol. 18, No. 1.

Hirschheim, R., H. K. Klein & K. Lyytinen (1995): *Information Systems Development and Data Modeling. Conceptual and Philosophical Foundations.* Cambridge: Cambridge University Press.

Hirschheim, R., H. K. Klein & K. Lyytinen (1995): Exploring the Intellectual Structures of Information Systems Development. A Social Action Theoretic Analysis. *Accounting, Management & Information Technology*, Vol. 6, No. 1–2.

Holbæk-Hanssen, E., P. Håndlykken & K. Nygaard (1975): *System Description and the Delta Language.* Oslo: Norwegian Computing Center.

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

Høyer, R. (1971): *Dataprocessing, Organization, and Working Environment.* Oslo: Norsk Produktivitetsinstitut. (In Norwegian)

Høyer, R. (1974): *Switching to Computers. . . .* Oslo: Tanum. (In Norwegian)

Iivari, J. & E. Koskela (1987): The PIOCO Model for Information Systems Design. *MIS Quarterly,* Vol. 11, No. 3.

Iivari, J. (1991): A Paradigmatic Analysis of Contemporary Schools of IS Development. *European Journal of Information Systems*, Vol. 1, No. 4.

Iivari, J. (1997): Perceived Usefulness and Perceived Ease of Use of Three Object-oriented Modeling Methods: An Empirical Test. *The Third International Workshop on Next Generation Information Technologies and Systems.*

Iivari, J. & K. Lyytinen (1997): Information Systems Research in Scandinavia. Unity in Plurality. In W. Currie *et al.* (Eds.): *Rethinking Management Information Systems.* London: Oxford University Press.

Israel, J. (1979): *The Language of Dialectics and the Dialectics of Language.* Copenhagen: Munksgård.

Ives, B., S. Hamilton & G. Davis (1980): A Framework for Research in Computer Based Management Information Systems. *Management Science*, Vol. 26, No. 9.

Jackson, M. C. (1982): The Nature of "Soft" Systems Thinking. The Work of Churchman, Ackoff and Checkland. *Journal of Applied Systems Analysis,* Vol. 9.

Jackson, M. C. (1983): The Nature of "Soft" Systems Thinking. Comments on the three replies. *Journal of Applied Systems Analysis,* Vol. 10.

Kaiser, K. M. & R. P. Bostrom (1982): Personality Characteristics of MIS Projects Teams. An Empirical Study and Action-Research Design. *MIS Quarterly*, Vol. 6, No. 4.

Kensing, F. & K. H. Madsen (1991): Generating Visions. Future Workshops and Metaphorical Design. In Greenbaum *et al.* (1991).

Kensing, F. & A. Munk-Madsen (1993): Participatory Design. Structure in the Toolbox. *Communications of the ACM,* Vol. 36, No. 4.

Kensing, F., J. Simonsen & K. Bødker (1996): MUST. A Method for Participatory Design. In J. Blomberg *et al.* (Eds.): *Proceedings of the 4th Participatory Design Conference.* Cambridge, Massachusetts.

Kensing, F., J. Simonsen & K. Bødker (1997): Designing for Cooperation at a Radio Station. In *Proceedings from European Conference on Computer Supported Cooperative Work.* Lancaster.

Kling, R. (1980): Social Analysis of Computing. Theoretical Perspectives in Recent Empirical Research. *ACM Computing Surveys*, Vol. 12, No. 1.

Kling, R. & W. Scacchi (1982): The Social Web of Computing. Computer Technology as Social Organization. *Advances in Computing,* Vol. 21.

Knuth, D. (1989): The Errors of TEX. *Software—Practice & Experience*, Vol. 19, No. 1. Reprinted in DeMarco *et al.* (1990).

Kozar, K. A. (1993): Adopting Systems Development Methods. An Exploratory Study. *Journal of Management Information Systems*, Vol. 5, No. 4.

Krasner, H., B. Curtis & N. Iscoe (1987): Communication Breakdowns and Boundary Spanning Activities of Software Design By Professionals. *Empirical Studies of Programmers. Second Workshop.* Norwood, New Jersey: Ablex Publishing.

