DIALECTICAL REFLECTION IN 
INFORMATION SYSTEMS DEVELOPMENT

GRO BJERKNES
Department of Informatics at University of Oslo
P.O. Box 1080, Blindern, N-0316 Oslo 3, Norway

Abstract

Unstructured and conflicting situations in systems development are often discussed in terms of bad planning and lack of control. In terms of dialectics, however, such situations can be regarded as a source of insight and development.

In this article, a problematic situation from a practical project is described. Different techniques that could be used in managing the situation are discussed. Finally, the situation is analysed according to the notion of contradiction.

Dialectical reflection permits problematic situations to be regarded as a result of and a source for development, instead of interpreting the situation as a result of bad planning. This can improve the understanding of processes that lead to a problematic situation. Thus the insight gained from dialectical reflection should provide a basis for applying well-known methods and techniques in information systems development.

Keywords: information systems development, project management.
1 Introduction

In information systems development, situations that are unexpected and confusing often occur. These situations usually result in changes, be it in the project plan, or in (parts of) the technical design. The changes are perceived as something unwanted, a deviation from a plan or a result of bad planning—in short, they are seen as lack of control of the systems development process.

One way of coping with lack of control is to introduce dynamic planning, such as risk driven project management, as proposed by Boehm (1988), or by evolutionary development, as proposed by Floyd et al. (1989). Better planning reduces uncertainties, and is one way of coping with complexity, but planning cannot eliminate all factors that contribute to confusion and change. Suchman (1987) supports this by writing that plans can be used as explanations or reconstruction of actions, but that actions seldom take place according to prescribed plans.

The goal of a system development project is usually ambiguous and unclear, and it changes during the process, as reported in (Curtis et al. 1988, Rolskcv 1990). This is because knowledge about the computer system and its use context develops during the system development process, as argued in (Brooks 1987, Floyd 1987). New insight results in new ways of seeing the computer system, in turn creating chaotic and contradictory situations. As Gould (1988, p. 762), puts it, “development is full of surprises”.

In addition to the uncertainties concerning knowledge and experience in systems development and in the application area, there are uncertainties concerning the familiarity with the chosen hardware and software, according to Davis (1982) and Saarinen (1988). Even though a certain familiarity is reached, hardware or software constraints can nevertheless result in feasible prototypes not being implemented, as reported by e.g. Avison & Wood-Harper (1991).

Last, but not least, systems development is not an isolated process. As Engeström (1987) argues, an organizational change process can be contradictory to other organizational processes, and thus be stopped due to organizational resistance. Keen (1981) and Kling (1987) argue that this organizational phenomenon also applies when systems development processes interfere with other organizational processes. Situations like this cannot be anticipated, and thus cannot be planned.

Systems development, as such, can be regarded as a change process, starting from some vague ideas of an information system, ending in a computer system in use. In this process, ideas about design of the system and how it should be implemented and introduced continually change. Changes can be regarded as lack of control. But changes can also be regarded as a result of development. This is the basic assumption in dialectical thinking. The notion of contradiction is central in dialectics, and it is imperative in order to understand change. Thus, dialectical reflection is a means for understanding systems development, especially
when changing conditions and new insight result in unexpected and confusing situations.

As Parnas & Clements (1986) argue, we should strive for a rational development process although we know that this cannot be achieved. In order to achieve a rational design process, the insight gained through dialectical reflection should be used for e.g., more realistic planning.

There are many different dialectical approaches. According to Elster (1985), dialectics is a methodology of social science. Dialectics is an analytical tool for explaining relations and understanding change in society.

Churchman (1971) has specific experience from dialectical inquiry from experiments within design groups. He discusses dialectics in relation to design, and he concludes that a dialectical approach will result in a never-ending inquiry process and no solution. Moreover, he argues that dialectics is a means for creating “a more knowledgeable political process in which the opposing parties are more fully aware of each other’s Weltanschauungen and the role of data in the battle for power.” (Churchman 1971, p. 185).

Elster is inspired by Marx and his materialistic dialectics, while Churchman is inspired by the more idealistic dialectics presented by Hegel. In this article, I apply a third dialectical approach, Mao’s dialectical approach, in order to analyse a problematic situation from the Florence project. Mao claims to form the basis of his work from the dialectical approach of Marx and Engels, but he differs from Marx by looking at situations, not at systems. Where Marx bases his work on the system of production, Mao approaches a problematic situation and lets what he calls the principal contradiction constitute a frame for understanding the situation. Moreover, Mao, in line with Hegel, argues that contradictions exist in the world, not only in the mind.

My aim is to show how dialectical reflection can be used for gaining insight into processes that lead to situations that are perceived as conflicting and confusing. The analysis stems from reflections which arose after the completion of a project. However, the notion of contradiction can very well be used for examining problems in ongoing projects, as demonstrated in (Bjerknes & Øgrim 1990).

The article is structured as follows: in Section 1, a situation from the Florence project is described; in Section 2, techniques for project management and methods and techniques for coping with unstructured situations are discussed in relation to the situation; in Section 3, Mao Tsetung’s notion of contradiction is presented and discussed, and in Section 4 the notion is applied to the situation, and my experiences from applying Mao’s notion of contradiction are discussed. Section 5 contains concluding remarks.

