Knowing in Action

22 video-analytical views "in situ" in the Development-department of an IT firm

by

Sisse Siggaard Jensen
associate professor
LOK Research Center
Institute of Management, Politics and Philosophy
Copenhagen Business School
ssj.lpf@cbs.dk

Abstract
This article is a descriptive and explorative analysis of reflective practices and knowing in action. It has been written on the basis of 22 video-analytical views in the development department of an IT company in the field of e-learning and multimedia. Video views are a qualitative method in video analysis and the analytical practice has been described concurrently with the descriptive analyses. Hopefully, this has been done in a way that encourages others to try the method in practice.
The analytical observation is directed towards dynamic knowledge processes when knowledge is applied in practice. They are 'the ones' which are of analytical interest, and it is difficult to observe them without allowing the live picture to take up the centre of the analytical practice. Therefore, various dynamic knowledge processes are described in 4 stories on the basis of the 22 video views. The subjects of the stories are research, when the code of the IT competitors is to be broken, *"When the Code Is Broken It Has to Be Checked"*; processes where a code is created through shared thought processes *"Thinking the Code Aloud Together"*; processes where a hidden or 'silent' code is made apparent, explicit and accessible through shared reasoning, *"Debriefing"*; and finally the story *"When Deadlines Have to Be Met"* gives an insight into the D-department's professional practice when taking stock and 'running a code' together.
The descriptive and explorative analysis of dynamic knowledge processes demonstrates that the developers work in a complex system where knowledge exists, is created, applied and shared together with learning through sparring, neighbour training, critical reviews, "intelligent methods" and debriefing. The analytical concept 'Knowledge Zones' is the starting point for the description, and the contours of the D-department's different zones are depicted in a drawing and on the basis of the descriptive analysis. It is concluded that the complex system of knowledge and learning unfolds on the basis of knowledge zones which constantly change because they are formed 'merely' in the mutual observation of the developers. However, the system or universe of knowledge zones in the U-department is entirely closed; it doesn't extend 'outwards' and does not include 'the others' in the company, such as staff in the media and sales departments, not to mention the company's customers. The eye is turned inwards towards the centre of the department's innermost knowledge zones and outwards towards top players of the same kind but at an international level.

Introduction

This article has been designed as a descriptive and explorative analysis of a series of *video views*, available as digital video data, and the purpose is to increase both the awareness of the video-analytical method and the knowledge of reflective practices. Video views is a term that describes the video-analytical practice which I have developed in practice since 1993[1]. I use this term to create associations to the qualitative interview rather than to video observation, which is the method most commonly associated with video-analysis. In brief, the difference, as I see it, is that video observation has objective registration as its ideal, an ideal which, like all ideals, is unreachable, whereas video views is a qualitative method which acknowledges the close relationship between observation and the object observed; between the recording and the facts recorded; a relationship which is so 'close' that the joint acts, concrete as well as symbolic, constantly interact as is the case with conversation and the form it takes in in-depth research interviews.
In other words, the article has been written to describe the analytical practice which I use when making and analysing video views, but also to speak up in favour of increased use of the practice and method. It is important for analytical practice that video data today can be made available in digital form. Not only for the actual situation in which the video view is recorded but also for subsequent analyses. The analytical software is now available which is able to satisfy the demands of qualitative methods to open and ongoing category development and coding[2] also in respect of live images. The mere difference of being able to be present at the scene with a digital camera *without* having to cover it all with spots, cables and microphones is an analytical gain which can hardly be overestimated. In this way technology plays a role in determining what analytical practice is possible. I am not saying that video can be set up and used as discretely and flexibly as a tape recorder, yet. However, in many instances it is possible with a hand-held recorder to get extremely close to practice in a certain area with the actors only remotely aware of the recording process.
In other words, it is my intention to describe my video-analytical practice in terms sufficiently concrete to allow it to be tested and used by others in practice while the descriptive analysis at the same time forges an initial path "through" my video

[1] Video views as a method are used in analyses of guidelines for communication via computer-media and reflective practices in network education, and the method is described in my PhD thesis: De digitale delegater: tekst og tanke i netuddannelse – en afhandling om hyperlinks i refleksiv praksis, der er face-to-interface. Multivers 2001(The Digital Delegates: text and thought in network education - a thesis about hyperlinks in reflective practices which are face-to-interface. Multivers 2001). The thesis is available in digital form on the Internet from which sections may be printed, see url: www.afhandling.dk, which can be found on the publisher Multivers' web site, see url: www.multivers.dk.
[2] The article has been written with users in mind who feel like tackling video-analysis without having at the same time to learn how to use video-analytical software. Readers who are interested in analytical software are referred to www.atlasti.de and the mail list www.atlasti.de/joinlist.html. The use of atlas.ti will be part of the articles below regarding video-analytical practice.

material. Seen in this light, the descriptive analysis in this article is an expression of an analytical process in which the purpose of the analysis, its theoretical references and the empirical material work together to form my interpretation of knowing and reflective practices[3].

The actual video views thus constitute empirical data material procured for analytical purposes where I have as aim *to describe and understand reflective practices and knowing in action in a field of practice.* The actual field of practice is a development department of an IT company within e-learning and multimedia. In other words, the analytical eye is focused on questions about 'what and how', when dynamic knowledge processes develop as an interplay between reflection and action, between different forms of knowledge and between knowledge and learning. Dynamic processes and interplay which take place in time and in live images. Time is the actant of dynamics, so to speak, and the analytical interest is therefore directed towards projection and comprehension of the relationship between knowledge when applied and time as it unfolds in practice.

I. Three Inscriptions and Video-Analysis in Practice

The following is an account of reflective practices, of knowing in action and of time as 'they' present themselves in a series of video views which were commenced one Thursday morning in February 2001[4], when the first test recordings and practical arrangements were made in the D-department or development department, which is the brain in an IT company within e-learning and multimedia. It is not always easy to take 'home' acceptable video views. It is therefore of decisive importance to first inspect the location of the planned recordings and to determine how many work stations are to participate in the recordings, how they are placed vis-à-vis each other, what the possibilities are to find an angle that covers the entire department and to get a close view of work methods and work stations of the individual developers - all with a view to light and sound, two factors which have turned out to be extremely critical in all my video views. They are right at the top of my internal checklist when I determine the angle, radius and movements of the recording. They colour my first view of the situation and they impact on the rough outline of the premises, which I initially make.

Space and Angle

The rough outline is one of the inscriptions which become my 'assistants' when I make video views. I call the inscription "Space and Angle" and I draw it as soon as I have a fair idea of the premises where we will be recording. In the actual video views the group of developers work in a large common room, a corner

[3] This paper is a descriptive analysis which is part I in a trilogy. It is a case study which is followed by a theoretical analysis of dynamic knowing in action based on the case study (II) and a methodological discussion of the video-analytical practice applied to the case study (III).

[4] The video recordings took place in the period between Thursday 15 and Monday 19 February 2001.

room, lit by 8 large windows. Together, the walls with windows constitute 'one angle' of the room. There are 10 workstations in total for the developers; 8 are placed 2 and 2 together to form 4 workstations along the two walls which each has 4 windows. Two workstations are a bit darker because they are away from the windows. Together, the 4 windows and 4 double desks form the basic structure of the room which is subdivided by 2 low bookcases. In the middle of the room is a large meeting table. It is used for group meetings where the individual employee merely needs to turn his chair around and roll it to the table in the middle of the room.

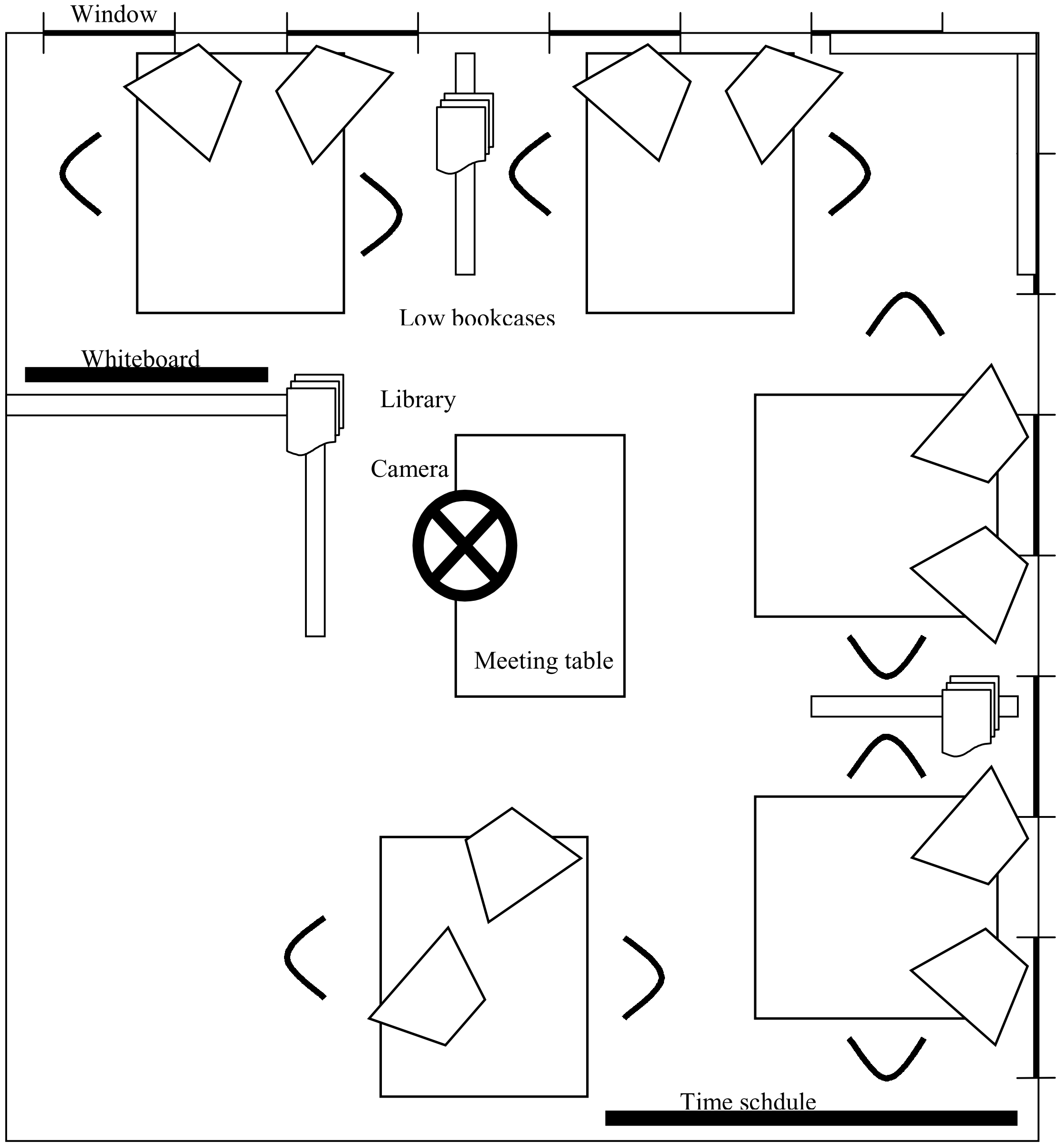


Fig. 1: A rough outline of the D-department in the IT firm; an outline based on the "Space and Angle" inscription.

Angles, radius and movements are almost pre-determined with this layout. The video camera can remain stationary in the middle of the room at the meeting desk and from here general views of all workstations can be obtained and at the same time it is possible to zoom in on the individual developers. It is easy to move the microphone around so the sound direction can be weighted differently. The room also allows for hand-held recordings. The two low bookcases, which sub-divide the workstations at the walls with windows, can serve as 'support' when close-up work with the hand-held camera[5] is required. An initial glimpse of the location shows excellent conditions for making video views.

Before beginning the recording of the actual video views, I take the opportunity to describe the premises as a whole and in live images[6]. The company is located in Østerbro and its close neighbours consist of other knowledge-intensive companies such as market research institutes, firms of consultants and NGOs. It is located at two floors with entrance from 2nd floor and an internal stairway leading up to the attic on the 3rd floor, which has large rooms with sloping walls. On the 2nd floor you enter at reception at the end of a long corridor with two well equipped meeting rooms on the left with glass walls and Venetian blinds and on your right four 'one-man' offices. The sales department is half way down the corridor, located in an open-plan office around the internal staircase leading up to the production department. Further down the corridor is a small kitchen, toilets, and copy room, before the long corridor ends in the development department, which is the stage of our upcoming video views. When mounting the stairs from the sales department, you find 'production' spread out over the entire 3rd floor - with a few exceptions. The manuscript writers are based in a room with a door that can be closed and the company's IT support is based at the end of the 3rd floor. This is where the company's heart of servers, PCs, cables, cords etc is found and several of the PCs located up here are used for running tests. It can therefore be said that the test department is an integrated part of the IT support department. You get a picture of the company's face to the outside world on the 2nd floor with reception, meeting rooms, individual offices and a sales department, and more out-of-the-way at the end of the corridor the company's brain, its development department, to which I will revert. Production is 'upstairs' and at the end the company's heart, the IT support, which works to the tune of the hum from the servers.