Kuhn, T. S. (1962): *The Structure of Scientific Revolutions.* University of Chicago.

Kyng, M. & L. Mathiassen (1982): Systems Development and Trade Union Activities. In N. Bjørn-Andersen (Ed.): *Information Society, for Richer, for Poorer.* Amsterdam: North-Holland.

Kyng, M. (1996): *Computers and Users.* Dr. Scient. thesis, Aarhus University.

Langefors, B. (1966): *Theoretical Analysis of Information Systems.* Lund: Studentlitteratur.

Lanzara, G. F. (1983): The Design Process. Frames, Metaphors, and Games. In U. Briefs *et al.* (Eds.): *Systems Design, For, With, and By the Users.* Amsterdam: North-Holland.

Lau, F. (1997): A Review on the Use of Action Research in Information Systems Studies. In Lee *et al.* (1997).

Leavitt, H. J. (1964): Applied Organization Change in Industry. Structural, Technical, and Human approaches. In: *New Perspectives in Organizational Research.* Chichester: John Wiley.

Lee, A. S., J. Liebenau & J. I. DeGross (Eds.) (1997): *Information Systems and Qualitative Research.* London: Chapman & Hall.

Lewin, K. (1947): Frontiers in Group Dynamics II. *Human Relations*, Vol. 1, No. 2.

Lundeberg, M. & E. S. Andersen (1974): *Systems Development—Information Analysis.* Lund: Studentlitteratur.

Lyytinen, K. (1987a): A Taxonomic Perspective of Information Systems Development. Theoretical Constructs and Recommendations. In R. J. Bo-

land *et al.* (1987): *Critical Issues in Information Systems Research.* Chichester: John Wiley.

Lyytinen, K. (1987b): Different Perspectives on Information Systems. Problems and Solutions. *ACM Computing Surveys*, Vol. 19, No. 1.

Lyytinen, K. (1989): New Challenges of Systems Development. A Vision of the 90's. *Data Base,* Fall.

MARS Project (1984a): *MARS. A Research Project on Methods for Systems Development.* Report No. 1. Aarhus University.

MARS Project (1984b): *Systems Development in Practice.* Reports No. 2–5. Aarhus University. (In Danish)

MARS Project (1985): *Improving Systems Development Practice.* Reports No. 8–10. Aarhus University. (In Danish)

Madabushi, S. V. R., M. C. Jones & R. L. Price (1993): Systems Analysis and Design Models Revisited. A Case Study. *Information Resources Management Journal.* Winter.

Madsen, K. H. (1989): Breakthrough by Breakdown. Metaphors and Structured Domains. In H. Klein *et al.* (Eds.): *Systems Development for Human Progress.* Amsterdam: North-Holland.

Madsen, K. H. (1994): A Guide to Metaphorical Design. *Communications of the ACM*, Vol. 37, No. 12.

Madsen, O. L., B. Møller-Pedersen & K. Nygaard (1993): *Object-Oriented Programming in the Beta Language.* Reading, Massachusetts: Addison-Wesley.

March, J. G. & J. P. Olson (1976): *Ambiguity and Choice in Organizations.* Oslo: Universitetsforlaget.

Markus, M. L. (1983): Power, Politics, and MIS Implementation. *Communications of the ACM*, Vol. 26, No. 7.

Markus, M. L. (1997): The Qualitative Difference in Information Systems Research and Practice. In Lee *et al.* (1997).

Mathiassen, L. (1976). *A Computer Book.* Copenhagen: Gyldendal. (In Danish)

Mathiassen, L. (1980). Systems Description as a Tool for Teaching Programming. *SIGCSE Bulletin*, Vol. 12, No. 4.

Mathiassen, L. (1981): *Systems Development and Systems Development Methods.* Ph.D. thesis, Oslo University. (In Danish)

Mathiassen, L., B. Rolskov & E. Vedel (1983). Regulating the Use of EDP by Law and Agreement. In U. Briefs *et al.* (Eds.): *Systems Design, For, With, and By the Users.* Amsterdam: North-Holland.