2 A Situation from the Florence Project

The Florence project was a research project concerning the development and use of computer systems in nurses’ daily work. The project can be classified as be-
longing to the Scandinavian critical tradition, as defined by Bansler (1987). In addition to exploring system development methods and techniques, the goal of the project was to build computer systems for nurses’ daily work. In the project, we analysed nurses’ work on two hospital wards by observation, interviewing and different kinds of activities termed as mutual learning. In addition to nurses and computer scientists, a social anthropologist participated in the project. The research method in the mutual learning activities was therefore inspired by anthropological research methods. In the project, we implemented two prototypes. One proved to be inadequate for practical use, and we learned how prototypes can be used to reveal misunderstandings of the use situation, see (Bjerknes & Bratteteig 1987). The other prototype was successful and was used by the nurses in their daily work, see (Bjerknes & Bratteteig 1988).

2.1 Crisis in the Florence Group at the Cardiological Ward

The following situation arose when the Florence project worked with nurses from the Municipal Hospital in Akershus. The situation is described according to how I experienced it as one of the computer scientists participating in the project.'

Imagine you are a fly on the wall at the following meeting of the Florence project.

The meeting has been called by the nurses in the Florence group on the cardiological ward, and the meeting takes place on the ward. Two computer scientists, and the Florence group, consisting of four nurses, are present. The current activity in the project is the implementation of a computer system suggested by the nurses six months previously. The computer scientists are implementing this on a computer at the University, 45 minutes away by car. There have been problems, and subsequently a time delay. It is a couple of months since the nurses and the computer scientists last met, and the atmosphere is gloomy.

Nurse: You see, motivation for going on with the project is rather low at the moment.

Computer scientist: How come?

N: It is difficult for us to defend a project when nothing seems to happen for months. The computer equipment is just standing there, and to us it seems to be a real waste of resources.

CS: Do you want to stop the project?

N: (no answer)

CS: If you don’t want to use the system, we can phone the manufacturer and ask them to fetch the equipment. There’s no point spending more energy and resources getting the computer system up and running if you don’t want to use it.

N: ... Well ... we didn’t exactly mean that.
CS: Do you think there's a chance of getting the project going again?

N: (looking at each other, hesitatingly) Well, there may be a chance.

(the nurses and computer scientists agree to try once more)

CS: How should we go about explaining the situation to the other nurses?

They then plan a strategy for improving feelings towards the project on the ward.

2.2 How Could This Situation Arise?

For the computer scientists, the situation that occurred was unexpected and confusing. On the one hand, they understood the nurses’ lack of motivation and their wish to give up. On the other hand, they had a strong personal interest in keeping the project going. Moreover, the computer scientists could see that there was no time for lengthy discussion as time delay was a key factor in the nurses’ frustration.

To understand the situation, it is necessary to understand something about its history. When the situation arose, the nurses and the computer scientists had worked together for almost one and a half years. The first six months were used for mutual learning between the groups. The nurses and the computer scientists decided, after a negotiation meeting, that the computer scientists should build a small computer system that could replace the notes taken by the nurses on scraps of paper during the report meetings and during a shift. This was conceived to be a small system which would not make too many changes to the daily working routines, and which would be fairly easy for the nurses to learn and use. In addition, most of them saw the benefits of this computer application. This indicated it could be a good way to introduce computer systems to nurses, and it seemed probable that the system could be built with the limited resources available to the project.

The summer was spent installing the computer equipment. Several months were needed to get the system working. There were some bugs in the installed software too, and the computer scientists used some time to find out whether it was their understanding of the documentation that was at fault or whether there actually was more serious trouble. They were offered courses by the manufacturer, but due to teaching duties at the University, it was impossible for them to attend these. During the autumn, they managed to make a suggestion for screen layouts by using a screen editor, and a suggestion for report layouts by making some proposals with the text editor. The nurses were happy with the proposals, and expected to have a working computer system within a short time.

The computer scientists were more reluctant. They were not satisfied with their working conditions, and they knew that their time on the project could be used more efficiently if they could work at the computer science department,
instead of borrowing a car and driving 45 minutes each way to the hospital. In
addition, they felt unwanted on the ward—there were few possibilities to call the
service department at the manufacturer’s due to only one phone on the ward,
and there was little space for discussing programming problems that turned up
in the process. Therefore they agreed to get duplicate software installed at a
computer at the University from the same manufacturer. When this agreement
was reached, even the computer scientists became optimistic, and attended a
course to learn the 4GL they had decided to apply.

It turned out that there were a lot of problems connected with installing the
software on the computer at the University. Some of them were due to ‘easy’,
though time-consuming errors, like receiving wrong diskettes or programs with
compilation errors. Other errors were harder to find, e.g. when they were suddenly
unable to install new software at all. This was due to a hardware error which
took about 1½ months to get fixed. Along with this, the computer scientists also
had technical problems in as much as that it was not possible to implement what
they wanted in an easy way with the given programming utensils (a 4GL and a
report generator). Much time was spent to exploring the limits of the utensils
in order to implement the desired solution. During this process, a lot of errors
naturally arose as these limits were stretched.

The computer scientists were convinced that the way information is presented
also conveys information, and that the nurses’ needs were best taken care of by
implementing their specifications. Therefore, the computer scientists rejected
the idea of implementing a solution that better suited the utensils, in spite of the
additional time needed to do so.

