ISBN 87-88715-12-4
ISSN 0105-8517

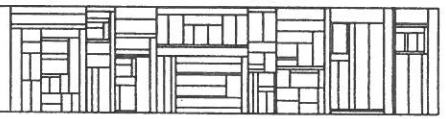
SYSTEMS DEVELOPMENT

Possibilities for and Obstacles to Changing Practice

Finn Kensing

DAIMI PB-195

Computer Science Department
AARHUS UNIVERSITY
Ny Munkegade – DK 8000 Aarhus C – DENMARK
Telephone: 06 – 12 83 55



MARS-report No. 12, July 1985

TRYK: RECAU (06) 12 83 55

SYSTEMS DEVELOPMENT

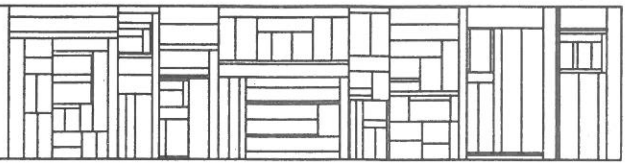
POSSIBILITIES FOR AND OBSTACLES TO CHANGING PRACTICE

by

Finn Kensing
Roskilde University Centre
Denmark

DAIMI PB-195

Computer Science Department
AARHUS UNIVERSITY
Ny Munkegade – DK 8000 Aarhus C – DENMARK
Telephone: 06 – 12 83 55



MARS-report No. 12, July 1985

Introduction

Too little time, too few people, and too poor techniques and tools are frequent explanations of why a system development project does not come up to expectations. The explanations may be true, but in a critical project situation they are insufficient because they do not point towards changing the prerequisites of the project: deadlines are often difficult or even impossible to postpone, adding man-power may be impossible because people are not available, or undesirable because it would create rather than solve problems, and finally it may be difficult to change techniques and tools half way through a project.

This article will discuss some other explanations as to why system development projects often fail to come up to expectations - be it expectations to the function and mode of operation of the final product, or expectations to the course of the project. The explanations center around two concepts: *approaching the product* and *handling project situations*. These concepts will be introduced and exemplified in section 3.

The article is partly inspired by Schön's (83) studies of how various professionals think when they work, and partly by my own studies of the working practices of computer specialists (MARS 84). Schön's ideas will be presented in section 2, where they - on the basis of my own observations - are put into a system development context.

Section 1 will describe a case which illustrates the necessity to change system development practices. This is not merely done by securing more resources and the newest techniques and tools. It is just as important that the system developers understand the computer based system - and that they understand their own working process - the system development process.

1. State of the Art - A Case

CTJ is a large Danish production company which has its own systems department. The company's dispatch department was about to replace its old batch-oriented inventory management systems, among other reasons because the old system was getting too complicated to maintain, the users wanted faster respons-time, and because the old computer was to be replaced. A user group consisting of middle management representatives, and a project team consisting of systems analysts and programmers were established to develop the new system. After the development of the inventory management system had been initiated, the user group suggested that a production control system be developed for the maintenance shops, the argument being that the slow execution of repairs could be traced back to the shop foremen's insufficient control where the work in the maintenance shops was concerned. It should be noted here that the maintenance shops were not represented in the user group. Both projects were estimated to a total of 12 man years.

When the inventory management system was about half finished the project team started to develop the production control system on the basis of the dispatch department's perception and description of the situation in the maintenance shops.

The system was to contain statistics of the time spent on maintenance, data on man-power plan, and data on current stock of spare parts. The idea was that the shop foremen, when receiving material for repair, input the type of repair which was to be carried out. The system would tell whether the necessary spare parts were on stock (or - if they were not - when they could be expected to be on stock), who was to carry out the repair, and when the repair was expected to be finished.

The project team found it difficult to discuss with the shop foremen how a computer based production control system could

be useful to them. The team found that one of the reasons for this difficulty was that production control had been very informal up till then, and that the foremen had not previously used computer technology in connection with their work. Another reason was that project team members knew nothing about the problems of production control. To improve their qualifications they read literature on production control in general, and studied a concrete computer based standard system.

The project team explained the functions and available display images of the standard system to the foremen by applying an overhead projector. The work of the shop foremen, and how it was carried out, was not discussed very much, although the project team did have a look at the paper routines, which they found rather complicated.