Mathiassen, L. (1988): Creativity and Discipline in System Design. *Data.* No. 6. (In Danish)

Mathiassen, L., A. Munk-Madsen, P. A. Nielsen & J. Stage (1993): *Object Oriented Analysis.* Aalborg: Marko. (In Danish)

Mathiassen, L., A. Munk-Madsen, P. A. Nielsen & J. Stage (1994): Combining two Approaches to Object-Oriented Analysis. In D. Patel *et al.* (Eds.): *Proceedings of the International Conference on Object-Oriented Information Systems.* Berlin: Springer-Verlag.

Mathiassen, L., A. Munk-Madsen, P. A. Nielsen & J. Stage (1995): *Object Oriented Design.* Aalborg: Marko. (In Danish)

Mathiassen, L. (1996): Information Systems Development. Reflections on a Discipline. *Accounting, Management & Information Technology*, Vol. 6, No. 1/2.

Mathiassen, L., A. Munk-Madsen, P. A. Nielsen & J. Stage (1997): *Object Oriented Analysis and Design.* Aalborg: Marko. (In Danish)

McFarlan, F. W. (1984): *The Information Systems Research Challenge.* Boston, Massachusetts: Harvard Business School Press.

McKeen, J. D. (1983): Successful Development Strategies for Business Application Systems. *MIS Quarterly*, Vol. 7, No. 3.

McKenney, J. L. & F. W. McFarlan (1990): The Information Archipelago. Maps and Bridges. *Harvard Business Review*, Vol. 60, No. 5. Reprinted in DeMarco *et al.* (1990).

Mingers, J. (1984): Subjectivism and Soft Systems Methodology. A Critique. *Journal of Applied Systems Analysis,* Vol. 11.

Mintzberg, H. (1983): *Structures in Fives. Designing Effective Organizations.* Englewood-Cliffs, New Jersey: Prentice-Hall.

Mumford, E. (1983): *Designing Human Systems: The ETHICS Method.* Manchester Business School.

Mumford, E., R. A. Hirschheim, G. Fitzgerald & A. T. Wood-Harper (Eds.) (1986): *Research in Information Systems.* Amsterdam: North-Holland.

Munk-Madsen, A. (1983): System Description With Users. In U. Briefs *et al.* (Eds.): *Systems Design, For, With, and By the Users.* Amsterdam: North-Holland.

Munk-Madsen, A. (1986): *Knowledge about Systems Development.* MARS Report No. 13. Aarhus University. (In Danish)

Munk-Madsen, A. (1987): *Evaluation of Systems Development Projects.* Ph.D. thesis, Aarhus University. (In Danish)

Munk-Madsen, A. (1996): *Strategic Project Management.* Aalborg: Marko. (In Danish)

Naur, P. (1972): An Experiment in Program Development. *BIT*, Vol. 12.

Naur, P. (1982): Formalizations in Program Development. *BIT*, Vol. 22.

Naur, P. (1983): Program Development Studies Based on Dairies. T. R. Green *et al.* (Eds.): *Psychology of Computer Use.* London: Academic Press.

Naur, P. (1985): Intuition in Software Development. H. Ehrig *et al.* (Eds.). *Lecture Notes in Computer Science 186.* Berlin: Springer.

Naur, P. (1992): *Computing. A Human Activity.* Reading, Massachusetts: Addison Wesley.

Necco, C. R., C. L. Gordon & N. W. Tsai (1987): Systems Analysis and Design: Current Practices. *MIS Quarterly*, Vol. 11, No. 4.

Newman, M. & D. Rosenberg (1985): Systems Analysts and the Politics of Organizational Control. *Omega*, Vol. 13.

Newman, M. & F. Noble (1990): User Involvement as an Interaction Process. A Case Study. *Information Systems Research,* Vol. 1.

Nielsen, P. A. (1989a): The Concept of Contradiction in Soft Systems Practice. An Illustration. In R. L. Flood *et al.* (Eds.): *Systems Prospects. The Next Ten Years of Systems Research.* York, UK: Plenum Press.