While the computer scientists were programming, the nurses were waiting.
Week after week passed, and no computer system appeared. They didn’t know
what was happening, since the computer scientists were far away, still program-
ning.

At the meeting, the computer scientists learned that after a while the nurses
felt left out. They thought the computer scientists had forgotten about their
existence. They couldn’t understand the delay: after all, they had chosen a
computer system that was very simple, and they had seen examples of both
screens and reports months ago. The nurses didn’t understand that a simple
solution in relation to the nurses’ work isn’t always a simple solution from a
computer scientist point of view, and that a demonstration of possible interfaces
may be far away from a working computer system. At the meeting, the computer
scientists also learned that there was an internal conflict on the ward. The conflict
primarily concerned the leader of the ward. She was also a member of the Florence
working group. Therefore the computer system—and the computer scientists—
became a symbol of the conflict. This made the delay even harder to explain and
defend to the other nurses on the ward.

In short, the time needed to get acquainted with unknown equipment and
unknown software was underestimated, and so was the time needed for exploring
and stretching the limits of the utensils. At the meeting, the computer scientists
understood that they had underestimated the importance of regular contact with
the nurses, especially in periods where no visible progress was made.

3 How Could the Situation Have Been Handled Better?

The computer scientists, unexpectedly, found themselves in a situation best de-
dscribed as a conflict between their own wishes of building a computer system
to be tested out by the nurses, and the nurses’ interest in stopping the project.
They could also see the danger of not getting the system in use due to the conflict
between the nurses. The best thing would of course have been to avoid the whole
situation—but that was too late. Given this situation, what kind of advice could
the computer scientists get from literature to help them handle the situation?

In this section I will first discuss whether techniques known from Quality
Control could have been used. I will then discuss Soft Systems Methodology,
and finally I will discuss whether the building of problem networks could have
been useful in helping the computer scientists handle the situation in a more
professional way.

3.1 Quality Control Techniques

Many techniques in the area of systems development management exist for evalu-
ating projects. Most of the techniques are drawn from the area of software
quality management. In this area we can distinguish between four types of evalu-
ation: (1) author-reader cycles, (2) walk-throughs, (3) inspections and (4) re-
views, described in e.g. (Sommerville 1989). Although these techniques are used
for evaluating and regulating projects, they are bound to the product, i.e. the
computer system, or to intermediate products, like project plans and design doc-
uments. Author-reader cycles are a document-driven technique that requires
either a version of the code or a version of the documentation. Walk-throughs
are also connected to discussion and an evaluation of the product. Inspections
use a check-list as a basis, and again software or an intermediate product is eval-
uated. Review is a formal session, the goal being to evaluate the quality of a
product. Thus the purposes of these evaluation techniques are (1) to take care
of the technical and/or functional quality of the computer system which is built
so that the system is easy to maintain, and (2) to keep track of productivity in
the project.

The positive effect of quality control is that computer scientists are urged to
do serious planning, to put effort into coordinating the project as a whole, and
to measure products against requirements.

The negative aspect of quality control is that the techniques require that parts
of the computer system and its connected documents are in such a state that it
is possible to evaluate them. Moreover, these kinds of techniques also assume
that there are criteria against which the products can be evaluated. Last, but not least, quality control is a means for preventing unexpected situations, not for handling them.

The question is whether quality control could have helped the computer scientists to avoid the situation, and whether quality control could have helped them handle the described situation. In this case, the answer to both questions is no. Quality control could not have helped to avoid the problematic situation, because it arose partly because of unpredictable events, such as hardware errors and tool limitations. But regular project evaluation could have forced the computer scientists to reflect on the problems earlier, leading to revised plans, instead of just delaying the delivery date. Although the computer system was close to a state where evaluation in terms of quality control would have been valid, it was not the computer system that was the problem. This meant that the described situation could not have been handled with any of these techniques. Nevertheless a systematic use of quality control techniques during the project could have made the computer scientists aware of their unrealistic planning.

3.2 Soft Systems Methodology

Soft systems methodology (SSM) and its theoretical foundation is described in (Checkland 1981). Checkland’s quest is to find out how to handle unstructured problems in organizations. He argues that hard system thinking is rewarding, but not sufficient for solving organizational problems, due to the fact that the problem is taken for granted. This is the basis for SSM.

When applying SSM, one starts with a real-world analysis to pin-point the problems. The real-world analysis consists of drawing ‘rich pictures.’ This looks easy when described in Checkland’s book, but actually much knowledge about the organization is required to draw such pictures and transform them into root definitions, which is a phase that belongs to the systems thinking part of SSM. A root definition is a hypothesis that concerns the improvement of the problematic situation, and it describes what the system is. The root definition is transformed to a conceptual model that expresses which activities the system must carry out in order to become the system named in the root definition. Having formed conceptual models on the basis of root definitions, we go back to ‘real world’ activities, and use the models for discussing feasible and desirable changes. The methodology is presented in sequential phases, but Checkland (1981) stresses that the phases do not have to be carried out in a sequence.

SSM has developed through use. There are some differences between SSM presented in (Checkland 1981, Checkland & Scholes 1990). The new version has e.g. a more thorough analysis of the problem situation before the system thinking begins. Moreover, Checkland & Scholes (1990) introduces two modes in which SSM can be used, the intervention mode and the interaction mode.