After six months of analyses, which mainly were carried out at monthly meetings of about two hours, the foremen and the project team agreed that they needed a more concrete basis for their discussions. They decided to develop a prototype. The project team estimated that the prototype could be developed within two months with an effort of six to eight man months.

They started to develop the prototype on the basis of their knowledge of other production control systems. Work with the inventory management system was temporarily suspended. The work proved to be much more extensive than anticipated. First the available equipment did not provide tools for rapid prototyping, and secondly the project team decided to develop tools for connecting display images and database for later use. Several times - with an interval of one or two months - the project team was forced to report that the completion of the prototype was delayed two or three months, and in the end it took eight months with an effort of 25 to 30 man months.

The project team decided that the prototype should be tested by the foremen. Very little time was spent on training the fore-

men in testing the system, and the project team neglected to explain the purpose of the test.

First, however, the foremen had to put some basic data into the system. They gave up every time there was an error. When they were about to start the test itself, they argued that they could not use the prototype because it did not contain data about available material on stock. When they got the data they complained that they needed a printer to see the results of their inquieries immediately.

The project team did not do very much to get the test restarted. One reason was that the team was getting behind schedule with the development of the inventory management system, another was that it took a long time to correct errors in the prototype due to the inadequate tools. The systematic testing anticipated by the project team was never performed, and the production control part of the project quietly died.

In spite of the apparent failure of the prototype it nevertheless proved to have one essential function: it revealed a disagreement between the dispatch department and the maintenance shops. The dispatch department claimed - as already mentioned - that there was insufficient control where the work in the maintenance shops was concerned, while the formen claimed that the dispatch department far too often ran out of the necessary material. The disagreement only surfaced because the foremen refused to test the system as they felt that its implementation would automatize all the interesting aspects of their job.

On first sight it seems obvious that things would have developed less unfortunately if the following circumstances had been avoided:

- The analysis process was scattered over a six month period with only one two-hour meeting a month.

- In wrongly estimating the development time of the prototype with a factor four, the project team lost their credibility in the eyes of the foremen.

- The project failed to support the foremen before and during the testing of the prototype.

This and similar courses of events at CTJ brought about two initiatives to improve systems development:

- Endeavours to get better prototyping tools, and

- endeavours to get a computer based project control system.

This article will not argue against the possibility that initiatives like that might improve systems development, but new tools are not enough. In the last couple of years one of the largest Danish software houses spent $ 200,000 on training its programmers and analysts in the Yourdon-DeMarco method. It was found that in several of the projects in which the method was applied, because management demanded it, it was mainly applied as a documentation technique to demonstrate to the steering committee that the standard had been met. The method was not applied to support the creative design process.

In another large Danish software house a project team - on their own initiative - decided to apply reviews as a technique to improve the quality of both product and process. When the time came to plan the review, the whole project team decided - using a tight time schedule as an argument - that a review was not necessary in that particular situation, and that an external reviewer in any case would be superfluous.

Many efforts have been made to improve the working practices in systems development by introducing new methods, but the above-mentioned examples show some of the difficulties which

arise, irrespective of whether the initiative comes from management or from the employees themselves. The next section will present a way of thinking which of course could be seen as a technique, only on a different and more general level from those mentioned above. In my opinion it is necessary in practical systems development to work for improvements on both levels: to improve the quality of the computer based systems, and to improve the system development process.

2. Systems Development: Reflection in Action

In his book "The Reflective Practitioner - How Professionals Think in Action" Schön processes a large number of observations made while studying representatives of a variety of different professions: architects, psycho-therapists, engineers, town planners, and managers. He finds a common structure which he calls *reflection in action* characterizing professional knowledge across professional borders.

Systems development is a young profession, and its performers have very different backgrounds: engineers, computer scientists, economists, many are self-taught, and many have a short education, mainly in programming. Reading Schön inspired me to take a closer look at the connection between what system developers think they do, what they say they do, and what they actually do. The discrepancies between these three realities are interesting both in relation to the system development process, and to the resulting product: the computer system.

My starting point for studying this connection is that many of the failures I have seen in practical systems development can be traced back to the system developers' erroneous or insufficient assumptions concerning the product and/or the process. Improvements in practical systems development must therefore be obtained through influencing the system developers'

assumptions. The presentation of Schön's ideas - here put into a system development context - serves this purpose. In the following section system developers' possibilities for and obstacles to practicing *reflection in action* will be discussed.

