# Corealisation:
# A Radical Respecification of the Working Division of Labour in Systems Development

Alexander Voss

Doctor of Philosophy

School of Informatics

The University of Edinburgh

2006

# Abstract

This thesis develops and assesses an alternative approach to developing IT systems in complex organisational settings, aiming to equip IT professionals with an orientation to design that allows them to create uniquely work affording artefacts that closely fit the working practices of those working with them. This involves a radical respecification of the working division of labour in systems development.

Regular reports of failing IT projects have lead to a sense of an ongoing crisis that persists despite the development of various candidate remedies over the past decades. This thesis starts with a critical appraisal of various issues encountered in systems development, conceptualisations of design work and a discussion of the problem of "informing design". A review of various approaches taken to address this issue reveals that the relationship between 'design' and 'use' and between 'designers' and 'users' is at the heart of the matter.

Drawing on ethnomethodology as a means of studying work as a socially organised, situated activity, I then introduce the notion of *corealisation* as a radical respecification of design. Corealisation aims to erase the boundaries between 'design' and 'use' by fostering a longitudinal partnership between IT and non-IT professionals orienting to the work on and with IT systems as a whole rather than as separate processes. It takes seriously the *ethnomethods* of all parties, calling practitioners to consider exactly what it is that they and their fellow members *know and use* in doing the work of IT design: *how* the work to be supported gets done in the here-and-now, with these resources at hand rather than according to some representation of how work gets done that is external to the setting and has little or no connection to the purpose at hand.

An ethnographic study of work in a manufacturing plant and of IT design in this setting provides the background for the subsequent discussion of the programme of corealisation, especially the notion of design *qua* member. The thesis goes beyond traditional research methodologies by documenting and reflecting upon the researcher's experiences as a corealiser of systems, working with other members of the setting. This highlights the importance of having a familiarity with the 'biography' of a place as a resource for design work.

Finally, the thesis discusses various aspects of corealisation, drawing out implications for the social organisation of design work, especially issues of participation, the use of representations in design work, aspects of dependability and, last but not least, the question of how widely the approach of corealisation may be applied.

# Acknowledgements

I wish to express my gratitude to my supervisors, Rob Procter and Robin Williams for their support, guidance and encouragement during the writing of this thesis. I would also like to thank Mark Hartswood, Roger Slack, James Soutter, Kate Ho and Jenny Ure as well as my colleagues within the Dependability IRC for their invaluable comments and the numerous and enjoyable discussions we have had.

Thanks are due to the workers at EngineCo whose work constitutes much of what is reported on in this thesis. Without them and their patience and support, this thesis could not have been written. It is their everyday activities, troubles and inventiveness that inspired this work and I am very grateful to have been allowed to be part of their world and to enjoy their company.

Work on this thesis was funded by the Engineering and Physical Research Council under studentship award number 00304580 and through the Dependability Interdisciplinary Research Collaboration (DIRC, EPSRC grant number GR/N13999). I would like to thank all colleagues from DIRC for their support and the inspiring discussions we had. I feel particularly thankful that I have been able to work with so many excellent people and to benefit from the coming together under one roof of so many different views and ideas. I would like to mention Stuart Anderson and Mark Rouncefield in particular.

I would also like to thank Bettina Törpel, Steffen Budweg and the other members of the Bonn reading group for their comments on my work and for their company throughout this effort.

Thanks, of course, are also due to my family and friends who have given me great support while suffering much neglect, especially through the last few months of writing up the thesis. Special thanks go also to the Watt family for their friendship and hospitality.

# Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. Some parts of the thesis and/or the data it refers to have appeared previously in publications listed in the appendix.

(Alexander Voss)

# Table of Contents


1 Introduction ..... 17

1.1 Software crises ..... 18

1.1.1 The Computer 'Reaches Out' ..... 22

1.1.2 An Ongoing Sense of Crisis ..... 24

1.2 Computer Science's Objects ..... 25

1.3 Accomplishing Computer Science's Objects ..... 27

1.4 Corealisation ..... 29

1.5 Chapter Outline ..... 30

2 Issues in IT Systems Development ..... 33

2.1 IS Methodology: Prescription, Process and Evolution ..... 34

2.2 Design for Collaborative Work ..... 37

2.2.1 Plans, Procedures and Situated Action ..... 39

2.2.2 Knowledge and Skills ..... 41

2.2.3 'Success' and 'Failure' ..... 43

2.3 User-Designer Relations ..... 44

2.3.1 'Users' and 'Designers' ..... 45

2.4 Design as Collaborative Work ..... 47

2.4.1 The Hidden Work of IT ..... 51

2.5 Innovation in Use: Envisaging the Future ..... 54

2.6 Dependability ..... 56

3 Understanding Practice – 'Informing Design' ..... 61

3.1 Ethnography for Design ..... 61

3.1.1 Doing Ethnography for Design ..... 66

3.1.2 Typifications and Patterns ..... 69

3.2 Participatory Design ..... 71
3.2.1 PD as a Political Project ..... 71
3.2.2 The Diversity of PD Approaches and Methods ..... 74
3.2.3 Ethnography in PD ..... 75
3.2.4 Cooperative Prototyping and Bricolage ..... 77
3.3 Contextual Design ..... 79
3.4 Configuration, Tailoring and End-User Programming ..... 82
4 CoRealisation ..... 87
4.1 Ethnomethodology ..... 87
4.1.1 Accountability ..... 90
4.1.2 Indexicality and Situatedness ..... 91
4.1.3 The Documentary Method ..... 92
4.2 Taking Ethnomethodology and Design Seriously ..... 92
4.3 Design qua Member ..... 95
4.4 Hybrid Knowledge ..... 99
4.5 Uniquely Adequate Systems ..... 101
4.6 Unique Adequacy of Methods ..... 103
4.7 Corealisation's Principles ..... 105
5 Case Study: Production Work at EngineCo ..... 109
5.1 Methodology ..... 109
5.1.1 A Note on Interdisciplinarity ..... 113
5.2 EngineCo and its production orthodoxy ..... 114
5.3 The Control Room ..... 118
5.4 Planning and the Contingencies of Production ..... 123
5.4.1 Plan and Reality ..... 123

5.4.2 Changing Production Layout ..... 126
5.5 Control Room Work: Doing Dependability ..... 128
5.5.1 Attending to the Normal Natural Troubles of Production ..... 128
5.6 Design in Context ..... 130
5.6.1 Information Systems Development ..... 131
5.6.2 The Plant Systems Management Group ..... 132
5.6.3 The Daily Reports System ..... 132
5.6.4 The Public Availability of Design Work ..... 134
5.6.5 Talking Design ..... 135
5.6.6 Design *qua* Member ..... 142
6 Corealisation in Practice ..... 147
6.1 Working with and on IT ..... 147
6.2 The Shift Book ..... 153
6.3 Systemwatch ..... 159
6.4 The Lift Station Case ..... 161
6.5 Ad-hoc Innovation ..... 165
7 Discussion ..... 167
7.1 The Work of Corealisation ..... 167
7.1.1 Breaking Down the Boundaries Between Design and Use ..... 168
7.1.2 Membership and Hybrid Knowledge ..... 169
7.1.3 The Use of Surrogates ..... 172
7.1.4 Prototypes, Mock-Ups and Working Configurations ..... 173
7.1.5 The Practical Politics of Design ..... 175
7.1.6 Opportunism Used Strategically ..... 177
7.2 Issues ..... 177

7.2.1 Dealing with Shifting Grounds ..... 177
7.2.2 Designing for Collaboration ..... 179
7.2.3 Designing for Skilled Workers ..... 180
7.2.4 Orienting to the Organisational Context ..... 182
7.2.5 Scale and Costs ..... 183
7.2.6 Dependability ..... 186
8 Conclusions ..... 189
9 Bibliography ..... 193
Appendix 1: Publications ..... 225
9.1 Papers in Journals ..... 225
9.2 Papers in International Conference and Workshop Proceedings ..... 225
9.3 Book Chapters ..... 227
9.4 Miscellaneous Publications ..... 227

# List of Figures

**Figure 1 Overview of participatory design approaches (from Muller, Wildman and White 1993) 74**

**Figure 2 The production plant layout (as of November 2000). Note the three areas where deliveries and pickups are made and the central location of the control room. 115**

**Figure 3 The Control Room IT systems from left to right: control host for autonomous carriers, general computing and assembly control host (via a limited number of terminal emulation windows, process visualisation system. 120**

**Figure 4 Engines stored outside the designated buffer spaces. Note the yellow line and the track marks which show the limits of the automatic carrier system. 127**

**Figure 5 The Excel version of the shift book. Buttons on the right hand side provide ways of inserting names and standard pieces of text quickly. They also act as reminders of entries that need to be made on a regular basis (e.g. about the 8 o'clock meeting). 154**

**Figure 6 The shift book application (list view, anonymised). 157**

**Figure 7 Content from the assembly control host 'cut-n-pasted' into the shift book application as text, making it searchable and retaining its formatting. 158**

**Figure 8 The Systemwatch visualisation (only top part shown). It displays, for every unit, the number of engines produced, the target for the current point in time and the target for the whole shift. 160**

**Figure 9 The visualisation for the lift station, the graphs track the number of transport orders to the lift station against the number of available carriers (red line) and the total number of carriers in the system (the flat line). 162**

**Figure 10 The lift station which feeds engines into varnishing (in the red circle). Also visible are the yellow carriers which supply engines to it from stationary assembly (to the right) and 163**

# Preface

All interpretation of this world is based on a stock of previous experiences of it, our own or those handed down to us by parents or teachers, which experiences in the form of "knowledge at hand" function as a scheme of reference. To this stock of knowledge at hand belongs our knowledge that the world we live in is a world of more or less well circumscribed objects with more or less definite qualities, objects among which we move, which resist us and upon which we may act. Yet none of these objects is perceived as insulated. From the outset it is an object within a horizon of familiarity and pre-acquaintanceship which is, as such, just taken for granted until further notice as the unquestioned, though at any time questionable, stock of knowledge at hand. The unquestioned pre-experiences are, however, also from the outset, at hand as *typical* ones, that is, as carrying along open horizons of anticipated similar experiences. (Schütz 1953, p.4)

# 1 Introduction

This thesis is concerned with the way we build and use IT systems in complex organisational settings – how we create assemblages of hardware and software so as to support the work people do. It is concerned with the relationship between systems, their design, the organisational settings they are embedded in, and the work they are built to support. As information technologies become a more and more pervasive feature of modern workplaces, understanding the relationship between these technologies and the social organisation of work, of which they are a part, becomes increasingly important for those involved in their creation, deployment and use. Traditional divisions of labour in IT systems design have tended to favour a separation of the design of information technologies and systems from their deployment, use and maintenance. To some extent this is inevitable and perhaps even desirable (from the perspective of technology suppliers) within a mature industry with relatively stable user requirements and established patterns of technology supply. After all, the aim is to maximise the reuse of candidate solutions to common problems, traded in the marketplace.

However, it leaves open the question of *adequacy* of solutions and how the specific, detailed working practices of workers can be effectively supported. Only when technologies get translated into systems, only when these get used 'in anger' and encounter the contingencies of the workplace, can one effectively assess their 'fit' with the work that gets done. This poses an important question given that 'design' and 'use' are often separated in time and space as well as being undertaken by different people with different skills, concerns and under different sets of constraints.

My aim is to look again at the social organisation of the work of building and using IT systems, to explicate the methods by which systems and working practices are elaborated and made to more or less 'fit' each other. From this starting point, I develop the concept of *corealisation* as a radical respecification of the work of designing and using IT systems based on an orientation to this project that breaks down the traditional boundaries of 'design' and 'use'. In order to locate this work in a larger context, I first wish to briefly review the history of the troubles that beset IT systems design. This then leads me to reflect on the objects of study that computer

science deals with, the ways in which it defines itself as a discipline and how it relates to practical use of information technologies and the IT industry.

## 1.1 Software crises

Following the use of early computers to calculate projectile trajectories during World War II, this new technology came to be used in the large corporations arising at the time and was heralded as the key to a new era of growth fuelled by the automated processing of information. Computers gave management the means to potentially achieve unprecedented increases in productivity and new levels of managerial control, especially of the increasing numbers of white collar workers. Early uses of computers in industry coincided with and supported the separation of jobs into front- and back-office functions. Consisting of largely routinised tasks, the latter would be assigned to temporary staff pooled in large open-plan offices (Greenbaum 1995). While the principles of bureaucratic control were not new, their automation and control through computers was a key change that enabled the further subdivision of tasks and therefore allowed managers to gain unprecedented control over details of work organisation in the office (cf. Braverman 1974).

While advanced technologies in the workplace have not led to an overall loss in task discretion as predicted by Braverman (1974), the positive impacts of automation predicted by some have not materialised either (Gallie 1996). This has less to do with inherent properties of technologies but with decisions made about their adoption and the ways in which they are applied (Liker, Haddad and Karlin 1999, Gallie 1996). Of course, it is not always in management's interest to apply control mechanisms even where this would be possible (Attewell 1987); other strategies can often be more effective and less costly or the potential gains may simply not be large enough to warrant the risks involved. For example, control by peers within teams is often more effective than direct control through technological means (Sewell 1998). Programmes advocating worker empowerment, such as flexible specialisation (Piore and Sabel 1984) or lean production (Womack, Jones and Roos 1991), depart from the earlier alienating control mechanisms of Taylorism and seek to enrol workers' inventiveness and abilities to further the organisation's aims, often using information technologies to make this move feasible.

However, while, arguably, the deskilling of work that characterised the early phases of computerisation has been replaced by a trend towards stronger task integration – the demise of data entry pools and rise of the 'knowledge worker' (Greenbaum 1988, 1995) – the intensification of work has continued. This has led to many groups of society being excluded from full-time, permanent employment while those in employment experience ever increasing pressure to become more productive in their work. The point I want to make here is that right from the start and up to the present day, use of computer systems was part and parcel of managerial strategies of work reorganisation and the politics of the labour market. Neglecting this wider role that information technologies play and focusing on their immediate function carries great risks.

As IT systems became more and more powerful and were applied to more and more complex problems, programmer productivity was soon the focus of attention as there was a lack of skilled workers familiar with the abstract logic and mathematical understanding required in the early applications of computer systems. In addition, in the early phases of computing, the tasks of developing a program, debugging it, running it on a machine and using its results were normally carried out by the same person, giving rise to the "programmer" as a highly skilled person with significant autonomy. As this job was taken over by predominantly white, male and young people with higher education degrees and a sense of being part of an avant-garde of the emerging information society, effectively managing this work under the rules of the corporation was difficult. Even with the advent of computer science as a discipline with a defined educational program in the early 1960s, there was a lack of education in what Bauer (1973) called "down-to-earth programming" as many students turned to what they saw as being more interesting fields such as artificial intelligence. The initial focus on establishing computer science as an academic discipline did little to help create a supply of skilled programmers (*ibid.*).

Not surprisingly, the principles of work reorganisation and management of the labour market that had affected office work were applied to the computing trade itself, leading to an increasing industrialisation of computer programming (Kraft 1979, Kraft and Dubnoff 1986, Greenbaum 1995). A complex division of labour was established, separating work content along similar lines as office work, which had

been separated into front- and back-office work. Job descriptions such as 'analyst', 'designer', 'programmer', 'coder' or 'operator' illustrate this vertical division of labour into jobs which differed significantly in terms of their status, the skill requirements and career structures.

However, the development of the computing industry brought with it significant problems that were not just related to problems of managerial control but were substantial and seemed to threaten the whole project. At the beginning, it was difficult for people in the industry to voice concerns but the realisation slowly sank in and at the 1968 Software Engineering Conference in Garmisch, Germany, the term 'software crisis' was perhaps not coined but openly debated for the first time (Naur and Randell 1969, Dijkstra 1972). Since then, the term has followed the computing industry wherever it went, has plagued industry and governments alike and it has been a cornerstone of academic debate ever since.

Repeated industry reports by various institutions such as the Standish Group (published biannually since 1994, www.standishgroup.com) or the National Institute for Standards and Technology (2002), reveal the lack of progress of the industry in its efforts to develop systems to specification, on time and on budget (cf. Charette 2005). With alarming regularity, these reports demonstrate that expectations of a reliable, predictable software development process are not met and that many projects fail to deliver effective, dependable or even working systems. Examples of IT systems failures include the large scale, bringing down whole national infrastructures such as the French railway system (Mitev 1996) or endangering an important public service such as the breakdown of the London ambulance system (Beynon-Davies 1995, 1999, Finkelstein and Dowell 1996) to failures directly endangering lives (MacKenzie 1996) such as the Therac-25 case (Leveson and Turner 1993), to developments that never get rolled out and 'merely' cost millions such as the Denver Airport baggage handling system (Gibbs 1994, Swartz 1996). At the moment, the UK National Programme for IT[1] (now renamed "Connecting for Health") provides a whole string of examples of projects that seem to be in serious

[1] A comprehensive programme to overhaul the use of information technologies in the UK National Health Service.

trouble even before they are rolled out (e.g., Hendy *et al.* 2005, Cross 2005). It seems that public sector procurement is particularly badly affected although that picture may be distorted because the private sector is not under the same obligations to make procurement processes transparent (cf. Shapiro 2005). Problems range from mundane coding errors to fundamental misunderstandings of the application domain (e.g. the London Ambulance case) or its potential unboundedness[2]. At the same time, projects are getting more and more ambitious and systems more and more complex as few systems these days are isolated ones but are interconnected as the drive for systems integration increases. Software projects are often subject to outside pressures such as politically motivated deadlines or cost-cutting exercises. The range of potential sources of trouble is vast. In addition, software is intrinsically fragile. As Charette (2005) notes:

> In a large brick building, you'd have to remove hundreds of strategically placed bricks to make a wall collapse. But in a 100 000 line software program, it takes only one or two bad lines to produce major problems. (*ibid.*, p. 47)

Despite repeated promises that the software crisis would be solved by some new method or technology, it has turned out that each partial, candidate solution only uncovered new troubles as it contributed to pushing the boundaries of IT use ever further. While initially the focus was on the productivity of individual programmers and their ability to intellectually comprehend programs and their execution (Dijkstra 1972), the focus soon shifted to the problem of supplying the industry with sufficient qualified staff, to the problem of managing the development of software 'in the large'. The solution to the previous problem, i.e. the establishment of a complex division of labour involving many people, had become the source of the next (Kraft and Dubnoff 1986). The focus shifted to the problem of effectively coordinating the various activities involved in building complex systems, breaking down problems in a way that the partial solutions could be effectively combined again to form an overall solution. The separation of "hand from head" inherent in the roles of 'analyst' and 'programmer' gave rise to problems of communication:

---

[2] Lehman (1998) notes the case of a CERN particle accelerator that was twice the size of its predecessor and failed to function as expected until someone noticed that the results were influenced by the phases of the moon.

> Because programmers weren't able to talk to customers directly, they found that they were coding and recoding the same programs in an unproductive attempt to make them do what was needed. But by splitting off analytical work, programming jobs were routinized and salaries were reigned in. (Greenbaum 1995, p 66)

While it became apparent that the division of labour was dysfunctional in this respect, the cost savings and managerial control achieved by it were enough to keep the system of work organisation stable. Various devices were invented at the time to address the problem, to improve the communication between analysts and programmers by introducing representations such as flow charts and data flow diagrams and emphasising the role of the requirements document in the systems development process. The 'rise of methodology' was brought about by the realisation that systems development did not scale very well as the coordination effort involved in managing increasingly large teams of IT professionals grew out of proportion (Brooks 1975). Projects often ran late and over budget and adding more staff just added to the problems, leading Brooks to formulate his law that "adding manpower to a late software project makes it later" (*ibid*. p. 25).

### 1.1.1 The Computer 'Reaches Out'

In the 1970s IT systems became more and more organisationally embedded and advances in electrical engineering and hardware design made unprecedented computing capacity available. In the 1980s the advent of personal computers brought IT to the 'front-office', directly exposing workers with relatively high status to the technology for the first time[3] (Greenbaum 1995, p. 68 ff.). For many people, use of computer systems was no longer indirect, mediated by lower status intermediaries but immediate. In addition, the ready local availability of computers which could be used interactively gave rise to new classes of systems that supported less well-defined tasks compared to earlier systems which dealt mostly with high-volume routine transactions such as payroll operations. Systems such as spreadsheets and word processors supported a wide range of business activities without being specifically designed for particular uses. While previously working practices were

[3] Additionally, the separation of tasks into smaller and smaller units had reached a limit and work dissatisfaction was rife, so new forms of work organisation were explored (Greenbaum 1988).

redesigned to fit and be enforced by IT systems, now the issue became more one of fitting systems to the working practices and skills of the new 'end user' or 'discretionary user': "people who saw themselves as having a job or profession that was not primarily geared to the computing medium itself, but who used it directly as a tool in their everyday work" (Bannon 1991, p. 32). Human-computer interaction emerged as a field studying the 'human factor' in computing or the way that people and computers 'interacted'.

With the increase in interactive use of IT systems and their application to more and more problem domains, issues "beyond the basic perceptual and motor processes" (Grudin 1990, p. 262) became important. Problems surrounding the definition of requirements for IT systems, the 'fit' of the technology to the workplace and wider organisational arrangements came increasingly into focus as did the relationship between technology supply and use (Friedman and Cornford 1987, 1989). The increasing availability of computer networks in the late 1980s and early 1990s as well as the increasing ubiquity of computers gave rise to a new class of systems, called 'groupware' that were specifically designed to support collaboration and brought with them a number of new challenges (Grudin 1994). Groupware was mainly designed as a new class of shrink-wrapped applications built by people who had previously been concerned with single-user applications. They were ill prepared to deal with these challenges, specifically the problem of understanding the complex, heterogeneous and evolving work environments (now extending beyond the single user – computer dyad) these systems were aimed at (*ibid.*).

In response to this, new (sub-)disciplines emerged in the mid-1980s that were related to the problem of understanding IT within its wider context. Within software engineering, the issue of requirements engineering received increased attention, and computer supported cooperative work (CSCW) emerged as a discipline that sought to understand how IT systems were used not just by individual workers but by multiple workers collaborating within an organisational division of labour. Ethnographic studies of workplaces were undertaken in this context both with a view to establish a body of knowledge of IT use in collaborative work and with a view to more directly inform the design of particular systems (cf. Plowman, Rogers and Ramage 1995). At the same time, participatory design (PD) gained wider prominence and was taken up

as a means to involve users in the requirements process. PD had originally been developed as part of an attempt by trade unions in Scandinavia to influence workplace relations by influencing technological design – thereby countering managerial uses of IT to redesign workplaces. The methods developed for the involvement of users in IT systems development were of more immediate use within software engineering as they could be used as a means for requirements elicitation in more-or-less traditional systems design processes and the original political ambitions of PD receded into the background to some extent. I will discuss CSCW and PD in more detail in chapter 3.

### 1.1.2 An Ongoing Sense of Crisis

However, despite the significant effort invested and the various ways that were developed to 'inform design', to control the development process, to provide various representation of IT systems and their context and to subject them to review and formal verification, a large percentage of systems development projects still fail outright, are not used effectively or suffer from various dependability problems. While the common causes of project failure are well known (Curtis, Krasner and Iscoe 1988) and documented and while candidate remedies exist for many of them, the success rates remain low and the costs of failure unacceptably high (e.g. Charette 2005). While more software is being created these days than ever before and often successfully so, our increasing dependency on information technologies has heightened our exposure to IT-related risks and therefore pushed IT systems dependability up the agenda (Neumann 1995, Center for National Software Studies 2005). Some would point to a lack of uptake of software engineering 'best practice' and a lack of professionalism in the industry (Abran *et al.* 2004, Royal Academy of Engineering 2004) as a main contributing factor to project failures, i.e. while the means are there, they are not applied properly. The recommendations resulting from such a point of view usually relate to issues of training, professional development and accreditation.

While I would not argue with the general case made, I believe that there are more fundamental problems which are yet unresolved that can *not* be tackled using the traditional methods of software engineering. *In fact, I will argue that getting to grips*

*with these problems will require quite a radical rethinking of what computer science as a discipline is about and how the business of 'doing IT' is run.*

## 1.2 Computer Science's Objects

Computer science is traditionally concerned mainly with the processing of information in an automatic manner, i.e. through machines or other mechanisms but not in a way that involves direct human action. At its core, computer science has its own internal logic and consistency. It does not face difficult epistemological questions (as evidenced by the fact that very few computer *science* courses include modules on the philosophy of science). One might say that the existence of the objects of study in traditional computer science and their nature is unproblematic. Formal modelling techniques are amongst the most important tools of the trade and a lot of effort is expended to develop and exploit them to their fullest extent. The relationship between computer science's objects and real-world phenomena, however, remains largely unexplored and introductory courses in computer science are still focused on computers, representations and their formal manipulation while the topic of what people do with computers is deferred to advanced courses in areas like human-computer interaction, computer supported cooperative work or computers in society[4]. At its heart, computer science is still very much about the computer and its internal logic rather than the real-world concerns and projects to which computers are applied.

The use of information technologies in more and more complex and less formalised domains and, especially, the advent of the computer on desktops and its use as an interactive tool have led to a shift of focus from programmer productivity and reliability of the technology itself as the primary concerns, to the relationship between design and use (Friedman and Cornford 1989, Grudin 1990). This change in focus brought with it a whole range of issues previously not encountered in computer

[4] Personally, I have studied computer science in Erlangen, Germany, for eight years and have never heard anything about the role computers play in what people do until I started taking courses in organisational psychology, workflow management and CSCW towards the end of my studies. Not that these courses paid much attention to activities of actual people, though – but at least the notion of people doing work entered the picture even if they were reified as abstract actors in workflow diagrams.

science and therefore poorly conceptualised in its traditional canon of orientations, approaches and methods. Consequently, computer science has started to interact with disciplines from the social sciences as well as areas of philosophy previously ignored. These cooperations have been most successful where there were similarities in concepts and methods and alignment was therefore relatively easy. While this engagement has let to the establishment of a number of interesting fields of study at the fringes of computer science, it has not led to a reconsideration of the definition of the field (despite efforts to reinvent it as Informatics).