The intervention mode is the one presented in (Checkland 1981), and it uses a frame of system ideas in order to inquire into and improve some parts of the
real world. Checkland & Scholes (1990, p. 60) say that “in this mode SSM easily is accepted as a consultant’s tool, making studies of problems that belong to others.” Thus this mode more-or-less presupposes a neutral change agent. It also requires a rigorous analysis due to the fact that intervention is the goal. The computer scientists could not have used SSM in this mode. After all, the study involved their own problems, and although intervention was needed, there was no time for starting an intervention process in the middle of the implementation activities.

The other mode, the interaction mode, “takes as its focus of enquiry the process of learning one’s way to purposeful improvement of the problem situations”, according to Checkland & Scholes (1990, p. 283). Thus the intervention mode focuses on the problematic situation, while the interaction mode focuses on the process of learning. Thus the interaction mode would have been more appropriate for the computer scientists (although unknown at that time). To a certain extent they did interact with the processes going on, but it would probably have helped a lot to have a framework for making the different expectations and priorities clear from the outset.

3.3 Problem Networks

Like quality control techniques, the building of problem networks is a technique for a systems development project group. Munk-Madsen (1986) is concerned with evaluating systems development projects in problematic situations, and he discusses the building of problem networks as a means of defining the problem. The technique is a further development of mapping, cf. (Lanzara & Mathiassen 1984). A network for a problematic situation will consist of problems and their causes and consequences, where one problem may be the cause or a consequence of another problem.

Munk-Madsen (1986, pp. 13–14) says that candidates for nodes to the network are statements that can be expressed as antagonistic relations. Moreover, a network node should be formulated as a contradiction between the situation and some theoretical statements, between the actual situation and the expected situation, or between the expectations of different people involved in the project.

The network in itself is not enough, however. In order to set the problem, a ‘central problem’ must be identified. Munk-Madsen argues that the real difficulty of setting the problem lies in the identification of the ‘central problem’. He gives hints on how to identify the central problem by describing some characteristics this problem node may have.

Thus the building of problem networks is a technique that doesn’t have to be used in advance in a preventive manner, like the quality control techniques. Nor is it necessary to do a rigorous analysis, which is proposed by SSM. Problem networks can be built because things did not develop according to the plan, or because the expectations were not clear.

This seems to be exactly what the computer scientists could have needed—
technique that can be used for evaluating a situation. Also the technique takes into account that it will be used by people with different interests in the outcome of the situation.

4 Dialectics

In dialectical thinking, the concept of contradiction is essential, but many philosophical writers take the notion for granted without explaining it. Mao Tseturig (1937), however, has tried to explain the notion of contradiction and some of the complex aspects of it.

In everyday language, ‘contradiction’ has approximately the same meaning as ‘conflict’. In this context, however, ‘contradiction’ can be translated as ‘relation’, it is not something bad or something that has to be avoided.

In this section, Mao’s (1937) notion of contradiction is presented. Different concepts are presented, and these are illustrated by an example of a contradiction in connection with the described situation. Next, a criticism of Mao’s notion of contradiction is discussed. At the end of the section, the approaches presented as possible candidates for helping the computer scientists to handle the problem are compared with Mao’s dialectical approach.

4.1 Mao Tsetung’s Notion of Contradiction

Contradiction is a notion that considers the totality of a relation. The notion also emphasizes processual aspects of a phenomenon.

A contradiction consists of two opposites. The relationship between the two is uneven—one of the opposites dominates the phenomenon. This opposite is called the principal aspect or opposite of the contradiction. The principal aspect of a contradiction will change with the development of the phenomenon.

Mao further says that there is identity and struggle between the opposites or aspects of a contradiction. The struggle between the aspects is what we often connect to ‘contradiction’ in everyday language. It points to the conflicting, or disharmonic, relationship between the two aspects. However, there is also identity between the aspects. The identity points to the interconnection between the opposites. Both struggle and identity are necessary qualities of a contradiction.

The fact is that no contradictory aspect can exist in isolation. Without its opposite aspect, each loses the condition for its existence.

. . . It is so with all opposites; in given conditions, on the one hand they are opposed to each other, and on the other they are interconnected, interpenetrating, interpermeating and interdependent, and this character is described as identity. (Mao 1937, p. 338)

Thus the identity and struggle between the aspects of a contradiction are related to the fact that the principal aspect of a contradiction is changing and that the two aspects coexist in a single unity.
One contradiction that occurs in the situation we are examining, is the contradiction between the computer scientists’ tasks in the Florence project and their other obligations. The identity between the opposites is that both tasks must be carried out within a limited time span. It was not acceptable to do only teaching, because then the project commitments would have been neglected. On the other hand, it was not acceptable for the computer scientists only to take care of the project, because this would imply a breach of their work contract. The struggle between the opposites was that it wasn’t possible to satisfy both sets of obligations within the existing working conditions, therefore it was always a question of giving priority to one of the two.

In the situation, the principal aspect of the contradiction turned out to be the work contract, i.e. the teaching part of work. This caused a problem because the computer scientists wanted the teaching part of the work to be less dominant than it turned out to be. The problem might have been solved by simply doing less teaching, but the computer scientists did not regard this as an adequate solution to the problem. Instead the computer scientists tried to do something with their working conditions, so they could spend more working hours on the project. In this way the contradiction triggered development.

A phenomenon does not only consist of one contradiction.