To illustrate what reflection in action is let us imagine a situation like the one described in the case above. The project team is stuck, and in an attempt to get the project back on its track they call in outside help in the form of a professional consultant.

Our consultant works according to three fundamental principles:

First he would approach the problem as though it were a unique case. In the case described above he would thus examine what was special about production control in these specific maintenance shops. This does not mean that he would act as though he had no relevant prior experience, or that it was irrelevant that the project team read literature on production control and studied standard computer systems. But the consultant's fundamental assumption would make him look for the peculiarities of the situation at hand, while the actions of the project team showed that they assumed generalities to be more important.

Secondly the consultant would not accept the problem on face value. He would have questioned that the only problem was the foremen's insufficient control with the work in the maintenance shops. Experience would have taught him that any situation he enters into is complex, uncertain, and ambiguous; and that it is a problem to identify the problem. This assumption might entail that the consultant for instance would initiate an analysis of the involved parties to clarify their respective objectives. Instead the project team's implicitly assumed that the dispatch department and the maintenance shops shared a common goal. This entailed that they apparently developed

a prototype for the foremen, but which actually seeked to fulfill the objectives of the dispatch department.

Thirdly the consultant would make himself part of the situation - in Schön's words: he would step into the situation. He would see that the project team was part of the conflict. In the eyes of the foremen the project team represented the antagonists: the dispatch department that wished to automatize the most interesting parts of their work. That was why they refused to test the system. Thus they directed their opposition at the project team rather than at the dispatch department which in reality formulated the idea with the system. Stepping into and making himself part of the situation the consultant is able to contribute to the solution of the conflict. Staying outside the project team lost this possibility, and instead played the traditional rôle of the neutral experts.

In addition to these three principles the consultant has a special way to obtain knowledge. Schön calls it a reflective conversation with a unique and uncertain situation:

The consultant might ask the project leader to describe the situation:

"The foremen do not want to test the system, and before this is done we cannot continue with the design."

The consultant would not take this problem definition for granted. To understand the situation he would ask some elaborating questions about to whom and how the prototype could be useful, to whom and how the final system would be useful, etc. The project leader might answer:

"As far as I understand everybody agrees that the repair time is too long. According to the dispatch department the reason for this is that the foremen are not good enough at planning.

And according to the foremen the reason is that the dispatch department never has the things they need."

The consultant would see the foremen's refusal to test the system as a result of a clash of interests. He would suggest that the problem should be seen as an organizational one:

"The foremen are afraid that the employees of the dispatch department will use the production control system to enforce an organizational change which will take planning and decision making - the most interesting part of their work - away from them."

The project team and the consultant might then start to examine the consequences of seeing the problem from this viewpoint, listing causes and consequences:

The causes included:

- that the dispatch department and the maintenance shops did not share a common interest in the system,

- that the sphere of authority relations were obscure, i.e. the project team did not know whom to listen to. Formally the dispatch department was higher in the organizational hierarchy, but in reality the foremen did have the power to sabotage the test.

The consequences were:

- that the deadline was not met,

- that there was still uncertainty whether the prototype was suitable or not,

- that the users were dissatisfied with the project team.

When seen as an organizational problem, the solution should, however, be found outside the project group: it should be found in the user organization. The systems department could contemplate to inform the user organization that the development of the production control system should be stopped until the internal disagreements had been clarified.

In order to illustrate what Schön calls the *reflective conversation with the situation* technique, we assume that the systems department does not feel strong enough for this solution. The consultant and the project team therefore try to reframe the situation. They see the problem as a problem of understanding within the project team itself: The project team had not understood the foremen's problems in connection with production control.

Again the project team and the consultant would examine the consequences of seeing the situation from this viewpoint, listing causes and consequences:

The causes included:

- that the project team had not examined the working routines, sphere of authority relations, etc. in the user organization,

- that the project team primarily had talked with employees of the dispatch department rather than with the foremen.

The consequences were:

- no inventory data in the prototype,

- no statistics on time consumption in the prototype,

- the prototype was of no use to the foremen.

The consultant and the project team would then try to propose alternative lines of action from this new understanding of the problem-cause-consequence interrelation. They would examine how they pictured the situation, and how they would react to the alternatives, i.e. they would evaluate the consequences of their alternatives before implementing any of them.

They might see two alternatives:

1. The prototype is shelved until the inventory management system has been developed. The prototype could then be tested again because the missing inventory data would be available, and the foremen could estimate the repair time themselves and give them as input into the prototype.