Nielsen, P. A. (1989b): Reflections on Development Methods for Information Systems. A Set of Distinctions between Methods. *Office: Technology & People,* Vol. 5, No. 2.

Nielsen, P. A. (1990a): *Using and Learning IS Development Methodologies.* Ph.D. thesis, Lancaster University.

Nielsen, P. A. (1990b): Approaches to Appreciate Information Systems Methodologies. A Soft Systems Survey. *Scandinavian Journal of Information Systems,* Vol. 2.

Nissen, H.-E., H. K. Klein & R. A. Hirschheim (Eds.) (1990): *The Information Systems Research Arena of the 90's.* Amsterdam: North-Holland.

Nonaka, I. (1994): A Dynamic Theory of Organizational Knowledge Creation. *Organization Science,* Vol. 5, No. 1.

Nunamaker, J. M. Chen & T. D. M. Purdin (1991): Systems Development in Information Systems Research. *Journal of Management Information Systems*, Vol. 7, No. 3.

Nurminen, M. I. (1988): *People or Computers. Three Ways of Looking at Information Systems.* Lund: Studentlitteratur.

Nygaard, K. & O. T. Bergo (1975): The Trade Unions. New Users of Research. *Personnel Review*, Vol. 4, No. 2.

Nygaard, K. & P. Sørgaard (1985): *Perspective. A Key Concept in Informatics.* In Bjerknes *et al.* (1987).

Olerup, A. (1989): Socio-technical Design of Computer-assisted Work. *Scandinavian Journal of Information Systems*, Vol. 1.

Parnas, D. L. (1972): On the Criteria to Be Used in Decomposing Systems into Modules. *Communications of the ACM*, Vol. 15, No. 12.

Parnas, D. L. & P. C. Clements (1986): A Rational Design Process. How and Why to Fake It. *IEEE Transactions on Software Engineering*, Vol. 12, No. 2. Reprinted in DeMarco *et al.* (1990).

Paulk, M. C., B. Curtis, M. B. Chrissis & C. V. Weber (1993): Capability Maturity Model. Version 1.1. *IEEE Software*, Vol. 10.

Polanyi, M. (1967): *The Tacit Dimension.* New York: Doubleday and Co.

Robey, D. & M. L. Markus (1984): Rituals in Information System Design. *MIS Quarterly*, Vol. 8, No. 1.

Robey, D. (1995): Theories that Explain Contradiction. Accounting for the Contradictory Organizational Consequences of Information Technology. In J. I. DeGross *et al.* (Eds.): *Proceedings from 16th International Conference on Information Systems.* Amsterdam.

Robey, D. (1996): Research Commentary. Diversity in Information Systems Research. Threat, Promises, and Responsibility. *Information Systems Research*, Vol. 7, No. 4.

Sandberg, Å. (1975): *A Question of Power.* Stockholm: Prisma. (In Swedish)

Salaway, G. (1987): An Organizational Learning Approach to Information Systems Development. *MIS Quarterly*, Vol. 11, No. 2.

Schein, E. K. (1985): *Organizational Culture and Leadership. A Dynamic View.* San Francisco: Jossey-Bass.

Schön, D. A. (1983): *The Reflective Practitioner. How Professionals Think in Action.* New York: Basic Books.

Schön, D. A. (1987): *Educating the Reflective Practitioner.* San Francisco: Jossey-Bass Publishers.

Selby, R. W, V. R. Basili & F. T. Baker (1987): Cleanroom Software Development. An Empirical Evaluation. *IEEE Transactions on Software Engineering*, Vol. 13, No. 9. Reprinted in DeMarco *et al.* (1990).

Simon, H. (1955): A Behavioral Model of Rational Choice. *Quarterly Journal of Economics*, Vol. 69.

Simon, H. (1956): Rational Choice and the Structure of the Environment. *Psychological Review*, Vol. 63, No. 2.

Simon, H. (1957): *Models of Man. Social and Rational.* New York: John Wiley.

Simon, H. (1982): *Models of Bounded Rationality. Behavioral Economics and Business Organization.* Cambridge, Massachusetts: MIT Press.