Actants and Parts

Back in the development department, I dig out a new inscription. It becomes a new helper. I now prepare a list of the actants who are going to be part of my video views. I name the helper or the inscription 'Actants and Parts' and both the human

[5] It is especially the microphone and the sound that limit how close you can get with the stationary camera. In the hand-held recordings we use the camera's built-in microphone.
[6] The recordings are available on Learn_1/ Tape_1

and the non-human actants[7] figure side by side on this cast list - as long as they play an important part in the stories told by my video views.
The first actants noted on the cast list are the ten workstations in the development department including the employees as well as their numerous artifacts. There are no obvious differences between the items on the list. The ten workstations are more or less the same in terms of hardware, equipment, facilities, capacity as well as furniture. A mere glance at chairs, desks or computers does not reveal who has the principal parts. However, after a while it becomes clear that Simon, the project manager, sits at one end of the room and that Poul, the informal head of the development department, is at the other end. The timetable hangs on the wall above the project manager and is the common inscription or the medium which concentrates the entire project in one synoptic glance as a whole or an oligopticon[8] as Bruno Latour might have called it. The informal manager sits at a whiteboard. In other words, the two principal players are placed at either end of the room with a full view of the entire room and in the vicinity of artifacts which can support and communicate the joint work processes involved in the project.
It turns out, however, that there are more layers than just managers and employees on the cast list of the development department. There are 'the newcomers', 'the green ones', 'the experienced', 'those who belong' and 'those who just sit here'. The newcomers sit at the two dark workstations. One of them belongs, although he is new, and is a very experienced programmer, a Nestor of the company, whereas the other belongs in another firm with which the company has just merged. He is based under their roof but he doesn't belong. He just sits in the department, however, there is an underlying hope that he may be assimilated and end up belonging. There is a need for at least two more employees in the development department, but they are impossible to find. The company has used all its contacts, and professional agencies have been asked to find them, but in vain. Therefore, everybody hopes that it is merely temporary that one of the newcomers doesn't belong. The other six developers are distributed evenly on the two groups of 'green' and 'experienced' developers. The green developers are studying for medium-length degrees such as computer science whereas the experienced developers have completed long-term degrees such as cand.merc.dat. (MSc in business administration and computer science), MSc, and engineering and computer science with physics as minor subject and in addition have previous project experience.
Little by little the cast list takes shape from the initial research work and imposes itself on the room and my vision and thereby on the foci of the camera; a cast list which varies according to the relations manager/non-manager, inside/outside, ex-

[7] An important conceptual couple in Bruno Latour's mind. For a more detailed explanation of the concepts, see Latour; B. 1991; Jensen, Sisse Siggaard: De digitale delegater, Del III, kap. 2 2001 (The Digital Delegates, Part III, Chapter 2 2001).
[8] Oligopticon and panopticon are opposite concepts which Bruno Latour uses to describe various forms of optics which show a whole. In a panopticon, you see everything and therefore only very little, whereas an oligopticon only shows very little, therefore allowing you to see everything.

perienced/inexperienced, old/newcomer, medium-term education/long-term education. Maybe the difference smoker/non-smoker should be included, too. Smoking takes place on the staircase, at the emergency exit. There is a lot of coming and going, and many tips, experiences and references are exchanged while smoking on the staircase.
The cast list is not easily determined by looking at the non-human actants in the room, as is otherwise quite often the case. It is determined by the glances with which the human actors observe and look at each other. The room, on the other hand, differs from the floor 'upstairs' in production. In the D-department there is light, air, room, comfortable chairs and desks and excellent facilities for group meetings. These facilities are in short supply 'upstairs in production' and in the sales department as well. There is no doubt that the employees in the development department consider themselves the principal players in the company as such.
The meeting table also plays a leading part and is therefore recorded at the same level as the ten workstations. Less prominent but nevertheless important parts are played by the library, which consists of books and the CD-ROMs on the bookcase in the middle of the room next to the meeting table. Here the developers gather literature of common interest together with the ' bibles' of programming. This is e.g. where the shared project documentation is found, an actant which also has a part to play.
The project timetable, which communicates tasks in relation to time, and in certain cases also in relation to employees, was mentioned earlier together with the whiteboard that can support work processes, but they both need to be mentioned again here as actants playing important parts.
However, the principal parts in the story are played by two actants, which are invisible, and which at first glance appear of an insubstantial nature. I am thinking of time and of reflective practices. They differ widely. Time plays an important role for everybody. It impacts on everything that goes on in the project and in the company. Deadlines are everywhere and everybody refers to them and to time. The reflective practices, however, are only mentioned by few, if any, but nevertheless impact on everybody in the D-department's knowledge-based practice.

Registration Card and an Overview

It is difficult to get an overview of many hours of video views, but an inscription once again comes in handy and assists; I call it 'Registration card'. This is where I summarise sequences, i.e. recordings, which are framed by a technical start and end of the actual recording[9], and I use them among other things to differentiate between various analutical 'views', which together make up the video-analytical views. An analytical 'view' is limited by a beginning and an end, however not of a

[9] Table 1 is an example of how I summarise video data on the basis of primarily technical registration data.

technical but of an analytical nature and which limit situations and types of situations[10].

Now we are getting to the difficult part. During the many processes, which transform and document my video views, I forge little by little one of many possible paths through the living video data; I often experience it almost like forging a way through dense scrub. When forging my way in this manner and creating the analytical paths in the material, my glance is constantly directed towards several things at the same time. The analytical purpose is my focus but I constantly have to remain open and ready to perceive the surprising, the unexpected, the controversial, the immediate impression which I get of the material. This requires training; it is hard and doubt constantly besets my mind. The question continuously arises whether the path is leading me astray or on long detours. The latter is probably unavoidable and it is necessary to have very clear analytical goals, clearly defined foci[11] and very simple and well-structured material. The complexity of video data rarely fails to surprise me, even when the situations appear quite simple, and so does the degree to which you influence the situation yourself during the recording. Already the choice of angle for the shooting, the presence of the camera and the many choices made during the recording process mean that I 'enter the picture' in the capacity of analytical observer and as camera holder and video recorder of the situation. In the subsequent transformation and documentation process, I moreover make a large number of choices, which in the most profound manner sort and weight my video data. Some data is included, some left out. Among the data included some are prominent, and some remain in the background.

Before I begin describing my analytical practices in detail, it is worthwhile briefly commenting on why I use the term 'views' rather than 'segmentations'[12] to summarise the analytical steps which constitute the analytical foreground of the material. The term 'views' emphasize that it is *'what is seen with the analytical eye'*

[10] Table 2 summarises the 22 views included in my video data and which thereby become an expression of the different insights into situations and types of situations which together constitute my analytical overview of the material.

[11] For a more detailed description of the heuristic and analytical 'foci' of the VIA method, see Jordan, B. & Henderson A., 1994 and Jensen, Sisse Siggaard: De digitale delegater, Del IV kap. 3 (The Digital Delegates, Part IV Chapter 3).

[12] The VIA method (Video Interaction Analysis) has inspired my analytical work. The heuristic of the method forms the basis of the preliminary analytical work. I do not use the term 'segmentation', however, as it implies something granted by the material itself and which is used to form a projection, whereas the term 'view' which I use instead of 'segmentations', emphasize that it is in the encounter between the material and the observer's observation of the material that the analytical insight is created. Another term, which I use with extreme caution, is 'structure', and the reasoning is the same as the one that applies to the term 'segmentations'. In the heuristic of the VIA method the underlying structure of the material is used for projection. In other words, the structure already exists as a given part of the material rather than arises out of the analytical encounter between the material and the observer. In order to emphasize the latter analytical point of view I use the term 'situations' or 'types of situations' instead of the term 'structure'.

which gives the material its shape, and it is *situations or types of situation'*[13] which are seen when viewed with the analytical eye. 'Views' are not properties inherent in the material itself; they are not structures which appear as a kind of 'deposits', creating chronologically structured layers in the recordings. Even if some of the foci which form part of the analytical eye are very concrete and noticeable even to the immediate eye, as is the case when a situation such as e.g. 'sparring in the form of joint research' begins with a manager moving his chair up to an employee and finishes when he moves it back again; or when the situation 'a break' begins when two colleagues go out and smoke on the landing and finishes when they place the packet of cigarettes on the desk and both sit down at their computers at opposite ends of the room. In the analytical process I always maintain that the analytical foci can only 'colour' my eye as they always have to appear on the analytical stage together with the purpose of the analysis. In this way, together, they shape what I call the analytical eye. It is the purpose of the analysis which in a decisive manner determines the analytical situations by allowing the differentiation of the observation on the basis of assessments of relevance, and it is the foci which give the eye the *points of view* which allow the observation to be differentiated. I hope this will all be much more clear once the analytical practice is allowed back in as part of the picture. Let me therefore do that.
As I have already mentioned, the registration cards help to record the first *insights* in the data material, allowing them to be summarised and hopefully allowing me to gradually create an overview over the video data. The time of recording has been chosen to cover the D-department's Monday meetings. These meetings constitute the regular status meetings[14] of the development group. In addition to status the meetings are used for 'debriefing' and to 'run codes'. The research phase therefore begins on Thursday in order to include the 'prelude' to the meeting in the analytical material, which means that recordings also take place the whole of Friday.
Generally speaking, the video data produced falls into three categories: test, background and analysis. Test and background recordings are made during the research phase on Thursday. They are mentioned above and don't really play an active part in the analytical process as such. The analytical material is subdivided into five sessions, which are recorded together with a number and a brief characterisation as well as date and tape number. The first summary already contains the first interpretations of video data embodied in the brief characterisation associated with each of these sessions and their very division. The summary of sessions and the characterisation is shown in Table 1.

[13] A situation may be described as follows: "A situation is a set of communicative and non-communicative acts that take place at a specific time and place. A situation is a social category, a "cultural unit" (cf. Eco 1977:66). This means that it must be conceived as a unit by some community of speakers, have a special name, be marked at the boundaries etc." (Bøgh Andersen, Peter 1990:54)
[14] This is not a practice which has been going on for a long time, but attempts have been made to implement it in the D-department during the past 5-6 weeks.

Table 1: Summary of Video Data: Test, Background and Analytical Material

Summary of Video Data	Date, Year 2001	Tape
Test (D-department)	Thursday 15-2	1
Background (The Company)	Thursday 15-2	1
Analysis (D-department):		
Session 1: Individual practice	Friday 16-2	1/2
Session 2: Sparring	Friday 16-2	2
Session 3: Individual practice and sparring.	Friday 16-2	3
Session 4: Meeting and debriefing	Monday 19-2	4/5
Session 5: Co-operation and sparring	Monday 19-2	5/6

At this stage, the preliminary 'exercises' in preparation for the video analysis have nearly been completed. The preliminary agreements for the recordings have been made, the time has been fixed according to the purpose of the recordings, which is the Monday meeting in the development group; the preliminary research with test and background recordings has been done, video views have been carried out and the preliminary work has created an overview by means of the inscriptions: 1) 'Room and Angle', 2) 'Actants and Parts', and 3) 'Registration Cards and Overview'. However, it is necessary to further work on the material as a whole in order to get it 'under your skin'. A few more steps need to be taken before it is time to put on the glasses that allow you to see with the analytical eye.

The chronology of the analytical steps, as described so far in the article, is a bit messy. In real life, i.e. in my analytical practice, the brief descriptions 'individual practice', 'meeting and debriefing', etc. which have been attached to the sessions in Table 1, are a product of continuous interplay between 1) the overview of the material which is a product of the technical registration data such as number of tapes, start and finish of the recordings, etc. and 2) the process which is described below.

It is a process which I call 'to skim' my tapes. The skimming consists of repeated examination of the material without sound and in 'ff-mode' (fast forward). At all the important points of change such as change of scene, change of movement patterns, change in orientation, change of actants whether human or non-human, I stop the recording in order to assess the situation and ask the question: Is this a new situation and therefore a new beginning or is it just a change in the recording rather than in the situation recorded? Little by little I record the picture of my video data, which emerges this way, in a new version on the registration cards. They therefore exist in two versions: one completed more a less concurrently with the recordings and one which is a result of the subsequent 'skimming' of the material. During the last examinations in 'ff-mode', I increasingly let the tape play in 'play mode' so that sound and speech gradually can enter the picture and contribute to the evaluation of the various situations so that their individual importance becomes more apparent. The situations, which thus determine the material, are evaluated simultaneously in the light of the analytical purposes, and in this way the first steps are taken on the path which gradually stands out as the direction and contours of the analysis. Often, a number of factors stare you in the face when you go through the material. These factors may be good indicators worth holding on to, but not necessarily. It is very easy to get a fixed 'view' of one's material. A view which it may be almost impossible to 'erase' in order to allow other 'views' to come forward. The slightly pedantic methods described above therefore don't represent an expression of an analytical point of view in which 'the individual parts' are at first identified and later joined together to form a whole. They are more an expression of a point of view which underlines the importance of *delays,* which serve the purpose of preventing perception and apprehension from quickly 'forming a picture' of video data; delays, which serve to ensure, to the extent possible, that more 'views' can be formed before the eye 'stiffens' in a view which is far too fixed.

A picture now emerges out of the many preliminary tape recordings of *22 analytical views* of widely differing practices in the D-department. A picture where the rhythm of the camera changes between panning and zoom-in, and where the camera is alternately hand-held and placed in a stationary position. The picture is summarised below in Table 2 together with the summary of video data and analytical views.