It has also not led to an end of the "software crisis", to an understanding of information technologies that are "in working order", matching the needs of the people who work with them. As the examples of IT system failures cited above show, these great and wonderful achievements of interdisciplinary work are paired with a seeming inability to design relatively mundane systems that are fit for purpose, support working practices and don't get in the way. Some seek a solution in the improvement of the formal models underpinning a system's design. According to them, current modelling techniques are inadequately equipped to model complex phenomena and thereby communicate what needs to be known about them in the design of a related system. This approach basically seeks a solution to the problem within the domain of the representations themselves or in terms of the *completeness* of the model and its match with real-world phenomena. The *nature* of the relationship between a real world phenomenon and its representation is treated as unproblematic. It is taken for granted that the real world 'functions' in the same basic ways, according to the same kinds of logic, as the models and that the models therefore capture the *essence* of what's going on in the real world – an assumption worth looking at more closely. Zemanek (1972, 1975) has done this as early as 1972 and has suggested a "philosophy of information processing" based on the realisation that the real-world problems that we face every day and our actions are not reducible to formal constructs. Zemanek points out that the gap between real worldly phenomena and computer representations is unavoidable and calls us to consider the consequences. I shall come back to this issue repeatedly throughout this thesis but for the moment suffice to say that in failing to attend to the nature of representations –

which I contend can be found by studying their *uses* – computer science significantly limits its horizon and thereby limits what can be achieved.

Representations are the means by which computer science and the computing industry achieve their aims and their manipulation is the object of study in computer science. Graduates from courses in computer science are readily familiar with representations and have learned numerous ways to store, manipulate and visualise them. They are, however, much less acquainted with the world these representations describe and questions about the relationship between one and the other. What they are lacking is an understanding of practical IT systems usage in real-world settings and over a period of time. How does IT design and usage happen over time and how does it relate to other phenomena in the real world? For example, does the workflow as modelled in the system describe what people have actually done? Are all the engine parts where the stock control system records them? Is the quality of the data recorded in a disease register sufficiently high to warrant certain claims about public health? Will the new hospital information system support the work of the various people involved in patient care as well as meeting the requirements of administration? It is these kinds of questions that are truly vexing. They are complex and involve phenomena that are in themselves difficult if not impossible to capture in formal representations.

In summary, it is important to note that computer science's objects are *representations and their transformation* rather than the real-world phenomena that natural or social sciences would be concerned with. In some sense, computer science is very similar to mathematics in that it deals with what has already come to be known, what has already been divorced from its real-world condition. Computer science, as a science of models, is itself internally consistent but becomes problematic in its application and real-world uses. It is where the representations that make up IT systems are connected to real-world phenomena that problems arise.

## 1.3 Accomplishing Computer Science's Objects

While computer scientists traditionally trade in a world of known objects, the practices through which objects come to be known and thereby *relevant*, escape their attention. In this way, computer science can be seen as being *incomplete* and in need

of a *complementary* extension. This extension, which we might call a practical epistemology for design, would be concerned with the *accomplishment* of computer science's objects and with the establishment of social *order* in systems development.

It is worth noting that for designers, these problems are of an entirely *practical nature* and are in principle amenable to *practical remedies*, i.e. they can, at least in principle, be solved for all practical purposes and in any specific situation. This sets them apart from the principled problems of epistemology that the social sciences (and with them computer science) face. However, and crucially, the problems mentioned are *endemic* and not amenable to a generic remedy. That is, they constitute a core *Problematik* in design work rather than merely a 'problem' that might eventually find its 'solution' in the sense of some technique or technology. However, as I will argue in this thesis, a practical way to come to terms with and address the problem *can* be found in the designers' orientation and their approach when faced with real-world phenomena.

It is ironic that communities studying IT systems development and use as activities embedded in real-world settings have developed *on the fringes* of the discipline. In order to understand the character of design and use, one has to attend to their real-world character which cannot be studied in laboratories. This orientation is most notable in research communities such as those studying computer supported cooperative work and participatory design. Researchers from these communities have made it their aim to investigate technologies *in context*, that is to study the relationship between technological artefacts, their design and their use in specific circumstances. Underlying this turn is a particular understanding of how we come to know about the world: "the real world of things concrete only exists *for human beings* as a result of human practice, and as practice changes so does our understanding of the real world of things concrete" (Crabtree 2004, emphasis in original). What does this mean? It means that we can know the world only by engaging with it. The best way for designers to know about the relationship between the representations and what they represent is to encounter both of them together. This can be achieved in a number of ways which I will discuss later on.

I will argue that at the heart of the problem lies a 'missing how', a lack of appreciation by most computer scientists and IT professionals of how people go about their everyday work, be it designing IT systems or using them. While this 'how' is sometimes captured in ethnographies produced as part of academic studies, it has proven to be difficult to bring into the design of technologies (e.g., Plowman, Rogers and Ramage 1995, Hughes *et al.* 1992, Hughes *et al.* 1995) as it is very difficult to distil requirements from these rich descriptions of practice. Various approaches to *informing design* have been developed that aim to address the problem of linking IT systems design and use by either bringing users into the design process in a stronger way (participatory design) or by making representations of use practice available to designers (ethnography for design). In effect, these existing approaches "build bridges" between the separated activities of 'design' and 'use' and the people involved. Arguably, they therefore fail to overcome the boundaries, especially the separation in time and space, which is at the heart of my argument.

## 1.4 Corealisation

I present a new approach to IT systems development called *corealisation* that aims to break down the boundaries between technology design and use (Suchman 2002) by bringing the systems designers and therefore the design process into the use situation. IT systems development is reconceptualised as an activity shared by IT professionals and other workers, located within the unfolding biography of a setting. This longitudinal engagement with the setting allows a radically different practice that encompasses traditionally separate activities and thereby breaks down boundaries not only between 'design' and 'use' but also between the various kinds of IT work such as 'analysis', 'design', 'programming', 'operation' and 'maintenance'. By taking seriously the situated practices of working on and working with information technologies, a new, non-prescriptive approach (with respect to the method) to practical systems development is established. It is this practical orientation to IT systems design and the related practices (the approach) that are the subject of this thesis. A case study of corealisation work is presented to illustrate how the observable features of work can provide a resource for systems design and to show how designers can make use of these resources to develop systems that are in

'working order'. The aim of the thesis is to formulate the principles of corealisation and to identify a programme of research.

## 1.5 Chapter Outline

Following this introduction, the second chapter of this thesis discusses a number of issues in IT systems design and the streams of academic research that have evolved around them. I will briefly outline how systems developers have traditionally gone about their business, the process models they have developed and one of the key problems they have encountered: evolution. As I am interested in systems in complex organisational settings I then discuss the nature of work in such settings which is usually characterised by various forms of collaboration between members of organisations. This changes the focus from the computer-user dyad to design for socially organised work. Having characterised the work people do, I turn to the notion of 'users' and 'designers' and the gap that has developed between them. It is this gap that underpins a number of problems we find with systems development and will be at the heart of my respecification of systems development practices. As systems development work itself is a collaborative activity, the principles applied to the study of collaborative work settings can be applied to it as well. Following on from this, I address the issue of innovation which is central to design activity. A discussion of what we can mean by the concept of dependability again shows how a consideration of IT systems in context leads to a shift of focus away from the technology itself to a concern with its role in real-world settings.

Having looked at the broad themes in IT systems design research, I discuss various bodies of literature, the work of various academic communities that have made these themes their object of study. Ethnography for design is an attempt to enlist social science methods as a way to bridge the gap between 'design' and 'use' by creating representations of work to inform design. Participatory design takes a different approach in that it aims to involve the 'users' themselves in the design project, often with a view to empower them, to give them more control over their working practices and environment. Contextual design promises to combine elements of ethnographic approaches with participatory elements to form a consistent approach that blends well with the practical needs to the design industry. Finally, I will discuss

configuration, tailoring and end-user programming as alternative ways of meeting the needs of workers.

In chapter four I introduce the notion of corealisation as a respecification of IT design work. First, I introduce ethnomethodology, a social science approach to explicate the methods that society members use to recognise and produce social order and to thereby conduct their everyday, ordinary affairs. Corealisation draws on insights from ethnomethodology both in terms of its underlying principles and in terms of the practical orientation it provides for designers: to take members' methods (i.e., the ways in which people go about their everyday business) seriously and use a familiarity with the way things 'get done' as a resource in systems development. In order to provide a basis for the discussion of corealisation, I discuss the notions of accountability, reflexivity, indexicality, situatedness, *ad-hoc*ing, the documentary method and membership.

The subsequent discussion of corealisation's programme builds on the themes introduced in chapter two and the approaches and orientations discussed in chapters three. The relevance of the orientation to studying the social world and the insights provided by ethnomethodology will be discussed and taken as a basis for a respecification of systems development as a ongoing process of corealisation work jointly undertaken by IT professionals and people who work use IT systems to do their work. There are three crucial elements in this respecification: first, design *qua* member, the idea that system design can and should be undertaken on the basis of a member's understanding of the setting (rather than on the basis of some external body of knowledge). An important aspect related to membership is the notion of hybrid knowledge, the requirement that IT professionals should be familiar with the practices of people in the setting within which they do their work and that they should use this knowledge as a resource in design. The core aim of corealisation is to produce 'uniquely adequate' systems, i.e. systems that afford working practices rather than reengineering them. From this aim follows the use of uniquely adequate methods, that is the maxim that the course of action taken should be chosen in correspondence to the situation at hand. The last section spells out corealisation's principles in a programmatic manner, so they can be easily referenced.

Chapter five explores the case of work at a EngineCo, a manufacturer of Diesel engines. Starting with a description of the setting and of work in the control room of the production plant studied, I then move on to look at work on and with information technologies in this setting and seek to explicate the role that members' knowledge of and familiarity with working practices plays in developing systems that are uniquely adequate for the tasks at hand. Following from this, I describe the implications of using corealisation as a practical orientation in and approach to IT design work. This chapter draws on my experiences in corealising IT systems at EngineCo. Throughout the discussion of the fieldwork material I will seek to relevance the findings to the project of corealisation.

The following chapter is based on the fieldwork material and aims to discuss various issues relating to the use of corealisation as a practical approach to systems development. I seek to demonstrate how corealisation addresses the issues defined in chapter two and how it builds on the previous work presented in chapter three. The important question of the scope and limitations of corealisation will also be discussed before I draw some general conclusions in the final section.

# 2 Issues in IT Systems Development

Developing IT systems involves more than technical work but is rather part of a larger socio-technical[5] domain of relations between technologies, people and organisational arrangements. The process of development itself is one that is not just straightforwardly technical but involves the alignment of various parties to the aims of the development project. Likewise, the subsequent use of the system developed takes place within a setting comprised of social relations and arrangements as well as technical parts. This chapter discusses a number of important strands of academic research as well as practical reasoning going on outside academia related to the sociality of IT systems development and use.

The first section discusses systems as necessarily evolving entities and the implications this has for systems development, operation and maintenance. I then discuss the relationship between information technologies and their use in larger social contexts where work is done not by individual users but collectively by a number of people collaborating more of less closely. Following this I turn to the issue of user-designer relations, problematise the distinction between 'design' and 'use' (and 'designers' and 'users'), and bring into focus the divisions of labour in IT systems development. I describe design itself as a socially organised activity and discuss the relationship between design's methodologies and practices. An important part of designing systems is the step of going beyond existing arrangements and envisaging what future practice might look like, how it might be arranged. This is the topic of the sixth section. As IT systems are a powerful means to enable as well as shape working practice, I will introduce the question how their design might impact on labour relations and what kinds of conflicts their introduction might raise or impact upon. The final section aims to establish a foundation for what we might understand as a system being dependable, how people might reasonably trust a system and what it means for an IT system to be a success by some measure. The

[5] On the various uses of the term 'socio-technical' and design approaches developed on this basis, see (Olerup 1989). In the context of this thesis, I mean to point to the fact that technical arrangements are reflexively tied to a range of organisational and societal arrangements without which they would not make much sense. This simple definition is sufficient for my purposes and the use of the term should not be seen as an alignment with any particular tradition of sociotechnical design.

basic underpinning developed here will form a basis for the rest of the thesis and will ultimately provide the yardstick against which the approach developed in this thesis will have to be measured.

## 2.1 IS Methodology: Prescription, Process and Evolution

No matter how well we design a system to match a set of requirements, there will always be a need for change. First, our understanding of the situation into which a system is to be introduced will inevitably be limited, bounded by our limited experience and subject to certain assumptions we necessarily make. Second, the introduction of the system itself will give rise to new requirements being formulated as people learn more about the potential uses of technologies and opportunities to change practices around the new socio-material arrangements. Finally, the situation of use changes constantly as the world keeps turning. We might say that requirements are a 'moving targets' and that change is an inevitable part of systems development[6].

Early approaches to systems development following the waterfall model (Royce 1987 [1970]) were designed to support a strict progression from analysis and requirements specification through design to implementation. Such models relied heavily on the formalisation of design documentation and the handover process that interfaced the phases of the development process. The danger inherent in such methods is that they get followed to the letter in a highly bureaucratic manner and end up getting abandoned because of the overheads involved and the unresponsiveness to circumstances. Use of systems development methodologies can become more of a ritual than a practical means for accomplishing work and can even be detrimental to the overall project (Wastell 1996). While systems development methodologies provide the means for management to gain (at least a sense of) control over systems development projects, they often fail to support adequately the work involved in doing the project's work, especially when they are followed rigidly. In the face of these problems, many organisations start 'tailoring' the processes to their

[6] This is the case for any kind of development although in some circumstances it may be able to control change, but this can be done only to an extent.

own needs, prioritising the aspects that are seen to add most value to the task at hand and picking the methods and tools that are readily appropriated (Wastell 1996, Sharma and Rai 2000). The common conception that methodologies can act as a vehicle for 'best practice' that can replace experience and skill and can provide a template for practice that can be readily adopted is highly questionable (Russo and Stolterman 2000, Nandhakumar and Avison 1999) as it ignores the work that goes into appropriating methods and tools and using them practically, according to the purpose at hand. I will discuss this issue further in section 2.4.

While linear systems development methods can be made to work even in the face of evolving circumstances, there is still a tension between the method's prescription and its practical use. In the 1980s it became increasingly clear that the linear approach to systems development did not work well in many circumstances. Consequently, models incorporating feedback loops, circular, iterative, evolutionary models (e.g., Boehm 1988, cf. Sommerville 2001) started to become popular. While retaining the notion of a phase and therefore addressing the problem of controlling the software process, they introduced the notion that systems development was a learning process and that what was learnt would have to be made available and factored into the design. The main rationale for the introduction of iteration into the software development process was the recognition that initial specifications were imperfect, that there was an inevitable drift between the static requirements specification and the changing world and that repairing these problems early saved effort and money. This realisation that software needs to evolve to meet the changing needs of a society that increasingly relies on it has led to the formulation of software evolution as a distinct field of study (Lehman 1998). Faced with the inherent incompleteness of finite systems in an unbounded domain[7], software evolution recognises that building a system always involves assumptions and is based on a leap of faith that the knowledge at hand is good enough for the practical purposes at hand. However, this is subject to potential future revision, hence the need for software to change. As Lehman puts it:

---

[7] Lehman (1998) observes that to be complete, a system would have to contain a model of itself and its operation in the problem domain. Obviously, a finite system cannot contain such a model.

> Code can be made flexible only to the extent that programmers specifically recognize uncertainty or a possibility of change and incorporate the appropriate tolerance, responsiveness, and switchable new mechanisms into the system's logic and its textual implementation. (Lehman 1998)

More recently approaches to systems development have emerged under the rubric of 'agile' or 'lightweight' that are explicitly aimed to support systems development in a changing world. Some of them have become quite popular as an answer to the rigidities of traditional phased design methodologies, the most notable being extreme programming (Beck 1998, 2000). Extreme programming (XP) focuses on the minimisation of risk during the systems development process and employs strategies that are often opposed to traditional systems development and project management wisdom. Key elements are a focus on working code involving early release and short release cycles, an incremental planning approach that allows changes to be made according to changing circumstances, a reliance on automatic testing tools to ensure quality and on tests, code and communication to represent system structure and intent, as well as a commitment to ongoing design throughout the system lifecycle (Beck 2000, p. xvii). By delivering value and spotting problems early and by emphasising simplicity, XP promises to deliver high quality systems and flexibility in the light of changing needs.

While agile methods like XP have become quite popular, it is not clear how well they scale and integrate with other methods (Boehm and Turner 2005, Turk, France and Rumpe 2002). It remains to be seen whether they represent merely a fashion or a rethinking of traditional, top-heavy systems development methodologies that will reshape the ways we build systems. In contrast to previous promises, XP at least seems to be a key departure in that it is not overly prescriptive and actively encourages designers to fit the method to their specific needs or to consider alternative approaches if the conditions are such that they would not allow XP to work. It would seem that agile methods at least provide occasion for the software engineering community to consider a new way of socially organising development work and relationships between IT professionals and their 'customers'. This consideration of practice will hopefully lead to a less fashion-driven, more considerate approach seeking to match methods to the problem at hand as well as the larger organisational context. There might be some resistance though on the side of

IT professionals to what might be seen as yet another change programme (cf. Nerur, Mahapatra and Mangalaraj 2005).

## 2.2 Design for Collaborative Work

The field of computer-supported cooperative work (CSCW) has emerged in the mid 1980s and early 90s both as a technical discipline concerned with exploiting networking technologies to enable computer-mediated communication and the development of distributed systems and as a discipline concerned with the sociality of activities involving computer systems. The latter aspect was a reaction to the way in which people and their activities were conceptualised in earlier work on 'human factors' and is perhaps best summarised by Bannon (1991):

> Part of the problem resides in an implicit view of ordinary people which, if surfaced, would seem to treat people as, at worst, idiots who must be shielded from the machine, or as, at best, simply sets of elementary processes or "factors" that can be studied in isolation in the laboratory. […] Understanding people as "actors" in situations, with a set of skills and shared practices based upon work experience with others, requires us to seek new ways of understanding the relationship between people, technology, work requirements, and organizational constraints in work settings. (Bannon 1991, p. 25)

His call to move on from "human factors" to "human actors" places the active worker at the heart of research and practice rather than the computer system with people as components that are somehow attached to it. Thinking about people using computers to do their work also brings their activities into focus and we find that they are normally collaborative activities involving a working division of labour (Anderson, Hughes and Sharrock 1989, p. 159).

Schmidt and Bannon make the point that collaboration[8] is not restricted to 'group-work' but is a feature of all work activities and that one should not restrict the meaning to a particular form of activity within particular work arrangements. Rather, it is the sociality of work in general that is of interest (Bannon and Schmidt 1991, Schmidt and Bannon 1992). It is therefore ironic that CSCW has developed at the fringes of computer science as it addresses a central rather than a marginal topic.

---

[8] I will use the terms 'collaborative' and 'cooperative' interchangeably to denote any socially organised work undertaken within an organisational division of labour.

The orientation to work as inherently social allows one to treat both collaboration in the sense of cordial work relationships and conflict as aspects of the same phenomenon, i.e., the production of social order. That is, seeing work as essentially collaborative does not mean that one negates the possibility of conflict (cf. Kling 1991) but points to the socially organised ways in which most conflicts normally get managed (if not resolved). In any real-world setting, one will find that people engage in collaborative work in that they orient to what others are doing, organised their work so that others can in turn orient to it (i.e., they make their work accountable, see section 4.1.1), they orient to their responsibilities within the organisation, appeal to others to discharge their duties, voice concerns and objections, etc. It is this wider definition of what constitutes 'collaborative work' that I'm interested in and that I will use in the following.

People routinely make decisions as to which aspects of their work to present to their fellow workers. That is, they will make the results of their efforts available but do not normally need to reveal details of their accomplishment (Suchman 1995a, Schmidt 2000). The observation that work is often 'hidden' is therefore not a surprising one[9] but it is interesting in terms of the implications for design and the extent to which requirements can be 'read off' various accounts of how work gets done. For example, Blomberg, Suchman and Trigg (1996) found that work in the litigation support department of a law firm was characterised as highly routinised and involving very little knowledge about legal matters. Subsequent observation of this work then revealed various practices that required significant understanding of how legal documents are organised. The authors found the distinction between routine and knowledge-intensive work unhelpful and suggest instead that "routine activities and the exercise of judgement coexist at all levels of the organization hierarchy" (Blomberg, Suchman and Trigg, 1996, pp. 255).

[9] This is not to say that people are involved in actively 'hiding' their work but that they will not normally put extra effort into conveying the details of their work to fellow members – who in turn will not *normally* have a need to learn about the accomplishment of the work in so much detail.

### 2.2.1 Plans, Procedures and Situated Action

In her book on "Plans and Situated Actions", Suchman (1987) took issue with the common conception that, in acting, people were following *plans* and that it was merely the *execution* of these plans in a particular context that constituted human activities. She formulated an opposing view arguing that the situation at hand was a crucial resource that people drew on to work up a course of action in and through their engagement with the setting. Plans, such as those 'embedded' in a photocopier's help system, do not in any strong sense predetermine peoples' actions but are rather available as a *resource* that may be oriented to by people (Suchman 1987, p. 100). Suchman's position has often been mischaracterised as negating the existence of plans, as claiming that there were only 'situated actions'. However, she has been at pains to point out the reflexive relationship between plans *and* situated actions, the fact that plans come about and are realised through situated actions. Her aim was to offer for consideration "*the relationship between the activity of planning and the conduct of actions-according-to-plan*" (cf. Suchman 2003, p. 301, emphasis in original).

There has been much debate about the relationship between plans and situated actions in the CSCW community (e.g. Suchman 1994, 1995, 2003, Winograd 1994, Schmidt 1997, Sharrock and Button 2003, as well as various comments in *Computer Supported Cooperative Work*, vol. 3, 1995)[10]. However, I do not wish to recount the debate as I believe that it is based largely on a fundamental misunderstanding of the position taken by Suchman (and others)[11]. The point is to treat plans as a *phenomenon* and to make their practical achievement the object of study, rather than treating people as 'judgemental dopes' (Garfinkel 1967, p. 66 ff.) whose actions are predetermined by the stable structures of society (or stable structures of cognition). I will say more about this orientation in my treatment of ethnomethodology in chapter 4.1. For the moment, suffice to say that the treatment of plans as structures that

[10] Note also the earlier debate in HCI on how to best conceptualise the human-computer dyad (Card, Moran and Newell 1983, Newell and Card 1985, 1986, Carroll and Campbell 1986).

[11] With regard to the politics that played a role in the debate (cf. Harper 1995), I wish to remain methodologically indifferent (i.e. not as an individual but as a researcher, see section 4.1 and 7.1.5).

predetermine peoples' actions has been shown to be problematic, i.e. worthy of study. As Sharrock and Button (2003, p. 260) point out:

> [The plan] is a device for the occasional organization of courses of action. [Cognitive science] has assigned it a theoretical role, one which it cannot fulfil, as a general mechanism which underlies the organization of all courses of action. Rather than a basis of action, a plan is a technique for the organization of action.

Consequently, rather than suggesting that plans do not exist, Suchman's work and that of others points to the ways in which they are practical accomplishments, how 'working up' a plan and acting in the light of its existence is a common feature of social actions. The finding that not all action is organised according to plans does not imply that no actions are organised in this way (Sharrock and Button 2003).

Of course, 'plans' exist at a number of levels and for a number of different purposes: my 'plan' to go and get a sandwich for lunch differs from organisational 'plans' in that the latter involve different kinds of commitments. While it seems problematic to say that my behaviour is produced through the existence of either of those plans, there is a sense in which organisational plans 'determine' my behaviour in that my activities are organisationally accountable, i.e. I have a certain responsibility to act in accordance with those plans. But even in this case the relationship between plans and actions is one of 'because-I-choose-to-comply' rather than 'how'. It is also worth noting that the relationship is an entirely practical one and is thus in contrast to the action determining 'plans-according-to-cognitive-science' (Sharrock and Button 2003). The crucial point is that plans are a resource, furnished by people for practical purposes and eminently revisable in the light of circumstances. Working out details of what a formulated plan can be taken to mean in a particular situation (e.g. Suchman 1983) and changing plans where the need arises (e.g. Rönkkö, Dittrich and Randall 2005) are common features of everyday life.

The meaning of procedures in any given situation needs to be established by first establishing if a particular rule applies and then working out the details of what following the rule might mean in the here-and-now (Suchman 1983). Designers often attend to the formal organisation of the workplace such as office procedures and the formal division of labour or work with the superficial understanding of the workplace that can be gained through a short visit, some interviews or meetings. The

danger, then, is that the very things that workers attend to in their work as important features get overlooked and are 'designed out' (e.g., Bowers, Button and Sharrock 1995). This is where the problem of plans and situated actions becomes a practical one, where system designs are influenced by misconceptions about the nature of peoples' activities.

Consequently, we need to attend to the use of plans as practical methods that people use to organise their actions and we need to consider plans as a part of how work is socially organised. Schmidt and Bannon (1992) describe some of the roles that plans can play as practical devices:

> […] a procedure may also convey information on the functional requirements to be met by the process as well as the product; it may highlight decisional criteria of crucial import; it may suggest a strategy for dealing with a specific type of problem (e.g., which questions to address first?); it may indicate pitfalls to avoid; or it may simply provide an *aide memoir* (such as a star procedure for a power plant or an airplane). Also, a procedure may express some statutory constraints where non-compliance may evoke severe organizational sanctions. More often than not, a particular procedure will express, in some way, all of these different functions. (*ibid*., p. 26, emphasis in original)

In summary, what is important is not to deny the existence of plans, nor to elevate them to any status other than that as a device occasionally used to organise activities.