There are many contradictions in the process of development of a complex thing, and one of them is necessarily the principal contradiction whose existence and development determine or influence the existence and development of the other contradictions. (Mao 1937, p. 331)

As there is only one principal contradiction, we should devote effort to finding it in order to understand the phenomenon.

It is important to note that although there is one principal contradiction at a specific point in time, the principal contradiction changes during a process. Mao says that a contradiction can be in a state where the identity between the opposites is dominating, i.e. that the contradiction does not represent any problem or conflict. On the other hand, the struggle between the opposites can dominate, and in this situation there will be a conflict. In this situation only a contradiction can be resolved, i.e. the process is transformed, and another contradiction will become the principal contradiction.

At the beginning of the process that led to the situation we are describing, the contradiction between doing project work and doing teaching did not appear as a conflict for the computer scientists, thus nothing was done with the situation. However, when too much teaching was done at the expense of the project work, the struggle between the opposites became too dominating, and the situation was changed. This resulted in the contradiction being resolved, i.e. that another contradiction became the principal contradiction. This will be discussed in Section 4.

Mao says that there are universal and particular qualities of all contradictions.
The universality or absoluteness of contradiction has a twofold meaning. One is that contradiction exists in the process of development of all things, and the other is that in the process of development of each thing a movement of opposites exist from beginning to end.

...In every form of motion, each process of development which is real (and not imaginary) is qualitatively different. Our study must emphasize and start from this point. (Mao 1937, pp. 316, 320–321).

The contradiction between doing different tasks within a limited amount of time is a quite common one in systems development. Therefore it is easy to see general aspects of the situation. However, the contradiction is particular in the sense that it is the only time it occurs with those specific persons involved, their particular experiences and their current working conditions. The two opposites can be described as carrying out the tasks of the project vs. carrying out tasks concerning other parts of the job, e.g. teaching.

Thus, the universal and the particular, the two aspects or opposites of a contradiction, the struggle and identity between the aspects, the principal aspect of a contradiction, and the principal contradiction are important concepts for understanding contradictions.

4.2 Why Mao’s Notion of Contradiction?

Mao’s notion of contradiction can be questioned from a philosophical point of view, and in this section some of the criticism raised against this notion is discussed.

1. What is the difference between dialectics and dualism?

Mao can be criticised for his rather dualistic version of dialectics. He introduces many pairs which cover a whole, e.g. internal and external contradictions, the universal and the particular, struggle and identity, the two aspects or opposites of a contradiction. As Israel (1979) points out, the important thing here is that there is movement. The movement is due to the unequal weight between the opposites in a contradiction, so that there is one principal aspect of a contradiction. Also, the different contradictions do not have the same weight, so there is one principal contradiction. And the principal contradiction changes. Again, the dualistic influence can be explained in terms of Mao’s cultural background and his audience, since the Chinese tradition has a long dualistic history (Yin—Yang).

2. Are all relations contradictions? And is there one principal contradiction?

Elster (1985) argues that Mao uses the notion of contradiction about nearly everything, while Marx used the term only about a certain (general) relation which Elster calls “the fallacy of composition”. The same criticism can be raised from an activity theory point of view, although the terminology is different: e.g. Engestrom (1987) just talks about contradictions
within a system, whereas Mao does not give any frame for discussing and understanding the contradictions. In this way the notion may become very subjective, although Mao warns strongly against a subjective analysis of a situation.

It is true that Mao calls nearly everything for a contradiction. The point is that all contradictions are regarded as candidates for the principal contradiction. The principal contradiction is important, because it represents a frame for an unclear situation. It can of course also be discussed to what extent an ‘objective’ principal contradiction exists.

3. Do contradictions exist in nature? And can contradictions be inherent in ‘things’?

‘Dialectics’ originally means ‘discourse’. In the modern Greek language it is translated as “understanding with words”. This seems to be opposite to Mao’s understanding of contradictions as something that exists in nature. In this question, Mao is in line with Hegel. According to Næss (1972), Hegel didn’t distinguish between contradictions that exist in nature and contradictions as a construct of the mind. For Hegel, these were basically two aspects of the same phenomenon. We should also remember Mao’s cultural background and his audience: in Taoism there exists a notion of ‘P’u’, which means something like “unshaped, natural block.” Thus the notion of internal contradictions in nature can be seen as a way of adapting dialectics to a taoistic way of thinking. Then it makes sense to say that all internal contradictions are the basis for all change, although external contradictions are the conditions for change.

Mao himself uses the notion of contradiction in a very pragmatic way—to solve political problems, to discuss the war against Japan, to evaluate problems that China has to solve. Independent of whether contradictions exist in nature or not, it is possible to use Mao’s notion of contradiction for framing a situation in a way that makes sense, cf. Section 4.

4.3 The Notion of Contradiction Compared With the other Approaches

In this section I compare the dialectical approach based on Mao’s notion of contradiction with the other approaches presented as possible candidates for handling the problematic situation of the computer scientists. I start with the most similar approach, the building of problem networks, and continue with soft systems methodology. At last I compare quality control techniques with a dialectical approach.
The Building of Problem Networks

Munk-Madsen (1986) is very inspired by dialectical thinking. He refers to Mao when he introduces the notion of contradiction. In many ways, Munk-Madsen’s approach can be seen as a way of transforming Mao’s vague prescriptions of how to do dialectical analysis into a technique that is adapted to evaluating systems development projects. It should be noted, however, that Munk-Madsen is very much oriented towards conflict. According to his definition, a problem node should be an antagonism. Thus contradictions in which struggle between the aspects is not dominating, will not be candidates of nodes for a problem network. Moreover, the description of network nodes are instances of the more general contradiction between reality and visions of the future. Therefore problem networks can help to find a specific kind of contradiction.