Table: Summary of Video Data: 22 Different Analytical Views:

Friday Morning:

Session 1/ Individual Practice / Analysis

1. Stationary view of 'studies' in corner 1 and 'coding, keyboard and thought' in corner 2. Tape 1.
2. Hand-held zoom-in and interview with Niklas (D2) about 'work methods'. Tape 1/ Tape 2

Session 2/ Sparring/ Analysis

3. Hand-held zoom-in on 'sparring in the form of joint research and evaluation' between Simon (D3) and John (D9). Tape 2
4. Hand-held panning in the entire D-department. Tape 2.
5. Stationary view of 'coding, keyboard and thought' in corner 2 and especially Theis (D4). Tape 2
6. Stationary panning on 'studies' in corner 1 and 'coding, keyboard and thought' in corner 2. Tape 2
7. Hand-held zoom-in on 'sparring in the form of joint research and evaluation' between Simon (D3) and John (D9) in corner 4 at the whiteboard. Tape 2

Lunch

8. Stationary panning on 'lunch with the screen'. Tape 3

Afternoon:

Session 3/ Individual Practice and Sparring / Analysis

9. Stationary view of 'sparring in the form of 'neighbour training' in corner 2 and especially of Theis (D4), Ib. (D8) and Simon (3). Tape 3
10. Brief stationary zoom-in on 'individual practice in the form of navigation' Søren (D7) and 'individual practice in the form of coding' Thomas (D6). Tape 3
11. Stationary view of 'sparring and neighbour training' in corner 2 with 'the three' from point 9 in focus. Tape 3
12. Brief stationary view of 'individual practice' in corner 3. Tape 3

13. Stationary view of 'individual practice in co-operation' in corner 2 - again 'the three' from point 9 and with brief visits to corner 3 and 1. Tape 3/ Tape 4

Summary:
A view of the situation as a whole in the D-department; a view of individual practice in the form of studies and coding where the voice of the keyboard blends in with the thought processes of problem solution; close up in two hand-held zoom-in situations: 1) an interview with Niklas about his work practices and methods and 2) co-operation and sparring in the form of joint research and evaluation between Simon and John; a stationary view of individual practice which takes place in close co-operation between 'the three': Simon, Ib and Theis.

Monday Morning
Session 4/ Monday Meeting and Debriefing / Analysis
The server is down. There is hectic activity to solve the problem. Tape 4
14. Stationary view of 'Monday meeting with status and debriefing' and agreements to 'run code in critical review' - interrupted by the staff meeting. Tape 4.
15. Hand-held zoom-in on 'status and debriefing' between Theis (D4), Poul (D10) and Simon (D3). Tape 4 and Tape 5.

Lunch

Afternoon
Session 5/ Co-operation and Sparring/ Analysis
16. Stationary view of 'critical reviews' in corner 4 with Niklas (D2), John (D9) and Thomas (D6). John is running the code. Tape 5/ Tape 6.
17. Stationary view of 'sparring in the form of joint problem solving' between Søren (D7) and Hans (D5). Tape 5/ Tape 6.
18. Stationary view of 'sparring in the form of neighbour training' in corner 3 with Ib (D8) and Simon (D3). Tape 5/ Tape 6.
19. Stationary view of 'briefing' between Hans (D5), Søren (D7) and Theis (D4) across low bookcase. Tape 6.
20. Stationary zoom-in on the project timetable which shows 'the relationship between time, tasks, people and thereby competences'. Tape 6.
21. Stationary view of 'sparring in the form of joint research and evaluation' in corner 1 between Niklas (D2) and Thomas (D6). Tape 6.
22. Stationary zoom-in on 'fire-fighting' in corner 2 and 3 with Ib (D8), Søren (D7) and Simon (U3). Tape 6.

Summary:
A view of the 'meeting culture' and the 'male culture' in the D-department; close up in a hand-held zoom-in on 'status and debriefing' between Theis, Poul, the head of the D-department, and Simon, in which Theis explains his coding which is vital; a view of several different forms of sparring 1) critical reviews between Niklas, John and Thomas, 2) joint problem solving between Søren and Hans, 3) neighbour training between Simon and Ib, 4) joint research and evaluation between Niklas and Thomas 5) fire-fighting between Simon and Ib; a view of briefing between Hans and Theis across the low bookcase. In addition a view of 'time' hanging on the wall -very beautifully presented in a timetable.

The descriptive analysis of reflective practices in the D-department, as it appears in my video views, takes on many different forms which enter on the analytical stage as dominant forms of practice in a professional developer community with knowledge-intensive work tasks. Thorough studies are undertaken amongst piles of thick books. Studies which take place in the work place and in the middle of the practice of the profession. Competitors' codes are broken and checked in order to find short cuts with all that this entails in terms of “intelligent methods”[15]. Codes are carefully scrutinized and are ‘run’ and checked with colleagues. Inexperienced colleagues are supervised in and through practice. They sit next to an experienced colleague who can keep an eye on each screen and assist if there are problems. The developers think aloud while media simulate computer screen and user interface. Inexperienced colleagues run codes next to experienced colleagues. Codes are run together with experienced colleagues who act as critical reviewers with the task of finding the gaps and problems with the code. Knowledge is constantly shared both between experienced colleagues and across levels so that inexperienced developers can learn from their experienced colleagues. The screen and the computer are actants which participate all the time and everywhere as vehicles for the continuous processes where knowledge is in action, when it is created, used, revised and shared.

The 22 video-analytical views therefore have to be studied once again, and the eye must search for high quality views, technically as well as analytically. That means views which are clear and unambiguous, where you can hear what is being said and see what is being shown, pointed at, etc, and where the content at the same time allows you to spot some of the different types of reflective practices, which take place in the department. The individual views are therefore examined once again in an analytical process in which ‘boards’ are made. It is a practice that consists of a process where the material is drawn, a practice I call *‘reverse storyboarding’*. In the concrete analysis, the individual views are not actually drawn but individual video frames, which are of primary importance to the story that is being told, are selected through the *‘grabbing of frames’*. In this process, *five stories* in particular enter on the analytical stage. These are the stories that together shape the odyssey in knowledge and thought processes which I describe in the following chapter “The great epos - an odyssey in knowing and thinking in action”.

The five stories begin with a view of a young and inexperienced colleague who has just started, who sits in the department but doesn’t yet belong. He is absorbed in studies and heavy books. In the next story *“When the code is broken it has to be checked”*, we see a very experienced colleague, who is also quite new in the department, and who works with methods which are very explicit and “intelligent” while being at the same time entirely intuitive. In the story *“Thinking the Code Aloud Together”*, we follow sparring where the project manager thinks the code through together with one of the developers in order to find the best solutions. In *“Debriefing”* we watch a session where the top developer of the department debriefs one of the experienced developers together with the project manager, and finally in the story *“When deadlines have to be met”*, we participate in the Mon-

[15] The concept of intelligent methods is here used with direct reference to Dewey. J. In the book ”How we think” (Boston 1933) he uses the concept to sum up the variation of methods in which thought moves from pre-reflective to post-reflective situations, from past to future and from disturbances, disorder and uncertainty to provisional clarity. See also Jensen, Sisse Siggaard: De digitale delegater, Del III, kap. 1 2001 (The Digital Delegates, Part III, Chapter 1 2001).

day meeting which has been in the analytical focus right from the beginning when the time and duration of the video views was arranged and planned with the department and the company.
In addition there are *three photo stories*. They are made 'anonymous'. It is not all knowledge, which becomes accessible, if it is to be shared. To be allowed into a D-department to have a look at the daily practices in the 'heart' of an IT company, is not that easy. It is a meeting between knowledge born with commercial ends in view and which therefore is subject to the requirement that it must serve the company and nobody else, and the knowledge which is born in the interest of research and therefore with a view to serving common goals and with a degree of openness and documentation which allows it to be subjected to critical reviews. The 'price' for the right to look over their shoulder at their practices has therefore been an absolute assurance that no video recordings will be made available to the public.
It is therefore only to the extent that the picture material in my recordings can be made anonymous, and this is obviously difficult, that the analytically vital pictures can become part of the actual story and not just function as the background of the analysis. The photo stories relevant in this context are *"The many lives of the code"* which ends the story "When the code is broken it has to be checked"; *"What if?"*, which ends the story "Thinking the Code Aloud Together"; and *"The code equilibrist"* which ends "Debriefing". They all consist of pictures in which the individuals cannot be recognised. Many, many other stories could be told with the countless pictures or video frames which together make up the 22 video-analytical views in the data material, but research ethics impose very clear limitations. Let therefore the story begin.

II. The Great Epos - an Odyssey in Knowing and Thinking in Action

Theis is a developer, and he is an equilibrist on his keyboard[16]. You can hear him talk when his fingers dance across it and see how he regularly talks to himself and his screen. He talks expressively, only without language. He is in agony when the problems cannot be solved immediately, when other helpers such as pencils and paper have to assist, when he has to allow his thoughts to flow and form in impatient waiting and without using the keyboard - all this is visible and audible on the Friday morning when I take a look at the D-department.
The room is buzzing. People are working and in a very concentrated manner. Theis' communication with his keyboard is the most audible activity. Kurt and Niklas, who sit right next to Theis, are also lost in their work, but their work consist more of studies and involve thick books. They are absorbed in books, paper and folders. If you didn't know better you would think they were studying. Kurt is completely new and he reads, reads and reads. He has to acquire a lot of new knowledge. With a technical background as shipbuilder he now 'sits' here in the

[16] All names in the account of the developers' great epos are fictitious.

development department where he learns a lot of new stuff in order to supplement his education and become a developer. At times he is also given minor tasks to solve. He is young, he is here at his own initiative, and the development department is his framework. Imagine being as good as competent developers like Theis! His voice is full of admiration when he talks about Theis and some envy when he says: "*…I know maybe 20% of this stuff, so it is hard to sit here and listen to Theis typing away on his keyboard*". Theis' juggling at the keyboard is obviously not music to his ears. It is more like a constant reminder of the goals he is so keen to reach and a reminder of how far he still has to go before reaching them. Kurt is completely new in the department, but he likes to show that he is not completely green and inexperienced. He advises an employee from 'upstairs', i.e. upstairs from production where they are working with graphic designs and interfaces. An employee who would like some advice and tips about how to get started with development and code writing. Together they look for material and books on the Internet, and Kurt explains and gives advice about how to get started. That he has experience and has written codes becomes obvious when he shows the other employee 'from upstairs' a system which he has coded. A display full of pride. In between, Kurt and Niklas exchange a few words, but otherwise the only sound accompanying the concentrated work comes from Theis' keyboard.

When the Code Is Broken it Has to Be Checked[17]
Niklas is also quite new in the company, but he is an older employee who describes himself as the company's Nestor. An experienced programmer who has been part of the game since the early days. He is not typing but juggling paper, pen and books. A maths teacher at high school busy correcting homework? You would almost think so. But no, he is a developer busy breaking a competitor's code. Nothing less. When the camera zooms in on his papers and his work, he stops me, and we exchange a few words about the need to keep my video data confidential. You cannot just film everything that is going on when someone is checking a competitor's code, but it is obvious that it is common practice. Many different kinds of knowledge come into play in the process of breaking a code. Some knowledge is 'at hand'; it is available and you can just reach out and page through it if it is embodied in the many very thick books lying on the desks, whereas other knowledge has to be created. The borderlines between knowledge as a product and knowledge as a process flow in a dynamic process. Is the competitor's code immediately at hand? Hardly. There is obvious proof that the code exists as knowledge because their software exists and it works. Seen with the competitor's eyes, the code is knowledge that exists, but it is not immediately available to the developer in the D-department. It has to be found and revealed, and this has to be done in ways which require experience in the field, experience of how to 'go behind the code and start working backwards', both when trying to find the examples of the

[17] Tape_1 time code [00.12.49 to 00:20:29] and Tape_2 time code [00:00:00 to 00:16:14].

code and when breaking it. In this process, the developer works backwards starting with the code which is given, which exists, because it has been ‘found’, and then he goes behind the code by imagining what it could be ‘based on’, what algorithms or mathematical operations the competitor has used; an imaginative process which is based partly on concrete, explicit and publicly available knowledge and partly, and to a great extent, on personal experience, which comes to light as intuitive knowledge or ‘nose’. To get an idea of what is behind the competitor’s code, you have to guess. What could the competitor possibly have been thinking? Guesswork, ideas, ‘suspicions’ and working backwards are factors which all play an active part in the process required to break the competitor’s code, when knowledge has to be found but is not at hand, and has to be made available so it can be used.

“Niklas:…[…]…and then I write a comment …, where I try to understand from their function names what it is they are doing here, and then read in the code, but is this really what they do, what are they actually doing, …up to a certain level, not in every detail, but to understand their principles; and then I had a suspicion that I thought they are doing the same as the guy does in this book, and in that case it would be nice to find out if it’s really true,…so that was the idea. They have a data structure which looks like this, it is called a matrix, but you don’t see trace of a matrix here, and there isn’t really any, it is just imagined, and then I guessed that I think this is the way, and then I check it here in the actual code and see how they do it and this was when it suddenly became evident…[…]…this was where…[…]…”[18]

The understanding of what the competitor has been thinking, how they have worked, the general mathematical principles involved and how they have been used, is formed in continuous interplay between the existing knowledge used and the way in which it is used. Ideas are constantly evaluated in the light of experience and nose in order to allow the many different forms of uncertainty such as guesswork and suspicion to *coax* the code and to create an opportunity for new knowledge to be created. This dynamic oscillation between certain and uncertain or potential knowledge, between knowledge as a product and knowledge as a process, between knowledge as something that exists and as something that is created, is a permanent feature when the code is being broken. However, it is clearly a requirement that the final result must lead to the finding that there is a solid and stable foundation of knowledge ‘underneath the code’.