### 2.2.2 Knowledge and Skills

Related to the problem of plans and situated actions is that of knowledge and skill. By locating the cause of action in plans, one locates the relevant skills in the drawing up of plans and therefore associates it with the person who does the planning. In contrast, an orientation to action-according-to-plan (Suchman 2003) as an achievement, one locates the skill in the practical accomplishment of a course of action that can be said to be 'in accordance with' a plan. Indeed, one can find knowledge and skill in *any* action, whether or not it is taken according to some form of plan.

Computer science has often sought to 'embed' relevant knowledge and skill in a technical system, either to automate or to give 'expert' advice. Additionally, the problem of establishing requirements for the design of systems is often associated with notions of how work 'ought' to be done, according to some concept of 'best

practice' which is often 'owned' by management (on 'owning' knowledge cf. Sharrock 1974). This emphasis on knowledge reified as technical expertise has traditionally been the basis on which computer scientists have pursued their work of design. Consequently, there has been a lack of appreciation of the skills and knowledge involved in practical, situated action. The boundaries between design and use coincided with the perceived boundaries between those who had the relevant (prescriptive) knowledge and those who did not. As Suchman (2002) notes:

> A crucial assumption underwriting these persistent boundaries is the premise that technical expertise is not only *a necessary*, but is *the sufficient*, form of knowledge for the production of new technologies (*ibid*. p. 93, emphasis in original).

Similarly, Anderson argues that:

> The artefacts we design should not subvert the practical logic of the routine world in the name of the rational view from nowhere. (Anderson 1994, p. 178)

Much work in HCI has focused on the novice user (Bannon 1991) rather than the skilled user who works with a system every day. HCI research usually takes place in laboratories under strictly controlled conditions where peoples' ability to demonstrate competencies is limited to their 'performance' of prescribed tasks, thus reducing their skills onto the dimensions of a single variable.

More recently, and notably in the areas of CSCW and PD, the concepts of knowledge and skill have been redefined and researchers have started to uncover the skills involved in the accomplishment of everyday work. In particular, studies have also uncovered "hidden skill in office computerization" (Clement 1991, also Wynn 1991, Green, Owen and Pain 1993, Suchman 2000), that is, they have found people to be more than passive recipients of technologies. Office workers were found to be involved in creating information systems and related working practices by building configurations of the technologies and other resources available to them. It is this realisation that has led to the discovery of the 'user' as a source of requirements or, as an active participant in the design process. The latter orientation treats workers as a knowledgeable and *interested* party in design (see the discussion of participatory design in section 3.2).

However, the interest workers show in technology is usually a practical one, i.e. they are not interested in it for its own sake but only insofar as it helps them to solve the practical problem of conducting their day-to-day work. As Ehn (1988a, 1993) points out, while workers are unlikely to have much understanding about computers *per se*, they are much more familiar with a particular system, particular artefacts and arrangements that have been developed to be used by them in their everyday work. The technology itself is merely a blank canvas and it is the picture, however partial and sketchy it might be, that a worker can recognise and make sense of. This is where the value of mock-ups or prototypes lies (see 3.2.4). As soon as there is enough of a resemblance will workers be able to bring their knowledge to bear and will engage with the design process.

This view of workers as competent members of the work setting with an interest in IT (as far as their work is concerned) has led to the formulation of a reconceptualisation of design work as a creative and communicative *process* (e.g., Ehn 1988a, Floyd 1987, Greenbaum 1991) rather than one where outside knowledge is embedded in IT systems (the product) in order to prescribe working practices.

### 2.2.3 'Success' and 'Failure'

If workers are not passive recipients of technologies defined in terms of a well-defined corpus of knowledge, how can one determine what designs will be successful? It may well be difficult even after the fact to clearly determine the 'success' or 'failure' of a particular project (Bowers 1994, Bowers, Pycock, Rodden and Dean 1994, Blythin *et al.* 1997). A successful technical implementation need not be a success with respect to the organisation's aims and *vice versa*. In addition, there will often be 'winners' and 'losers' with respect to the system implementation. As Grudin (1988) points out, while any multiuser system raises this issue, groupware systems in particular are likely to bring them to light. Early CSCW implementations were often rejected by groups who bore the costs without reaping the benefits. As groupware systems are 'organisationally optional', there was often not enough management support to 'push through' their implementation (*ibid.*). Systems like electronic email, which distribute the costs and benefits evenly amongst the user population fared much better than, say, electronic calendars or voice mail systems.

The lesson from the history of groupware systems is important in that it highlights an important aspect of system design and implementation. Even if few organisational systems will be openly rejected and abandoned on the basis of an uneven distribution of benefits, it *is* important to take this aspect into consideration when designing systems. The question "what's in it for me?" is organisationally relevant even through it is often ignored by management who are usually the sponsors of system implementations. Bowers (1994) describes the various ways in which people appropriate CSCW systems and how they deal with the tradeoff between the extra costs of working collaboratively through the system and the benefits thus realised. It is important to note that these considerations do not necessarily lead to a blanket acceptance or rejection of the technology but to various ways of working with or around the system. In addition, what might be seen by some as a limitation of a particular solution might for others be a way to enforce a particular way of working (e.g. Bowers 1994, p. 291).

The practical arrangements people make around their use of the technology, their orientations to is, their making sense of it, the way that technologies get factored into organisational life, etc. are all important aspects of the appropriation of technology which lie 'beyond design' (Procter and Williams 1996). The best that designers can do in a traditional, a-priori design process is to try and anticipate what might happen. This has led researchers in CSCW to take an interest in cooperative prototyping practices and facilitation as a means to overcome the various troubles that arise as people grapple with the technology and the contingencies of its use (Bowers 1994, Bowers *et al.* 1994).

## 2.3 User-Designer Relations

Communities such as human-computer interaction, computer-supported cooperative work and participatory design have produced a number of studies that emphasise the social organisation of design work (e.g., Button and Sharrock 1995, 1995a and 1996, Bowers and Pycock 1994, Sharrock and Anderson 1994, Woolgar 1991, 1994). While they focus on different aspects, such studies have in common that they often discuss the relationship between 'users' and 'designers', for example, as they

collaborate to work up requirements for design or focus on the organisation of work within a design project.

Most approaches today share a common set of understandings and assumptions about the nature of IT systems development and the relationship between the development process, IT professionals (designers) and non-IT professionals or workers (users). Drawing on a similar list of common understandings by Greenbaum and Kyng (1991a), I would contend that the following are more or less universally accepted in the disciplines named above:

1. Work is socially organised and takes place within a specific situational context and, consequently, so does IT use.
2. Design needs to be grounded in an understanding of what the context is.
3. Participation of users in the design process is generally beneficial.
4. Users have knowledge and skills relevant to the design process.
5. The role of various skills changes when IT systems are introduced.
6. IT systems should support working practices and
7. they should support quality work, not merely quantity.
8. The design process is political with at least the potential for conflict.

While there is little debate about these matters in principle, different academic traditions have their respective specific takes on them. However, the big question is how to practically address and deal with these insights. It is here that differences start to appear most clearly. While some approaches focus on integrating the 'user' – as a 'human factor' or 'human actor' (Bannon 1991) – into an existing design process, others go further in their suggestions that the design process itself has to change in more or less fundamental ways.

### 2.3.1 'Users' and 'Designers'

When we use the terms 'user' and 'designer' we run the risk of implying rather more than we want to express. As Greenbaum and Kyng (1991a) point out: "to system designers, the people who use computers are awkwardly called 'users', a muddy term that unfortunately tends to focus on the people sitting in front of a screen rather than

on the actual work people are doing". What is in danger of getting lost from view here is these peoples' main occupations which are not necessarily centred around a computer screen and the skills they have that allow them to do their day-to-day work. Ignoring their skills in this way and placing the IT system at the centre of attention leads to a view of users as being naïve or, even worse, as incompetent. While they may not know much about IT systems, they are not naïve but are rather competent in their respective areas of work – it is normally the IT designer who is 'work-naïve' (Bannon 1991).

Suchman (2002) has voiced a similar critique of the term 'user' and its counterpart 'designer' which problematises the relationship between them:

> To move beyond simple dichotomies in our understanding of who and where we are within the divided terrain of technology production and use, we need to begin by problematizing the terms 'designer' and 'user' and reconstructing relevant social relations that cross the boundaries between them. Attempts to avoid this conclusion lead to various sorts of surrogates, proxies, stand-in's for 'the user,' designed to allow the creation of usable technologies in the absence of these other relations.
> […] developers must give up control over technology design (which is in any case illusory), and see themselves instead as entering into an extended set of working relations for which the question at each next turn becomes: How do we proceed in a responsible way? (Suchman 2002, p. 94)

Furthermore, by using these simple categories, we elide the fact that the groups so described are by no means homogenous, that there are different 'users' and different 'designers' who may well have different jobs, pursue different aims, possess knowledges and skills, work under different constraints, etc.

Nevertheless, the terms exist and are used not only by academics but by ordinary society members as well and it is their understanding and use of these terms that I wish to attend to. One might say that the glossing done by the term points to the fact that systems development attends to the organisation-as-user (or in some cases department-as-user) rather than the user *per-se* (Slack 1997) or that 'users' are treated as *types* rather than concrete individuals (Sharrock and Anderson 1994). As Woolgar (1991, 1994) points out, 'users' are often actively 'configured', i.e. designers work up a view of how users would (or should) use a system and aim to place restrictions on users' actions in order to make design more manageable. Rather than suggesting that these uses of language are in need of repair, we should study

how the different notions of 'user' and 'designer' are used in practice, by researchers, IT professionals, workers, managers, trade unions and others to place restrictions on how a particular artefact might be used in practice. That is, while recognising the fact that the use of the terms 'user' and 'designer' is problematic, *as researchers* we should not seek to respecify them but should seek to investigate their uses. However, we should be mindful of our own use of the terms in *our* projects and seek to use them in a careful way that avoids implying more than we wish to say.

We find significant differences in the way in which the term 'user' is used in the academic literature. A lot of work in sociology or economics orients more to the organisation-as-user and the relationship with external suppliers or to the department-as-user and its relationship with the IT department (e.g. Friedman and Cornford 1987). In contrast, participatory design and ethnographically informed design deal with the worker-as-user, the individual "end-user" who directly works with the system. Some work also considers the indirect users of systems, e.g., members of the public using a service underpinned by IT systems. These distinctions are often not made explicit as researchers follow the usual, everyday practice of using the term as a convenient gloss[12].

For the practical purposes of this thesis, I will use the terms 'user' and 'designer' as a convenient gloss for what are complex phenomena and I will sometimes place them in quotes where I feel it is worth reminding the reader. As I am primarily concerned with IT systems in context, the term 'user' refers to the person who uses the system directly to get their work done, rather than, e.g., anonymous 'users' of packaged software. I will use more specific terms where other kinds of 'users' are referred to.

## 2.4 Design as Collaborative Work

Developing IT systems is itself a collaborative activity carried out within a particular working division of labour. One might therefore assume that there would be a substantial amount of qualitative studies of development work to be found in the

[12] It is important to note that critical traditions such as participatory design do not resolve the problem of just what is mean when the notion of a 'user' is invoked although they tend to recognise its existence. I would contend that ethnomethodology's focus on the activities involved in 'using' is helpful here in unpacking this issue.

software engineering literature, providing an insight into the practices of the various professionals who involved in design projects. Unfortunately, this is not the case. Of those studies that do exist, most report on *exceptional* circumstances such as the application of a particular methodology within a research project or a laboratory setting or are based on survey methods (e.g. Curtis, Krasner and Iscoe 1988).

To what extent such studies can be relied on to provide us with an understanding of design work 'in the wild' is questionable as the researcher intervention necessarily distorts the picture. As Bansler and Havn (1991) have noted: "At best, comparison and evaluation of different systems development methods is based upon, what you might call 'laboratory experiments'. […] Although these types of experiments yield valuable insights into the virtues and shortcomings of the tested methods, they do not tell us very much about how systems development is practiced". I would like to add to this comment that such laboratory experiments can only find what has been made findable as part of the experimental design. Given the repeated complaints that practitioners fail to adopt 'best practice' appropriately, it is surprising that studies of actual development *practice* are still so thin on the ground and are to be found within the CSCW literature rather than within software engineering.

Studies of design work have focused on different aspects such as the organisation of talk in problem solving (Button and Sharrock 2000), the organisation of source code and the use of guidelines (Button and Sharrock 1995a), the work of 'recomposition' (or integration) and the management of source code dependencies (Grinter 2003, Sharrock and Anderson 1993), organising the work as a project (Button and Sharrock 1994, 1995, 1996), working up and negotiating requirements (Bowers and Pycock 1994), maintaining an overview over both the product and the process (Bjerknes and Kautz 1991), dealing with plans that do not work out (Rönkkö, Dittrich and Randall 2005) and instantiating the rules of a methodology within an organisation context (Button 1993a, Button and Sharrock 1998).

It is important to recognise the detailed, situated practices that people doing systems development engage in. These practices *establish* the more or less routine ways in which systems development is normally undertaken. While development work may be officially undertaken under the regime of a particular methodology (as mandated

by the organisation), it is not the rules of the methodology that provide for the orderliness of the development process but peoples' everyday ordinary activities. For example, Grinter (2003) describes a number of practices aimed at overcoming the problems of managing dependencies in software projects and attending to the problem of delays. In one of the settings she studied, overall progress was discussed at weekly meetings and where delays were caused by dependencies involving other departments, the department head would follow up by arranging meetings with her peers:

> When the monthly review did not immediately follow the weekly review, she would return to her office and begin scheduling individual appointments with her peers to discuss delays. She did this face-to-face over lunch when she could, but in some cases her peer was in another state or continent and a phone call had to do (*ibid*. p. 307).

It is the work of preparing and holding the meeting, and following up with meetings addressing particular issues that allowed the organisation to achieve one of the core aims of software development, namely "measured progression" (cf. Button and Sharrock 1996). The extract also shows the department head going about her business in a way that is appropriate in the light of the circumstances. Rather than employing more formal means, she seeks to resolve issues in a non-confrontational way 'over lunch' but resorts to phone calls where necessary.

The organisation studied had a policy of non-escalation, i.e. there was a strong preference to seek a resolution to problems locally, amongst peers before calling on someone higher up the hierarchy to arbitrate:

> The process of engaging a more senior manager was known as "escalation" and implied that all possible negotiations among peers had not resolved the dependency, and that there were problems with that dependency that had to be made visible to management. Sometimes team leaders self-reported that they were delaying other people with particularly difficult problems. Open admissions tended to occur when a delay had only just emerged. Early admission, with a precise technical description of the complexity of the problem, often appeared to encourage other team leaders who depended on this code to help out with suggestions for possible design solutions (Grinter 2003, p.308).

Again, we see how the principles of good practice are realised and, moreover, how people orient to the organisation's 'moral order' (e.g. the rules and responsibilities

around the use of technologies) and how they demonstrate competence in the work they are doing. By acknowledging the existence of a problem early on and making the details of the problem available to their colleagues, team leaders can demonstrate their competence which lies in recognising the existence of the problem in the first place and describing its features. Offering the work of finding a practical solution to be undertaken in a collaborative way is a means to show professional respect and gives others the opportunity to raise issues themselves (e.g. how a proposed solution may impact their work).

In the work of Rönkkö, Dittrich and Randall (2005) we can see how plans are oriented to as objects of negotiation, how changes are made in the light of circumstances. However, neither does this mean that the original plan is simply abandoned, nor is the process chaotic. Rather, changes are made in a systematic way in the light of a need to come up with a viable and agreed plan for the future. Various issues are raised and discussed, for example, the degree to which the prerequisites for doing a particular job will be met: "will the prototype be in a state to make testing both feasible and meaningful?" Resources are also an issue as their planning will need to be matched to the workplan. In the action of changing a plan in the light of circumstances we can see clearly the relationship between plans and situated action (cf. section 2.2.1). As Sharrock and Anderson (1993, p. 159) put it:

> The carrying out of work is a matter of constant estimation: how much work is there to do, who is going to do it, how many people, for how long, doing what, needing what, with what assurance of success, and with what eventual product? It frequently turns out that the work does not go as estimated, very typically that it takes longer, is more uncertain of outcome, is more problematic, requires different personnel than have been estimated and resourced, but finding that the carrying out of the work is problematic is another of the 'normal natural troubles' of this work.

Recognising the existence of a problem and finding a practical way to deal with it is part and parcel of the work, a routine activity which does not normally occasion anything other than the normal practices of dealing with problems that are recognisable as the kinds of problems that have been faced before and have been overcome for all practical purposes under the given circumstances.

Button (1993a) describes the case of the development of a photocopier system involving hardware and software design which were mutually dependent but were

following different trajectories. In order to allow the hardware engineers to do their work, the software team had to release software which was not developed according to the strict principles of the mandated software development method (Yourdon and Constantine 1979, cited in Button 1993a, p. 36). Instead, they followed a strategy that allowed them to produce interim releases of the software that could be handed over to the hardware engineers:

> In developing the software in this fashion they fully recognised that they were compromising Yourdon principles. Yet in compromising Yourdon principles they were able to organise its use within the circumstances of the development. In so doing they were, in the activities of using it within the strategy they had devised, making the use of the method organisationally accountable (Button 1993a, p. 37).

By using this strategy, the software team can be said to have preserved the integrity of the Yourdon method while attending to the particular circumstances they faced. That is, while at any point they were aware of the ways they were 'breaking the rules', they were also aware of what needed to be done to re-establish a state of affairs which could be seen as being in line with the requirements of the method. In this way, they could make their work organisationally accountable as work done under the Yourdon regime.

The studies cited above contain many more perspicuous examples that illustrate how an orderly systems development process is produced through various mundane activities and the fieldwork material presented in chapter 5 will provide more. The point here is to note the existence and availability for study of methods of producing order in software development that are not rules prescribed by some methodology but are indigenous, locally relevant methods (or *ethnomethods,* see chapter 4.1) of systems developers. Orienting to what people know and use, how they draw on various resources in their everyday work, lies at the heart of understanding the collaborative achievement of a system's development.

### 2.4.1 The Hidden Work of IT

While a number of studies exist that describe systems development practice, the literature on the work of systems administrators, "the unsung heroes of the information age" (Barrett *et al*. 2004) is noticeably thinner. This is despite the fact

that a considerable part of the costs involved in the ownership of IT systems is not spent on their acquisition but on their ongoing operation and maintenance (cf. Patterson *et al.* 2002). Additionally, today's IT systems are usually complex configurations of various hardware and software components from multiple vendors and in any setting will usually be a collection of these, all partially linked together. There is a growing body of work concerned with the total cost of ownership of technologies[13] (e.g. Smith David, Schuff and St. Louis 2002, Wild and Herges 2000) and approaches are being developed to make systems more self-managing (e.g., Kephart and Chess 2003, Kephart 2005) or to reduce the recovery-time after failures (e.g., Brown and Patterson 2001, Patterson *et al.* 2002). However, studies of the actual working practices of system administrators are very rare, notable exceptions are Barrett *et al.*'s (2004) study of system administrators in large corporate compute centres and Bowers' study of IT facilitation work in an organisation making use of CSCW applications (Bowers 1994).

The work of system administrators involves a number of related tasks ranging from installing new equipment, monitoring, attending to routine tasks such as making backups, taking proactive measures such as moving database table spaces to larger discs, bookkeeping, to responding to troubles of various kinds. Not surprisingly, it exhibits similar features as other work. System administration is collaborative work that involves contacts with various parties such as fellow system administrators, technical specialists, system developers, suppliers, users, etc. Working within a complex working division of labour requires the maintenance of situation awareness, i.e. workers need to orient to what other people are doing as well as to the overall state of the systems. Working on multiple tasks at the same time and coping with interruptions (which may be an important part of the work rather than a nuisance, cf. Rouncefield *et al.* 1994) is another feature of system administration work that it has in common with other kinds of work.

There are, however, a number of features of system administration work which, while not unique, are still worthy of special attention. Planning activities in advance

[13] Studies mostly focus on general computing infrastructure desktop PCs or peripherals rather than systems.

and rehearsing them, possibly within a sandbox environment, is an important aspect of system administration work (Barrett *et al.* 2004). Any significant intervention that could potentially affect operation will also be timed if possible and then undertaken during allotted time windows.

As system administrators are usually highly skilled IT professionals who are often familiar with scripting languages, the development of monitoring tools and scripts for various kinds of jobs is also often part of their work. Such tools are often locally developed and maintained but can find wider application within the organisation (Barrett *et al.* 2004). One might say that their use of such tools is an example *par excellence* of end-user programming (see section 3.4).

Not only is the work of system administration largely absent from the academic and trade literature but it is normally invisible to most users as well and intentionally so. In a network administrator's words: "if you are successful, your work becomes invisible; if your work is visible, it suggests the network is failing" (Bowers *et al.* 1994, p. 14). The purpose of system administration is to 'keep things running smoothly', to hide the work involved in operating a system from its users so that they may concentrate on *their* job rather than having to struggle with the technology.

Rogers' (1992) study of distributed troubleshooting of a local network arrangement points to how 'users' get involved in the ongoing work of operating and maintaining systems. Her account concentrates on the potential troubles involved in this, how their activities are subject to various contingencies and to the various rules and responsibilities arranged around the use of the technology (the 'moral order'). It shows some of the ways in which troubleshooting activities are socially organised, e.g. in the local rule that potentially disruptive actions should be announced to fellow workers or in the shared use of resources. People oriented to the plot queue as a relevant account of system state and the state of the plotter itself as a its relevant real-world counterpart and tried to establish what the problem was by first taking some action (resenting a plot file) and then comparing the status of the queue and that of the plotter (as having or not having produced a plot).

The kinds of activities described above are what I would call 'the hidden work of IT', activities that are a routine part of operating and managing IT infrastructures and

systems but remain largely unacknowledged. I would argue that it is only through these kinds of activities that organisational uses of IT are made to work and that they therefore constitute an important but largely neglected aspect of the business of IT.

## 2.5 Innovation in Use: Envisaging the Future

Designing working systems involves the crucial step of moving from work as observed to a vision of how work might be done in the future, using new technological artefacts and organisational arrangements. Traditional approaches to IT systems design assume that systems are designed by IT professionals, that at the end of their design process their implementation is finished and their properties fixed, at which point they are 'handed over' to users. This reliance on 'prior design' has turned out to be problematic. Supplier offerings rarely provide solutions to the specific problems encountered in a setting and they need to be appropriated, fitted, extended, configured, grafted onto existing practices, etc. This observation is at odds with traditional supply-driven concepts of innovation which saw finished artefacts emerging "from the research and development laboratory as 'black-boxed' technical solutions, already corresponding to user needs, that could simply be diffused through the market to potential users" (Williams, Stewart and Slack 2005, p. 12). This presents a simplistic view of technological innovation that ignores the contributions of a wide variety of players, (end-users and various intermediaries) as well as the possible conflicts and significant uncertainties involved. The 'design fallacy' lies in "the presumption that the primary solution to meeting user needs is to build ever more extensive knowledge about the specific context and purposes of various users into technology design" (*ibid*. p.67).

The social learning perspective (Sørensen 1996, Williams, Stewart and Slack 2005) provides an alternative analysis that includes activities during implementation and use and highlights the diversity of players, their active role in technological development as well as their interactions and negotiations around design and use of technologies. Fleck (1988, 1993, 1999) coined the term 'innofusion' (innovation in technology diffusion) to describe the "processes of technological design, trial and exploration, in which user needs and requirements are discovered and incorporated in the course of the struggle to get the technology to work in useful ways, at the point of

application" (Fleck 1988, p. 3). In a similar way, the related concept of domestication draws attention to the various ways in which technologies are made sense of and accommodated or made 'at home' within larger socio-material arrangements (Williams, Stewart and Slack 2005, p.56-58). This involves, *inter alia*, acquiring necessary skills, exploring possible uses and developing practices and routines.

Attempts to improve the requirements gathering process and thereby the 'fit' of technologies with working practices during design will inevitably be frustrated by the changing circumstances of use – brought about in part by the introduction of the system itself. Systems are often used in different ways than originally intended by designers and the adaptations that people make range from the seemingly trivial to organisational innovation processes of significant scope (e.g. Bowers, Button and Sharrock 1995). Such innovations may be traded locally, within organisations or may, in the extreme case, be traded in the marketplace or be fed back into design through mechanisms such as user groups or other forms of user-supplier relationships.

Underpinning this view is a departure from the traditional understanding of requirements as somehow pre-existing, as something that can be 'captured' through appropriate 'requirements gathering' methods (Jirotka and Goguen 1994, esp. Woolgar 1994). Rather, requirements are seen as being constantly evolving and in need of being 'worked-up' and regularly revised in the light of the situation at hand. Any step in the design process and any event might potentially lead to changes, to new or changed requirements being formulated. Requirements as an outcome of social activities are also not unambiguous but inherently complex, reflecting the different interests various people have. Working up a set of requirements that can be used to inform the development of a system therefore involves negotiations and bringing into alignment various parties, technological and organisational arrangements. This is inevitably an ongoing concern rather than something that can be done once and for all.

It is therefore important for researchers to investigate the appropriation and use of IT systems and to attend to the potential for innovation during this phase. Attending to the local contingencies of technology appropriation and use offers a way to find

candidate solutions to overcome the problems of local fit of generic offerings discussed above. However, this can only be achieved through a long-term commitment to developing and supporting local configurations of technological arrangements, through a partnership between IT specialists, end-users and other organisational stakeholders (cf. Hartswood *et al.* 2000). Such a partnership can make the work of envisaging and realising new technological options more achievable, through stepwise design and experimentation. Research exploring this idea can be found in the more recent[14] PD and CSCW literature and I will discuss some examples in section 3.2.2.