2. The development of both inventory management and production control systems are suspended while the project team gets itself thoroughly acquainted with the working routines in the user organization. From this a new design should be developed, the main idea of which would be that the system showed the foremen the necessary data. The decisions would continue to be made by the foremen themselves. They could now require an output with the results of their decisions.

The consultant and the project team would then assess the probable consequences of the two alternatives, and continue in this manner until they were satisfied with the solution of one of the formulated problems, or until they could not spend more resources, thus having to choose one of the solutions they had found so far.

The following section will discuss the possibilities for and obstacles to moving from *state of the art* to something similar to *reflection in action*.

3. Changing Working Practices: Possibilities and Obstacles

Section 1 described a case which I - from my observations - claim not to be atypical where systems development is concerned, and section 2 described what I see as a more ideal way of working. This does not mean that methods for systems development, like those recommended by e.g. Yourdon, Jackson, and advocates of prototyping are of no use. Discussions and experiments with new systems development methods are important activities in a systems department, and I have myself contributed to this discussion (Kensing 84a) and (Kensing 84b). This article focuses on other prerequisites for changing working practices. It looks at the qualifications of the computer specialists, their atttitudes, and the type of activities they initiate or get involved in. This section will comprise the discussion into two concepts: *approaching the product* and *handling project situations*.

3.1. Approaching the Product

A project team's understanding og the product they are constructing, and how this understanding is acqired and used, is crucial to the project's success. This is true irrespective of whether the success is measured in terms of the user organization's satisfaction with the product, or in terms of time and money spent. We will comprise the project team's understanding, and the techniques and tools for acquiring and using this understanding, in the concept *approaching the product*. To obtain this understanding in the analysis and construction processes, the project team uses concepts and models. We will focus on these concepts and models in the following.

To further qualify the content of the concept approaching the product we will draw up a number of options between which the project team - implicitly or explicitly - will have to choose.

The choices are not a question of either/or, but should rather be seen as extremes at both ends of a scale. However, in the MARS-project we have observed that project teams tend to operate towards the extremes illustrated in the left-hand column in the figure below. In the following I will show the limitations this tendency entails.

The intention of introducing the concept *approaching the product* is to draw attention to the right-hand column as an option, and to argue for the expediency of taking these possibilities into account.

The project team has a range of options between these extremes:

The final product is seen as a computer system	( ) (	The final product is seen as a computer based system
The product is specified in implementation-oriented terms	) ( ) (	The product is specified in application-oriented terms
The applied models capture data or information flows	) ( ) (	The applied models capture working routines
Establishing a technically qualified basis for the construction	) ( ) (	Creating a comprehensible basis for the user organization to understand the functions and mode of operation

The observations about the project team's tendency to operate towards the left-hand column should be seen in the light of two other observations: First, contact between the system developers and the user organization is often weak. Secondly, the tool-kits of system developers lack concepts and models for capturing the application-oriented perspective illustrated in the right-hand column above in situations such as analysis, construction, and tests or reviews of sub-products.

One consequence of this is that the project team will find it difficult to understand "the soul of the product", as a project member participating in one of MARS' analysis put it.

This furthermore makes it difficult for people in the user organization to see the link between the project team's specifications and their own practices. And this again has the consequence that the users find it difficult to respond to the various kinds of specifications. Hence the project team does not have a basis for testing whether crucial aspects of the user organization's routines are captured in the specifications or not.

I claim - and this is an essential criterion for evaluation in this analysis - that to ensure a good product the project team must have:

- knowledge of the user organization's specific functions and working routines which are to be supported by a computer system, (e.g. how is production control carried out here?),

- knowledge of the abstract function which is to be supported, (e.g. what is production control all about?),

- knowledge of available standard solutions, (e.g. function and mode of operation of various standard systems for production control),

- knowledge of the available computer system, (e.g. which software tools are implemented?).

The project team in our case basically assumed that it was possible to analyse and construct the system without a thorough understanding of the present working routines in the user organization, and of the future application of the system.

The project acquired a purely abstract understanding of what production control is by studying literature and a standard system. They did not try to understand the specific problems in the maintenance shops, and they did not examine the efficiency of the available software tools.

The project team chose a prototyping strategy, but as the team's vision of the new system was only vaguely anchored in the actual conditions in the user organization, the prototype proved unserviceable. The intention of the project team was to create a comprehensible basis for the user organization, so that the latter could make decisions concerning the functions and mode of operation of the new system. But the model of the future system - represented by the prototype - was based on a data-flow oriented understanding of production control, while the users' understanding was based on the everyday working routines.