“Niklas: …[…]…and then here they describe how they are going to put two of those together in order to make a transaction and a rotation; which makes it a transformation, … and then you can add one more to it, and one more, and by

[18] When the Code Is Broken_2 time code [00:01:11]

doing that you get an object which moves along a path of these transformations with corresponding rotations which ensure that the object remains parallel to the tangent to the code. It is those transformations they have gathered here to turn into one, and then I can guess from the way in which they do it that it is actually the same he is writing here in this book, that when they write a,b,c,d, then you have to imagine that they are placed in a matrix in exactly that way, and then imagine that it reads 0,0,1 to form a nice matrix of 3 x 3, so what I have done here is to check, as I know the rules for matrix multiplication; then I can see,.... is that what they have done here too, and I have checked that here.... Yes, indeed, that is it…[…]…"[19]

It is not merely intellectual pleasure but also aesthetical enjoyment which shines through the account of the work practice of breaking and checking a code. Indeed, checking plays a very important role. Preparation, analysis, the detailed design which is required before a code can be written on solid ground, oscillates between breaking and checking. It is a practice which unfolds in the shape of questions such as 'how does the competitor do his coding?', 'what do the large players do elsewhere in the world?', 'what do the textbooks say?', 'how does all this tie up?' and 'what are we going to do next?'.

Solid knowledge in the form of a textbook or one of the 'bibles' is also on the developer Niklas' desk. It is big, a thick book of about a thousand pages about the programming of computer games. It is fascinating reading and "some of the best", says Niklas. The author has almost 20 years' experience in the programming of computer games, which have to be fast and at the same time full of life and action, with space ships in movement and other fantastic but very demanding operations. Unqualified professional admiration radiates from Niklas when he talks about the book. It is obviously an esthetical enjoyment and a pleasure to read it; to be able in this way to gain an insight into how a true code artist can 'create' complex chains of events, movements, figures, environments in almost live action by means of his code, which in fact is extremely simple. A code that optimises the relationship between an event or component and the time and energy that is required to make it a reality. Optimal code. The joy of seeing a simple code that solves complex problems is almost like the experience a reader gets when life's complexity is distilled by a poet into the simple form of a poem where extremely little can express an endless amount. In fact, that is exactly what the preparatory work is about. To manage to express as much as possible in a code which is as simple as possible. The pleasure of preparing extensively and thoroughly lies in this. It is pure joy and pride for a professional to be able to say to the project manager and the other developers "there won't be much to the code". Listen:

[19] When the Code Is Broken_2 time code [00:02:49]

"Niklas: …[…]… The next step is then to check against the current standard for things like this, let me see, I think I saw something similar, and I wouldn't be surprised if it is exactly the same. The next question then is how exactly do they do it, and if that ties up, then the next step is to look at our own code and see how do we actually do it today, and does it tie up. It will be interesting and if it really ties up, then it's all sorted out, isn't it, then I can write in a document "Dear friends, I have found out"
Sisse: Do you then write that in a document?
Niklas: Yes, I write that in a document. I have read this and that document and checked what we talked about and what's happening is this and this and this …[…]… and it looks good. It's really exciting. So I hope that on Monday at our status meeting, I can say that now I am ready to begin coding and by the way there won't be much to it, because of this and that, that's how simple it is. …[…]…"[20]

To get to something that is simple and reliable therefore requires many different angles of approach, detours, secret paths, nose, experience and a 'feeling of certainty' about one's own knowledge together with the will to venture out into uncertainty by guessing, sensing, getting suspicions and in other ways coaxing one's way to the thoughts through a set of uncertain ideas, presumptions and guesswork. Handling a work practice as complex as this requires well developed systems and intelligent methods together with dynamic oscillation between situations where there is access to knowledge and information and situations where there is no such access; where there is no access to knowledge and new information but rather a possibility of 'digging' and delving deeply in what is already available. Niklas' system is based on such a division and in a form where he changes between working in the company and working at home. In this way he has developed a system of dynamic oscillation between having access to knowledge and information and being cut off from this access. At work the focus is on gathering as much 'input' as possible, to obtain the knowledge, the information, the codes, the overview which require further scrutiny, and at home, evening and night in clouds of cigarette smoke, which is not allowed at work, it is time to dig into the material which means to imagine what others have thought, to suss out and trace differences and similarities, to deduce abstract figures of thought 'behind' the concrete appearances, to guess which mathematical rules and laws apply and to follow one's 'nose' for 'what appears to have been done in this code'.

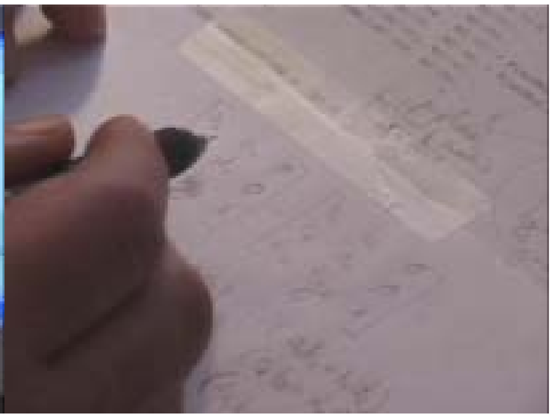
"…[…]…Sisse: And then you happily went to bed?
Niklas: Yes, by then it was 2 o'clock at night…
Sisse: I can believe that!

[20]When the Code Is Broken_2 time code [00: 06:37]

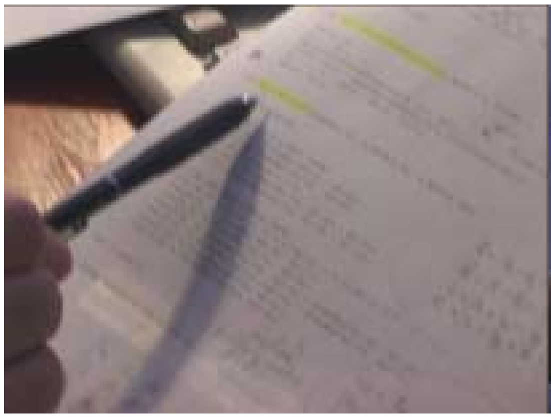
Niklas: Well, you see…[…]… I am so comfortable there at home and can smoke as many cigarettes as I like…
Sisse: Without having to get out the emergency exit…
Niklas: Also without having the possibility of looking for new information, a new overview, because for that I have to use the computer, which is here, so there I can switch off,… then I dig deep into the stuff I have on paper, yeah.
Sisse: So there you switch off?
Niklas: Yes,…So it showed us that the way the competitor does it and the way the book does it are the same, we have understood that, so then we can do something similar, if we want to, can't we.
Sisse: Do you often work like that at home?
Niklas: Yes! I do that a lot.
Sisse: Have you got a computer at home?
Niklas: Yes I do. At home it is not so essential, at home, unless I absolutely have to download something from the net which I need, if I don't have the book then maybe I can download something, or sit and scroll through and print some of the pages which are essential, mark them and add some comments. At work I have the tools at my disposal and have to make sure I get sufficient input, partly to get it into my head and get it to take shape, but then I can take it home with me.
Sisse: And then you take it home?
Niklas: Yes, in order to really dig into it. Yes, that's what I do. …[…]…"[21]

[**The photo story** *"The Many Lives of the Code"]*

Body language gives life to the code


…as does the constant search for underlying rules…

[21] When the Code Is Broken_2 time code [00:04:10]


Here it is! The rule for matrix multiplication.

It is all recorded in the note book!

Thinking the Code Aloud Together[22]

While Niklas and I sit and talk, Simon, the project manager, and John have repeatedly consulted each other in front of John's screen. They sit with their heads deep into the code. Now they tackle it together and try to think through a set of components in the editor seen 'from the outside' or from the interface perspective. They don't converse in the usual sense of the word but rather think aloud with each other and with a medium; not at the computer, but together and at a whiteboard. It represents the third party. In this way they are forced to 'think aloud together' without 'expressing themselves' by typing and coding. The new medium wipes 'the board clean' so it forces the thoughts to come out as conversation instead of as code. However, at the same time the board 'plays' the part of a computer, because Simon uses it to simulate an interface and in this way the use of the board creates a focus for everyone's thoughts as a basis for the conversation.

Simon begins by feeling his way. He tries to find his way into the thoughts behind the codes John has made and asks probing questions in a way that forces John to explain his own thoughts. He has to rethink the thoughts he had when he made the codes, and he has to make them explicit. It is quite difficult to remember one's own thoughts in this way. Often they were thought without being perceived as thoughts, but rather 'made' in the shape of a code. Now the thoughts, which were not thought but rather made, have to be translated into spoken language. A number of translations and transformations come into play in such an 'exercise' which appears quite simple. The question is also whether it is possible to think the same thoughts twice? The mere fact that they have been thought creates new conditions for the thoughts that are being rethought. In fact, John's rethinking of his codes gives rise to many more questions than answers. The exercise is nevertheless the prerequisite for thinking the logic of the code through together. As was to be ex-

[22] Tape_2 time code [00:33::03 to 01:03:03]

pected, the initial conversation quickly progresses to a joint evaluation of the possible consequences of different approaches.

Two particular ‘modes of thinking’ are involved. The two developers work with two main types. In practice they often appear merged, but in the following I will nevertheless try to keep them separate, as each momentarily dominates from time to time. The first appears as a thought process where each developer in turn tries to understand the other’s thoughts. One thought follows the other; they proceed side by side for a while until a given moment when a certain thought causes the developers to bring out the thought and its consequences in the open, so they both have to stop and honour the direction of the thoughts. It may be an idea they get, which appears to solve a problem, or maybe they discover that an imagined procedure doesn’t work after all. We start the odyssey in the thoughts of Simon and John, listening to a ‘what if’ thought process. The first odyssey begins with Simon trying to follow John’s examples of codes and the thoughts behind them. He then points out a problem which it turns out John doesn’t really know what to do with either, but towards the end of the quotation we see glimpses of conceptual breakthrough in both developers, especially in close association with the ‘what if’ projection.

“…[…]… (Simon leaves his diagrammatic example on the board and joins John to look at an example of a code on paper).
Simon: What are you giving it?…This “add menu”, what is… there you are giving a menu description, aren’t you, what does it look like?
John points at the paper.
Simon: This one?
…
John: We could, of course, create a level here somewhere, couldn’t we, find a parameter; I mean, when it comes to menus there will also be some popdown… at level, but on the other hand if we say that the level, the menus differ, then…
Simon: But then when you have to, if you should enter a ‘project’, ‘project’ is what it’s called here, what has it then filled in?… of these…?
John: Well, er… what is necessary for…the bar out there, it’s the icon and the ID number.
Simon: The ID number?
John: Yes…because the ID number is mailed, … so that, in the event which is attached, the type or a number here, you see don’t you, it is attached to that one, when the event is mailed to you from …, then you will get the ID number as well.
Simon: Mmm
John: The one it has inserted itself, then you can control the icons, so you only need one object to handle them all, that’s really what the intention is, … the icon itself…, icon, ID and then this popback or event …, there, you see?
Simon: Yes

Simon: How do you then decide on the inscription? Now we call this one pop…, so you write that up here or what?
John: Mmm, let me…I think I have stored it in… (checks his computer)… Mmm
Simon: Somewhere you have to indicate what it's going to be called.
John: Yes…that's true, I have also thought about it, er… I was thinking of parsing it, maybe, but, er, we could do that but I think it would be a bit misleading…, on the other hand if we are going to use the same…, then the path doesn't really have to be coded
(Simon goes back to the board and starts drawing again)
Simon: No, but what if this one out here, this registration, it's called 'project', it's this one. When it then registers one called 'project' forward slash something
John: Yes, yes! I follow you, yes, I also got that idea… at the same time… just now, then call it 'type-view'!
Simon: Yes! 'type-view' …[…]…"[23]

At first Simon tries to follows John's thought processes and understand his examples; then it turns out that they are both stuck with a problem which, however, is solved by the two simultaneously and in a breakthrough. It is almost like following a dramaturgical example where tension is built up and resolved. The next odyssey into Simon and John's thought processes reveals another procedure when the two work together. This time it takes the form of a dialogue in which they shift between 'translating' each other and thinking each other's thoughts until finally - in this particular example - they end up rejecting a train of thoughts because of the consequences their thinking has revealed.