The building of problem networks doesn’t make the substantial part of the dialectical analysis easier, but it contributes to finding contradictions, or problem nodes, and to determining whether the principal contradiction, or central problem, is identified or not.

Soft Systems Methodology

There are similarities and differences between soft systems methodology (SSM) and a dialectical approach.

The most important feature both of these approaches have in common, is that they are based on an actual, problematic situation which is used when building a frame for understanding it, in contrast to assuming a predefined frame and adapting the situation to this frame, or system, as recommended by e.g. Engestrom (1987). Both approaches emphasize that it is not trivial to frame a problematic situation in a fruitful way.

In addition, both approaches focus on change, although in different contexts. A dialectical approach regards change and development as the central property of the material world, and dialectical reflection is a means for understanding processes. SSM, on the other hand, regards change in the context of learning, which is a more idealistic approach.

One main difference between the approaches concerns the focus on analysis vs. design. A dialectical approach emphasizes analysis. In order to understand why a situation is as it is, (processes of) the past is used to understand the present. The main point of dialectical reflection is to find the principal contradiction of a process in order to understand a situation. This insight can of course be used for creating visions of the future, but focus is on analysis of the present situation.

SSM on the other side, focuses on visions of the future. Although an analysis of the present is required, the main concern of the methodology is the improvement of a problematic situation. The focus is on how a future situation, which is both feasible and desirable, would look. Thus SSM is much more design-oriented than the dialectical approach.
A second difference is the notion of world. SSM distinguishes between two worlds, the real world and the systems world, whereas the dialectical approach does not make such a distinction.

The systems world of SSM is very different from a dialectical approach since the world of SSM is regarded as being stable and harmonic, as pointed out by e.g. Lilienfeld (1978). Mathiassen & Nielsen (1989) try to embed dialectical reflection in the systems world of SSM by extending the methodology with a new kind of root definition that expresses dialectical relations. Due to the fact that the assumptions of the systems world view are antagonistic to those of the dialectical world view, dialectical root definitions do not fit very well in the systems world of SSM.

In the real world of SSM, the goal is to improve understanding and communication in order to decide upon feasible and desirable changes of an organization. The means for achieving this is the use of systems thinking. The discussion of, and choice between, different root definitions and the related conceptual models makes us aware of different Weltanschauungen. Contradictory conceptual models may exist which are discussed and compared. The notion of contradictory root definitions of Mathiassen & Nielsen (1989) could probably help to improve the discussion by relating the different root definitions to each other.

SSM has much in common with a hermeneutical approach, which describes understanding as a dialectical process, according to Linge (1976). In this context ‘dialectical’ means an idealistic dialectical approach, i.e. the dialectical relations are mind constructs. Thus the notion of dialectical that can be associated with SSM, differs from Mao’s notion of dialectics by not sharing the profound assumption that the material world is developing, and that contradiction is the basic notion for understanding these change processes.

A third difference between the two approaches is the notion of principal contradiction. This central concept of the dialectical approach is lacking in SSM, due to the focus on learning in SSM. If there were a principal contradiction, it would not be a ‘fair’ learning process, thus the notion of principal contradiction seems to be impossible within the framework of SSM. Instead SSM uses the notion of feasible and desirable changes, which is a rather weak concept. According to my personal experience, however, a root definition and its corresponding conceptual model that expresses feasible and desirable changes, can express what in a dialectical approach would be called a better balanced principal contradiction. Thus it seems that the two concepts may have some similarities in practical use, even if they cannot be compared on a theoretical basis.

In spite of their different theoretical backgrounds, the dialectical approach and SSM seem to offer improvements to each other when it comes to practical use. On one hand, the notion of principal contradiction and the notion of dialectical relation as such can contribute to the real-world part of SSM. On the other hand, it seems that the systems-world part of SSM can contribute to finding contradictions, as shown by Mathiassen and Nielsen (1989), and in this way support dialectical reflection.
Quality Control Techniques

Quality control techniques force system developers to plan, to determine goals, and to evaluate whether they reach the goals or not. Thus quality control techniques can help to avoid problematic situations by improved project management. Dialectical analysis, on the other hand, is a technique for understanding processes that prevent the goals from being met. Thus both kinds of techniques are necessary, and complementary, in systems development—quality control for making and following plans in order to reach the goals, and dialectical analysis for understanding why the actual course of the project did not contribute to achieving the goals.

5 The Situation Can Be Understood in Terms of Contradictions

The meeting described in Section 1, and the gloomy atmosphere, can be described as a result of several contradictions and how they developed. Some of them were solved in the course of the process, even though the solution led to other contradictions.

In this section, the different contradictions that were perceived are presented, along with how they relate to each other. Following this, the contradictions are described in detail. Finally, the problem of finding contradictions is discussed.