The consultant's assumptions were different. This was reflected in his questions to the project team. He focused on the application situation rather than on data flows when he asked for whom the prototype and the final product would be of use. He focused on the organization rather than on the computer system when he suggested that the project team's problem was an organizational problem. The project team had to learn the hard way that this was a reasonable approach: They spent eight months on developing a prototype, only to have it rejected.

3.2. Handling Project Situations

Project teams working with computer based systems of the size and complexity we talk about in this article must expect turbulent situations. Situations which perhaps are difficult to anticipate - situations which require intervention such as changing central prerequisites like: deadlines, basic design decisions, technical equipment, etc. The project team's understanding of situations like that, and the techniques and tools they employ to obtain and apply this understanding, may be comprised in the concept *handling project situations*. I.e. while the concept *approaching the product*

focuses on the *product* resulting from the system development process, the concept *handling project situations* focuses on the *process* in which systems development is carried out. We will focus on how the project team makes itself or others capable to form the system development process.

Again we will list a number of options between which the project team - implicitly or explicitly - will have to choose. And again the choice is not a question of either/or, but should be seen as extremes at both ends of a scale.

The project team has a range of options between these extremes:

To see themselves as external in relation to the process	( ) ( )	To see themselves as part of the process
To consider users/steering committee/others as responsible for the process	( ) ( )	To consider themselves as partly responsible for the process
To make a point of following formal guidelines when making reports	( ) (	To make a point of obtaining and communicating insight into/about the process

The MARS-project has found that project teams show a tendency to operate towards the left-hand column in the table. As was the case with *approaching the product*, the intention with introducing the concept *handling project situations* is to question the expediency of this tendency, and to argue for also considering the possibilities in the right-hand column.

The project team - which ought to have the insight into turbulent project situations - often does not have this insight. And if they do have it, they often have no formal right to make the adequate decisions. This would not be a problem if the insight were made explicit, and therefore could be communicated to the decision makers, but this is rarely the case.

All this will lead to "floating" project situations where the project team will find that decisions are made at random, or not at all. We will return to this and other causal relations later, and continue with describing what I see as necessary conditions for handling project situations reasonably:

- Critical prerequisites (e.g. planned consumption of man time per week, anticipated qualitative requirements to test equipment, important deadlines, anticipated response from the users, etc.) must currently be checked with the actual realities.

- Parties interested in the project, their spheres of interest, and their competence must be identified.

- The process must be reviewed regularly.

When - in our case - problems developed in connection with the testing of the prototype, the project team blamed it on the fact that the foremen refused to test the system, and that they could not continue the design before the test had been executed. In assuming that the problem was caused by the foremen, the project team placed themselves in a position from where they could not contribute to solving the problem - the team placed the problem outside their own sphere of action.

A more constructive way to handle project situations would be for the project team to realize that they were victimized by the situation. They should - in Schön's words - have *stepped into the situation*. Combined with reflections of the kind the consultant made in section 2 (i.e. run-throughs of the type: situation-problem-consequences-solutions-consequences-assessments of whether the consequences would be preferable to the present ones), would enable the project team to act themselves out of a turbulent situation. Action based on explicit attempts at understanding the situation and the possibilities it offers.

Why do project teams tend to place the problems outside their own sphere of action? By working with the reasons - and not by merely providing techniques and tools - can we hope to improve the working practices in system development.

In our case the misfortunes were blamed on the formen. This was perhaps a correct assessment. On the other hand, being unable to enforce the test, the project team should have stepped into the situation and considered possible actions. As it were the project team did not take any constructive steps to solve the problem. This in itself might seem trivial. To create the posiibilities for the project team to stop and consciously try to understand the situation and act accordingly is not trivial: Communicating insight to the interested parties, or taking action directly is difficult - as indicated earlier.

Other difficulties include:

Most projects face the problem of *time pressure*. However, this is a dogma which hinders reasonable project management. "We have problems, but we are too busy to do anything about them". Of course this is an absurd statement, but it captures the reality of many projects.

There are no *techniques and tools for situation analysis* in the tool kits of most system developers. Only a minority of system developers know what they should do to systematically get an overall picture of a turbulent situation. That is why they choose the ostrich way out.