"…[…]… Simon: In the case we are going to do now, 'project' is going to call Explorer plug in's 'add menu' four times, four times: first time that one, and here three times both…
John: Okay
Simon: What happens then …, when you press over there…
John: When I click on it
Simon: Then you get the message that …er, now the process is…
John: Then the main programme writes the corresponding sub menu
Simon: Yes
John: And the sub menu sits over there, of course, and then, and then what?
Simon: Then it has to tell 'project' that now it's active
John: That shouldn't be a problem; it has an event stored
Simon: Yes, that's fine

[23] Thinking Aloud Together_2 time code 00:36:30

John: Then it gets the message that you have clicked out left, because it knows that, doesn't it, because it has attached the ID itself, the ID which comes from out there, hasn't it?
Simon: Yes!
John: Oh, it's reversed (comment to own body language)
Simon: And then it has to choose
John: Then it has to, yes, when it starts up, then it can find out which type view it wants to attach
Simon: This one then somehow has to be able to send a message back about what kind of menu it is
John: Like this, so that it is activated
Simon: Yes…this one has to be pushed down
John: Yes, that's obvious
Simon: But, what's also interesting is that, … now we have pushed this one down, it has been inserted
John: Yes … there you can go in and make a 'remove' and then an 'add menu' with a new image, can't you
Simon: Mmm no, that's not so good
John: No, that's er…
Simon: No, that's not good at all
John: No …[…]…"[24]

In the other main form or thought process, Simon and John suddenly think aloud together. One starts a thought process which the other picks up and the first continues; then the conceptual breakthrough takes place where their thinking takes a new turn and new ideas are born. It is as if the thoughts suddenly stare both of them in the face and this happens in close association with the board and the way it is being used. Simon constantly draws the thoughts and thought processes on the board, draws the path taken by their thoughts and the contexts and consequences which this creates. The diagrammatic flow, which in this way is communicated, simulates a computer with interface and this creates a common direction for the two developers' thoughts; it creates a shared view of the situation. This has already been shown in the previous two odysseys, where conceptual breakthroughs occur in both cases, but in the third odyssey, which now follows, we get another look at how the two developers *by their thought processes generate ideas and conclusions in each other's mind.* At first they progress side by side, then the thought processes accelerate in a solution with 'wooden structures' which they take turn to generate only to dismantle again - once again by helping each other and taking turns.

[24] Thinking Aloud Together_2 time code 00:45:33

"…[…]…John: It's just that we still have to write 'project' here
Simon: We could write 'project' over here
John: Then that's the name of the icon you click on, isn't it? When you click on the icon, the name of the top layer will be written out there, on…it means that if you have registered it as 'project' then that's the one which will be written out there, on…hmm… the icons to the right…isn't it, subtitles, names?
Simon: Yes
John: And the others?
Simon: Well, what you do is that when you've got
John: Then you build a tree on the basis of that
Simon: You've got to have some kind of tree structure inside this one, which controls the menu,
John: Tree and tree, then, there aren't that many levels (John laughs)
Simon: No, no…it's a relatively small tree, isn't it. But you enter it here, are called by 'add menu', find out that, oh, it's called 'project', I have that one already, so this is a submenu to 'project'
…[…]…
John:… We don't need a tree; there is only one level
Simon: Well, you can make a map, and then, er… a map of these, can't you
John: Yes, yes,… it's
Simon: It has some kind of structure which contains a child level, a map or a vector, a catalyst,
John: The question is probably how much we need to do about it; after all it is…we could just make a general map, a map like
Simon: It's just; it's just a few
John: There aren't really that many, so that's why I'm thinking why, why start making such a giant …out of this
Simon: Not very many
Simon's phone rings
John: I think we can do this…[…]…"[25]

It is amazing how easily they follow each other's thoughts. It is as if they share a world where everything is implied. There are very few instances where they find it difficult to follow each other's thoughts, and only one instance where they disagree. Let's first listen to one of the few sequences where they are unable to follow each other's thoughts. Simon has not caught onto John's thoughts and is a bit uncertain about whether John's procedure will work:

"…[…]…Simon: We could have a look at…that this one registers over here, it says 'add menu' over here
John: Well…

[25] Thinking Aloud Together_2 time code 00:41:11

Simon: What it receives is the message that it has to register a view, it will call in here and say 'add menu' …
John: Then it doesn't have to be defined, then project can say that now it has to be this type, it has to be another type, and then it sends it on to the main program when it sees what type it is, oh…ok…then we make this one…I mean different types
Simon: I didn't get that
John: I mean depending on which menu, which virtual menu, if you give it a type…in the type where it has to go, the…this menu type, it's an icon or…
Simon: It just can't say that it's…
John: No
Simon: Because it has to, it has to be in the way that it, it
John: Then we have to make some flags somewhere, at some level, some levels or something like that
Simon: We can indicate some kind of structure where we say that what we say is that we have two types of flags, we have one, which is a top level, and another, which is a child level, can't we? How should it then look in the user interface…[…]…"[26]

Finally we go on an odyssey where the question of 'object orientation' is close to separating the two developers' thought processes. For the first and only time in this view, this think aloud session, disagreement is visible, where Simon and John confront each other's thought processes. The confrontation 'comes to a head' only to dissolve again when John, somewhat reluctantly, gives in. It doesn't look like he is quite convinced.

"…[…]…Simon: Take this one … now we have inserted 'project view' here, then the user presses… menu, then 'project' has to be told that it has to deactivate
John: Yes, that's obvious
Simon: So it needs both a
John: Yes, that … But yes, yes that's, but, I mean, it could be that… when you look at the project you have to enter yourself and say I just have to tell the others that they have to close all the other views, but of course, it can't know that
Simon: Then this one only needs two messages
John: Yes
Simon: One that's called 'open view' or something like that, right …'show view' I suppose it's called
John: Yes
Simon: And hmm 'close view'; because then it gets the message from out here that it has to both open and close

[26] Thinking Aloud Together_2 time code 00:34:34

John: Then you have to create something, because hmm, it's a bit difficult because when we do object orientation like this then… on the buttons too, then it's a bit difficult because if you click on another button than the one which actually has the object, then mmm then you don't get any message.
Simon: No, …it has to hit there
John: But how?
Simon: Well, that's really fairly simple because with the outlook bar out here you will get two messages; you get a message both about that one instead of
John: Well, ok
Simon: And that's…when that one is pressed, isn't it
John: That's right…Or maybe you can put it… You can, you can then create an event which is just called 'activated' and 'deactivated', when it's clicked…or something, I don't really know what, it…I mean every time you have clicked it you have to make sure that all the others get a message that they haven't been selected…then you don't have the menu functionality, then it's not really a menu any more; with a menu something only happens when you click it…then you have to send to all, don't you, if you have a normal menu then, I mean the pulldown menu, then you also have to send a message to all the other menu points that they have not been selected.
Simon: No
John: You do that don't you?
Simon: Mmm no, you only have to send to the one which has been selected now, not the one which is active, in here … has to receive a message that now you are no longer active, now you have to deactivate, and then this one has to…, and then you have to send this one a message that it has to activate, so that's two that have to receive messages
John: You can say that it's the main program that has to do it, but it sort of has to ask to have it done
Simon: It's the main programme itself, yes, or not the main program; it's the Explorer plug-in
John: Well
Simon: It's the Explorer plug-in that controls these
John: It inserts them and removes them but it doesn't control them
Simon: It has to, it's those up there that have to control it
John: But listen, what happens when you click on them, the main programme has nothing to do with that
Simon: It has to…
John: Hm
Simon: So when you press it, it has to know that it has to de-allocate the one it's pointing at
John: Well, ok then …[…]…"[27]

[27] Thinking Aloud Together_2 time code 00:46:57

[**The photo story** *"What if?"*]

How did you code it?

I think I thought from here to there.

Or maybe as far as here?

But what if …?

Debriefing of the Code Equilibrist[28]

We met the keyboard and code equilibrist, the developer Theis, already at the beginning of our odyssey. Several people have to be present when he is debriefed, and he has to be debriefed. The company has once experienced what it means when knowledge about the code is stuck in the head of a top developer because it is personal, implicit and silent so that it can 'only' be expressed by running the code which then either works or doesn't work. They don't want to be in that situation again. Poul, the top developer, who is the man behind the idea and the key person in the project which the D-department is working on for the moment, is the informal head of both the D-department and the company as such, and together with the project manager he now sits down with Theis, the equilibrist, in order to debrief him. Only a developer at top level can debrief a developer like Theis. It

[28] Tape_4 time code [00:41:25] to tape_5 time code [00:39:00]

requires experience and knowledge to understand what he has coded, and even more to be able to see the possibilities for optimisation of the code in order to make it even more simple and elegant, but most of all to discover the pitfalls. This time, it is nothing less than the control code for the development process, which Theis has just managed to get up and running. That's all very well, but he is unable to log on to his own control code; a problem which immediately occupies him and Poul the minute there is a break in the debriefing half way through the discussion. There they sit, three strong in front of Theis' code, in order to understand the thought processes which have led to the discovery of his version of the code. The code is the obvious centre of the debriefing and the screen flickers while Theis points and explains how the thought processes behind the data model and data structure have been defined in a code. The explanations of the code are very expressive. The code is visualised using body language, which constantly gives shape and direction to the account, as the language of the body is an integrated part of the expression of thought. It is as if the hands shape the thoughts in the account, and the code serves as stage set for the visualisation process. The account is accompanied by acknowledgements from Poul, the top developer. He nods and says yes. The project manager listens attentively but mainly in order to learn something. The top developer has the final say the whole time.

"…[…]… Theis: Well, that's really what I am trying to do by having - my only principle is really that everything is a node - i.e. whether it is a template or a node or meta data, it is still a node. It contains more or less the same components but centered around an image and then a few things which control some versions and … have to do with and that sort of thing - and then I have to take into account that it has got template ID -that doesn't really matter because it can be used more generally
Poul: mm
Theis: .. and it doesn't really matter if they are there; I just have to make sure they have a …and that kind of thing. So that's really the foundation of the data model and then the fact that structure and data actually are separated.
Poul: mm
Theis :.. and then I try to place this stream which operates these things; I place it on … to the extent it is possible - I mean - I just get rid of these sentences or place them or… in the …procedures.
Poul: mm
Theis: .. and there are two reasons for that, first of all it's easy to access, just to change the … procedure, I just call it … procedure and secondly it'll be … all things being equal it will get done a bit faster.
Poul: mm …[…]…"[29]

[29] Debriefing_4 time code [00:42:48]

Several times the code is confirmed when Poul finishes off Theis' sentences. He thinks together with and across Theis' account, its thoughts and coding. Theis does the same when Poul comments on his code. He then thinks Poul's thoughts through. In that way the two developers constantly create each other's thoughts and a mutual understanding.

"…[…]… Theis: The size is really - it follows across, I mean, it lies in the nodes in the data model all the time.
Poul: mm
Theis: Everything has a total size and a total length, which is the sum of all the rows - each row contains - is limited to something; in the registration database I have some kind of maximum size per row of image fields, so that's where the 17 different rows are located in the database, and there's the total, that's the sum…
Poul: of all these rows (points at the code) …[…]…"[30]

Simon also participates actively in the debriefing process but especially in the way that he sees the code with the eyes of the user. How will the code which Theis is creating appear to the user; that is a question which he asks several times although in different ways. Let us listen to an example:

"…[…]… Simon: I also think that to define it, we say that it can only undo/redo once, I mean you can only go one step back, you cannot in inverted commas go backwards forever.
Theis: But we could really go further back because we have the entire history of the revision. Therefore, I can construct…
Simon: From the user's point of view because it will be a terrible mess because nobody will be able to understand it
Poul: No
Simon: Because from a user's point of view you won't be able to understand how you can undo something someone else has done, or what the heck do we do when someone else has tampered with the file over there, while I have done something here, so suddenly they get enmeshed, and then I undo
Theis: I think we should have a look at that, though, that if we can show the user what has happened because I should be able to show, I mean, I can see what has happened with a node overall and I can also see, show or return what has happened to that user on that node and then somehow - the more you tell the user, the better
Simon: Maybe we can arrange that you can at least - that's not something we have to do now -
Theis: No

[30] Debriefing_4 time code [00:59:44]

Simon: If in the long run we did something to the effect that you can somehow get access to … lock where you can see, where the user can see what it is that has happened, the changes you have made
Simon: We could also say that you can only undo as long as you are the last person who worked on this node. If someone else did, I mean now I move this one over here, then I can undo, but as soon as someone else takes it and moves it elsewhere, then my undo is gone. Then I can no longer undo, then it becomes…
Poul: That's a better solution …[…]…[31].

A decision can be as straight forward as that. As mentioned, the debriefing is interrupted half way through because the project manager has to deal with some other issues. That changes the form of the debriefing. Poul, the top developer, immediately looks ahead and turns the debriefing towards system facilities and upcoming coding. The thread from before is resumed when the project manager is back but now with an even greater emphasis on challenging Theis` code. In this way the debriefing shifts between the code, which is challenged, and the problems that are identified and thought through by the developers as a group until a solution has been found.