## 2.6 Dependability

The notion of dependability has traditionally been defined quite narrowly in terms of aspects of the system under consideration itself. The aspects of dependability discussed in the traditional literature include (Laprie 1995):

- Availability: readiness for service,
- Reliability: continuity of service
- Safety: absence of catastrophic consequences
- Confidentiality: absence of un-authorised disclosure
- Integrity: absence of improper system state alterations
- Maintainability: ability to undergo repair.

Laprie presents a view of dependability as determined by the presence or absence of 'impairments' which he defines as follows:

> A system **failure** occurs when the delivered service deviates from fulfilling the system **function**, the latter being what the system *is aimed at*. An **error** is that part of the system state which is *liable to lead to subsequent failure*: an error affecting the service is an indication that a failure occurs or has occurred. The *adjudged or hypothesized cause* of an error is a **fault**. (Laprie 1995, p. 2, emphasis as in original)

[14] Of course, the early PD projects were also aimed at changing technology supply through building long-term partnerships. However, the scope of what they could hope to achieve in terms of design was limited in comparison to the more recent projects enabled by the wide availability of technologies today.

The fault-error-failure model of dependability is a good example of the assumptions underlying much work on systems dependability. At the heart of this work lies a concern with faults, i.e. the problem is defined in terms of properties of the IT system and, consequently, the means for making systems more 'dependable' are similarly system centric. Laprie (*ibid.*) describes them as fault prevention, fault tolerance, fault removal and fault forecasting.

An increase in the reliability of underlying hardware and the increased application of fault-tolerant systems in critical IT systems has led to a recent focus on "design faults" and on "human error" as contributors to IT system undependability (*ibid.*). This move is an important one as it brings into view the development and use of IT systems rather than their properties as artefacts. In addition, there is a growing recognition that traditional approaches such as the use of formal methods for the verification of software systems (Jones 2003) have their limits as they can only be used to reason about formal representations but leave their relationship with real-world phenomena unaddressed. While formal methods are of great importance in the design of a particular class of systems – safety or security critical systems such as nuclear power plant control systems or payment systems – their uptake in other areas is very limited (Cleland and MacKenzie 1995). While verifying that the implementation of a system corresponds to a formal specification is important (and a significant task in itself), the question of the fitness for purpose of the specification itself lies outside the domain of formal models and reasoning (cf. Jones 2003, p.39).

This widening of the agenda to encompass human activity in working on and with IT systems is accompanied by an increased focus on larger socio-material arrangements. Rather than focusing on safety critical systems – which usually have well-defined boundaries and are employed in relatively well-defined contexts – recent work has started to consider more mundane applications of information technologies which are less critical and where undependabilities are relatively frequent and are dealt with in a routine manner (e.g. Clarke *et al.* 2002, Clarke *et al.* 2003, Voß *et al.* 2002). So, while the traditional concerns continue to be important, they do not describe *all* that could be meant by 'dependability'. As systems are increasingly embedded in complex organisational and larger societal arrangements, further aspects become important that are concerned not with the system *per se* but with the way in which it

plays a role in the larger socio-material arrangements in which it gets used. Issues such as fitness for purpose or usability come to the fore as do larger managerial considerations such as cost or control.

Despite the sense of crisis that has plagued the industry and despite all the outright failures and catastrophic consequences, a lot of software does get built and used in practice. However, even systems that are 'successful' and serve their purpose well suffer from problems in a number of ways relating to at least some aspects of dependability, at least for some people (cf. section 2.2.3). I would argue that it is next to impossible to develop systems that 'tick all the boxes', that can be said to be dependable in all respects. People have developed ways of compensating for various kinds of undependability exhibited by the *more or less* dependable systems they work with (Voß *et al.* 2002). Examples include detecting errors[15] before they lead to service failures or compensating for failures by working around them. In this way we usually manage to make larger sociotechnical systems (assemblages of people, technologies and organisational arrangements) quite robust despite these *normal natural troubles*[16] (Voß *et al.* 2002, cf. Garfinkel 1967, p. 191, Sharrock and Anderson 1993, p. 159). The everyday undependabilities that we are faced with, not just in terms of problems with technical systems but also with other troubles (people calling in sick, etc.), are quite frequent but are not normally catastrophic. Rather, they are ordinary, mundane events that occasion situated practical (as opposed to legal) inquiry and repair. We face them often and have routine ways of dealing with them. In this respect, they can be said to be "part and parcel of the work being done" (Button 2006).

The point is not that these troubles are not important and can be ignored. Indeed, if there were simple solutions that could prevent them from occurring, these measures would be taken. However, in most circumstances there are no simple solutions but only ways of *avoiding, repairing or working around* these problems rather than

[15] I am not following Laprie's terminology here but am rather using these terms to describe phenomena within larger socio-material arrangements and not restricted to the technical system *per se*.

[16] I call them 'troubles' to point to their normally mundane nature – of course such 'troubles' can potentially become 'normal accidents' *a la* Perrow (1984).

making their occurrence *impossible*. The kinds of normal natural troubles described above normally receive attention only when they occur too frequently or when events give reason for concern (Clarke *et al.* 2006). We learn to trust technology through our experience with it and we learn in what respects to trust it. This trust becomes part of a background of expectations, of what we take for granted and in this sense, becomes noticeable only in its absence. As Button puts it:

> Computer systems are placed within the everyday world in which commonsense understandings of trust prevail, not into a world in which technical definitions of dependability rule. Like anything in that everyday world, they are therefore subject to commonsense everyday judgements with respect to their trustworthiness or their dependability. (Button, forthcoming)

Developers of IT systems therefore need to consider the use of systems *in context*, i.e. they need to attend to the way that people deal with systems that are only more or less reliable. This involves, amongst other things, attending to how people make sense of a system's operation (cf. Hartswood *et al.* 2003), how they can come to trust it to an extent and what implications that has (cf. Clarke *et al.* 2003a) as well as how the system's operation meshes in with its wider organisation context and the working practices around it (cf. Hartswood *et al.* 2003a). This opens up new areas for research into dependability as an *accomplishment* rather than an attribute of a system.

# 3 Understanding Practice – 'Informing Design'

Before discussing corealisation as a respecification of IT systems design, I wish to introduce a number of related, more or less traditional approaches that have important resonances with corealisation. This list is not aimed to be an overview of existing literature touching on the issues discussed above; such an overview would be necessarily incomplete. Rather, I select a number of approaches that are both relevant in systems design (both in practice and in the academic debate) and have had an important influence (one way or another) on the development of the concept of corealisation.

## *3.1 Ethnography for Design*

Ethnography is a form of study reportage developed in the social sciences that presents data about social life, usually obtained through observations, in a way that makes it available for analysis. This is achieved through first rendering the phenomena reported on strange and interesting, and then recognisable through examining their orderliness (Anderson 1994). There are many forms of ethnography developed in different disciplines within the social sciences which have in common their emphasis on producing a naturalistic description[17] of social life and its conduct in the natural context, taking an 'appreciative stance' (Rouncefield 2002, p. 71) that aims to recover the way things are seen by the members of the setting. However, approaches differ significantly in the way in which data is obtained and how it is analysed. Some forms of ethnographic analysis are based on categorisations or theoretical frameworks that are defined *a-priori* by the researcher. As Randall, Hughes and Shapiro (1994) point out:

> […] the appropriation of ethnography as a method of investigation in CSCW has not so far been accompanied by the necessary attention that needs to be given to issues such as *what kind* of ethnographic practice might be suitable for the task of gearing into the procedures of eliciting requirements, or how its analyses and descriptions can be related effectively to systems design. (*ibid*. p. 242, emphasis in original).

[17] I choose to avoid the term 'thick description' *à la* Geertz (1973) because of its various connotations which are not of interest in the context of this thesis (cf. Ortner 1997).

I do not wish to elaborate this issue at this point but for reasons that will become clear in my discussion of ethnomethodology in chapter 4.1, I find it important to aim to preserve the integrity of the phenomena observed by adopting a non-ironic stance that does not substitute them with categories defined *a-priori*. Such an approach avoids the problem of losing the phenomenon itself and finding only the analyst's objects[18]. Therefore, when I speak of ethnographic studies or ethnographies in the following, I mean by this a specific form of study informed by ethnomethodology which is based on a concern for identifying and explicating the ways in which social order is produced (the social organisation of work) and how people can act in meaningful and mutually intelligible ways as they go about their everyday business. This particular analytic mentality (Schenkein 1974) has been widely adopted in studies of computer supported cooperative work[19]. One should be careful to distinguish between ethnographies as a particular form of reportage and ethnomethodological studies which may or may not involve ethnographies but are conducted under particular analytic auspices.

Ethnographic studies of work in various settings have been instrumental in uncovering the seen-but-unnoticed aspects of work that have so often escaped attention in requirements gathering exercises and have therefore not been supported in the resulting systems designs. Even worse, systems designs are often in direct conflict with the organisation of work, as demonstrated by Bowers, Button and Sharrock's (1995) study of the introduction of a workflow system on a print industry shopfloor. These studies have been of great scientific and educational value, sensitising people to the kinds of phenomena of everyday work that are so easily missed in IT design. They have demonstrated the social character of workplace activities (even seemingly solitary ones) and have put the issue on the agenda once and for all. The success of ethnographic studies of work in this respect has instilled an interest in their use as a means for requirements capture. However, the problem of

---

[18] Which would be like starting an observation of life on a farm by looking for milk, steaks, burgers or sausages. Such an approach runs the risk of failing to spot the cow in the field (thanks to Roger Slack for the metaphor).

[19] This is not to say that there are not significant disputes within ethnomethodology and within CSCW about the significance of particular analytical choices (cf., e.g., Schmidt and Bannon 1992).

how best to incorporate findings from studies of work and technologies into IT systems design processes remains a matter of ongoing debate (e.g. Hughes *et al.* 1992, 1993, 1994, 1995, 2000, Randall, Hughes and Shapiro 1994, Plowman, Rogers and Ramage 1995, Schmidt 2000). Blomberg *et al.* (1993) point to the different projects of ethnography and design:

> While the ethnographer is interested in *understanding* human behavior as it is reflected in the lifeways of diverse communities of people, the designer is interested in *designing* artifacts that will support the activities of these communities. The current challenge is to develop ways of linking these two undertakings. (*ibid.*, p. 123)

In addition, there are common misconceptions amongst many wishing to employ ethnographic studies as a means to inform design. In the social sciences, ethnography is a particular 'form of reportage' (Anderson 1994) rather than a method for data collection or an analytic approach. In contrast, the use of the term in areas such as HCI and CSCW often presents it as combination of data collection, reportage and mode of analysis without further specifying either of those elements. Designers and many researchers are largely ignorant of the nature of ethnography in the social sciences and in practice, these matters do not impact on their work (cf. Anderson 1997). Some authors give some indication as to how the research was done by referring to their approach as "ethnomethodologically informed ethnography", indicating that the data collection is based on observation, usually involving some recording device, and that the approach to the analysis of the data is informed by ethnomethodology's study principles.

Very often, ethnographic studies, while providing rich descriptions of working practice as it exists, leave the question "so what?" unanswered (Plowman, Rogers and Ramage 1995, Fitzpatrick 1998, p. 17). Design implications, if any, are often quite vague in nature and few projects have managed to bring ethnographic observation and technology design together in a convincing manner. The problem of envisaging the future is notoriously difficult to solve as the real implications of any design decision can only be revealed over a period of time and through using the system in anger. While ethnographies provide an account of current working practices, they do not help with the work of envisaging the future. As Jirotka, Gilbert and Luff (1992) have put it:

> […] although ethnographic analyses of interactions in the workplace can highlight systematic, and often robust, features of work practices, they do not and cannot conclude either that these features *should* be preserved or that they *will* be preserved when new technology is introduced. (*ibid*., p. 112, emphasis in original)

This conclusion should not be read as saying that it is not possible to make practical judgements on the basis of the ethnographic analysis, rather, that what the ethnography *itself* can offer for systems development is at best a partial solution, which needs to be worked into practices that allow the implications of technological intervention to be worked out and for decisions to be made on that basis. It is interesting to note that the field of human-computer interaction faced similar problems in its early days (Bannon 1991, p.37). Many studies have drawn attention to the fact that technologies rarely emerge from nowhere and that they cannot be simply adopted but have to be appropriated by their potential users (Procter and Williams 1996, 1996a, Williams, Stewart and Slack 2005, Stewart and Williams 2005). This appropriation involves work to make technological offerings 'fit' their intended purpose and context. Innovation therefore happens long after the 'design' of a system has finished as people grapple with its affordances while facing the contingencies the world confronts them with. This work of appropriating technologies is inherently a social one as various studies of computer used in real-world settings have shown (e.g. Nardi and Miller 1990; Williams, Stewart and Slack 2005).

What is it, then, that ethnographies can provide for design in the light of these circumstances? Indeed, the question to what extent designers should be interested in *analytic ethnography* (Anderson 1994) is an interesting one – what is it that designers are interested in and should they burden themselves with the different interests and commitments that social scientists have? I would argue that rather than trying to appropriate analytic ethnography as practiced in the social sciences, designers should take seriously their own aims and commitments, they should take an interest in what people know and use, how they go about their day-to-day work and how they make sense of the actions of others. But, and this is crucial, they should pursue this interest on their own terms, to pursue the aim of informing design decisions. As Anderson (1994, p.155) puts it:

> It is simply that you do not need ethnography to do that; just minimal competency in interactive skills, a willingness to spend time, and a fair amount of patience.

Realising that the projects of the social sciences and of design are different but may require a similar attention to how people do their work, one can then develop a programme for the study of such activities which is *adequate* in regard to the object of study *and* the interests of design. One such approach, namely corealisation, is the object of this study and I will lay out its principles in chapter 4.

All this is not to say that ethnographic studies of work do not have a role to play in the project of informing design but they are useful in a different way.

> What ethnography may offer designers concerned with productivity is not just detailed description of work routines and daily life with which to fix the features of the design, but an opportunity to open up the overall problem-solution frame of reference in the context of some proposed solutions to specific identified problems. […] In other words, the contribution that ethnography may make is to enable designers to question the taken-for-granted assumptions embedded in the conventional problem-solution framework. (*ibid*., p. 170)

Working up requirements, then, is still a problem to be resolved. Ethnography has something to say about 'is' but cannot provide how things 'ought' to be. The point, then, is not to treat ethnography for design as a requirements capture method that will provide a specification but to treat it as a device for fostering what Anderson (1994) calls 'design sensibilities':

> [There is a] presumption that to be of value to designers, any description must be couched in a formalized or semiformalized notation of some kind: as if design consisted in jigsaw-puzzle solving and only certain shaped pieces were allowed. The age-old (and tired) prescription versus description debate, with the ethnographers staunchly appearing to refuse to be prescriptive in the face of designers' demands for requirement specification. What seems to be being missed here is the extent to which design involves sensibilities as much as models and predictions, programs and prescriptions. (*ibid*., p. 152-153)

In order to make these sensibilities useful for a design process, it has to allow for them to be brought to bear. This cannot be done in the form of a one-off process of writing a formal requirements document of the kind referred to above by Anderson. Conducting an ethnographic study only to then lose the phenomena so uncovered in the process of writing up a formal specification would be a waste of time. Of course,

various representations may be created for practical purposes, for example as an *aide memoir*, etc. (cf. Schmidt and Bannon 1992) but it is important that the design itself is informed by the rich understanding gained rather than by an impoverished version of it. One possible way of doing this is demonstrated by Blomberg, Suchman and Trigg in their approach to case-based prototyping (see section 3.2.3) which relies on an iterative process of ethnographic observation and design work. Similar work has been undertaken by Büscher *et al*. (2000, 2001, 2004) who draw on a wide range of participatory design methods to establish a process of long-term engagement between designers and (potential) users of novel technologies developed as part of the project.

### 3.1.1 Doing Ethnography for Design

There are practical problems related to the study of work in the context of IT systems development which those conducting an ethnographic study will have to tackle. While the use of ethnographies to inform design is advocated by many, there is a clear lack of guidance on how to actually conduct such a study (for whatever reason). This is especially problematic as many who will conduct studies with a view to inform design will not have a social science background but rather a technical one. This is not to say that 'doing' an ethnographic study would require skills which are not available to these people. After all, ethnographers rely on mundane skills we all have – observing, recording, sorting, formulating – but since those using it to inform design will not have been exposed to ethnographic accounts, they will find it even more difficult to know what to look for and how to go about conducting their study. It is therefore worth reviewing what various people have written on this matter.

Harper (2000) makes some recommendations for people using ethnographic methods for design. His first recommendation is to attend to the flow of information as a means to investigate the scope of the study and to ensure that no important activities are missed. Note that in making this suggestion Harper provides us with a practical means to achieve a particular purpose but does *not* suggest that the 'information flow' is necessarily an essential feature of the fieldwork or the analysis, nor is it all that is going on. In this his recommendation differs from recommendations made by, for example, Beyer and Holtzblatt (1997) who use 'information flow' as an analytical

concept rather than a practical device. The observation that information is being worked up, passed from one person to another, sorted, rewritten, recipient-designed, discarded, etc. is a recurrent feature of work in most settings (Harper 2000). This is why the flow of information in its various forms can be used as a guide to investigate a setting, to gain a broad understanding of what is going on – but it should not be confused with the phenomenon itself, i.e., with 'just how' people go about these activities.

A second recommendation, relating to 'ritual inductions' has as much to do with the problem of gaining access and being accepted as an 'insider' as with gathering information about the setting. While taking part in routine activities is the basis for an engagement with the setting and the basis for coming to be seen as an 'insider', there are opportunities to take a further step and to demonstrate commitment, respect and a genuine interest in what people do. 'Doing a nightshift' is an example of this, as is accompanying members on 'missions' that take them outside their normal place of work in the organisation. Researchers have found that taking part in such activities can help the ethnographer to gain acceptance, to get access to aspects of the work previously inaccessible and to learn more about what members would consider important. In general, sharing peoples' concerns and taking part in their activities (e.g., Christmas parties) as members of the setting is an important part of becoming an 'insider' and getting accepted and trusted. The ways in which this can happen will be specific to the setting and may be more or less obvious.

Harper's third recommendation is to pay close attention to detail. Details of how work is accomplished can often be retrieved through direct observation. When an ethnographer interviews members of a setting or asks them questions while observing their work, they will often comment that their work is not interesting, that the ethnographer would not want to know about it because it is so mundane. This leads to the ironic situation that the 'stuff' that the ethnographer is interested in, namely the details of work's accomplishment, is deleted from accounts. Members will instead often talk about versions of 'how things should be' or will use glosses to cut short what might seem to them to lead to a complicated and tedious account. It is therefore an important task for the ethnographer to remind interviewees that the details *are* of interest and to ask them to unpick what they say and provide more

detail. However, again in contrast to Beyer and Holtzblatt's recommendations (cf. section, 3.3), this should in no way take the form of disregarding or correcting the informant's accounts. Rather, the task is to convey to members of the setting that their work is taken seriously and that they are invited to provide as much detail as they care to provide, i.e. they can suspend the usual courtesy of not bothering others with the details of how they accomplish their work.

Another question that arises in relation to how ethnography for design is done concerns the choice of fieldwork method used. In some settings, participant observation will be possible (one can probably take over some office work under instruction and guidance) but in others, such as in medical work or air traffic control, this might not be possible and one might be limited to an observational role. Sometimes it will not even be possible to use observation (participating or not) at all, for example for ethical reasons and other means of gathering data will need to be found that are appropriate (cf. Hemmings *et al.* 2002). In addition, the degree to which the ethnographer might partake in the work can vary depending on circumstances, e.g., it might depend on workloads, the complexity of a case, etc. The question what to observe, for how long, when to participate, when to ask questions or interview someone, etc. can only be answered by referring to the situation studied and the purpose of the study, be it to add to academic knowledge or to inform design or some other intervention.

Where the ethnography is produced by social scientists, the problem of collaboration between them and designers arises. As Randall, Hughes and Shapiro (1994, p. 248) suggest:

> [This involves] procedures that, while not *ad hoc*, are, nevertheless, to be viewed as practical responses to problems encountered in an evolving collaboration between sociologists and system designers. In the absence of a universally acceptable method of subsuming descriptions of cooperative work into the requirements analysis process, it could hardly be otherwise.

There is no 'silver bullet', no machinery that will translate ethnographic accounts of working practices into requirements for design. One approach is collaboration between designers and ethnographers aiming to work up requirements which can be said to be grounded in the ethnographic account. Randall, Hughes and Shapiro offer no universally applicable method but by offering their experiences as an example,

they invite others to use this as a resource in reflecting on their own situation and developing methods that are adequate to the situation they face.

### 3.1.2 Typifications and Patterns

One way in which ethnographies can inform design is through the use of typifications, either of technologies or of phenomena observed. Trigg, Blomberg and Suchman (1999) provide an example of the former: they discuss a number of issues that arise in the introduction of a particular class of systems, namely document management systems. While the specific findings provided by their study are interesting for CSCW researchers, they do not provide requirements that might inform the design of document management systems in general or an implementation of such a system in another setting. However, the authors provide a set of questions one might ask in relation to a setting in which such a system might be introduced. These questions are motivated by the observations made within the study setting but are presented as question one might ask of any 'typical' setting in which documents are managed. By presenting the study setting as one that is a 'typical one', i.e. other settings will be roughly similar but will differ in detail, the step is made from a specific study to something that is more generally useful. It is not that requirements are formulated in a ready-to-use form but the formulation of questions emerging from the study might allow a designer to relate the questions or perhaps even the details of the study to the setting they are interested in.

A related approach is to distil from ethnographic studies a set of *patterns* (Erickson 2000, Martin *et al.* 2001, Martin and Sommerville 2004) in order to make the findings of ethnographic studies available as a "background for understanding or characterizing work in different settings" (Martin and Sommerville 2004, p. 62). By making findings of ethnographic studies available to designers in a standardised format and pointing out to what extent they are *repeated findings*, patterns aim to make the body of workplace studies more accessible and useable for the purposes of design. As with other forms of indexing or abstracting, the value ultimately lies not in the systematic presentation itself but in the fact that it makes a more substantial resource more easily accessible (cf. the ACM Guide to Computing Literature). Consequently, Martin *et al.* include in their patterns pointers to the original studies.

Patterns are orienting and organising devices that provide topics for investigation and ways of relating findings to similar findings from other settings, opening up the possibility of comparison and contrasting.

The way that a setting is arranged will always be reflexively tied to the way work is accomplished and workers will always have ways of coordinating their work. However, as Martin and Sommerville (2004, p. 63) point out:

> […] In any given setting *just how* coordination is achieved in relation to *what*, and in *what* ways layout affects, facilitates, or constrains activities still remains to be discovered.

The question to what extent these typifications have any purchase, how it is that they are relevant (or not) to the situation at hand still needs to be answered by those who use patterns to help them inform their design activities. Being able to access workplace studies that can be seen to be of relevance to the situation at hand can serve as a sensitising device that makes certain features of work activities more readily available. The patterns that Martin *et al.* (available on the patterns website at polo.lancs.ac.uk/patterns) have collected demonstrate this principle. For example, the notion of 'artefacts as audit trails' points to the

> […] way in which an artefact can serve as a stratified record of work. In this way the artifact serves as a means of coordination between workers allowing them to locate who has done what work and therefore assisting in remedying problems and so forth. It focuses on how amendments and attachments to the artefact, such as comments, date stamps, post-it notes, other documents and so forth, are accountable to the personnel within a setting. These annotations are accountable in that they readily afford information to these competent members about the process through which the artefact has progressed in the workplace. Actors are able to recover the process through viewing the artefact, seeing who has carried out work, when and why using their local knowledge of the setting and work practices. (http://polo.lancs.ac.uk/patterns/ArtefactAsAuditTrail)

In addition to this description, the pattern further contains a short note on where and how the observed pattern may be of relevance, what its implications for the dependability of systems might be as well as pointers to two studies that informed the formulation of the pattern and provide further resources. These are presented in abstracted form as 'vignettes' which summarise the findings and link to the original publications of a study of work in an entrepreneurial firm (Anderson, Hughes and Sharrock 1989) and work in air traffic control (Hughes *et al.* 1992, 1993).

## 3.2 Participatory Design

Originally developed as part of a strong trade union movement in Scandinavia as an approach to further workplace democracy by influencing technological development[20], participatory design (PD) has developed a number of approaches spanning a wide spectrum of design activities (Bjerknes, Ehn and Kyng 1987, Floyd *et al.* 1989, Greenbaum and Kyng 1991, Schuler and Namioka 1993, Muller, Wildman and White 1993, Clement and van den Besselaar 1993, Kensing and Blomberg 1998, Törpel 2005). While some have tried to remain true to the roots of the movement and its political ambitions, others have taken a more pragmatic stance and have developed participatory design methods with an aim to improve the quality of the resulting IT systems, their fit with the working practices and their acceptance by users but not necessarily taking side with the users in a political arena.

### 3.2.1 PD as a Political Project

Initial studies in the context of the Scandinavian trade union movement were focused on the aims of the unions to gain influence on the development of working practices by influencing technological innovation in the workplace. By forging close collaborations with their employer counterparts and academic partners, the unions were able run a series of projects such as the NJMF project (Norwegian Iron and Metal Workers Union, Ehn 1993), DEMOS (Ehn 1993) and DUE (Kyng and Mathiassen 1982). The approach and its theoretical underpinning developed in these projects became known as the collective resource approach (CRA) or simply the "Scandinavian Approach" (Floyd *et al*. 1989). "Action-oriented and trade union based strategies" (Kyng 1994) were combined with a concern for the social relations within technology production and the aim to achieve cooperative design (Greenbaum and Kyng 1991) in which the 'users' would play an important part. While the feasibility of the initial projects was heavily dependent on the particular industrial relations within Scandinavian countries, there were also significant barriers to a

[20] PD was quite deliberately divorced from preceding sociotechnical approaches as it was felt that they did not allow the pursuit of workplace democratisation and trade union based participation (Ehn 1993, p. 49 ff.). I will limit myself to the discussion of PD as it encompasses relevant aspects of sociotechnical approaches but develops them further.

wider adoption of the approach, even within Scandinavia[21]. The trade union based part of the collective resource approach was at odds with the established formal negotiation structure which gave workers little effective power to object to individual management measures (Kraft and Bansler 1992, Greenbaum and Kyng 1991, p. 11).