The third and last reason to be mentioned here is that *system developers tend to concentrate on the system* they are to construct, and they spend very little effort on understanding the process in which this is to be done. This is, among other reasons, historically determined by the fact that a tradition has been established among professional computer specialists

which makes the product the all-important element. But with the increasing size and complexity of the products - and the number of people involved in developing them - it will become necessary to spend more time and energy on the process to ensure a qualitatively satisfactory product. The traditional division between those who perform system development, and those who control it is another barrier because in the eyes of the project team, process related assignments is the responsibility of "the others".

3.3. New Activities, Qualifications, and Attitudes

The findings of the MARS-project (MARS 84) indicate that system developers need to change their attitude to the products they construct, and rather see the product as an integrated part of the user organization (a computer based system), than merely as a computer system.

System developers also need to change their attitude to their own rôle in the process. They should rather see themselves as co-actors in changing the user organization, than merely as suppliers of a product which the user organization is solely responsible for implementing. The main argument for this change in attitude is that systems change into becoming increasingly complex, and increasingly integrated in the human working processes.

Even though change in attitude is a neglected topic both in literature on systems development, and in the courses offered in this field, this is naturally not enough to change the practices of system developers. New types of activities must become a natural part of system development projects. I will point out two types of activities in relation to the product and the process, respectively.

In relation to the product the working routines which have to be performed in order to get the computer system to function in the user organization are more or less neglected, both where analysis and construction are concerned. In addition there is a need for activities which aim at describing or presenting the changes which will take place in the user organization. These are activities which concern the part of the computer based system which lies outside the computer system, and which is the part that is given low or no priority at all in practical systems development.

In relation to the process, project establishment activities (evaluation of the prerequisites of the project, and - on this background - regulation of the project to fit the prerequisites, or regulation of the prerequisites to fit the project) are often neglected. In addition there is a need for regularly measuring the status of the project, i.e. activities of the type we have described as *reflection in action*. Activities which facilitate a conscious regulation of the project while it is carried out.

Change in attitude, and initiation of new types of activities are intimately tied to changes in the qualification structure of those who participate in the system development process. Here technical qualifications are generally good. There is, however, a lack of qualifications enabling system developers to design the working processes and assess the organizational consequences of introducing various system proposals. There is also a lack of qualifications concerning project management.

As already mentioned earlier on in this section, discussions about, and experiments with new system development methods are also important prerequisites for changing practice. This article, however, has not dealt with this discussion, and only mentions it for the sake of completeness.

The suggestions for change listed here will face many obstacles in practical systems development. Time and other scarce resources always obstruct change, but also the fact that the suggestions would push system developers to the brink of their current qualifications is a major impediment. It is thus not surprising that well-known activities, at which one knows one is good, are given higher priority. Finally the low priority given to training and personal development characterizing many organizations, also plays an important rôle. The actual fact is that too little money is set aside for this, and the money which is set aside is primarily spent on objectives like teaching new methods for programming, mastery of program languages, or training in a new operating system.

Changes in working practices in systems development will depend on different conditions, according to the kind of organization in which they are to take place. In our experience they are easier to implement in organizations which have their own systems department, and hardest to implement in software houses which primarily make their living by selling standard systems. Our current research, which is carried out in co-operation with practitioners, indicates that - in spite of the obstacles - there is a growing interest in discussing and changing working practices in systems development.

References

DeMarco, Tom: Structured Analysis and Systems Specification, Prentice Hall, Englewood Cliffs, 1979.

Jackson, Michael: System Development, Prentice Hall, 1983.

Kensing, Finn: Towards Evaluation of Methods for Property Determination, in Beyond Productivity: Information Systems Development for Organizational Effectiveness, T.H. Bemelmans (ed.), North Holland, 1984.

Kensing, Finn: Property Determination by Prototyping, in Approaches to Prototyping, R. Budde et al. (eds.), Springer Verlag, 1984.

Lanzara, G.F. and Mathiassen, L.: Mapping Situations within a System Development Project, University of Aarhus, 1984

MARS-project: Reports No.s 1-5, Aarhus University, 1984.

Munk-Madsen, A.: Practical Problems in Systems Development, Proceedings from the 7th Scandinavian Research Seminar on Systemeering, Helsinki 1984.

Schön, D.: The Reflective Practitioner, Basic Books Inc., 1983.

Yourdon, E.: Managing the Systems Life Cycle, Yourdon Press, 1982.