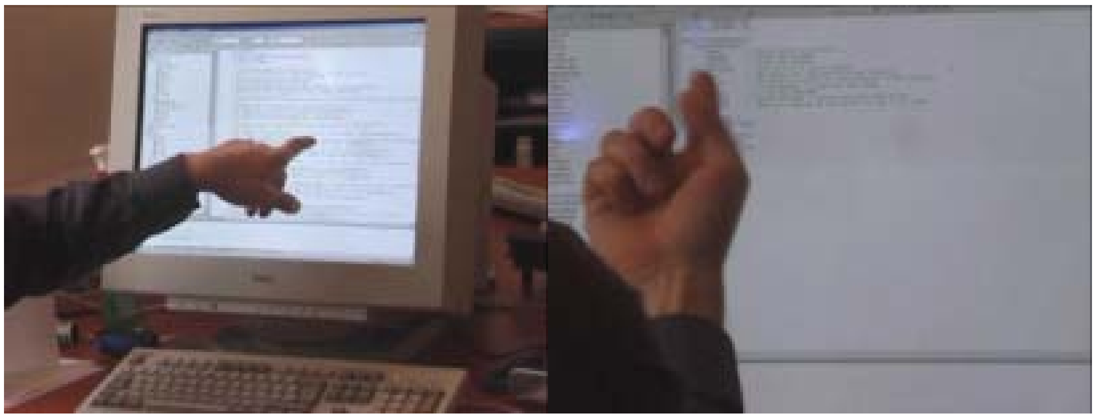
~…[…]… Poul: Is there also a remote user?
Theis: Not yet (Theis laughs) so I haven't spent time on that. I have actually imagined that the administration interface and the client interface which has a do XML, which takes a string or a stream of XML; and then I use that when I run in a web server, because then I give it an XML envelope, … pack the envelope and parse it - find out which call, parameter, make the call and get something back, wrap it up in XML and return it
Poul: I.e. we make another table…
Theis: Yes, yes, it becomes a bit like soap because soap is something like that, but there are many things I don't like about soap, so I choose to make our own, because we can always log it to soap, if we want to, later on …[…]…[32]

Theis only makes notes once during the entire debriefing process. It happens when another user problem is discovered, which he has to take into account in his code. Otherwise he appears to keep all the consequences of the debriefing in his head. A few times he mentions his doubts, but they are not followed up. In one case he asks Poul if they can code together. Poul is obliging but vague when he answers, "*It will be a few weeks"*. Obviously, Poul's time only suffices for 'vital' tasks such as debriefing the developer who is at a level that only Poul can match.
The four actors in the debriefing, the code and the author of the code, the top developer, who is also the informal leader, and the project manager all have a part to

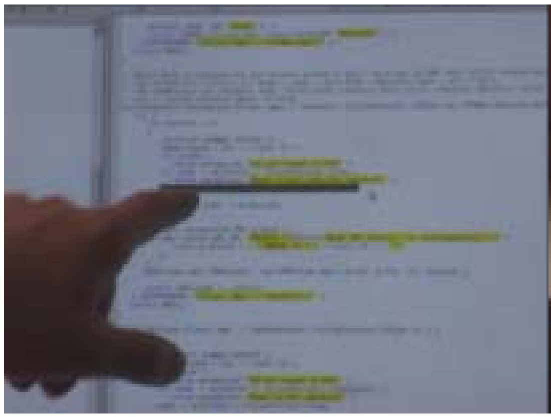
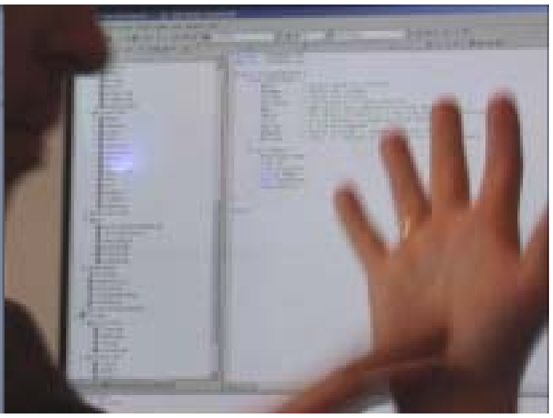
[31] Debriefing_5 time code [00:09:14]
[32] Debriefing_5 time code [00:25:51]

play, and the entire session has its roots in the company's experience from the previous project. The code has to be brought to light, it cannot be allowed to remain hidden as personal and silent knowledge of the author of the code, as was the case in the company's first large project. It is not just that it creates uncertainty when the company's foundation can be removed the minute the person who knows the code leaves it. In the previous project Poul, the top developer, who is also co-founder of the company, was the author of the code. He was responsible for the entire development process to the extent that it almost cost him his health. It is therefore not only that kind of nervousness which is behind the systematic debriefings, which are now being carried out. Experience has shown that unless these debriefings are held, it becomes impossible to plan and estimate work load and time, not to mention strengthen the team of developers with further manpower if it turns out to be necessary - and feasible. Therefore, the thought processes behind the code have to be thought through in a group which can follow them and at the same time challenge them. The top developer can do that, and he does. On the sideline, however, is the project manager who in this way gains insight and new knowledge, but who also challenges the code as seen through the eyes of a user. The author of the code explains with obvious pride and joy; he listens, is open-minded and completely trusts his own understanding of the consequences of the debriefing. They are, in fact, lodged in the thought processes of the players. Nothing is made explicit except as an amended or new code.
The session is terminated when an employee from the sales department enters the D-department to find out who has an appointment with a potential customer. He doesn't succeed and it remains unclear while the developers joke a bit about "these customers who always come and disturb".

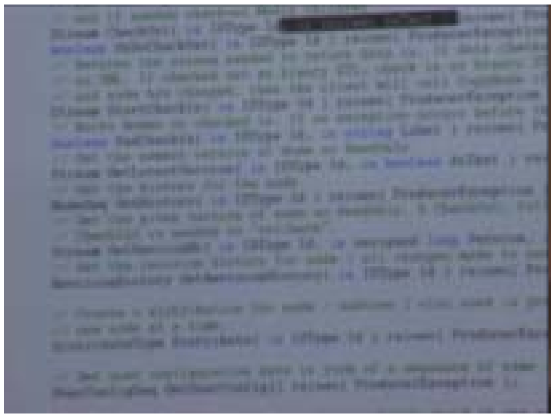
[**The Photo Story** *"The Code Equilibrist"*]



This is how the code was made! *We have to remember this …*

I have made an exception here. *We have to keep all this in mind.*



This is how dense it is! *And how beautiful it has been done!*

When Deadlines Have to Be Met[33]

The developers' great epos or the 'project' is already running behind schedule. That is strikingly visible from the timetable of the project which hangs on the wall behind the project manager. Only very few fields have been coloured yellow, the colour that shows that the task has been completed. Many of the other tasks are being worked on, but the development department is nevertheless behind schedule. Monday morning the D-team is gathered for the regular Monday meeting around the table in the centre of their department. The project manager puts time and deadline on the agenda as a problem right from the beginning of the meeting. In the introductory statement, time and work morale are mentioned as an acute problem because the project manager feels that work morale is too low on the D-team when compared with time. This is done in the form of a 'minor dressing-down'.

[33] Tape_4 time code [00:00:00] to [00:41:25]

"…[…]… Simon: Okay. It's damned important that, it's not quite, I am really glad that Theis began the meeting the way he did[34] *(everybody laughs), because I actually feel like giving you a minor dressing-down, because I simply feel that we are not really gunning for it. We also talked about that last time, and the way I look at it, it hasn't improved one bit during the week. We are way behind and nevertheless you are, er, it looks like you are treading water. It er… We'll never ever complete this project unless we get moving. It's also that everybody seems to start thinking that unless he or she goes and fights to get these things done then this project is not going to materialise, and this applies to everyone. It simply applies to everyone that, er. If just one person sits back and says thinks, someone else will probably, well, then it catches on. So we have to fight together to get this up and running. Right now we, well, we actually have a deadline. It's approaching very rapidly. We don't have much time to do it. We actually have less than a month before delivering and we don't have very much…yet*
Theis: We have…, haven't we?
Simon: And er, it's quite essential, because if we don't deliver it, the user test goes out the window, … , and, it's about to become, isn't it, so if we deliver too late it has an impact on others; then their deadline is affected. It also means that if you get behind with things like these then you have to slog to catch up. Of course, it's obvious that if it becomes a big problem, a case where we constantly have to sit and try to catch up, then we have to correct the timetable, but if it requires a week or two of overtime work, of slogging, then we bloody well have to do it. I am quite serious about that, that each of us has to decide what we can do to help getting this project back on track. It is unfortunate that we had to make a lot of changes to the plug-in structure during the past week, but then it's really good that at least we are not afraid of dropping some work and go back and say that we went down a path which was wrong. There's nothing we can do about that; then we have to go back; but that' also proof of courage. Its' also daring to change things. However, as we have made that decision we are now even further behind with the plug-in structure. Therefore that's an area where we have to work really hard to catch up. We also have our dear little Lab server which we really need to get up and running because we need some data to work with …[…]… [35]

On the face of it, they all seem very intense and concentrated on their work when you enter the developer team as an outsider. However, things are nevertheless falling apart. A glance at the project timetable and it becomes difficult to imagine the work morale which can cope with being behind right from the start and have to race all the way up hill from a place at the back of the field. The fight is against one of the multi media giants on the world market. The developers' great epos is

[34] Theis begins the Monday meeting - with ironic detachment - with pep talk and battle cries, and the other developers join in with the battle cry "Bush", "Bush" - also with ironic detachment (the recordings were made before 11 September 2001).
[35] When Deadlines Have to Be Met_4 time code [00:01:05]

nothing less than an alternative to Macromedia's 'Director', a well-known product well placed on the market. Together they are writing the second version of a normal set of multi media tools which on several points differ considerably from the first version, because it turned out that the application, which had been developed together with the tools, impacted too much on the development tool itself. This limited the application of the product. In the second version they have created a multifaceted solution, which consists of a general multimedia tool plus an e-learning perspective planned as a series of templates which accompany the general tool.

What do you do, though, with a situation like the one facing the D-department already now? How do you create motivation and thereby a work morale which is strong enough to race all the way up hill in a battle against time which is an overpowering adversary? How did David conquer Goliath? The project manager's answer is slog and team spirit, which reminds you of sport and creates associations to elite sport at international level with battle cries and fighter spirit. The developers simply have to continue slogging until the timetable looks realistic again, even if it means working evenings and weekends. It is up to the individual to realise his responsibility towards the entire team and the project in order to be able to make the necessary effort. It also counts if the work is meaningful, if it is fun and exciting. This is a point of view which becomes evident in Theis' attempt at explaining why the situation is as described by the project manager. He says:

"…[…]… Theis: I think part of the problem is that it's difficult to see something concrete, to imagine a goal, when you can't…When you get going, then you can say, okay, I just need to add another feature before, …No, I just want it to work before I go home; but as long as, I mean, when… then it's a bit, er … er, well, I need to…we can look at that tomorrow, can't we, I suppose that's how most feel, as long as you can't see…, as soon as the first concrete…
John: I don't think I feel like that
Theis: You don't? Well, okay…
John: The things I start I also want to see work
Theis: You are working on something concrete, that's different
John: That's true
Theis: The editor isn't something that
Simon: We would all like to have something concrete
(Everybody is talking at the same time)
Theis: Yes, yes, concrete, something cool which you can SEE develop, which becomes real, which is alive and…
Simon: You mean that the boring prompt with mistakes, that's not
Theis: If you can't combine something which makes sense, which, er, then you become a bit, orhh, something is missing…

Simon: That's true, and, of course, it's quite logical, and it's probably something that happens in the early stages, but the problem is that the boat is leaving and if we don't get on board now, well, then we'll miss it.
(Prolonged silence where they all look at each other or stare at the table) …[…]…[36]

It is difficult to pull yourself up by the hair and slog if it doesn't' make sense; isn't exciting, if you can't see that you are creating something which is alive and substantial. However, that is not the only reason for the current situation. The project manager points out that it has been necessary to drop a large amount of work which had been done, with the effect of bringing the project even further behind than it already was, when it was decided the week before to begin afresh with the plug-in structure. Other developers point to network problems. They have obviously experienced a lot of them. The Monday meeting did in fact begin late because the company's server had been down with staff working frantically to get it up and running again. A combination of many factors has therefore contributed to the situation where *time has taken on a principal role right from the beginning of the developers' great epos.*
The situation is, however, even more difficult than it appears. Poul, who is top developer, co-founder of the business and informal head of the D-department arrives late at the Monday meeting and immediately asks to make a comment. It is about the staff meeting which also takes place on Mondays.

"…[…]… Poul: I have a comment to make
Simon: Yes
Poul: I need to talk to you all before you leave for the Monday staff meeting
(General muttering)
Simon: Then it had better be soon… Does it have to be now?
Poul: Søren is going to talk about cost cutting, and er, at the Monday meeting, and I just want to say,…it won't have any effect on what we are doing,… …so we talked about having to cut down…
Simon: it will still have an effect anyway, because, er, because, it means that we can't just spend money as we please. We can't very well sit here as a small department and then (makes a gesture of someone pouring money out), buy steaks and er…
Poul: We have the budget that we have had all the time, and we still have it. So that was the only comment I wanted to make, when we talk cost cutting and…, then it doesn't apply to us…
Theis: That means it could be a very long Monday meeting?
Poul: It could, yes
(General muttering) …[…]…[37]

[36] When Deadlines Have to Be Met_4 time code [00:05:45]

The staff meeting doesn't take long as far as the developers are concerned. The D-team quickly returns to their status meeting where they agree on the details of the joint and critical code reviews, which are to take place later in the day. The meeting is resumed in high spirits with jokes, laughter and fooling around. It is obvious that they are all affected by the briefing about the company's financial difficulties and the necessity for retrenchments and cost cutting, but they all try to make light of it. It is only visible in a very indirect manner that the developers are aware at some level that the company's financial situation will impact on the D-department as well. It is Theis who more than hints at this in a joking conversation about pay rises and coffee consumption:

"…[…]… Simon: Err, as Poul also says, then this savings drive, err, it doesn't really involve you, there is no-one in the development department who is at risk of being fired, so you can just relax… We have in fact a situation where we're still in need of people, aren't we, contrary to the production department, so there is no reason to panic. Of course, as we just briefly discussed, it also means that we now, that there will be a lid on to some extent, on what is spent on external costs. It will be something like … seminar (expressive body language), I am afraid, you know, but no. When there are minor liquidity problems like these, you know. As starting point we have the budget, as we have had all the time, for this project and for this department. So in principal it has no impact on us
Hans: …not drink so much coffee
(Laughter)
John: and pay rises - that will have to wait
(Laughter)
Simon: no, there …, there won't be any coffee restrictions
John: coffee and pay rises
Simon: it's half a pot, half a pot each per day, half a pot
Theis: and what if the project takes three times as long? then…?
(Loud laughter) …[…]…[38]

Theis hits the nail on the head with his comment. If the developers are already far behind schedule and if the budget at best is the same, then it is easy to see that the ends won't meet. Regardless of the fact that everyone tries to make light of the situation and believe that it is not going to affect the D-department. If nothing else, then the struggle with time, and thereby the need to slog, becomes that much harder. However, the Monday status meeting continues as usual without further discussions of retrenchments and cost cutting. They sum up where they are with the great epos which they are all working on; they discuss how they can best co-

[37] When Deadlines Have to Be Met_4 time code [00:22:45]
[38] When Deadlines Have to Be Met_4 time code [00:31:30]

operate and the best way of creating a simple and elegant code, they conduct critical reviews and they debrief.