The second generation of projects, e.g. Utopia (Bødker *et al.* 1987, Ehn 1988, Ehn 1993), or Florence (Bjerknes and Bratteteig 1987), consequently focused on the second instrument, that of establishing collaborative design practices and it is this strand of the work which has inspired most of the later work on participative design. A third generation was mainly concerned with finding ways to embed participative design practices with organisational arrangements to make them sustainable in the long term (Bødker 1994). Other approaches, most notably the MUST method (Kensing, Simonsen and Bødker 1998, 1998a, Bødker, Kensing and Simonsen 2004), are concerned with finding ways to reconcile the aims of participatory design with the changing landscape of technology supply strategies, especially the increased use of off-the-shelf products.

It is the later two phases which are of primary interest to this thesis. However, while there has certainly been a shift in emphasis, the larger political aims of the collective resource approach and the critical stance regarding various issues such as the conceptualisation of skill and knowledge (cf. section 2.2.2) have continued to influence later work. Arguably, this turn to the internal organisation has made participatory design more widely applicable and it is worth noting that it was at this point (in the mid 1980s) that it became better known outside Scandinavia. At the same time, computing underwent its critical transformation from a centrally managed resource to a local tool – potentially opening up opportunities for much smaller scale and grass-roots efforts to change technological development. The idea of designing tools for skilled work was first explored in projects such as UTOPIA or Florence, and in a number of studies investigating cooperative prototyping (Bødker and Grønbæk 1991a).

---

[21] Consequently, questions as to the wider impact and prospects of success of the collective resource approach were asked, especially with regard to the question of uptake within the US (Kraft and Bansler 1992, Bansler and Kraft 1994, but cf. Kyng 1994).

There has been much debate within the PD community as to whether there is an inherent, unavoidable conflict between workers and management or if there is at least a potential for collaboration. Shapiro has recently suggested that the question to what extent changes in working practices leading to increases in productivity are against the interests of workers is an "empirical question with open possibilities today" (Shapiro 2005). Without discarding the political ambitions of participatory design or denying the underlying conflict of interest, one might say the interests of workers and management in *particular* circumstances *may* be aligned far enough to make cooperation possible. In addition, the almost ubiquitous nature of information technologies now means that the group of 'users' is much more broadly defined and includes people who may well be in a situation that allows them to shape their own working practices. In addition, traditional ways of increasing productivity fail to be effective more and more often and it turns out that previous 'successes' of rationalisation were based on a fiction, that work processes that were presumed to be routinised still depended to a large extent on workers' resourcefulness (e.g. Suchman 1983, 2000). Managers often simply do not know what the best way of organising work may be – even in their own terms – so in many instances, they may well depend on the cooperation of their workforce and may well be looking towards workers themselves for answers. Where these opportunities can be exploited to improve efficiency or to provide a better service without intensifying work (e.g. reducing the amount of rework done), room for collaboration and for PD emerges. What designers need to tackle, then, is the practical politics of the particular place(s) they are designing for (Büscher *et al.* 2002). I will return to this issue in section 7.1.5.

The picture over the past 30 or so years is certainly mixed. PD has certainly established itself as an academic discipline with its own bi-annual conference and significant overlap with other disciplines such as HCI and CSCW. There is a continuing stream of studies and the ideas behind PD have influenced systems development methods. More recently, trade unions in Sweden and Germany have started certification programs for software products, trying to replicate the worldwide

success of TCO certifications[22] (Sundblad, Lind and Rudling 2002, Walldius *et al.* 2004, 2005).

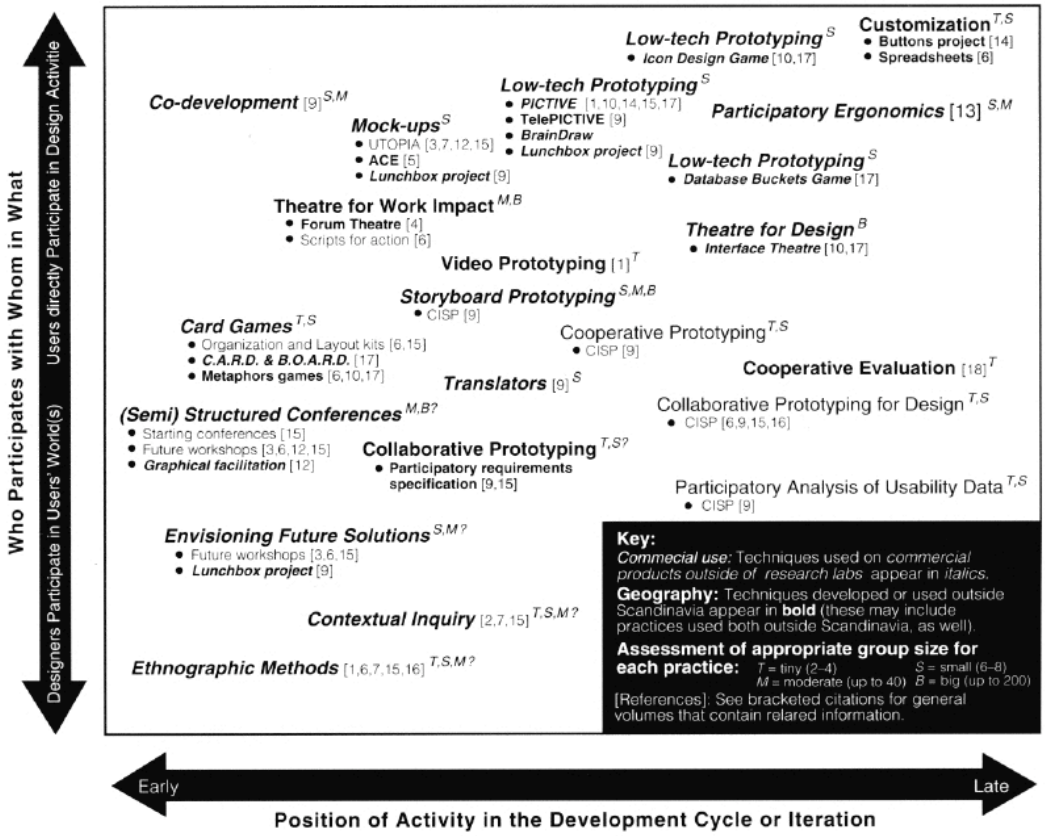


**Figure 1 Overview of participatory design approaches (from Muller, Wildman and White 1993)**

### 3.2.2 The Diversity of PD Approaches and Methods

The change of focus from traditional forms of trade-union bargaining and towards the process of systems development itself has led to the emergence of a wealth of approaches to and methods for user involvement in systems development processes. An interesting overview of participatory design methods is given by Muller, Wildman and White (1993) who arrange different approaches on a two-dimensional grid along the dimension of degree of user participation and the stage in a system lifecycle in which user participation takes place. It is interesting to note that they seemingly find little problem in categorising approaches in this way, i.e. each finds a place on the grid quite easily (see Figure 1). Approaches concerned with

[22] See www.tco.se, www.usersaward.se and www.usersaward.de

requirements gathering and vision building can be found on the left side of the diagram with prototyping approaches dominating the middle ground and the right hand side being populated by evaluative methods and tailoring approaches. While each individual method seems to find its place, it is also worth noting that the whole spectrum is covered. The wealth of participatory design approaches, methods, tool and techniques demonstrates the various answers that can be found to issues such as the rationale for participation, the requirements for participation (Bødker 1994), the scope of participation as well as barriers to achieving it (Bødker 1994, Axtell, Waterson and Clegg 1997).

I wish to focus on two strands within participatory design which are of particular interest in the context of this thesis: the use of ethnographic approaches and cooperative prototyping.

### 3.2.3 Ethnography in PD

In the 1990s, the use of ethnographic methods in PD projects became quite popular as an means to enable researchers/designers to better understand the daily activities of the workers they were designing with (Simonsen and Kensing 1997, Crabtree 1998). Ethnographic fieldwork is useful in uncovering issues that workers might not talk about, either because they are potentially problematic, because their relevance to the project is not immediately apparent or simply because they feel that they should not bother others with the details of how their work is accomplished. Simonsen and Kensing (1997) note that ethnographic observation can provide useful input for subsequent interviews and design activities. I would add that observation also provides an opportunity to clarify open questions and to give workers further opportunity to comment and clarify. As Crabtree (1998) points out, ethnographic fieldwork can also counter the potential problem of focusing too early on future possibilities, especially during prototyping, and losing sight of the social organisation of current practice.

Research undertaken by Büscher *et al.* (2000, 2001, 2004) explores long-term technological development in partnership with landscape architects, exploring advanced technologies such as virtual collaborative environments in real-world settings. Suchman, Trigg and Blomberg have explored novel document management

technologies in a law firm (Blomberg, Suchman and Trigg 1996) and a civil engineering office (Suchman, Trigg and Blomberg 2002, Trigg, Blomberg and Suchman 1999). These studies combine workplace studies with the development of prototype technologies to explore these technologies in use. In combining the analytic interest of much CSCW research with the interest in collaborative design practices developed in PD, these studies aim to move beyond mere user-orientation and to realise user-led cooperative development through successive rounds of prototyping informed by observational studies of work. They demonstrate how working practices change in response to the introduction of IT systems and how these changing working practices in turn result in the emergence of new requirements.

For example, Trigg, Blomberg and Suchman (1999) describe how document coding practices changed in response to the introduction of a document management system that could store meta-data alongside the document files and allowed for various different ways of document retrieval. Before the introduction of the system, coding was closely linked to the storage location of the document and workers had to make difficult decisions as using the wrong code might result in the document becoming difficult to retrieve. The development of the electronic document management system with its support for multiple meta-data elements meant that coding now became associated with the process of scanning the document and was done in part by a student intern. Engineers would then amend or change the codes as necessary. In a next step, a hardcopy of the electronic coding form was introduced as a means for engineers to code up documents before passing them on to he intern for scanning. Thus, in a sequence of steps involving changes to the document management system, the introduction of a new paper artefact as well as organisational changes, a new configuration of working practices and technologies was elaborated through the cooperation between designers and users. Designing for a changing world means "co-developing a complex alignment among organizational concerns, unfolding trajectories of action, and new technological possibilities" (Trigg, Blomberg and Suchman 1999, p. 349).

As the studies mentioned above demonstrate, the combination of ethnographic observation and participatory design methods can be highly effective in developing

systems in ways that fit in with the (changing) ways in which people go about their business. In addition, there is at least the potential that such studies will be of more general interest and will be a valuable resource for researchers and designers concerned with other settings. The specificity of such studies can be a resource rather than a hindrance for designers wishing to learn from them as "the richness of their contents and the observable details of their use connect in varied and evocative ways to a wide range of other settings and practices" (Blomberg, Suchman and Trigg 1996, p. 258). In this, the studies differ from some approaches based on 'scenarios' which subject the phenomena to a process of representation in an attempt to make them "more discussible by developers and users" (Chin, Rosson and Carroll 1997, also cf. Muller 1999).

### 3.2.4 Cooperative Prototyping and Bricolage

Mock-ups allow experimentation and envisioning to be extended beyond what is immediately available by introducing placeholders for actual resources. For example, the Utopia project made use of simple cardboard models and hand-drawn images to simulate a graphical workstation for newspaper layout (Bødker *et al.* 1987, Ehn and Kyng 1991). This enabled them to explore, for example, the idea of having a very high-resolution screen that could display a whole newspaper page in one go although the technology was not available at the time.

Bødker *et al.* point out that although the basic resources like cardboard are cheap and can be quickly assembled, the approach also has drawbacks as it is difficult to capture the dynamic behaviour of a system and the drawing of images is time consuming. Also, being able to use the system under development helps with envisioning its future use and to spot potential problems early. In this respect, working prototypes are superior to paper mock-ups as they better visualise the dynamic aspects of the system (Bødker and Grønbæk 1991). However, computer models or prototypes have their own problems. For example, modifications of the prototype may require programming which may need to be done outside the prototyping session or they can blur the distinction between what is a prototype or mock-up and what is a working system (Bødker and Grønbæk 1991a). There is also a danger that a computer implementation may appear more finished than it actually is

and therefore discourages people from making suggestions for change. In order to achieve a reasonably fluent process, "the designer must know the prototype and the prototyping environment well enough to recognize the complexity of a change and not start programming activities that destroy the users' understanding of the situation" (*ibid*. p. 326). In a prototyping session, it is the responsibility of the designer to manage the situation in a way that does not require users to wait for the designer to finish lengthy tasks in which users cannot participate. The challenge is to maintain the users' interest despite a number of potential 'breakdowns' the cooperative prototyping process might suffer. Bødker and Grønbæk (*ibid.*) point to a number of practices for making this happen, e.g. the use of tools allowing direct manipulation of the prototype rather than programming, postponing changes that seem to require extensive work, or making preliminary changes which get 'fleshed out' at a later time.

Cooperative prototyping sessions need to be carefully planned and managed in order to be effective. It is necessary for those planning the process to attend to a number of issues such as: availability of users to participate, choice of resources to make available and tools to use, achieving a mutual understanding of the aims and limitations of the endeavour, avoiding premature lock-in to particular configurations, avoiding modifications that take too long and disrupt the flow, managing users' expectations while making sure they remain 'in charge' (Bødker and Grønbæk 1991a). Grønbæk (1989) suggests that one should also consider whether the functionality of the prototype is sufficient to allow users to be engaged in a meaningful way or if its nature *as* a prototype with limited functionality limits this engagement. He suggests that, compared to 'horizontal' prototypes which represent just a thin layer of the system (e.g. its user interface), 'vertical' prototypes offer the potential for evaluation in real-world settings and with real data. Building more functionality into a prototype can be expensive, especially if the prototype is not intended to be developed into a working system, but the benefit of a more meaningful engagement with it may outweigh these costs. Grønbæk (1989) suggests a number of practical ways of reducing the costs, focusing on the use of fourth generation languages (or application generators). Clearly, attempts to write prototypes 'from scratch' would not only be prohibitively expensive but would also

potentially lead to frustration if the low level of abstraction involved meant that productivity is low and therefore turnaround times long.

The point about rapid prototyping is that the prototypes have the potential to provoke a reaction to them that will be of use to those aiming to explore the potential uses (Mogensen and Trigg 1992). Such a reaction will not normally be a direct statement of a requirement – "we need this" – but will rather be offered by 'users' in the form of "queries as to what is possible, doable, within their capabilities or within those of the designer or machine" (Bowers and Pycock 1994, p.303). The interactional phenomena observable in prototyping sessions are interesting in that they reveal some of the details of how requirements are 'worked up' rather than merely 'uncovered', that they are the outcome of a process of making sense, envisaging and formulating. Bowers and Pycock's (1994) study also points to how the interactions are managed by both 'users' and 'designers', how 'designers' may resist the formulation of a requirement or try to change the way in which it is formulated. This is not to say that there is some sort of hidden agenda on their part but it demonstrates how they orient to their own concerns which have to do with the practical problems of implementation, generalisation, etc.

Crabtree (2004a) examines the use of the concept of a breaching experiment (Garfinkel 1967) as a way to explore "technological innovations that have little grounding in current practices". Using the example of a mobile mixed reality game, he demonstrates how the unfamiliar context of the game leads players to build up a working stock of knowledge that allows them to play the game despite various, potentially disruptive troubles. He also demonstrates how the experience of using the system in its current, imperfect state can give rise to ideas for a redesign or modification. In Crabtree's example this feedback into design is facilitated by ethnographic observation of the game play by the researchers.

## *3.3 Contextual Design*

Although contextual design (Beyer and Holtzblatt 1995, 1997, 1999, Holtzblatt 2005) is often subsumed under the rubric of participatory design, I will discuss it in a separate section for three different reasons: first, on superficial reading, contextual design can be seen as a synthesis of participatory design, ethnographic approaches

and more traditional software engineering approaches. Second, it differs from many other approaches discussed above in important ways and therefore provides an interesting contrast both to them and to corealisation. The third reason is that contextual design has achieved immense popularity and is therefore of such significance that a more extensive treatment is warranted. Holtzblatt defines contextual design as:

> A set of techniques to be used in a customer centered design process with design teams. It is also a set of practices that help people engage in creative and productive design thinking with customer data and it helps them co-operate and design together. (Holtzblatt, in Preece, Rogers and Sharp 2002, pp. 313).

Contextual design consists of a number of consecutive steps concerned with working up requirements for systems design and developing prototype user interfaces (*idem.*): contextual inquiry, interpretation and modelling with cross-functional teams, consolidation of information, visioning about work practices and the development of storyboards and finally user environment design. The promise of contextual design is one of a comprehensive, well-described and generic method for user-centred design of (interfaces for) interactive systems. From Holtzblatt's comments cited above it would appear that contextual design developed out of a concern with usability and the tradition of participatory design. It would appear that one of the motivations was dissatisfaction with the way that "all this qualitative stuff" (idem. p.314) became sidelined in systems development and usability engineering practice.

Contextual design rightly shows a concern with understanding the context in which a system will get used as a prerequisite for system design. In this it parted from traditional usability engineering (which still focuses on laboratory-based investigations) and aligned itself with more recent approaches to requirements elicitation coming out of the CSCW and PD communities. Recognising that end-users, their practices and other aspects of the context need to be taken into account is a laudable achievement. Providing designers with a teachable, adaptable set of methods that allow them to keep their eyes focused on the context while envisaging future working practices is equally important. Contextual design is a significant achievement and its commercial success attests to its usefulness in practice. However, I wish to draw out a number of important features of contextual design that

set it apart from other approaches developed within CSCW and PD, e.g., ethnography for design or case-based prototyping.

First, contextual design produces what might be called 'thin description'. Perhaps this is in response to the inevitable pressures and resource limitations imposed on most projects, but I suspect there is more at play. To avoid a possible confusion: I do not suggest that merely collecting more data would solve the problem. Rather it is a question as to how one approaches the setting and the people one studies. The analytic mentality is of crucial importance and I would suggest that contextual design encourages a particular way of 'seeing' the world it studies, namely as one that consists of a number of essential features that can be quickly captured and made available for design.

Additionally, contextual design subjects the fieldwork data to an array of transformative mechanisms which reify it in terms of a number of predefined category systems, contextual design's process models. While on the surface this seems to make the data more amenable to uptake within design, the question remains as to what is won through such a transformation. Even more crucially, one needs to ask what is *lost* on the way? It is ironic that Beyer and Holtzblatt (1998) call for the person conducting the contextual inquiry interview to try and discourage their informants from speaking in general terms:

> Even in the workplace, customers easily slide into talking about their work in the abstract. But there are signals that indicate the customer needs to be brought back to real life. […] The best cure is to pull the customer back to real experience constantly. Every time you do this, you reinforce that concrete data matters, and you make it easier to get concrete data next time. (Beyer and Holtzblatt 1998, pp. 48-49)

While the abstract process descriptors that come with the method and the activities of the design team are seen as valid ways of condensing and taming the bewildering host of observations to make them more amenable to design, the generalisations that 'customers' readily offer when talked with are seen as distractions from the "real data". This stance is methodologically ironic (cf. section 4.1) in that it systematically replaces members' accounts of their working practices with objects from an ontology provided by the method.

Finally, I wish to discuss contextual design's *raison d'être*: to build systems that "fit into the fabric of everyday life" (Beyer and Holtzblatt 1998, p.1). This would seem to be in line with the kinds of concerns of CSCW and PD as discussed in the previous sections. However, this program also carries with it some significant undertones as contextual design also "develops the details of business process redesign" (*ibid.* p.73). While any design effort aims at effecting some sort of change in the work environment, the aim of contextual design is based on a particular premise, namely:

> There is no explicit intention on the client's side to change the way they work in any major way. Introducing a new system to automate the inefficient ways that things are done currently is a waste. The challenge is to move the design team and the client together to invent ways to improve the work. (Beyer and Holtzblatt 1998, p. 72)

Underlying contextual design's agenda is an assumption that change is needed, that current working practices are inefficient and, finally, that the people working with or procuring the system are unaware of better ways of organising work. In summary, if the aim is to replace working practice, why engage in a costly but rather thin descriptive exercise?

## 3.4 Configuration, Tailoring and End-User Programming

One way to respond to the problem of various, evolving requirements is to defer certain decisions about a design and make a system configurable or tailorable. Another, related approach is to include facilities that allow end-users to extend the system themselves in complex ways not foreseen by the designer using some (visual) programming language (Nardi 1993). While most systems these days are configurable in many respects, the degree to which people can modify the system varies widely. Simple changes such as selecting program settings and defaults are quite common but the use of more sophisticated technologies such as task-specific programming languages or visual application frameworks remains limited.

An interesting exception, through, are spreadsheet systems where the development of computational structures is an integral part of their intended use. Not every user necessarily gets involved in the development of spreadsheets, many people merely enter data and use the computed results. However, the development of spreadsheet applications in many organisations is *not* the preserve of IT specialists but is rather

done by domain specialists. Various studies have been undertaken to uncover the various practices and socio-material arrangements underpinning and supporting the use of spreadsheets (e.g. Nardi and Miller 1990, 1991). Spreadsheet development is a collaborative activity, involving a practical division of labour between those more intimately involved in the development of a spreadsheet and those who 'merely' use it as a tool but are not interested in its internal workings. However, the division of labour is not one of 'designer' and 'user' but usually a more complex one where people take on responsibility for different parts of the overall 'design' (Nardi and Miller 1991, p. 166). Nardi and Miller (*ibid.*) found that the close cooperation between people with different respective skills found in many instances allowed non-programmers (those not interested in the spreadsheet's workings) to widen the scope of their activities as they could call on colleagues to contribute parts they could not handle themselves and they could learn from more experienced colleagues. They are also likely to retain ownership of the spreadsheet, farming out jobs they cannot do themselves to more experienced people but remaining in charge of the overall development:

> With spreadsheets, problem solving is distributed such that end users do not rely on programmers as the indispensable implementers of a set of specifications; instead end users are *assisted by* programmers who supply them with small pieces of complex code, or with training in advance [*sic.*] features, as they build their own applications. (Nardi and Miller 1991, p.168, emphasis in original).

Another aspect of the use of spreadsheets is that their development provides an occasion (and a resource) for workers to reflect on their working practices. Trading spreadsheets can be a way to disseminate domain knowledge (*ibid.*). One might say that work *on* spreadsheets is part-and-parcel of people's work *with* spreadsheets.

It is not that the work with spreadsheets is without its problems. Concerns have been raised about the correctness of spreadsheet calculations (e.g. Brown and Gould 1987, Panko and Halverson 1994, Panko 1996), and their long-term viability. However, Nardi and Miller's (1991) work shows that these potential problems are recognised by practitioners and that they have practical ways of addressing them. An important means for doing this is their understanding of the problem domain which allows the search for errors to be prioritised according to the potential impact. Users also draw

on their experience to perform "reasonableness checks" (*ibid*. p. 173) and use more general strategies such as checks against manual calculation or checks on partial results to increase their confidence that the overall calculation is correct[23].

In another study, Gantt and Nardi (1992) have observed the work of CAD[24] system users who were involved in various ways in adapting and extending the system to fit their purposes. Again, they found a practical division of labour between those who were not primarily interested in the configuration of the system and those who took on a support role as "tailors", "translators", "gatekeepers", "gardeners", "facilitators", or "gurus". Trigg and Bødker (1994, also cf. Bødker 2000) found that these kinds of roles can become established and even find official recognition with the organisation. Thus, instead of being merely an *ad-hoc* response to local needs, the work of tailoring can be pursued more strategically and in correspondence with, indeed furthering, the organisation's overall aims (also cf. Mackay 1990). Tailoring can be a way for providing the interaction, the necessary grist between those involved primarily in the design of technologies and the users of these technologies. In this way, through what Schmidt and Bannon (1992) called 'articulation work', the organisation's needs for a degree of standardisation of practices and tools can be achieved. However, rather than being pre-planned, such alignment is emergent and constantly changing, an interplay between local and more general solutions, constantly evolving. Rather than seeing tailoring as a threat to interoperability and maintenance, one might argue that it is precisely the ability of systems to be configured that enables them to play their role in linking different practices (cf. Trigg and Bødker 1994).

Suggestions that the 'problems' of end-user computing should be solved using traditional software engineering approaches (e.g. Taylor, Moynihan and Wood-Harper 1998) miss the point of many end user computing activities: end-user developed systems are often worked up to serve a particular purpose that does not

---

[23] While the practical methods are sufficient in some cases, there are undoubtedly measures that could be taken on the part of the designers of spreadsheet programs to better support the design and validation of spreadsheets. However, such measures need to be consistent with the way spreadsheets get used and should not destroy the main value of this type of system: their flexibility.

[24] Computer Aided Design

warrant or is unlikely to attract the substantial resources needed for a more formal development project involving IT staff. Indeed, the very usefulness of end-user developed applications rests on their rough-and-ready-ness and a short time to realisation. Subjecting end-user computing to the practices of software engineering like change control, data normalisation, etc. is unlikely to achieve the desired results and may well be counterproductive as it runs the risk of stifling end-user innovation.

# 4 CoRealisation

This chapter describes a new approach to IT systems development that differs in important respects from the approaches discussed in chapter 3 but is also inspired by them and aims to synthesise them into something new. It is a radical respecification of IT systems design as a collaborative activity engaged in by members of a setting over a course of time, focusing on their competentices as members of this setting as a crucial resource in their doing the work of design. This respecification of design as "inter-subjectively constituted, lived experience" (Hartswood *et al*. 2002, p. 10) is an important precondition for the development of corealisation as a practical approach.