5.1 The Situation in Terms of Contradictions

At first, the contradiction between the project commitments and the work contract was regarded as the principal contradiction by the computer scientists. This contradiction was solved when the computer scientists could start programming at the University, because the duplication of software meant a significant improvement of their working conditions. Then another contradiction emerged as the principal contradiction: the contradiction between software made for traditional applications and the use of this software for non-traditional solutions. This contradiction seemed to be resource demanding, but solvable. Consequently the computer scientists spent much time on programming. Since the equipment was placed at their ‘normal’ working place, they lost the regular contact with the nurses during this period. The concentration on programming in favour of keeping in touch with the nurses can also be expressed as the contradiction between the primary and secondary tasks of the project. The primary task of the project was to test out computer applications for nurses’ daily work. To do this, however, it was necessary to have a working computer system, and this can be regarded as a secondary task. On the ward, there was already a conflict between the nurses. The conflict became a part of a contradiction, the contradiction between negative attitudes towards the computer system and the need for the goodwill of all the nurses on the ward in order to fulfill the objectives of the project.
While programming, the computer scientists conceived the contradiction between software made for traditional applications and the use of this software for non-traditional solutions as the principal contradiction in the project. This was not quite right, however, as they were later to experience at the meeting described in Section 1. As the problematic situation arose, they did not think in terms of contradictions to solve the problem. However, they tried to find out what to do to motivate all the nurses on the ward, and they reestablished regular contact with the nurses again. Stated in terms of Mao's notion of contradiction: before the meeting, the computer scientists conceived the contradiction between software made for traditional applications and the use of this software for non-traditional solutions as the principal contradiction. However, at the meeting, the computer scientists changed their minds, and came to regard the contradiction between the Florence working group and the other nurses on the ward as the principal contradiction.

From a more reflective perspective, neither of the two turned out to be the principal contradiction of the situation. The principal contradiction in the situation was the contradiction between primary and secondary tasks. When the computer scientists started to concentrate on the primary task, the nurses and the computer scientists started taking action to solve, or at least minimize the conflict between the Florence working group and the other nurses on the ward. This illustrates (1) that it is important to do something about the principal contradiction in order to change the situation, and (2) that it is far from obvious what the principal contradiction in a situation is.

The contradictions presented so far have all appeared as conflicts, i.e. the struggle between the opposites dominated the relation. There were also some contradictions that represented latent conflicts in the situation, and which turned out to be no problem at all. For instance, there could easily have been problems to reach the goals of the project, i.e. use a computer system in the nurses' daily work, if the exceeded time limits had resulted in no extra working hours for the nurses on the ward. This would have had the character of a contradiction between the resources in the project on the one hand and the goal of the project on the other.

5.2 The Contradictions in Detail

I will now go into detail of the different contradictions that were perceived. The aim of doing this is to give some more examples of the notions of identity, struggle, and the principal aspect of a contradiction. The contradiction between the project contract and the work contract for the computer scientists, has already been presented, therefore it is omitted from this section.
The contradiction between software made for traditional applications and the use of this software for non-traditional solutions

The identity of the opposites in the contradiction was that the software was installed in the computer, and without software it would be difficult to implement anything at all, whether traditional or not. The struggle of the opposites was that the software was not designed for programming the kind of programs the computer scientists wanted to implement in the project.

To avoid the problem connected to the application area of software, it would have been possible to use traditional programming languages, like Pascal or Cobol, a screen handler and a database system. This set of utensils would have given wider opportunities for the implementation of the system. Then the resources would have been tied to educational costs, to learning the screen handler and the database system. It is not easy to judge whether this would have been more efficient. In the beginning, the principal aspect of the contradiction was traditional software used for non-traditional applications, in the sense that the computer scientists evaluated it to be easier to learn the 4GL and work around it, than to work with a 3GL, the screen handler and the database. This remained the principal aspect, partly because it would have been impossible to change to 3GL and the database system due to time constraints.

The contradiction is connected to the contradiction between restrictions and freedom of action. Usually, the different kinds of generators are made in order to standardize programming work, by looking for similarities between a class of common programs, and thereafter making a template for efficient programming of these kinds of programs. However, the template may be an obstacle for adjusting computer systems to new application areas. Nevertheless, it is possible to transcend the inherent assumptions of the generators.

This illustrates that the same phenomenon can be explained by different contradictions, and that the difference in phrasing is connected to what is conceived as most important.

The contradiction between the primary and secondary tasks in the project

The primary task in the project was to develop, test and evaluate a computer system for nurses’ daily work. To do this, a computer system had to be implemented. When seeing the project as a whole this was a secondary task. The identity between the opposites was the impossibility to reach the goal of the project without building a computer system. It would be meaningless to build a computer system if nobody was going to use it. The struggle between the two was that when too much energy is spent on building the computer system, this may destroy the possibilities for applying it, both due to too little time for use, and due to the development of negative feelings towards the system. On the other hand, it is not possible to use a computer system which is not yet finished. During the project, the principal aspect of this contradiction changed. When the
programming started, the secondary task was the principal aspect, but when the
time limits were exceeded, the primary task was the principal aspect.