Simonsen, J. & F. Kensing (1997): Using Ethnography in Contextual Design. *Communications of the ACM*, Vol. 40, No. 7.

Stage, J. (1986): Evaluation of Systems Development Methods. Approaching a Conceptual Framework. In K. Fuchs-Kittowski *et al.* (Eds.): *System Design for Human Development and Productivity. Participation and Beyond.* Amsterdam: North-Holland.

Stage, J. (1989): *Between Tradition and Transcendence. Analysis and Design in Systems Development.* Ph.D. thesis, Oslo University. (In Danish)

Stage, J. (1990): Analyzing Organizations in Systems Development. In Bjerknes *et al.* (1990).

Stage, J. (1991): The Use of Descriptions in Analysis and Design of Information Systems. In R. K. Stamper *et al.* (Eds.): *Collaborative Work, Social Communications, and Information Systems.* Amsterdam: North-Holland.

Stolterman, E. (1991): The Hidden Rationality of Design. A Study of Methods and Practices in Systems Development. Ph.D. thesis, Umeå University. (In Swedish)

Stolterman, E. (1992): How Systems Designers Think about Design and Methods. Some Reflections Based on an Interview Study. *Scandinavian Journal of Information Systems*, Vol. 4.

Sørensen, C. (1993): *Introducing CASE Tools into Software Organizations.* Ph.D. thesis, Aalborg University.

Tan, M. (1994): Establishing Mutual Understanding in Systems Design. An Empirical Study. *Journal of Management Information Systems,* Vol. 10, No. 4.

Vidgen, R. & K. Braa (1997): Balancing Interpretation and Intervention in Information System Research: The Action Case Approach. In Lee *et al.* (1997).

Vitalari, N. P. & G. W. Dickson (1983): Problem Solving for Effective Systems Analysis. An Experimental Exploration. *Communications of the ACM*, Vol. 26, No. 11.

Vitalari, N. P. (1985): Knowledge as a Basis for Expertise in Systems Analysis. An Empirical Study. *MIS Quarterly*, Vol. 9, No. 3.

Waltz, D. B., J. J. Elam & B. Curtis (1993): Inside a Software Design Team. Knowledge Acquisition, Sharing, and Integration. *Communications of the ACM*, Vol. 36, No. 10.

Weinberg, G. M. (1982): Overstructured Management of Software Engineering. *Proceedings of the Sixth International Conference on Software Engineering.* Tokyo, Japan. Reprinted in DeMarco *et al.* (1990).

Weinberg, G. M. & D. P. Freedman (1982): *Handbook of Walkthroughs, Inspections and Technical Reviews.* Boston: Little Brown and Company.

Weinberg, G. M. (1986): *Becoming a Technical Leader. An Organic Problem Solving Approach.* New York: Dorset House.

Welke, R. J. (1981): *IS/DSS: DBMS support for Information Systems Development.* Hamilton: McMaster University.

White, K. B. (1984): MIS Project Teams. An Investigation of Cognitive Style Implications. *MIS Quarterly*, Vol. 8, No. 2.

White, K. B. & R. Leifer (1986): Information Systems Development Success. Perspectives from Project Team Participants. *MIS Quarterly*, Vol. 10, No. 3.

Williamson, O. (1975): *Markets and Hierarchies. Analysis and Antitrust Implications.* New York: Free Press.

Wilson, B. (1984): *Systems Concepts, Methodologies, and Applications.* Chichester: John Wiley.

Wynekoop, J. L. & N. L. Russo (1993): System Development Methodologies. Unanswered Questions and the Research-Practice Gap. In J. I. DeGross, R. P. Bostrom & D. Robey (Eds.): *Proceedings from 14th International Conference on Information Systems.* Orlando, Florida.

Zmud, R. W. (1980): Management of Large Software Development Efforts. *MIS Quarterly,* Vol. 4, No. 2.

Züllighoven, H. (Ed.) (1992): Special Issue on Prototyping. *Information Technology & People*, Vol. 6, No. 2–3.