I will first introduce ethnomethodology as it provides the analytic mentality that corealisation adopts (and that I use in my study of its practices) before considering what it means to apply this mentality to the project of systems development. I formulate the notion of 'design *qua* member' as a respecification of the divisions of labour in systems development and the related concept of hybrid knowledge. Following this I will discuss the notion of unique adequacy of systems and of methods for building them. I conclude this chapter by formulating corealisation's principles.

## *4.1 Ethnomethodology*

Corealisation as an approach to IT design is inspired by ethnomethodology (Garfinkel 1967, 1996, 2004, Garfinkel and Wieder 1992, Maynard and Clayman 1991, ten Have 2002, Sharrock and Anderson 1986, Sharrock 2001, Bergmann 2005), and its philosophical framework grounded in the work of Schütz, the later Wittgenstein and Husserl. Ethnomethodology's foundational question is: "how is it that we as society members come to perceive this society as orderly, how is that order created and how is it recognised?" It aims to answer this question by observing peoples' methods in everyday situations (such as in their workplace) and explicating how the routine, ordinary activities they engage in allow them to go about their business in what would appear to be a routine and unproblematic manner.

Society members go about their daily business without constantly questioning the state of affairs (as scientists might do) but taking for granted the existence of an order

in their everyday lifeworld (Schütz 1953) with which they are familiar, that they can recognise and that they contribute to through their own behaviour. The intrinsic order of the social world as taken for granted by ordinary society members acting in the "natural attitude" (*ibid.*) is opposed to the "scientific attitude" which suspends and questions the taken-for-granted nature of worldly phenomena.

In contrast to, for example, Parsons' (1937) view, ethnomethodology's incommensurate (Garfinkel 1988) stance is that order exists *within* and is produced by society members' everyday, ordinary activities and that it is therefore massively available for study rather than being elusive and available for scrutiny only through sociology's methods. As Garfinkel (*ibid.*) puts it:

> "For ethnomethodology the objective reality of social facts, in that and just how it is every society's locally, endogenously produced, naturally organised, reflexively accountable, ongoing, practical achievement, being everywhere, always, only, exactly and entirely, members' work, with no time out, and with no possibility of evasion, hiding out, passing, postponement, or buy-outs, is *thereby* sociology's fundamental phenomenon." (emphasis in original)

Ethnomethodology sees itself as being an incommensurate alternative to formal analytic sociology (Garfinkel 1991) and refuses to be folded into the traditional canon of sociology as some people have tried to suggest it should be[25] (cf. Garfinkel 1988, Button 1991, Lynch 1993). It is incommensurable in that it chooses as its matter of investigation the methods whereby any activity is made possible and recognisable. In doing that it poses a very different question than formal analytic sociology (Button 1991) – it has no interest in the structures of society as sociologists might find them, but in how they are achieved, how they come to be through society members' ordinary actions, how they can be recognised by society members as being part of the orderliness of the social world and how society members can come to perceive the world as generally orderly. Ethnomethodologists take a stance of methodological indifference (Garfinkel and Sacks 1986), that is, they don't "grant a particular party an interpretive warrant for what it is they are looking at, nor do they attempt to apply some theoretical/Political program or remedy to the phenomenon

[25] However, Pollner (1991) argues that ethnomethodology *has* settled down 'in the suburbs of sociology' and has thereby abandoned the 'radical reflexivity' that characterised early studies.

being studied" (Slack, R., personal communication). Lynch (1991) points out that indifference is not equivalent to denial – ethnomethodologists do not deny the existence of, for example, various forms of oppression but (*qua* ethnomethodologist) they attempt to explicate how such mechanisms work. As Paul ten Have puts it:

> For the DURKHEIMian strand in classical sociology, and social research more generally, the ultimate goal is to investigate 'social facts', and their determinants, where 'social facts' have the twin characteristic of being both 'external' and 'constraining' to the actions of individuals" (ten Have 2002, capitalisation as in original).

In contrast, in ethnomethodology, social facts are treated as accomplishments (Garfinkel 1967). As Pollner (1974, p. 27) puts it:

> …where others might see 'things', 'givens' of 'facts of life', the ethnomethodologist sees (or attempts to see) *process:* the process through which the perceivedly stable features of socially organized environments are continually created and sustained.

Social facts therefore exist only as a result of ordinary society members' practical actions, giving the latter pride of place and making them the focus of investigation. It is this interest in the endogenous methods of realisation of social phenomena from which ethnomethodology gets its name (Rawls, in Garfinkel 2002, p. 5). An important corollary is that ethnomethodology does not claim a privileged view of the world as its accounts are produced through the same methods that society members use to furnish accounts[26]. Its stance is not an ironic one that treats real-world accounts as being in need of (sociological) remedy. Formal analytic sociology produces accounts that compete with those provided by society members. Ethnomethodology's principles mean it does not theorise the matter, it does not substitute its own categories for those of society members. In sum, social order exists at all points (Sacks 1984). It is recognisably produced and oriented to by members.

In the next few sections, I will discuss a number of concepts that are at the heart of ethnomethodology and of great importance to corealisation. Even though it is not the main purpose of this thesis to provide an ethnomethodological investigation, I will

[26] While the same holds true for sociology (a fact pointed out consistently by ethnomethodologists, e.g. reference here), the discipline has so far found it impossible to accept this.

introduce the vocabulary and grammar of ethnomethodology. Its analytic mentality informs the study and practice of corealisation.

### 4.1.1 Accountability

When society members are involved in some activity they usually go about their business in a way that is observable and intelligible for others. This means that others can normally understand what someone is doing without having to ask, i.e. the activity itself is enough to account for itself. When we see a number of people standing at a bus stop in a way that they are arranged in a sequence, we can safely assume that this arrangement is not mere coincidence but is produced as people engage in the activity of queuing for a bus. We can then orient to this arrangement *as* a queue and act accordingly, i.e. if we, too, want to catch a bus, we will *join the queue* at the end. In turn, our joining the queue would be recognisable as such by the others who are already in the queue and so on. These very basic observations are what motivates the notion of accountability. As Suchman, Trigg and Blomberg (2002, p. 164) put it: "Our viability as members of the social world turns on our ability to make sense of the actions of others, and to make ourselves sensible to them". This is what Garfinkel (1967, p. vii) has described as activities being "visibly-rational-and-reportable-for-all-practical-purposes".

Our understanding of the world rests upon this character of everyday activities and at the same time underpins them, that is, they are reflexively related. Reflexivity[27] is central to Garfinkel's

> […] recommendation […] that the activities whereby members produce and manage settings of organized everyday affairs are identical with members' procedures for making those settings "account-able". The "reflexive" or "incarnate" character of accounting practices and accounts makes up the crux of that recommendation. (Gafinkel 1967, p. 1)

That accounts are reflexively tied to the social phenomena they describe is an empirically findable matter. In fact, the accounts members produce are constitutive of the phenomena they describe. Members' actions depend on and make use of this

[27] Reflexivity is a concept widely used and applied in the social sciences. It is important to note that for ethnomethodology, reflexivity is a members' practice first and foremost (cf. MacBeth 2001).

as reflexivity is the means whereby social phenomena are elaborated and made seeable as relevant for the purposes at hand.

### 4.1.2 Indexicality and Situatedness

Words such as 'you' or 'next year' are indexical as they derive their meaning from the context in which they are used. While in linguistics only certain kinds of terms are called indexical, Garfinkel (1967, also cf. Wieder 1974) has extended the concept to encompass all expressions, pointing out that pronouns are merely the most obvious examples of indexicality but that *all* expressions are indexical in the same sense if perhaps not in the same obvious ways. Tying the concept back to the discussion of queueing, we see that actions as well as expressions in a language are indexical, that they can only be understood as what they are in the context they appear in. For example, we might find it difficult to see a group of people standing behind each other in a line in the middle of the desert as a queue because an essential element of what a queue is about is missing: the something that people queue for. If we add a water hole to the picture we can start to see these people as people queuing although we might find it strange that they should proceed in such an orderly manner.

When ethnomethodologists speak about action they often refer to it as 'situated' (e.g. Suchman 1987) to denote the fact that they are speaking about the action taking place in a particular context, under particular conditions, with particular resources, etc. While any particular action can be described as an instance of a more general concept, it is the particulars that constitute the action, not its belonging to a class of actions. The phenomenon does not come into existence as an instance of an abstraction but is, if you will, worked up from scratch, *ad hoced* (Garfinkel 1967, p. 22), worked out in the here and now within this particular context. This is what is meant by 'situated action'.

The situatedness of action and indexicality of language pose a problem for sociologists (as well as others, computer scientists in particular) as they create an "essential tension" (ten Have 2002) between abstract concepts and particular, situated instances of action and language use. Sociology attempts to 'repair' this tension by substituting "objective (context free) for indexical expressions" (Garfinkel 1967, p. 4). Indexicality and situatedness cannot be "repaired" in the sense that the

world could be rendered more amenable to formal descriptions by, for example, inventing and imposing an 'ideal language' as envisaged by the earlier Wittgenstein and his colleagues in the Vienna circle and worked out by Wittgenstein in the Tractatus logico-philosophicus (Wittgenstein 1984). Wittgenstein himself later refuted his own, earlier work in the Philosophical Investigations (Philosphische Untersuchungen, Wittgenstein 1984). Wieder (1974) discusses the problems that structural semantics runs into in this respect:

> The sense of the title uncle is thus not restricted by the criteria for properly using the term but can instead be used as a vehicle for "recovering whatever can be said about uncles by the parties who say and hear the term. How members go about the task of seeing what the other is saying by relying on and actively developing a sense of what the other could be talking about has been a principal phenomenon of ethnomethodological interest (*ibid*., p. 133).

Indexicality is repaired for all practical purposes by society members with respect to the situation and purpose at hand. Members do not have a need for a generic remedy as their practical actions and their ability to say things in only so many words turns on the notion of indexicality.

### 4.1.3 The Documentary Method

When society members encounter a phenomenon that corresponds to a pattern, they take it as evidence or document of that pattern. As Garfinkel puts it:

> "The method consists of treating an actual appearance as 'the document of', as 'pointing to', as 'standing on behalf of' a presupposed underlying pattern. Not only is the underlying pattern derived from its individual documentary evidences, but the individual documentary evidences, in their turn, are interpreted on the basis of 'what is known' about the underlying pattern. Each is used to elaborate the other." (Garfinkel 1967, p. 78)

The mutual elaboration of the pattern and the evidences of the pattern (their reflexive relationship) is the crucial point here as it establishes how it is possible for ordinary society members to practically reconcile the tension between abstract notions or patterns and real-worldly phenomena, how a sense of social structure is possible.

## *4.2 Taking Ethnomethodology and Design Seriously*

In this section, I will relevance the core concepts of ethnomethodology discussed above to the work and study of IT systems design. Before I turn to the discussion of

corealisation, I wish to briefly mention the related work of Button and Dourish on what they called "technomethodology" (Button and Dourish 1996, Dourish and Button 1998). Like corealisation, their work aims to respecify the relationship between ethnomethodology and design, to change ethnomethodology's role from a practical requirements elicitation service to a foundational role in systems development. However, unlike corealisation, technomethodology is aimed at a respecification of our understanding of the human-computer dyad and how design might orient to it. It attends to the problem of how people using a system make sense of what its status is and the resources they have for doing this.

Corealisation takes seriously the challenge of technomethodology to make ethnomethodology an underlying orientation in systems development rather than just an add-on remedy to a specific problem. It does this by considering the role that ethnomethodology's analytic mentality and its programme of workplace studies can have in the process of systems development. This is how it differs from the idea of technomethodology which was concerned with the *product* of design and how it might incorporate ethnomethodology's principles.

With the focus on the social organisation of the design process itself, I now set out a programme for corealisation (Hartswood *et al*. 2002) with the aim of taking ethnomethodology's contribution seriously and radically respecifying what IT design work is about. The changes will not be cosmetic as has been the case too many times before but will be foundational ones that alter the *grammar* of what it means to be engaged in design work.

At the heart of this will be the question what an orientation inspired by[28] ethnomethodology can provide for people involved in IT systems development. The observation that people act in ways that make their behaviour observable, accountable has important implications for anyone interested in the social arrangement of working practices. Studies of work settings have illustrated this point numerous times and in very different contexts such as city dealing rooms (Heath *et*

[28] But not equivalent to. This difference is crucial as the concern here is not to further ethnomethodology as a social science project but to make its orientations and approaches relevant to IT systems design.

*al.* 1995), air traffic control rooms (Hughes *et al.* 1992, 1994, Harper and Hughes 1993, Suchman 1993), wastewater treatment plants (Bertelsen and Nielsen 1999, Bertelsen and Bødker 2001), underground line control rooms (Heath, Hindmarsh and Luff 1999, Heath and Luff 2000), general practitioner practices (Luff, Heath and Greatbatch 1994), or manufacturing plants (Robinson, Kovalainen and Auramäki 2000, Voß *et al.* 2002). These studies also highlight the fact that people use a number of resources to make their activities accountable, some directly connected to the task they are involved in (such as flight strips in the air traffic control room) and some designed specifically for this purpose (e.g. the shift books or diaries found in many manufacturing contexts).

This is in contrast with the view that there is an unavoidable difference between the understanding of workers (of their work) and designers (of information technologies) that is in need of some intermediary, either in the form of some representation they can both understand or through mediation by people who can somehow 'bridge' this divide (e.g., Williams and Begg 1993). Since this view is quite a common one that underpins a lot of work within PD and CSCW, it is worth spending some time considering its assumptions and implications.

Underlying this view is an assumption that workers and IT professionals are in possession of different bodies of knowledge relating to their relative areas of expertise. Further, it is assumed that these bodies of knowledge are not available to the other; I will refer to this as the 'communication problem' in design. While the reasoning underlying these assumptions may be an entirely practical one that recognises that neither party has the time (or interest) to acquire the other's expertise, it does remind one of the struggle within the social sciences to incorporate 'the native's view' in ethnographies. Sharrock and Anderson (1982) have discussed this assumption and its implications and have demonstrated how ethnography's problem disappears if the assumption is not made. Instead of insisting on the unavailability of the other's view, they suggest that one should study the ways in which understanding is achieved routinely, both amongst 'the natives' and between 'strangers' (in the Schützian sense, see Schütz 1973, pp. 134ff). I would suggest that a similar move can be of immense value for the project of IT systems development but it requires a rethinking of the associated practices and divisions of labour.

Underlying the formulation of the communication problem is the view that the specification of requirements needs to incorporate an understanding of working practice and that this understanding needs to be as good as possible (the user-centred design view). At the same time, divisions of labour separate 'design' and 'use' of systems, thus *establishing and maintaining* the communication problem by depriving designers of the chance to become familiar with the working practices of 'users' and the role the technical system plays in these. 'Designers' and 'users' remain 'strangers' (*ibid.*) who can communicate only through cleverly designed methods or mechanisms. Corealisation aims to break this self-sustaining circle by starting from radically different assumptions and working out their implications.

## 4.3 Design qua Member

At the heart of corealisation lies the assumption that *familiarity* can be achieved and *membership* established. Following on from the discussion of user-designer relations and design work as cooperative work in the previous chapters, I now aim to extend the argument by looking at the ways in which IT professionals as *members* of a setting can mobilise a stock of knowledge (Schütz 1973) that is not normally available to designers working in more traditional divisions of labour and how this knowledge can be brought to bear on and is, at the same time, constituted by the work of corealising systems in collaboration with non-IT professionals. I seek to investigate how members mobilise what they know and how their understanding of and being part of the 'natural history' of the setting also affords better opportunities for non-IT professionals to engage with the design process and to participate in meaningful ways.

One might refer to this as the 'pragmatics of design', not in a linguistic sense but as a placeholder for the situated everyday practical activities involved in doing design work which is reflexively tied to the unfolding biography (the history of events) of a setting. It is this reflexive tie that enables these designers ('corealisers') to do their work *qua* members. At the same time this very work is an essential part of their status as members of the setting, recognised by other members, knowledgeable and accountable. I seek to explicate the pragmatics of design by attending to the ways in which members in the workplace interact with each other, using their knowledge of

the ‘natural history’ of the organisation and its exigencies. This is an attempt to examine and unpack the ways design work is profoundly and irreducibly grounded in ordinary interaction. These interactions are reflexively tied to the ‘natural history’, i.e. they are at the same time based upon and constitutive of that understanding of how things have come to be the way they are and what paths for future development are open. I argue that a more detailed understanding of these situated practices of ‘doing design’ and the role of membership therein will be to the benefit of IT systems design, not just within ‘progressive’ design settings such as PD but wherever design work takes place.

Put simply, orienting to what people, both users and designers, know and use in and as a part of their work is essential to any discussion of design work but has been largely ignored in favour of a focus on process descriptions and the artefacts produced within design processes (UML diagrams, projects plans, etc.).

The ways in which a system comes to represent the real-world phenomena it is intended to deal with, i.e. the work of design, depends in fundamental ways on everyday, ordinary activities such as reviewing, sorting, prioritising, etc. It is a common fallacy to attribute the quality of a system to a prescribed or presumed methodology rather than the work involved in its production which draws on the method as a resource for its accomplishment. In a sense, the actual activities of producing a system get deleted from the system (cf. Suchman 2002) and its documentation and are replaced with generic descriptors defined in terms of a generic classification scheme representing a particular orthodoxy of design, i.e. some development methodology.

Instead of providing a ready set of categories and a “single, asituated, master perspective” (*ibid*.), the idea of design qua member is built on “multiple, located, partial perspectives that find their objective character through ongoing processes of debate” (*ibid*.). It is the familiarity with the natural history of the setting and the design that establishes order in the work of design. With this focus comes a consideration of peoples’ relationship with the setting and their relationships with others. Of crucial importance for current purposes is the notion of *membership*. The use of the term membership in ethnomethodology refers to the everyday, ordinary

and mundane competencies we all have that allow us to meaningfully interact with other people, i.e. it refers to our membership in the ordinary, immortal society that we all know. The notion of membership as used here is tied to Schütz' discussion of intersubjectivity which is summed up by Heritage (1984, p.54) as follows:

> Thus, rather than treating intersubjectivity as an essentially philosophical problem for which a determinate in-principle solution must be found, Schutz [sic.] treats its achievement and maintenance as a *practical* 'problem' which is routinely 'solved' by social actors in the course of their dealings with one another.

To be a member in/of a *particular* setting (a workplace, a particular group) then does not mean that one has to be officially part of that setting in some formal sense, e.g. having some membership card or being an employee of the organisation, but rather that one's presence in the setting is not surprising for others and can be accounted for by them in terms familiar to the setting. This does not mean that it is an unproblematic status conveyed simply by the act of being introduced but it is acquired over some period of time. The important point is not that a person achieves universal acceptance but that a sense of familiarity is achieved where the rights, obligations and reasonable expectations are established and where a degree of intersubjectivity is achieved.

For example, the notion of a student in a workplace, observing and learning about the work done in this setting, is one that is familiar to many people and it is therefore relatively easy for one to become accepted in a setting in this role and the rights, obligations and reasonable expectations are also easily established. Rights include such things as protection from harm (in dangerous environments), appearing and leaving at irregular times, being allowed free access to the workplace, its people, artefacts and events. Examples of obligations are non-interference with the workplace's normal business, politeness and confidentiality. Lastly, reasonable expectations in this example means that while a student may be expected to make coffee and help out occasionally they would not normally be expected to take on a full workload as their main business is observing and learning.

It is on common understandings like these that the notion of membership is built. Its achievement lies in a mutual display of an awareness of these conditions and a willingness to comply with them. Should a person transgress their roles in any way

that is not easily handled by the usual sanctions such as reminding people of their obligations or administering a reprimand, the membership of a person is threatened. Exclusion may follow in a number of forms such as being cut out of certain groups, not being given certain information or being excluded from the setting entirely.

The way that membership is used here can be potentially criticised as being naïve because it does not conceptualise the notion of conflict. Indeed there are many ways in which membership can be problematic and in which exclusion can occur. For example, my experience in the fieldwork setting might have been different if my skin had been of a different colour, if I had been a woman, if my abilities had been different or even just if I had not been already reasonably familiar with the local dialect people spoke. However, this is entirely speculation and misses the point of how people do become a member, how they establish whether or not they are welcome, how familiar they are with the setting or if they meet their obligations. While the details of the encounter might have been different, its grammar would have been the same and would have been recognisable to people as it is part of the normal ways in which we are all members of society.

In this way my use of the term membership is consonant with the way ethnomethodologists use it. The notion of membership does not denote some special status achievable by some and not by others but it describes the ways in which people are normally members of the everyday ordinary society with which we are all familiar. This society is decidedly not a theoretical one such as the information society or any other society construed by some sociological theory but a practical one that is empirically and massively available to all of us. In particular, phenomena such as discrimination and inequality are not merely concepts sociologists define and trade in but are part of the same life-world as are understanding and collaboration. As such, they should not be ignored when encountered but investigated and explicated. However, they should also not be seen as some magical analytical tool (like a sociological electron microscope) through which everything can be seen or by which everything can be explained. Rather, they are phenomena that are findable in the world we study and they can be explicated in the same way as other phenomena in this world.

## 4.4 Hybrid Knowledge

In order to do corealisation, does one require special training, does one need to become an ethnomethodologist or some other kind of social scientist? The answer can be found in ethnomethodology's treatment of methods: as there is no time out from members' methods, all activity is based on these, especially the activities of ethnomethodologists. They consist of the ordinary activities of observing, talking to people, taking notes, highlighting, ordering, reviewing, asking, etc. This is why ethnomethodologists would suggest that anyone can 'do' ethnomethodological investigations (Francis and Hester 2004). The methods by which people go about their daily business are massively available and they are available to everyone who has the opportunity and patience to study them. No special scientific equipment is needed, no electron microscope or radiotelescope. Neither are theories or special means for representation: ordinary explication in natural language of what is observably, reportably the fact is sufficient to uncover the social organisation of work[29].

What is required, however, is that corealisers acquire some of the knowledge and skills that members in the setting have acquired and use to do their work. Garfinkel refers to this as hybrid knowledge (Garfinkel 1986, Garfinkel 2002, p. 100 ff.). While some ethnomethodologists have invested significant time and energy to become competent in areas such as mathematics (Livingston 1987), law (Burns 1986, cited in Garfinkel 1991), playing jazz piano (Sudnow 1978) or Buddhist debates (Lieberman 2003), for the corealiser it will normally be sufficient to get acquainted with the relevant subjects 'on the job'. After all, the aim is not for them to become competent practitioners themselves[30] but for them to be able to appreciate what that competence would consist of, i.e., the knowledge and skills required by the practitioner to do their work competently (from the point of view of other members).

---

[29] This is not to say that what can be observed is all that is going on or all that is interesting, the skills involved in peoples' work are often not directly visible in any particular situation but can usually be uncovered over time in the same way that members learn over time what is involved in the skilful and competent accomplishment of work, i.e. through repeated observation in a range of (slightly) different situations.

[30] A similar orientation is formulated in Bødker et al. (1987, p. 263.).

While training may help in understanding complex areas such as microbiology or astrophysics, it is the practices observable in the setting that are of most interest and it is these that are often *not* made available in formal training. Consequently, the concept of hybrid knowledge should not be misread as meaning that what is called for is people who are IT professionals and have studied another subject such as microbiology to make them bio-informaticians or some other 'x-informaticians' (as much as that might be a potential advantage). Rather, what is called for is that IT professionals show greater *appreciation* of what goes into the work that other people do, of *what it is that they know and use.*

In a similar vein, corealisers are not called to produce ethnographic accounts of their observations that would add to the academic literature, i.e. they do not need to be ethnomethodologists. They should, however, engage in a practical form of ethnomethodologically inspired observation, in what Anderson (2000) called a 'practical sociology'. An important part in this is that they should adopt, as far as possible, a stance of methodological indifference towards concerns of sociological theorising, for example, the discourse around business process redesign (BPR) and its attendant benefits (cf. Randall, Rouncefield and Hughes 1995). With regard to BPR this means that *qua* corealiser one should not reify what goes on in the setting in terms of BPR's notion of 'process' as BPR does not provide, in any way, a privileged way of seeing the world.

The aim is to not ironicise members' methods and their common-sense understandings but to take them seriously and treat their achievement as the phenomenon of interest. This kind of orientation will help corealisers to gain the kind of detailed, appreciative understanding of what people do and what they know and use in their doing their everyday work that I would suggest is ethnomethodology's unique contribution to the project of corealisation.

## 4.5 Uniquely Adequate Systems

Corealisation aims to develop 'uniquely adequate' and thereby work affording systems[31]. That is, systems that enable rather than hinder work and are designed on the basis of an understanding and appreciation of how work gets done. This does not mean that there are hard and fast rules about what a uniquely adequate systems is or, indeed, that a system that is uniquely adequate today will be seen in the same way tomorrow. There is, however, an underlying grammar as to what can be seen as a uniquely adequate system and what can not. The important point about 'unique adequacy' is that it is a judgement 'from within' rather than an attribute of a system that might be assessed by outside experts (cf. Garfinkel and Wieder 1992, p. 182). It builds on an understanding of just what doing a job consists of, just how a fellow member of a setting would recognise a particular activity and its skilled accomplishment by another member.

The CSCW and PD literature gives us some ideas of the kinds of concerns people will orient to when making a judgement about a system's adequacy: Does the system get in the way? Does it hinder the way things get done around here? Have we been involved in its development? Are we in control of its operation or do we trust the people who are? Do the benefits justify the costs? Are the compromises we made reasonable? Are we happy with what we see as the likely future development of the system? These different questions open up a space of concerns far beyond the normal concerns of a-priori systems design and 'expert' evaluation. It is a corealiser's job to orient to these concerns and to bring to bear their understanding of peoples' working practices, their concerns and the history of the place.