**The contradiction between negative attitudes towards the computer
system and the need for the goodwill of all the nurses on the ward in
order to fulfill the objectives of the project**

The computer scientists were working with a small group of nurses. The nurses
were needed in the project to suggest useful computer applications, and to in-
troduce the computer system to the other nurses on the ward. The computer
scientists as well as the nurses in the Florence working group had an interest in
developing and using a computer system for nurses’ daily work. This goal could
only be reached if all the nurses on the ward were willing to use the system.
This is the identity between the opposites. The struggle is related to the con-
flict between the nurses. As the project became the symbol of a conflict, it was
unreasonable to expect all the nurses to be positive to the system. This con-
diction was suddenly conceived at the meeting described in Section 1, and the
negative attitude to the computer system was regarded as the principal aspect.
The Florence working group and the computer scientists tried to minimize the
contradiction by letting the computer scientists, instead of the nurses, take the
responsibility for explaining the situation to all the nurses, and in this way stress
that the time delay was not due to the actions of the nurses on the ward. This
was also an attempt to change the principal aspect of the contradiction, by cre-
ating more goodwill among the nurses. Anyway, the contradiction disappeared
when the leading ‘negative’ nurses found themselves other jobs.

**The contradiction between the goal of the project and the available
resources**

There has to be correspondence between the goal and the resources of a project.
This means that there is an identity between the two: without a goal, there is no
need to spend resources on a project, and without resources, there is no need to
set up a project. The struggle between the opposites is connected to a mismatch
between the two. The goal cannot be reached if the resources are insufficient,
or, if the resources are insufficient, there may be a need to adjust the goal. As
this contradiction did not represent a conflict, it is hard to evaluate the principal
aspect and how, or if, it changed during the process. Moreover, the analysis of
this contradiction can be used for making a positive statement, but a further
analysis will not bring more insight into the problematic situation.

**5.3 How Do We Find Contradictions?**

Although Mao describes in detail what a contradiction is, he doesn’t say much
about how to find them.
According to my experience, we can find, and name, many contradictions in all situations. Some of them will be intertwined, like the contradiction between software made for traditional applications and the use of this software for non-traditional solutions and the contradiction between restrictions and freedom of action. What are regarded as contradictions will be related to what is conceived as important and relevant in the situations.

There are two ways of finding contradictions: the first is to examine what appears as conflicts. The exploration of a conflict will often result in many contradictions. Problem networks, presented in Munk-Madsen (1986), is one means of exploring conflicts in a systematic way, thus this technique supports dialectical analysis. The second way is to compare and discuss the situation according to recorded examples of contradictions, as they appear in empirical studies or in reports about general contradictions in system development, as presented in e.g. Andersen et al. (1986), Ehn (1988), Mathiassen (1987), and Stage (1989). However, the use of SSM as described by Checkland & Scholes (1990) can contribute to finding context dependent contradictions, by extending the methodology to looking for dialectical relations between root definitions.

In the situation described in this paper, the contradictions were found by examining conflicts. According to my experience, it is easier to discover the struggle than to discover the identity between the opposites of a contradiction. The identity is often taken for granted, thus the struggle between the aspects of a contradiction is easily stressed. Moreover, the knowledge of contradictions that are not causing conflicts may be interpreted as positive statements about a situation.

The principal contradiction of a situation is not obvious. As illustrated by the example, reflection was needed to find the principal contradiction.

My experience is that dialectical reflection, in this case Mao’s notion of contradiction, can be a means for understanding situations. A discussion of the relation between contradictions, the principal contradiction and the principal aspect of a contradiction is a way of framing a situation. In Schön’s (1983) terminology, looking for contradictions is a kind of conversation with the situation. The conversation allows us on the one hand to take care of the specific qualities of the situation, on the other hand to apply general knowledge or previous experience. In this way, contradictions allow a broader understanding of a situation that is regarded as problematic. The principal contradiction will thus constitute a frame for suggesting actions to change a situation.

### 6 Concluding Remarks

An organization in which a computer system is introduced or changed is in a process of organizational change, and usually both the computer system and its use setting is so complex that the development of a computer-based information system cannot be totally prescribed and preplanned.
Although unexpected and conflicting situations are often regarded as a result of lack of control, and of bad planning, unstructured situations are not always a sign of bad management. They can also be a result of the uncertainties and the complexity of a systems development process, leading to plans not being followed.

Dialectical reflection is a general approach for identifying causes of instability and change when unexpected and conflicting situations emerge, and this approach can be applied in systems development too, as illustrated in this article. An analysis of a situation in terms of contradictions is a means of framing a situation, even if it is not trivial to find the principal contradiction when examining a situation.

The insight gained from doing dialectical analysis should be used as a basis for applying other techniques in systems development, i.e. to make more realistic plans, according to advice from quality control, or to make expectations and goals transparent, according to the goals of soft systems methodology.

Acknowledgments

I would like to thank Karlheinz Kautz, Lars Mathiassen, Trevor Wood-Harper, Leikny Øgrim, and the anonymous referees for constructive comments and encouragement during the work with this article.

Notes

1. The Florence project lasted from 1984 to 1987, and it was initiated by the Department of Informatics, University of Oslo. The project was financially supported by the Royal Norwegian Council for Scientific and Industrial Research, Akershus Municipality, and the Social Department, the Municipal Hospital of Akershus (SiA) and Norsk Data.

2. The discussion is reconstructed by means of personal records and the memory of the two computer scientists.

3. In the Norwegian translation of Hoff (1982) the notion used is “den rå blokk” or “den utilhugne blokk”, referring to possibilities that exist in nature, or to natural abilities.

References


76