There is an obvious difference, and a surprising discrepancy, between the way in which the developers handle their knowledge and experience in their daily professional practice and the way they handle the crucial problem that the company is not doing well because of financial problems. Problems that are so huge that cost cutting and retrenchments are necessary.

The 22 video views give an insight into a complex system of knowledge creation and learning, sparring and co-operation, studies and sharing of knowledge, and a picture is drawn of a competent professional practice which is reflective; where many forms of reflective practices are communicated via the computer screens and the common epos. Reflective practice in knowledge-intensive work. A practice where strategic actions are taken in line with department goals as long as the developers deal with a simple and optimal code which can compete with top players worldwide.

The Monday meeting, on the other hand, draws a completely different picture. It ‘looks’ more like a traditional organisation with wage earners, where the project manager is the shop foreman who upholds authority and discipline. The meeting begins with a ‘dressing-down’ of the developers because the project is behind schedule; they won’t meet the deadline if the developers work as they do, according to the project manager. They have to boost work morale, buckle down, work overtime when necessary, work weekends or evenings as and when required by the project.

When co-operation takes place round a meeting table and not at one of the many computer screens, when the culture is a ‘meeting culture’, where the reflective practices involve the group as a whole and their relationship to ‘the others’ in the company, the picture becomes quite different; then style, mood, form as well as culture change.

The project manager stresses again and again that retrenchments and cost cutting measures are matters which the developers, the project, the department don’t need to deal with. The budget for the project won’t be touched; it remains unchanged. Nobody in the D-department will be affected. There is almost an atmosphere of relief and high spirits at the Monday meeting after the staff meeting about the company’s problem. Nobody considers whether the D-department can do something; nobody asks how it will impact on the way they will be working on the project. That is surprising. It had just been pointed out during the first half of the Monday meeting that the D-department has problems as they are already behind schedule. Indeed, the D-department is behind precisely because the company has had financial problems. It is the reason why the developers have completed a lot of ad hoc tasks in order to bring in money. Nobody now asks how the D-department is supposed to solve the problems they already have with time in the project if the company is still in financial difficulties. Will they have to complete ad hoc tasks again in order to improve finances? Will the project have to be sidelined, or will less staff be allocated to the task while they are making money? The situation is more than likely to arise and once again impact on the project and on time. Nobody asks for “intelligent methods” to solve the problems with time, for shortcuts or new strategies. The solutions are based on ‘raw’ brain power and proven methods: work discipline has to go up, they have to buckle down and work harder, even if it means working overtime during weekends and evenings.

Conclusions

For the developers on the D-team, there is no doubt what knowledge is. It is being able to ‘code’. It implies not only knowing a lot of examples of good codes, but rather the ability to make the code come alive so that a simple code can create a multitude of possible actions and meanings in the shape of functionality and experiences in the software they are developing, the great common epos. The ability to code in general is not of much use. The developers’ knowledge takes the shape of ability in relation to the exact project at hand. In the D-department, therefore, the proof of knowledge is ability, which has a concrete expression. The code either works or it doesn’t work. The code is a symbolic construction just like an epos, but it differs from an epos in that ‘the code can be run’, and in a way which gives actual proof of whether it works or not, whether it ‘is stuck’, self-oscillates or has other unintended effects. The code is written in an alphabet which is both symbolically and mechanically active as is the case with all computer text[39]. These texts can both express and function. In this way the D-department’s knowledge is tied to practice through the creative processes of coding. However, it is a knowledge that exists in many forms. It is in the form of thick books, or shear bibles, found everywhere on the desks and on the book shelves in the library, but, also in the form of the code which has already been created and which is still being created on an ongoing basis during their daily work. It is not of much use, though, neither as knowledge which already exists, nor as knowledge which is being created, if it doesn’t make sense, in reality and in their shared practice. This is a rather obvious statement when we are talking about knowledge in the form of shear bibles. Being able to ‘quote’ examples of codes without being able to turn them into practice is obviously of no relevance. However, it is probably less obvious that the rule also applies ‘the other way round’ for the code which is created in practice. To be able to code without knowing why and how, and in practice this means without being able to explain and document the code, is not only of no relevance; it can create great problems for the project and the company. If the code only exists as personal and silent ability, nobody can help develop, understand and test it. This problem becomes more serious with increasing complexity, when the

[39] This is a point of view which has been expressed most succinctly by Niels Ole Finnemann in his thesis about the computer’s symbolic properties ”Tanke, Sprog og Maskine” 1994 (“Thought, Language and Machine” 1994).

code becomes difficult to handle without being made explicit in shared practice. This is an insight which has been gained through bad experiences during the company's first large development project. Experiences on which the D-department now builds its professional practice.

Knowledge exists outside the actual context and is at the same time a part of the knowledge which is created in concrete form and in context; explicit knowledge goes hand in hand with intuitive and silent knowledge; certain knowledge with uncertain knowledge. In other words, the many different forms of knowledge continuously merge in dynamic processes; they unfold as variations of simple themes where knowledge constantly undergoes transformations and translations. Knowledge which comes from *the outside* in the shape of known top players' code, is transformed and translated to knowledge which is *inside*; knowledge which exists *out of context* in the 'shear bibles' of coding are brought *into context* and given life in the coding of shared experiences in the great epos; *personal, intuitive and tacit* knowledge, such as "nose" when the competitor's code is broken, becomes *explicit and public* knowledge about 'how simple the code has become' on the Monday status meetings. In the 22 video views, the observing analytical eye gets so close to the various forms of knowledge, as 'these' unfold and merge in practice, that they appear as close-ups. When the eye in this way is focused on dynamic knowledge processes in practice, it becomes of decisive importance to understand the phenomenon of dynamics. In other words it becomes important to identify and understand the chains or spirals of translations and combinations that constantly arise between knowledge that exists and knowledge that is being created, when knowing is in action, and thereby also the interplay between the many well-known forms of knowledge, including not the least the many quasi-forms or intermediate forms, which knowledge take when 'it' is viewed as a dynamic phenomenon. In addition, an analysis, which is focused on knowledge as dynamic processes, also has to deduce the different relationships between time and place on which the dynamics depend. These theoretical considerations lie outside what is possible in the descriptive analysis but play the principal parts in the article *"Knowledge at the Right Time and the Right Place"*, which will follow.

If we turn the eye to the descriptive analyses once again, an obvious fact should be mentioned which is that the D-department 'runs code' every Monday, so that abstract knowledge and practical ability can merge in varying movements, translations and combinations, while they take stock at the same time of each developer's coding and of the department as a whole. The code is therefore also run in many different ways. The practices of the department are reflected in the contours of various *knowledge zones*. Not because there is an action plan for these activities or because they are part of work or job descriptions. They are also not 'written into' the equipment and facilities of the room. Knowledge zones are, however, a social universe which is created in the reciprocal glances of the developers. The way they look at each other creates, shapes and reshapes the contours of the knowledge zones which play a major role in practice. There is no doubt about the zones that exist; and it is because of these zones that the code is run differently *in* the different zones and *between* the zones. In this way they form the basis of learning and knowledge sharing in a practice community.

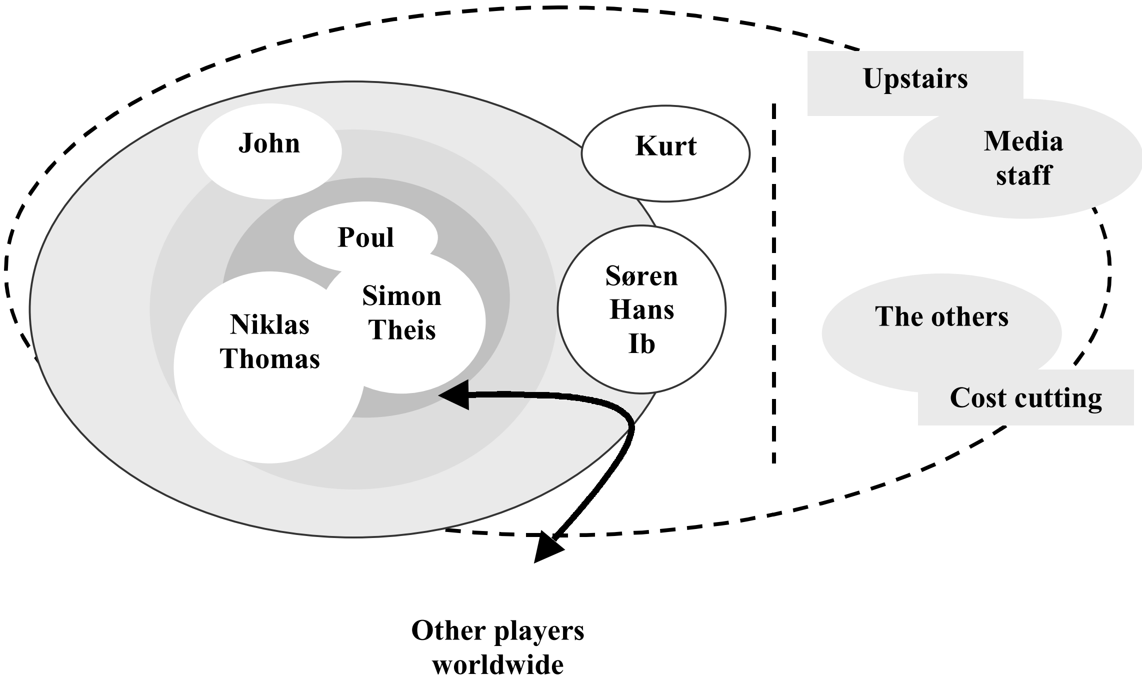


Fig. 3: Knowledge zones in the development department of the IT company. Snapshots of a 'social universe' which is constantly changing while being created through the developers' mutual observation. The knowledge zones form the basis of a complex system of knowledge creation, knowledge sharing and learning in a professional practice community.

Kurt does not participate in running the code. He sits in the outer periphery of the department and absorbs its style, atmosphere and tone while studying to become a developer instead of a boat builder, for which he is trained; however, he doesn't yet belong. His eye is purposefully focused inwards towards the centre of the department and its inner circles. He takes time to advise the media employees who comes down from upstairs and seek advice about how to get into the D-department 'downstairs'. In that way he becomes part of it himself. Although no doors separate the D-team from the rest of the company, it is nevertheless experienced as a world apart in which everybody's eyes are turned inwards towards each other and the common code, but only to a limited extent outwards towards the others or upstairs. When Kurt directs his eye to 'upstairs', it is not in order to find out what happens 'up there' but in order to become part of what is inside, downstairs, in the D-department.
Ib runs the code, but he does so next to the developer Theis, not on Mondays but on Fridays because he has to attend lectures as part of his education in computer science on Mondays. This fact is accepted reluctantly and with forbearance. The

code is run next to Theis so that he can help if the problems become insurmountable. Supervision and support are needed, and in this way Ib participates in a kind of neighbour training. Søren and Hans sit facing each other, and they help each other. They are having problems. Especially because Hans really is busy writing a compulsory essay as part of his education, also in computer science, or perhaps more correctly put, he 'ought' to be busy with it, because it remains unclear whether he is actually writing it, has finished with it, or what? Hans keeps a low profile. He tries to avoid attention, but he has a good friend or colleague in Søren, who silently but loyally assists. All three sit in the outer knowledge zone of the D-department, but they belong.
John runs code together with the developers from the innermost zones both in thought as described in the story *"Thinking the Code Aloud Together",* where he thinks together with the project manager Simon, but also in the critical reviews where the two experienced developers, Niklas and Thomas, sit down on either side of him and almost cover him while they run code together. John is not educated and also not studying. He is autodidact and brings with him experience from previous jobs. He doesn't belong to the innermost zones.
Niklas and Thomas work intensely on research and on explaining and documenting the code. They belong to the innermost circles, but are not right in the centre. They are both fairly new, so it is merely a question of time before they move right into the centre. They have both completed long-term education and Niklas has almost an entire working life of experience as developer behind him. They both work very independently. Niklas has his own system of intelligent methods as described in the story *"When the Code Is Broken It Has to Be Checked"*, but they also participate in critical reviews both of their own codes and of the other D-department developers' codes.
In the centre is Poul, the top developer. He is autodidact, co-founder of the company, informal head of the D-department and of the company and also the man behind the first version of the project. He enjoys undisputed authority in the department. The project manager, Simon, is also in the centre of the knowledge zones. He is the formal head of the department, and he contributed to making the first project a reality. He has therefore gained some of the hard-earned experience which is the reason behind the Monday status meetings with codes that are continually run, thought through, explained and documented in order to be brought to light. Finally, the developer Theis is also in the centre. He enjoys great authority because of his 'natural gift'; he really has a background as engineer. Only Poul can match him. When Theis' code has to be run it is not because it has to be reviewed but rather in order to debrief him as described in the story *"Debriefing".*
The creative talent for making a simple and systematical code places Poul and Theis in the inner centre of the knowledge zones. Poul's view of education can be condensed into a few words: 'self-taught is well taught'. It is the combination of ideas, vision, creative talent, ability and ruthless slog which creates good developers according to Poul. However, the problems inherent in this point of view be-

came apparent in the company's first project. It nearly went off the rails because the code remained hidden in that it was personal and silent. Only Poul could interpret the breakdown of the code or the other problems that materialised whenever others ran the code. A situation which was completely untenable. The project manager Simon was part of the team which was employed to get the code 'out of Poul's head and body', so it became explicit. Experience has shown that the waters separate where documentation begins. The ability to document a code and think it through with others, make it explicit as part of the shared knowledge in the D-team, is therefore approaching the centre of the knowledge zones. That is largely the reason why John is not in the centre, although he has several years' experience as developer. It is difficult to get a clear answer to how and why he has coded the way he has. In a great epos like that of the development software, which the D-team is writing, explanation and documentation together with recall and rethinking therefore become necessary prerequisites. At the Monday meeting the project manager does, in fact, regularly make the comment that *"…[…]… it is easy to see who the academics are in this team…[…]…"* This is a direct reference to the issue of documenting the project as a whole and in particular the code. The D-department was originally based on the ideal of the developer with 'a natural gift' but through reflective practices it has turned out that the 'schooled' ability, which is required to document and explain, is necessary to handle the complexity of the great common epos, if it is to work.