While it may be relatively easy to make a judgement about the adequacy from a *local* perspective, there will inevitably be situations where wider concerns need to be considered. Formulations of requirements and implementations of systems are usually outcomes of negotiations between different parties with different concerns. There are situations where the possibility to influence decisions may be limited, for

[31] I use the term 'affording' in the same sense as Anderson and Sharrock (1993) that stresses the role of the social organisation of our environment that allows us to see the objects that populate it as *meaningful* objects.

example where developments are driven by outside requirements or where the rigidities of existing arrangements leave little room for choice. Corealisers have to draw on their understanding of this wider context to bring about closure so that development can become possible. The aim should be that such closure is based on an agreement by the parties involved that the way to progress is a reasonable one. It is important to note that corealisation does not call IT professionals to become 'advocates' of a particular cause. It asks them as professionals to make their own decision but suggests that this decision is best made on the basis of a thorough understanding of the setting and of how the orderliness of its work is produced. This is in contrast to, say, BPR's assumptions that practices found are either inefficient and in need of remedy or instances of 'good practice' that *can* and should be extracted and codified (Randall, Rouncefield and Hughes 1995).

The concept of unique adequacy acknowledges that there is no ideal solution to any given problem and that views about a particular choice may change. It is up to the corealiser as a responsible professional to reflect on the adequacy of arrangements and to take appropriate, responsible action. A uniquely adequate system will then be a system that is *demonstrably related* to a sequence of choices made on the basis of a member's understanding of the situation at the point when the choice was made (i.e. these choices can be seen as being reasonable ones by other members). Again, this should not be seen as a commitment to accept the system in its current form once and for all, in the sense of 'sign off', but as an eminently revisable statement that will be seen in the light of the question "what next?" and will allow people to progress in their work.

It is not that the concept of unique adequacy makes new, extraordinary demands: rather, traditional approaches have ignored them or have separated them out into different disciplines such as design, operation, maintenance. Neither is it the case that the concept of unique adequacy ignores traditional concerns such as usability – rather, it puts them back into context and demonstrates the ways in which they become practically applicable. Examples of systems that are directly at odds with working practices are available in the CSCW literature (e.g. Bowers, Button and Sharrock 1995 or Heath and Luff 2000, chapter 2). The concept of uniquely adequate systems aims to provide corealisers with an orientation to systems design that will

help to take practical action towards configurations of artefacts and working practices that are more work affording.

## 4.6 Unique Adequacy of Methods

The means by which unique adequacy can be achieved vary and will have to be chosen according to the 'case at hand'. Rather than planning activities in advance, without knowledge of the context and employing a predefined methodology, corealisers are called to constantly reflect on the methods employed and ask themselves: "what's next, how can we progress from here, given these circumstances?" This is inspired by Garfinkel's maxim of unique adequacy of methods (Garfinkel and Wieder 1992), which requires studies to

> […] operate *from within* the competency systems they describe. Accordingly, their descriptions of orderly and socially organized inquiries do not present an opposition between the practices described and the practices which make such description possible. (Lynch 1985, p. 6)

One might say that the phenomenon of interest provides the methods for its study and that any methodological apparatus brought in to transform and to replace what the phenomenon provides would only serve to produce a *methodologically ironic* version (Anderson and Sharrock 1983) of it, i.e. the account produced would compete with the naturally available accounts in and as of the phenomenon itself.

I wish to illustrate the principle of unique adequacy of methods using three practical examples: the use of representations, the use of a 'sandbox' to provide the space for corealisation in a controlled environment and choices around the negotiation of requirements.

The usefulness of a representation in design does not necessarily depend on how closely it matches what it represents but it depends on how it is used. A rough sketch used *as* a rough sketch evoking associations and connecting with members' rich understanding of the task at hand and the context, the biography of the place, is infinitely more useful than an detailed, abstract (detached) model of, say, the data flow. This is not to say that models are superfluous but one has to think about how they are used.

Button (1993, p. 38) makes the crucial distinction between the use of representations to support the "process" of system development and their use to support the documentation of that process and its outcomes. In his study he also observed that the developers used only about 10% of the functionality of a CASE tool. While one might argue that this was due to their lack of familiarity with the tool, an alternative interpretation is that it demonstrates the developers' reflection on its adequacy for the task at hand. After all, they did have a dedicated person on the project to support them in its use by providing on the job training.

It will not be possible for corealisers to be intimately familiar with all the places where a system might get used or know all the people involved. Not all IT professionals will be involved in corealising. The usefulness of representations of working practices is therefore undiminished and the artefacts and methods developed in the PD community should be part of the repertoire that corealisers can draw upon to explore the "unfolding horizon of design questions" (Trigg, Blomberg and Suchman 1999). However, they should be treated as what they are: "interpretations in the service of particular interests and purposes, created by actors specifically positioned with respect to the work represented" (Suchman 1995a, p. 58).

Trigg (2000) describes the use of a 'sandbox', a prototype system containing 'real' data. This kind of approach allows workers to experiment with new possibilities without fear of upsetting existing arrangements. The change from an existing system to a new one can occur at a time and in a way that the risks involved are minimised. Trigg (*ibid.*) describes how in his case the change was aligned with a time in which there was known to be a dip in activities in the organisation and how arrangements were made to keep parts of the existing system in place, allowing members to change over gradually and have a backup should things not turn out as hoped. Choosing a particular course of action has to be done in the light of the circumstances and an evaluation of the likely outcome under those circumstances. This is where corealisers can bring to bear their knowledge of the setting which can be checked against the understanding of other members.

Another example of unique adequacy in the choice of method provided by Trigg (2000) concerns a choice concerning the management of requirements and how they

get negotiated. Some members of the organisation had an urgent requirement for personalisable letters to be generated by and stored in the system while others were in favour of arranging meetings to discuss the design of such a facility. By opting for a solution allowing the 'owner' of a particular document type to determine the extent to which documents should be personalisable, Trigg effectively got around the potential problems of going ahead quickly without achieving universal consensus first. Knowing (in a practical sense) what will be acceptable and under what circumstances can enable corealisers to make choices about how to progress.

The above examples demonstrate how corealisers can be called upon to make practical decisions about the methods they employ (using representations, bracketing off technological development from operation, managing the ways that requirements get negotiated). At the same time, they demonstrate how the knowledge of the setting and the availability of its resources enable corealisers to make precisely these choices in a way which is adequate (Hartswood *et al*. 2000, Trigg 2000).

## 4.7 Corealisation's Principles

In summary, these are corealisation's principles: it places the IT professional into the work setting as a member amongst others and hence locates the work of developing IT systems in this context and makes it a normal, ordinary feature of it (Hartswood *et al.* 2000, 2002). Through 'being there' in the workplace and sharing the experiences and concerns of fellow members, corealisers are able to realise the vision of 'design in use' (Henderson and Kyng 1991). Instead of trying to provide a (necessarily narrow) channel linking design and use as separate activities, it leverages day-to-day interaction in context and operationalised the everyday skills of designers as tools in the design process: observing, reporting, asking, demonstrating, and so forth. It is this this commitment to a long-term, *direct* engagement between design(ers) and use(ers) that sets it apart from the use of ethnographic approached in design as well as some participatory design methods that rely on mediation or sporadic participation in workshops or focus groups. This is not to say that simply 'being there' will solve the problems of systems development, nor that it will replace activities such as drafting and agreeing requirements specifications. Taking these activities into the context of use will enable corealisers to find answers to the practical question *par*

*excellence*: “what next?” Working up these answers can be done in a way that is accountable, i.e., it takes place in a context in which all stakeholders are involved and where they have the resources to participate in the process of considering options and making decisions.

Corealisation is not a methodology but a principled orientation to design based on participatory design and ethnomethodology. It is agnostic with regard to development techniques and tools but recognises that in calling for a uniquely adequate solution to a workplace computing issue, the methods chosen must also be uniquely adequate – it is, therefore, realistically permissive and allows the use of any method that is consonant with corealisation’s principles (i.e. extreme programming but not business process reengineering). Corealisation requires the IT professional to become a member of the workplace setting, i.e. to acquire some of the mundane competencies that other members of the setting have, to know what others in the setting know and use. As explained in section 4.3, the requirement is one of familiarity, not one of complete (in some sense) knowledge. Design should be undertaken on the basis of this familiarity, using it as a resource and further developing it in cooperation with others.

It calls for work to be afforded: changes in working practice should not be pushed or enforced by technology, design should not be an exercise in re-engineering work. Rather, it should seek uniquely adequate solutions for particular settings. Unique adequacy refers to the social order that can be found in the setting (whatever social order people have established) rather than prescriptions of how work ought to be organised. The assumption is that workers know about their work and that systems can be developed in partnership predicated on what workers know and use. Corealisation encourages reflection on work and artefacts that afford it, attending to the ways in which work with and on IT gets done and using this as a resource. It does not privilege the ‘designer’ but recognises the different knowledges and skills that both IT- and non-IT workers have and contribute to design.

Politics are a members’ problem and should be treated as another worldly phenomenon in the same way as other phenomena. As an approach, corealisation does not offer a political programme but it places emphasis on responsibility and

accountability, not in a formal sense but an everyday one of being responsible towards fellow members and accountably, seeably doing ones' work properly within a working division of labour. It opens up a space where trust can be fostered, where commitments can be accountably made and risks taken on the basis of a shared understanding of what they are and how they are managed.

Corealisation is not a panacea. It has the same problems that other approaches have but strives to create awareness amongst members of them so that they might practically cope with them. It does not present a methodology but makes the consideration of methods central to its concerns. Corealisation requires space, both in terms of a physical space where people can work with each other and an organisational space where it is possible to make design a pervasive activity rather than one that is confined to meeting rooms or project spaces. It makes the process of doing design more explicit and includes in the notion of design activities traditionally seen as being outside its scope such as operation and maintenance.

This is based on a recognition that the true implications of design can only be learned in use and that any solution is a temporary one that may need to be revised in the light of what is learned or in response to changing circumstances. Consequently, corealisation tries to maximise feedback from use and make it available as early as possible by encouraging the use, in everyday practice, of prototypes and partial systems that get incrementally refined. While it does not aim specifically to develop solutions that travel beyond the context in which they are originally developed, it acknowledges that successful solutions in one place will often be taken up and appropriated elsewhere.

It is in this sense that corealisation takes ethnomethodology's approach and findings seriously and brings together its analytic mentality with a practical, participatory design orientation in order to achieve 'design in use'. In this respect, corealisation can be seen as a principled synthesis of ethnomethodology and participatory design (Hartswood *et al.* 2002).

# 5 Case Study: Production Work at EngineCo

This chapter presents findings from an ethnographic study of everyday work practices in a manufacturing plant. After introducing the methodology, I first present an account of work in the control room of the plant, focusing on the work of organising production. This then provides a basis for an account of IT work in the setting. Local arrangements of IT systems provision and management are arranged so that IT professionals are located within the production plant and are involved in its day-to-day operation. I will show how they make use of their knowledge as members of production work to organise the work of building and operating IT systems. This orientation to what people know and use will help to uncover the 'hidden work' in IT systems design, the kinds of practices that are still not acknowledged by much of the textbook literature. These two elements of the ethnographic study combine to provide a background for the discussion of my own work as a corealiser in the setting which is presented in the next chapter.

## *5.1 Methodology*

The fieldwork conducted as part of this thesis[32] has two different but related strands: ethnomethodologically inspired ethnographic observation and corealisation. The former is concerned with producing a rich picture of the setting and the phenomena observed while the second incorporates an element of intervention, so the researcher loses his or her passive stance and actively engages in and in fact changes the setting. I would argue that the two stands are actually well placed to inform each other. Ethnomethodological enquiry does not presuppose that the researcher be 'a fly on the wall' but even encourages researchers to devise methods to illuminate a phenomenon and make it more amenable to investigation, an approach most famously taken by Garfinkel (1967) and his students in devising 'breaching experiments' (*ibid.*). Corealisation, on the other hand, places much emphasis on the role of the IT professional (or researcher) making use of their mundane competencies in learning about a setting and using the knowledge thus acquired to design systems.

[32] To avoid possible confusion: this section is concerned with the methods of *studying* work practices rather than with corealisation's methods.

While I would argue that there is no in-principle problem in combining an ethnomethodologically informed ethnography and the work of corealisation, this does not mean that anything goes. In addition, there are entirely practical considerations that will have an impact on the practical choices made about the adoption of methods. Although the study had a design aim from the start, the first few months were spent conducting an ethnographic study and not undertaking any design activities, i.e. the interest to inform the design of some system was suspended (cf. Randall, Hughes and Shaprio 1994, p. 246). There are two basic motivations for doing this: first, one would not want to start designing without having established an understanding and appreciation of the work that does on in the setting. Second, as corealisation is grounded in the notion of membership, this status needed to be achieved first and while it may be difficult to say at what point membership may be said to be achieved, it is certainly not possible to take it for granted that just getting introduced is enough. Acquiring the status of a member involves building up a degree of trust and mutual understanding and an ability on the part of the designer/researcher to demonstrate some of the competencies that members have. Tied to this is the work involved in understanding not just what things are like just now but also how they have come to be and being able to recognise what needs to be known (for all practical purposes, as a competent member) and what can be bracketed off.

Consequently, the interest initially was to find out about control room work *per se* rather than as work relevant to design. The method employed was ethnomethodologically inspired ethnographic observation (see section 3.1, chapter 4.1, also cf. Hughes *et al.* 1992, Rouncefield 2002), which calls for the fieldworker to become involved in the setting and to acquire an understanding of its phenomena that is, as far as possible, identical with the understanding that members have of their own activities. This means that observations should not be subjected to researchers' reformulations in terms of theoretical preconceptions (i.e., methodological irony should be avoided, cf. chapter 4.1, especially page 89). The ethnographer's job is to

> listen to the talk, watch what happens, see what people do when, anywhere, to write it down, tape it, record what documents can be recorded, and so on. The sorts of things that can be collected and recorded include: memos, notices, graffiti, transcripts of meetings, war stories and more. It is not that such

> materials have any intrinsic value; the material is valuable insofar as it can be made relevant or useful for what it can say about the social organisation of activities. (Rouncefield 2002, p. 76)

While direct observation was the main activity, it also was a prerequisite for other activities such as asking questions as it enabled me to develop the sense to ask them (cf. Whyte 1973, p. 303). As Rouncefield (2002) points out, the problem is not so much one of uncovering something that is hidden but to cope with the mass of observations in a way that allows one to become familiar with the setting. He suggests a number of rules of thumb that can help with this:

> …what kinds of things do [members] take for granted or presuppose in going about their work, what kinds of things do they routinely notice, what kinds of things are they 'on the lookout' for, how do they 'tune themselves in' to the state of being 'at work', what are the constituents of their 'serious frame of mind', how do they react to the things that occur within their sphere of attention, what objectives are they seeking to attain in their reactions to whatever occurs, and by what means - through what operations - will they seek to accomplish those objectives in adaptation to these unfolding circumstances. (*ibid*., p. 83).

The researchers' understanding will be built up over the course of the engagement with the setting, will be revised in the light of new observations and how they relate to observations already made. At any point in time, what is known about the setting and the phenomena it provides is potentially revisable, so part of the work of conducting an ethnographic study involves checking the understanding gained thus far against new observations or against members' understanding.

In line with ethnomethodology's unique adequacy requirement (Garfinkel and Wieder 1992, p. 182, Garfinkel 2002, p. 124), the methods chosen at each point were chosen in the light of the circumstances and the phenomenon that I was focusing on. For example, because work with the central production control system was difficult to observe in real-time because of the speed with which workers went from screen to screen in the application, a video recoding was made and analysed 'offline' by the researcher at a later time[33]. Similarly, some phenomena were not directly or fully

[33] Because the control room workers were doing shift work and could not easily attend meetings, I was unable to include control room workers themselves in the process of analysing the recordings (cf. Karasti 2001). However, I generally made use of the availability of informants to check my understanding of their work.

available (e.g. phone conversation) to the observer, so additional means had to be employed to investigate them. This could involve impromptu discussions, interviews or the study of various documents to retrieve aspects of the phenomenon not available through direct observation.

Following the initial period of observation, discussions with control room workers established a number of areas of interest for design. That is, the choice of what kind of design intervention should be made was informed by the understanding gained of current work practice and was taken in collaboration with control room workers and in the light of their interests and concerns. Development of the shift book application was undertaken in the control room itself to make the work of design available for control room workers and give them the opportunity to get involved in making design decisions. The process was different from cooperative prototyping approaches in that by locating the design work in the control room, opportunities were created for more routine and occasioned rather than planned interaction.

The work of observing and recording working practice was continued throughout and was used as a means to reflect on the state of play and inform further design decisions. Wherever possible, use was made of opportunities to get involved in the organisation's work and help out in small ways by, e.g. going down to the shop floor and 'hunting down' missing parts. These activities were only sometimes instructive but they served to establish and maintain a role within the organisation that was crucially different from that of the scientist or the management consultant. In this respect, being available, reasonably competent and willing to help with all sorts of problems was a key part in establishing a trust relationship and maintaining 'access'.

In early 2002, I was asked to relocate from the control room to the office of the plant systems management group. The reasons for this were to do with staffing issues and concerns about my role in the organisation. It was felt by the control room manager (who is also head of the plant system management group) that my presence and my activities would be easier to account for organisationally if I moved. The control room was by no means 'off limits', so I was able to continue my work on the shift book, although the affordances of 'being there' were lost to an extent. At the same time, the relocation brought with it the opportunity to get more involved in the work

of the plant systems management group which is concerned with the use of IT *throughout* the plant. Consequently, the scope of my work widened as I started getting involved in the design of system used outside the control room, for example, on the shop floor.

In summary, the methods employed and the particular aspects studied changed a number of times. In a longitudinal study this is not surprising and the unique adequacy requirement calls us to match what we do and how we go about retrieving phenomena and making them available for analysis to the situation at hand.

### 5.1.1 A Note on Interdisciplinarity

As is by now the accepted rule rather than the exception in areas like HCI, CSCW or PD, this thesis draws on the insights gained in the social sciences. This opens up interesting problems to do with the commitments of and boundaries between various disciplines which are often contested. Shapiro (1994) raised the question to what extent social sciences would have to give up their 'essentialist' positions and make various kinds of compromises when they get involved in any way in a project like design. In the end, for the designer (or researcher in computer science), what counts in design are the practical findings rather than the promise to provide 'essential truths'. In so far as this is the case, one can reasonably draw on the findings of studies, especially those which are 'methodologically modest' (Slack, personal communication), especially where they answer questions arising in relation to design that are not easily or sufficiently answered by ethnomethodological studies. This does not necessarily require a commitment to the methodological stance on which they are based and should be careful about making sweeping statements. Despite all that labour process theory has to say about them, managers and workers are not inevitably tied up in conflict and they can both the found to be quite sophisticated in working up arrangements that make day-to-day work not just possible but also bearable. At the end of the day, what may be necessary is that designers and computer scientists themselves become more sophisticated and knowledgeable about the phenomena of social order and human behaviour so that they may divorce their project from that of social science disciplines and get on with it rather than getting

entangled in border disputes and vendettas of other disciplines. This kind of hybrid knowledge is what underpins corealisation's project.

## 5.2 EngineCo and its production orthodoxy

EngineCo is an independent manufacturer of diesel and gas powered engines with a long and rich history going back right to the earliest days of motorisation. After a phase of expansion into various areas including the production of agricultural and construction equipment, industrial production plants, and even aircraft engines, the company has had to scale back significantly and now concentrates entirely on the core business of producing diesel and gas powered engines from 4 to 7,400kW. The thesis describes work undertaken in the 'new plant', a plant built in the early 1990s building diesel engines in the 11 to 190kW range[34], based on a small number of basic engine 'types'. These engines are used by EngineCo's customers in a variety of applications such as trucks, tractors or generators and get specifically tailored to these particular needs and according to customer specifications. The following sections describe the production process as originally envisaged and as it is inscribed in the gross physical structure of the plant and in the various organisational arrangements and IT systems. While the basic layout of the plant has not changed, recent years have seen a number of extensions to the plant, some of which have affected the path an engine might take through the production process. Some of these extensions will be discussed in section 5.4.2.

[34] These statements refer to the state of the plant at the point of observation in ca. 2000/01.

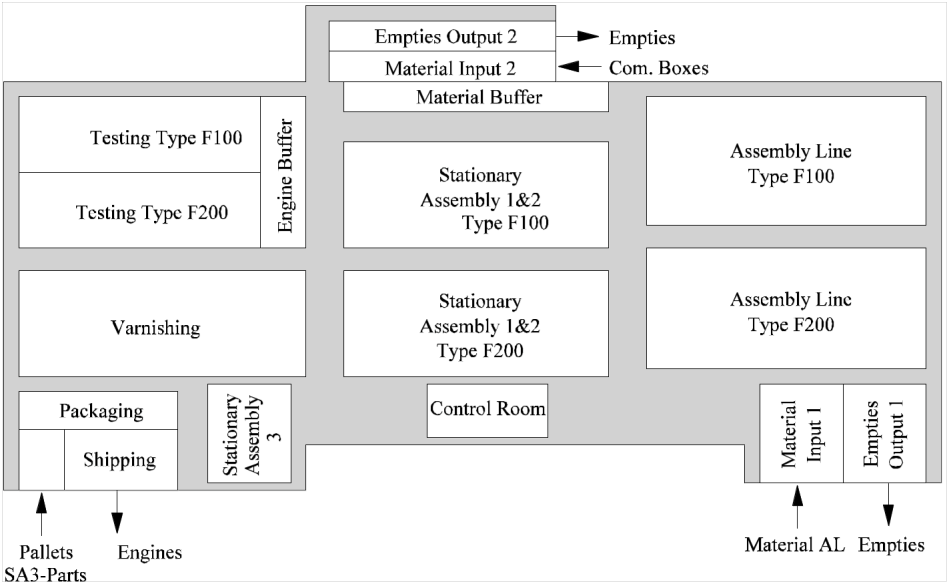


**Figure 2 The production plant layout (as of November 2000). Note the three areas where deliveries and pickups are made and the central location of the control room.**

In order to achieve both economies of scale and a product closely matched to customers' requirements, the chosen production approach is one of 'mass customisation' where all engines are built based on the same basic engine block produced in a highly automated production line but are configured according to specific requirements in a next, largely manual (and thus flexible) production step. This work takes place at workplaces in 'stationary assembly' where individual workers complete the engine configuration. Subsequent steps such as testing and painting are again automated to a larger degree, underlining the underlying philosophy of combining the benefits of automation and automated production control with the skills and flexibility of well-trained production workers.

The overall goals of efficiency, flexibility, quality and reliability of production are translated into a number of specific goals relating to different aspects of such diverse areas as marketing, product development, logistics, acquisition of parts and components as well as production layout and practices. All these areas are affected by the principles at the heart of EngineCo's production orthodoxy:

- The products are expected to be characterised by low variance in the basic engine and customer-specific configurations tailoring the product to the specific requirements of the application context. Customising has to be extremely flexible and responsive to meet customers' high demands.

- At the same time, the number of distinct components parts is to be reduced to a minimum by a strict parts management regime that would seek to eliminate unnecessary variations in parts by encouraging the use of parts already sourced by the company for similar purposes.
- Parts and material are delivered to the plant 'just-in-time' by a logistics subcontractor serving parts from a high-shelf storage facility nearby, thus largely removing stock control from the shop floor and placing it under central, computerised control. The plant itself has only limited buffer space for component parts that lasts for about half a shift (4 hours).
- A highly trained workforce together with the co-location of managerial functions within the same building (in offices overlooking the production plant) is to lead to short communication paths between the shop floor production processes and related planning, administrative and managerial functions. The 'new plant' embodies a vision of a new culture for the organisation.
- Routine tasks are largely automated and the overall production process overseen by a central assembly control host system that also links the plant with the enterprise resource planning (ERP) system that controls stock levels and the administrative processing of customer orders.

The layout of the plant can be seen in Figure 2. Parts and material are delivered to the plant by the logistics provider who also picks up finished products and empty containers. Their roll-on / roll-off trucks can be emptied or filled within a matter of minutes, normally only requiring the driver to initiate the process. The transfer is then automatic from a conveyor belt in the truck to a receiving belt in the plant or vice versa. All delivered goods are automatically checked in and delivered to their destinations within the plant by infrared-guided autonomous carriers. Parts for the basic engine are normally delivered to buffer spaces running alongside the assembly lines from which they can be readily delivered to the workplaces. Parts for customer-specific configurations are 'picked' by the logistics provider, i.e. the exact number of parts required for a particular engine is placed in a commissioning box which is then delivered to the plant and stored in a material buffer until needed.

Production starts with the assembly of all basic moving parts on one of two assembly lines. At this stage, the number of different engine configurations is relatively low and so assembly can be automated at many stations. The engine is moved from the assembly line to its next station in stationary assembly I/II by autonomous carriers. Stationary assembly I/II consists of 70 workstations where workers configure engines according to customer orders, reflecting the specific requirements of the various applications engines will be used in. All parts that need to be fitted before an engine can be tested for its conformance with the specification are attached here before the engine is taken (again, by autonomous carriers) to the testing field. It is subjected to a test cycle that allows not only to establish that an engine functions properly but also allows parameters to be adjusted to meet the specification. As engine performance is guaranteed in a range of different use situations (e.g. in a truck or in a generator) and engines are tailored to achieve optimum performance in these, the activities in the testing field are not only important for quality control but constitute an important part of the production process itself. After testing an engine may require additional parts to be assembled and this is again done in stationary assembly I/II. On the way to and from the testing field, the engine passes through the engine buffer, the only designated place in the plant where engines are stored between consecutive production steps. Now, the engine is ready to be varnished and is taken to the varnishing section, where it is first prepared (i.e. by removing oil and sealing access points so that varnish or cleaning fluids can not enter the various internal sub systems such as cooling or lubrication). It is then varnished – first by a robot and then in detail by a worker – and dried. The last production step is normally packaging and shipping. However, some engines may need to be audited or may have defects that need to be repaired. Such engines can be moved to stationary assembly III where workplaces are available for such cases. The engine leaves the plant on the same kind of truck that originally brought the component parts and is taken to the logistics provider's high-shelf storage facility before being shipped to the customer.