There appears, however, to be almost watertight vertical blinds between the D-department's complex systems of knowledge zones and the reflective practices, which they outline, and the rest of the company. The coding goes hand in hand with the expertise of the specialist, the artist, the musician with refined techniques, "intelligent" work methods and reflective strategies. The project as such, however, with its timetables, deadlines and agreements, is something completely different. The project has a manager who, like a shop foreman, gives dressing-downs, demands more work, improved discipline, better work morale, more overtime work and 'stay for the weekend' if necessary for the project; and it has staff who accept being reprimanded, who disclaim responsibility for finding strategic solutions to critical problems, who put up with 'hard work' and 'slogging' when time and agreements are the issue, but not the code.

It looks like the limit to reflective practices is found where the computer screen and the common code define the knowledge zones as opposed to the situations where the D-department is part of the company as a whole. In this way, the D-department's system of knowledge zones 'is closed' unto itself; it is a system which is self-sufficient. This is not a knowledge strategy that applies to the company as a whole but a practice in the D-department, and the system's primary references are more likely to be players of the same kind elsewhere in the world.

References

Alrø & Kristiansen: Mediet er ikke budskabet in: Alrø & Dirckinck-Holmfeld (Eds.): Videoobservation, Aalborg, Universitetsforlag, 1997.
Alrø, H. (red.): Organisationsudvikling gennem dialog. Aalborg Universitetsforlag. Aalborg. 1997.
Alrø, Helle & Lone Dirckinck-Holmfeld (eds.): Videoobservation. Aalborg Universitetsforlag 1997.
Argyris, C. & Schön, D.A.: Organizational Learning II. Theory, Method and Practice. Addison-Wesley, Reading, Massachusetts, 1996.
Argyris, C. & Schön, D.A.: Organizational Learning. A Theory of Action Perspective. Addison-Wesley, Reading, Massachusetts, 1978.
Argyris, C. & Schön, D.A.: Theory in Practice. Increasing Professional Effectiveness. Jossey-Bass Publishers. San Francisco, London. 1987.
Bateson, G.: Steps to an Ecology of Mind. New York: Ballantine Books. 1972.
Benesh Ronald & Joan: Reading Dance – The Birth of Choreology, Condor Book, Souvenir Press (E&A) Ltd, 1977.
Borgnakke, K., (1996) Procesanalytisk teori og metode, I & II. Danish University Press. Copenhagen.
Brown, J., Collins, A. & Duguid, P.: Situated Cognition and the Culture of Learning. Educational Researcher, Vol 18 No 1, 1989.
Castells, Manuel: The Information Age: Economy, Society and Culture, Vol 1, The Rise of the Network Society. Blackwell Publisher 1996.
Christensen, Allan (red.): Den lærende organisations begreber og praksis. Læring – Refleksion – Ændring. Aalborg Universitetsforlag. Aalborg. 1997.
Christensen, Søren & Molin, Jan: Organisationskulturer. Akademisk Forlag. København. 1983, 1987.
Christiansen Ellen, Jensen Sisse Siggaard, Nielsen Janni and Orngreen Rikke: Learning Happens – Artifacts, Space and Body – rethinking video. Working Paper, Copenhagen Business School 1999. Unpublished
Christiansen Ellen, Jensen Sisse Siggaard, Nielsen Janni and Orngreen Rikke: Learning Happens – Rethinking Video Analysis; in Dirckinck-Holmfeld, Lone m.fl. (Ed.): Learning in Virtual Environments 2001. Research Network on Learning and Multimedia (forthcoming)
Christiansen Ellen: Explorations into other ways of doing videoanalysis, working paper, unpublished 1999.
Christiansen, Ellen: Hvad sker der egentlig på gangene? in: Alrø, Helle & Lone Dirckinck-Holmfeld (eds.): Videoobservation. Aalborg Universitetsforlag 1997.
Dewey John: The Sources of a Science of Education. New York: Liveright 1929.
Dewey, J.: How we think. A restatement of the relation of reflective thinking to the educative process. D.C. Heath and Company. Boston. 1933.
Dewey, John: Experience and Education. Kappa Lecture Series. Collier Books, New York and Collier Macmillian Publishers, London 1938.
Dewey, John: Freedom and Culture. New York: Capricorn Books 1939.
Dewey, John: Lectures in the Philosophy of Education: 1899. New York: Random House 1966.
Dewey, John: Rekonstruktion i filosofien, i Hartnack, J. & Sløk, J.: John Dewey. De store tænkere. Berlingske Forlag, København 1969.
Dewey, John: The School and Society. The University of Chicago Press, Second Edition 1915.
Dewey, John: Theory of Valuation. International Encyclopedia of Unified Science. Vol II No 4. The University of Chicago Press 1939.

Eco; Umberto: Kant og næbdyret. Hvor ved vi fra, at en kat er en kat? En bog om sprog og erkendelse. Forum, København 1997.

Engeström, & Midleton (Eds.): Cognition and Communication at Work. Cambridge University Press, 1995.
Engeström, Yrjö. Training for change: new approach to instruction and learning in working life. Arbline. Arbetslivsinstitutet.1998.
Eriksen, Sara: Knowing and the art of IT Management. An inquery into work practice in one-stop shops. Ph.d. afhandling fra Ronneby Högskola 1998.
Finnemann, N. O.: Tanke, sprog og maskine. En teoretisk analyse af computerens symbolske egenskaber. Akademisk Forlag, Kbh 1994.
Gherardi, Silvia & Nicolini, Davide: The Sociological Foundations of Organizational Learning; in Dierkes, Meinolf; Antal, Ariane; Child, John & Nonaka, Ikujiro (Eds.): Handbook of Organizational Learning and Knowledge. Oxford University Press 2001.
Goffman Erving.: The Presentation of Self in Everyday Life. New York, Anchor 1959.
Goodwin C. & Goodwin M: Formulating planes: Seeing as situated activity. In: Midleton & Henderson, A. & Jordan B.: Interaction Analysis: Foundation and Practice. Xerox Palo Alto Research Center and Institute for Research on Learning, Unpublished Paper, 1994.
Hart J.: The Art of the Storyboard - Storyboard for Film, TV and Animation, Focal Press, 1999.
Hockney, David: That's the Way I see It. Cronicle Books 1987.
Jacobsen, J. (red.): Refleksive læreprocesser - en antologi om pædagogik og tænkning. Politisk Revy. København. 1997.
Jacobsen, J.C. (red): Autopoesis. En introduktion til Niklas Luhmanns verden af systemer. Politisk Revy. København. 1992.
Jacobsen, J.C. (red.): Autopoesis II. Udvalgte tekster af Niklas Luhmann. Politisk Revy. København. 1995.
Jensen Sisse S.: "The Matrix: practice, foci and design - building shared frames of reference in interdisciplinary research on new media, learning and work" i "New media, work, design and learning". Documentation from an international research workshop. National Institute for Working Life & Departement for Art and Communication. Malmö University, 25-27 February 1999.
Jensen Sisse Siggaard: De digitale delegater: tekst og tanke i netuddannelse – en afhandling om hyperlinks i refleksiv praksis, der er face-to-interface. English Summary. Website access to full-text version, a pdf-format for local print and a print-on-demand version: www.afhandling.dk Forlaget Multivers www.afhandling.dk København 2001.
Jensen, Sisse S.: "New Media, Adult Learning and Work Practice". A report on teamwork sessions. "New media, work, design and learning". Documentation from an international research workshop. National Institute for Working Life & Departement for Art and Communication. Malmö University, 25-27 February 1999.
Jensen, Sisse Siggaard & Mønsted, Mette: Managing uncertainties in R&D processes in small IT-firms. Proceedings EGOS at E.M. Lyon July 2001.
Jensen, Sisse Siggaard: Kairic Rhythmicity in the Turing-Galaxy. Proceedings. International Conference on Timing and Spacing: Rethinking Globalization and Standardization. Palermo, November 2001.
Jensen, Sisse Siggaard: Når tiden foldes i en gråzone. En beretning om vidensdeling, arbejdsdeling og refleksiv praksis i en IT virkdsomhed indenfor e-learning; in Mønsted, Mette & Poulfeldt, Flemming (Eds.): Tid og timing. Forskningscenter for Ledelse, Organisation og Kompetence & Institut for Ledelse, Politik og Filosofi, Handelshøjskolen i København 2001. (forthcoming).
Suchman. Lucy & Judith: Six Readings of a Single Text: A Videoanalytic Session, Panel, Procedings CSCW 98, Seattle Washington 1998.
Koschmann Timothy, Anderson Anne, Hall Rogers, Heath Christian, LeBaron Curtis, Olson

Latour Bruno.: Technology is society made durable. In Law (Ed.): A Sociology of Monsters: Essays on Power, Technology and Domination, 1991

Latour Bruno: A Few Steps Towards the Anthropology of the Iconoclastic Gesture, in Science in Context, Vol 10 No 1 1997
Latour Bruno: Artefaktens återkomst. Ett möte mellan organisationsteori och tingens sociologi. Studier i företagsekonomi 5. Nerenius & Santérus Förlag 1988.
Latour, Bruno & Wolgar Steve: Laboratory Life: the Social Construction of Scientific Facts. Sage, Los Angeles 1979.
Latour, Bruno: Aramis or the Love of Technology. Harvard University Press. Cambridge Mass 1996.
Latour, Bruno: Artefaktens återkomst. Ett möte mellan organisationsteori och tingens sociologi. Studier i Företagsekonomi 5. Nerenius & Santerus Förlag 1998.
Latour, Bruno: Do Scientific Objects Have a History? Pasteur and Whitehead in a Bath of Lactic Acid, in Common Knowledge, Vol 5 No 1 1996.
Latour, Bruno: On Interobjectivit,y in Mind Culture, and Activity, An International Journal, Vol 3 No 4 1996.
Latour, Bruno: Pasteur on Lactic Acid Yeast: A Partial Semiotic Analysis in Configurations, 1992.
Latour, Bruno: Science in Action, How to Follow Scientists and Engineers through Society. Harvard University Press, Cambridge Mass 1987.
Latour, Bruno: Technology is Society Made Durable in J. Law (Ed.) A Sociology of Monsters. Essays on Power, Technology and Domination. Sociological Review Monograph, No 38, 1991.
Latour, Bruno: Trains of Thought: Piaget Formalism and the Fifth Dimension, in Common Knowledge, No 3 1997.
Latour, Bruno: We Have Never Been Modern. Harvard University Press. Cambridge, Massachusetts 1993.
Nielsen Janni: Pictures and words – rethinking video analysis. Working Paper 1999. Unpublished.
Nonaka, Ikujiro; Toyoma, Ryoko; & Byosière, Philippe: A Theory of Organizational Knowledge Creation: Understanding the Dynamic Process of Creating Knowledge; in Dierkes, Meinolf; Antal, Ariane; Child, John & Nonaka, Ikujiro (Eds.): Handbook of Organizational Learning and Knowledge. Oxford University Press 2001.
Orngreen, Rikke: The Movement Perspective in Video Analysis. Working Paper 1999, Unpublished.
Qvortrup, Lars: Det hyperkomplekse samfund: 14 fortællinger om informationssamfundet. København, Gyldendal, 1998.
Schön, D., (1987). Educating the Reflective Practitioner. San Francisco: Jossey-Bass.
Senge, P.: The Fifth Discipline. The Art and Practice of the Learning Organization. Century Business. London. 1990.
Suchmann Lucy & Randall Trigg. Understanding Practice: Video as a Medium for Reflection and Design in: Joan Greenbaum & Morten Kyng: Design at Work. Cooperative Design of Computer Systems, Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey 1991.
Tillich Paul: On Art and Architecture. Crossroad, 1973.