The whole production process is controlled by the assembly control host which is at the heart of the plant's IT infrastructure and links the various subsystems that control individual functional units within the plant. It also connects the plant to outside

systems such as the company's enterprise resource planning system. The assembly control host receives packages of production orders from the ERP system with a normal lead time of 1-2 working days. These orders are then locked within the ERP system which means that they can no longer be modified by parties outside the plant (e.g. the sales department) and are completely under control of the assembly control host. This mechanism of hand-over of control establishes the boundary of the plant's IT infrastructure and the rest of the company's systems. Before production of an order commences, the assembly control host orders the required parts to be delivered by the logistics provider who sends an acknowledgement and thereby guarantees that the parts will arrive at the plant within four hours. When the goods arrive at the plant, the local system at the goods entrance sends a message to the assembly control host which then orders their further transport by the carrier system. Every production station receives the data it requires from the assembly control host which also initiates the transport of material. All subsystems operate asynchronously, making the overall system less vulnerable to failures and easier to change.

## 5.3 The Control Room

Overlooking much of the shop floor is the central control room where three workers monitor processes in the plant (see Figure 3). Their tasks are to monitor the flow of engines and material and react to incidents, regulate the processes in the plant, make adjustments to scheduling when necessary and to forward information to maintenance, foremen, and the logistics provider. They also create statistics about problems and liaise with Assembly Planning, Disposition, and other departments. The overall aims are to sustain a high and even utilisation of production capacity while meeting all delivery dates and reducing work-in-progress to a minimum. Note that these aims are often in conflict, so practical trade-offs need to be made at any point but with a view to likely future developments. The control room workers are supported in this task by a number of IT systems:

- The assembly control host contains all relevant information about production orders and component parts. With this system, control room workers can influence the flow of engines through the plant. For example, they may adjust

priorities or stop engines that require attention, ‘parking’ them in the engine buffer.

- A process visualisation system collects and visualises about 15,000 signals that are generated on the shop floor. It is the main means for detecting process breakdowns.
- The control host for the autonomous carriers is a separate system that allows workers to monitor and, if necessary, modify transportation processes in the plant.
- In addition, control room workers have access to the central ERP system on the company’s central mainframe computer.
- PC-based office software including text processing, database management and spreadsheet functionality are also available. A number of smaller systems and local artefacts are realised on their basis.

Control room workers are engineers or skilled workers who have worked in other positions in the plant. Training new members of the group is done on the job since a large part of the relevant expertise is not easily described outwith the context of its use. Since production processes in the plant are largely automated or delegated to autonomous workgroups, the flow of engines through the plant is normally controlled by the assembly control host only, requiring no human intervention other than by the assembly workers. Under normal circumstances, orders that enter the plant have all their preconditions met and are routed through the plant by a priority-based scheduling algorithm. Since the scheduling decisions to be made are relatively simple (they concern only usage of testing stations, buffers and varnishing), the automatically generated schedules are suitable under normal circumstances.

**Figure 3 The Control Room IT systems from left to right: control host for autonomous carriers, general computing and assembly control host (via a limited number of terminal emulation windows, process visualisation system.**

Nevertheless, there are a sizeable number of orders that cannot be handled according to the automatic schedule as they deviate from the normal case in some way and thus require attention. The handling of such special cases, monitoring production, and responding to breakdowns make up much of the daily activity in the control room. The following entries from the shift book[35] illustrate this:

> *Engines #179864, 179869, 179791 control units for these engines are in Mr Muller's office, engines stopped before varnishing*
>
> *As soon as crankcases for 4-cylindes are available, schedule order number 785654 (very urgent for [company])*

Control room workers have to ensure that the control units are assembled and that the engines are re-introduced into the normal production process afterwards. Some orders have tight deadlines and thus need priority treatment. Also, certain customers'

[35] Entries have been translated from German and anonymised.

orders may receive priority treatment for a variety of reasons, a fact that is not recorded in the IT systems but is part of the workers' stock of knowledge.

> *Info for Peter: part no 0659255, crate was empty on delivery, so I booked 64 parts out of the inventory*
>
> *Engine 179867 built twice: Mr Meier downloads a similar order to the assembly control host, new engine number 179890 has to be engraved, Mr Schmitz will see to it*

Here a worker responds to a breakdown in material supply by re-establishing the match between stock count on the shop floor and the recorded information. The second entry illustrates another situation in which the fit between the shop floor situation and the information recorded needs to be re-aligned as an engine is built twice by mistake.

> *Repeated problems with carriers: #37 runs out of battery before getting to the charging station*
>
> *Please move as much material as possible into the plant during the late shift: roll-on/off blocked tomorrow 8:00 to 11:00 (see fax)*

The above entries show how control room workers have to respond to general engineering and organisational issues that affect the processes within the plant. These examples give a flavour of the kinds of problems that control room workers attend to.

While there is no official functional division of labour in the control room – everyone has in principle the same day-to-day responsibility – there is a working division of labour (Anderson *et al.* 1989) which is negotiated locally and subject to revision according to needs. A practical division of labour is normally established at the start of a shift or beforehand when the shifts are planned (who will work when), a process that takes into consideration a number of factors such as a need to ensure that at least one senior member of staff is on each shift. Certain kinds of tasks can be assigned, for example, responsibility for one of the two strands of production, relating to the two production lines (see Figure 2, p. 115) and the types of engines produced there.

A task that usually gets assigned to a single worker is that of scheduling trucks delivering parts and picking up empty containers and finished engines from the plant. The trucks and drivers are provided by the logistics provider but transports are

scheduled by the control room workers. A simple paper form is used keep track of transports, when they were scheduled, what type of transports was requested, when the truck arrived at the plant, if the truck was sent directly to another plant rather than to its base, driver notifications of traffic jams, etc. The layout of the form affords 'seeing at a glance' (cf. Heath and Luff 1992), i.e. by looking at the last few entries, workers can quickly pick up 'what the situation is'. This is important for the person responsible for scheduling as this task is an ongoing concern throughout the entire shift which requires only occasional action and it interleaved with numerous other tasks. It is also important for colleagues who will provide cover during breaks or other absences from the control room. The scheduling form can be seen towards the bottom right in Figure 3 (p. 120), it is the single sheet of paper on an otherwise empty desk that has a highlighter placed on top of it. Note its massive public availability; the positioning of the scheduling form is quite deliberate and important as control room workers will need to pick it up when they receive phone calls about truck scheduling (from the drivers or the logistics provider) or when they see that a truck has docked at the plant (this is available through the visualisation system).

One might say that control room workers maintain an awareness and an overview of what the state of play is in the production plant (cf. Bjerknes and Kautz 1991). Overhearing conversations or noticing people doing things, events happening etc. are the means by which awareness is achieved (cf. Heath *et al.* 1995, Heath, Hindmarsh and Luff 1999). It is also important that workers assume that their colleagues maintain awareness and therefore communication can be very compact, can take the form of what Heath *et al.* (1995) have called 'outlouds' – which are themselves a means for establishing awareness and giving opportunity to talk.

Different tasks lend themselves to different ways of allocating duties. While monitoring tasks are best performed by all control room workers at the same time, the job of producing a production schedule for each of the two production lines is best left to one individual who will then take responsibility for this task and will "keep an eye" on how this particular part of the overall plan pans out over the course of a shift. While the working division of labour is still very much oriented to the group having overall responsibility, the workers make practical decisions as to how the duties can be assigned so as to make the overall task feasible. This division of

labour is not a static one as people will cover for each other (e.g. when a colleague is busy elsewhere in the plant or is taking a break) and the daily arrangements also depend on who is working a particular shift. The details of the division of labour are sometimes arranged at the start of a shift as well as being discussed in relation to the shift plan for coming days.

## *5.4 Planning and the Contingencies of Production*

Assembly Planning is the department responsible for the planning of production within different timescales from the short term measured in days to the medium term measured in months. Planning is done with a view to the goals of: adhering to negotiated times of delivery, maximising the utilisation of the plant and supplying production with packages that have all preconditions met. To this end, Sales, Disposition and Assembly Planning and production need to coordinate continuously throughout all stages of planning. Flexibility for customer change requests has to be maintained for as long as possible and guaranteed until the last week before production. Responsibility for order-related data lies with the sales department and cannot be changed by Assembly Planning.

The creation of daily assembly packages is the effective interface between Assembly Planning and production. Daily packages consist of orders that are submitted to the assembly control host with a normal lead time of 1-2 working days, enabling the timely scheduling of material transport and creating a buffer of spare orders in case some orders cannot be built. Assembly Planning is responsible for the 'buildability' of all orders submitted to the plant in terms of availability of material, information, machines, and workers.

### 5.4.1 Plan and Reality

As the planned work packages are confronted with real day-to-day production, it turns out that important assumptions often do not hold: customers' demands have lead to an increase not only in different product configurations but also in variant parts and the availability of material cannot always be ensured in advance. In order to keep production running, orders need to be downloaded despite these troubles in the hope (rather than with a guarantee) that the material will arrive in time. The

autonomous carrier system has become a bottleneck partly because the performance promised by the supplier was not reached and because the performance required rose as the plant produces more engines than it was originally designed for.

These problems have led to modifications in the production layout. Some of the customising normally done in stationary assembly have been moved to the production line so that today all F100-type engines are moved from the production line directly into testing using an overhead monorail system rather than by the older carrier system. The downside of this change is that it has introduced more complexity into the production line, the part of the production process that was supposed to be kept simple. Tradeoffs like this are part of the daily work of a number of professionals working to keep the plant in line with the demands of its operating environment. Engineers, production planners, workers, managers and others work to find solutions within the restrictions set by numerous factors such as the physical plant layout, established social relations and outside demands.

Higher than expected production figures have led to a shortage in some self-produced parts, especially crankcases. The single line producing crankcases (located in another plant in another part of town) is running at maximum capacity and there are no ways of further increasing throughput other than installing a second line which is not economically feasible. In order to minimise downtime caused by reconfigurations of equipment, EngineCo has increased the size of batches of each individual type, leading to improved overall productivity of the line. However, this means that the supply of crankcases is ill-matched to the demand of the assembly plant which produces batch-sizes down to a single engine. If Assembly Planning continued to work with the hard criteria for buildability as originally defined, the plant would at times run out of buildable orders and production would have to stop.

EngineCo has addressed this problem by increasing the horizon of visibility of Assembly Planning to include parts that are known to physically exist but are not recorded in the ERP system. This solution, in effect, changes the interface between Assembly Planning and production as orders are downloaded to the plant that are not buildable in the strict sense. Orders downloaded to the assembly control host may now be either ‘green’, ‘orange’ or ‘red’:

- Green: the order is buildable in the strict sense
- Orange: Material is known to be available but has not yet reached the logistic chain.
- Red: Material is unavailable but is expected to become available, engines may also be downloaded to create a buffer of orders.

This change means that it is now the responsibility of the control room workers to ensure that everything needed for producing an engine is available when it is scheduled. In practice, orders that are either 'orange' or 'red' will have only a single missing part so that monitoring the buildability of orders is relatively easy and therefore compatible with the other duties of production control. They do, however, require special attention, therefore monitoring such special cases is a routine part of control room work and their existence is an organisationally accountable matter as the following two entries from the shift book demonstrates:

> *Production order 34-5766-5-6689 from green to red (manually) missing part 0596 0452 (bypass)*
>
> *Order 34-5768-5-6690 from red to green, part no. 0486 0735*

By redefining details of the organisational division of labour, EngineCo has addressed a difficult situation that was impossible to predict during the original planning of the plant. Although the change involves a modification of the notion of buildability, a key component of production management, this does not mean that the grammar of buildability as initially envisaged is abandoned. Members understand that, ultimately, all prerequisites need to be in place for production to progress and so the original idea is still being oriented to. It is, for example, not uncommon for control room workers to complain about the number of engines that are 'red', thereby reminding others, especially Assembly Control, what constitutes a 'good' state of affairs and that the current situation is less than satisfactory. On the other hand, they may ask Assembly Control to download engines even if they are 'red' towards the end of office hours as people working in Assembly Control go home while production still has hours to run, normally finishing at midnight or even 1am. By asking for a larger number of orders to be downloaded, they increase the potential for

buildable engines being available and so increase the chances that production can be kept running, the ultimate aim or production planning and control.

The notion of buildability has thus been turned from being a simple relationship between engines and component parts available for production to become a matter of negotiation between Assembly Planning and the control room. Control room workers will normally be aware of the circumstances around the states of orders, e.g., why parts are late or cannot be used, need rework, etc. Understanding these circumstances and being able to relate them to the state of production on the shop floor as well as to the likely unfolding of events over the course of one or more shifts constitutes an important skill that control room workers need to have. It is part of the stock of knowledge that they draw upon.

### 5.4.2 Changing Production Layout

The example of the monorail fast-tracking engines into testing, has already demonstrated how the order and layout of production steps is potentially subject to revision. Another example is that of stationary assembly III, a workplace originally designed for auditing and where defective engines could be dealt with. Engines leaving varnishing were assumed to be completed. However, changing customer demands and new engine configurations mean that some parts can only be assembled after varnishing or that other work may be needed. Since engines can be moved to stationary assembly III simply by marking them as engines to be audited in the assembly control host, control room workers have started to use this feature to move engines that require attention to this workplace. As the simple audit tag does not allow a distinction to be recorded between different reasons for moving the engines there, a text field was introduced in the assembly control host system for control room workers to record an explanation why an engine is moved to stationary assembly III. Later, the simple tag was replaced with an enumeration of the different reasons. This case is only one of many that illustrate the ongoing debate about what is an acceptable workaround at a particular time, facing a particular problem and what gets officially sanctioned and more strongly embedded in organisational arrangements.

Another example involving a rather larger modification to the production layout is the relocation of large parts of stationary assembly I to a space behind the engine buffer, on top of the testing field. This move has not only enabled modified and improved logistics in the plant, taking more load off the autonomous carrier system but has also freed up space on the shop floor which is now used for providing more buffer space in the plant. In the past, there situations where engines had to be stored outside designated buffer spaces were not uncommon (see Figure 4) and as this measure led to manual handling of engines using forklift trucks, the practice was labour intensive, involved an increase risk of accidental damage and was outside the scope of the assembly control systems.


**Figure 4 Engines stored outside the designated buffer spaces. Note the yellow line and the track marks which show the limits of the automatic carrier system.**

Other alterations to the normal production process become necessary as a means for 'repair work', i.e., improvisations required to maintain the consistency of data or the fit between production data and actual circumstances. Such situations arise because of failures in the computer systems or in the attached sensors, because workers make

mistakes operating controls, because incoming material is wrongly labelled or otherwise faulty, or because defects of completed orders are recognised after the engines have left the plant. In all such situations, consistency has to be re-established manually. While control room workers have some means to do this, they have to rely on help from the systems operators for certain kinds of repair work (e.g., when an order has to be reintroduced into the plant under the same unique production number). An example of repair work that occurs daily is the correction of stock records.

## 5.5 Control Room Work: Doing Dependability

If it is not the production orthodoxy, planning and rigorous execution of the plan that make the plant work, how *is* production realised? To answer the question, I would like to explicate in a bit more detail the day-to-day work in and around the control room of the plant.

### 5.5.1 Attending to the Normal Natural Troubles of Production

As discussed earlier, control room workers have effectively taken over ultimate responsibility for ensuring a smooth flow of production by making sure that the individual steps of production can be carried out when needed. This involves making sure that when an engine arrives at a particular station in the plant all parts are available as well as all information required. This concern for practical buildability is illustrated by the following entries from the control room shift book:

*Engines are red even when only accessory bags are missing.*

In this case, the control room worker notes that engines are marked as 'red' in the system even when just an accessory bag is missing, something that will not hinder the production of the engine *per se* but should be attended to before it goes to the customer. In bringing this case to the attention to other members, the control room worker contributes to the ongoing adjustment of practice to circumstances and helps to work out the practical meaning of the categories 'red', 'orange' and 'green'.

*Received pistons in advance, 238 pieces, part no. 54636765*

Here, a part has been delivered 'in advance', i.e. without being handled by the company stock control system in the normal way. This is a frequently occurring

workaround that allows workers to ‘fast track’ certain parts, effectively bypassing the strict regime of the stock control system and allowing an exceptional handling of the parts involving manual interventions afforded by the loose coupling between the enterprise resource management system and local systems in the plant. Occasionally, parts would even be handed to the driver (if they are light enough to be carried) who would bring them to the control room from where they are then brought to the specific point in production where they are needed. This manual handling is much faster than the automated handling system which is designed to work well under an even utilisation and produce a constant supply of parts while handling high volumes. Taking such exceptional action is costly in terms of staff time and holds the potential for error. It also leads to a temporary mismatch between information recorded in the IT systems and the situation on the shop floor. However, control room workers are able to judge the implications of such action and they make the fact that exceptional action was taken accountable to fellow members who may be working on a different shift and encounter subsequent troubles. Also, the practical division of labour extends beyond the control room and working out the details of a particular workaround requires the cooperation of and collaboration with other departments such as goods receiving or quality control:

> *We are getting 1 case of crankshafts (part no, 54676656) back from [other plant], are all OK. Been checked there after fall at [logistics provider], call goods receiving on 5785 how they should get booked in, case is on ASM1R386*

This cooperation across departmental boundaries extends beyond the boundaries of the organisation itself as it may involve the suppliers, logistics providers or even customers. Maintaining an understanding of the working practices of closely collaborating parties elsewhere in the plant or outside is an important part of the work of control room workers as they have to make judgements about what constitutes a reasonable request, a big or a small favour or a plausible refusal. Being seen to be doing one’s work in a professional manner is what such cooperation turns on and the grounds for this judgement are constantly worked out in practice.

The study of control room work has uncovered a series of expectable, ‘normal’ or ‘ordinary’ troubles and candidate solutions that are available to members in, and as

part of, their working practices (cf. Auramäki *et al*. 1996). That is, such problems do not normally occasion recourse to anything other than the usual solutions. Each problem encountered may or may not have connected to it a set of used-before-and-seen-to-work candidate solutions. These troubles and their candidate solutions are part of the stock of knowledge that is available to members.

## 5.6 Design in Context

I now turn to consider the work of IT workers and the ways in which their work is tied in various ways to the plant's operation and working practices of other members of the setting. Button and Sharrock (e.g. 1995, 1996) have pointed out the importance of studying design work in practice and have explicated how the prescriptions of the project management and software engineering literature are translated into practical actions. This section has a similar aim. However, the conditions under which IT systems development in the EngineCo plant is taking place are quite different from the usual project organisation. Many studies, e.g., Button and Sharrock's (1995, 1996), focus on the development of products, work that often faces problems such as the threat of cancellation, the absence of 'the user' or the pressure to abandon 'good practice'. In contrast, here IT systems development is an ongoing activity oriented to the practical work of keeping production going, continuously updating and improving the arrangements made. That I have been studying an instance of in-house development is not the important point here: I wish to explicate the practical work involved in IT systems development and the arrangements that support this work. The crucial element is the longitudinal aspect of the involvement of IT workers in the setting, not their position within vs. outside the organisation. At EngineCo, IT and non-IT staff work within working divisions of labour that are oriented to the particular organisational contingencies that they face rather than to some general schema for user-designer relations (such as found in traditional software engineering approaches or, indeed, participatory design methods).

Such working relations are a matter of practical, situated politics. That is to say all members in this setting have their own perspectives and concerns but, crucially, these are recognisable to other members and can be oriented to in interactions. This is far from the dystopia of "design from nowhere" (Suchman 2002), even though it

may not fulfil all the dreams of researchers working in the participatory design community. I would argue that this is not an isolated example and that such practical arrangements in IT systems design are common in the everyday work of "doing IT". Further, they are constitutive of the invisible work of IT professionals which is recognised neither by official reward structures nor by academic research as it falls outside the project-oriented perspective seen as the norm for IT work. Nevertheless, this work gets done and I would argue that a consideration of it, of what it means to be "doing IT", might be a good starting point for rethinking user-designer relations.

### 5.6.1 Information Systems Development

Development, maintenance, and operation of the assembly control host is outsourced to an external service provider (ITCo) that guarantees quality and availability of service to EngineCo. They have operators and systems developers on site and work in close cooperation with EngineCo's engineers and IT staff. IT systems development at EngineCo has inherited some of the characteristics of the engineering culture surrounding it. Production processes are continuously changed in detail in order to keep up with customers' demands, to improve efficiency as well as to respond to various troubles. The development of IT systems has to keep up with these dynamics. Accordingly, although design work is contracted out to an external service provider, it is not arranged as a one-off design project with a handover to the customer and some subsequent maintenance but it is rather arranged as an ongoing service provision contract, renewable every couple of years. In addition, the service provider took over staff who had previously worked for EngineCo and were therefore already acquainted with the company and its situation. Whereas many outsourcing deals involve the centralisation of IT work in some remote (from the customer's point of view) location, arrangements were made here to co-locate the service provider's staff within the production plant itself where they are involved in the day-to-day running of the plant and not just in the running of the IT systems. Note, however, that while this is the case for managers and operators, programmers are not directly involved and that different individuals may have more or less involvement in the daily business. These arrangements were motivated by entirely practical considerations, they constitute an adequate solution that allowed EngineCo

to reduce the costs of running its own IT infrastructure while keeping the relevant knowledge of its IT systems and other organisational arrangements close by and, at least in principle, readily accessible.

### 5.6.2 The Plant Systems Management Group

Outsourcing IT has not meant that EngineCo has no IT staff at all. A core IT department was retained that oversees corporate-wide IT provision. In addition, there is a group responsible for IT systems within the production plant described above. It is located within the plant itself and works closely with ITCo staff who are also located in the plant. It is the activities of this local IT group that I want to explore in the following as their direct involvement with the day-to-day running of production puts them into a very interesting role.

### 5.6.3 The Daily Reports System

The following fieldwork extracts present interactions between a worker from the plant systems management group and two *Meister*[36] relating to the design of a system to support daily reporting of production statistics and shop floor events such as breakdowns. The design of this system is part of a current focus of activities on improving the documentation of and reporting on the production process. Currently an Excel-based system is used by the *Meister* to produce daily reports containing details such as staffing levels, production figures and information about breakdowns. The reports generated are discussed in a briefing meeting held each morning at 8 o'clock, to which all stakeholder groups in the plant send representatives. Information from these reports is routinely typed into other systems for analysis and this double data entry is the main driver motivating the development of a new system that would store the data in a central database and would thus provide integrated support for reporting and analysis. Whilst the development of the daily reports system is subject to the interests of a number of distinct, yet connected, user constituencies, I shall attend here to interactions between the *Meister* and the plant

[36] The 'Meister' in Germany are shop floor workers with a predominantly supervisory role; for a discussion of the role of such 'production supervisors' in different national contexts see Mason (2000).

systems management group, because they perspicuously instantiate the notion of design through membership.

While the main driver of this is management, there is also a clear interest by various shop floor workers to use such documentation for their own purposes. In modern divisions of labour in manufacturing, the work of looking for improvements in production has been devolved to some degree to shop floor workers such as the *Meister* or team leaders (Mason 2000). My concern with the organisational working division of labour leads me to argue that the practical arrangements of this work are part of the biography of an organisation or particular setting which unfolds through a complex interplay of interests. Workers are not merely recipients of managerial dicta but actively participate in the unfolding biography of the place using the various means at their disposal to comply, obstruct, work around, etc. (Roy 1954).

The existing Excel-based system allows data entry into a WYSIWYG form, which automatically updates derived data. It would be costly to reproduce the same functionality using custom-built software and the use of Excel or a similar generic application as a client-side programming environment is seen as being technically challenging and possibly locking the company into proprietary software solutions in strong ways. The web-based approach that has been chosen, in contrast, promises to offer an open architecture built on readily available open-source components such as Apache Tomcat[37] and Cocoon[38] that can be grafted onto an existing relational database (the possibility of moving to an open source database system was also a key consideration on EngineCo's part). This is seen as a major benefit by local IT staff as it offers the opportunity to have the system developed and operated by their main IT subcontractor, ITCo, while keeping open the option to re-deploy the software on their own servers using an open-source infrastructure at some point in the future.

[37]Tomcat is a web application container, see http://jakarta.apache.org/tomcat

[38] Cocoon is a web application framework, see http://cocoon.apache.org

Also, it allows local IT staff to contribute to the project using tools and skills readily available to them or easily acquired[39].

Barbara, a member of the plant systems management group and the researcher decide to visit the *Meister* to discuss the possibility of using a web browser as the client interface. Barbara has concerns that the limited user interface offered by the browser-based solution, in particular the loss of WYSIWYG-ness and responsiveness, would make it unacceptable for the *Meister*.

### 5.6.4 The Public Availability of Design Work

In order to find out if the browser-based architecture would be acceptable for the *Meister* and others involved in the creation of the daily reports, Barbara and the researcher decide to discuss this issue with them. Barbara suggests that representatives of all groups should be consulted but she also suggests that it would be fruitful to speak to certain people first as they are readily familiar with the biography of the existing artefact because they have contributed to its development. This should not be read as some kind of prioritising of some people's concerns over other's but as a practical arrangement made for doing the consulting as efficiently as possible by visiting peoples' workplaces instead of convening a formal meeting. In arranging the consultations in this way Barbara trades on her practical knowledge about the availability of certain persons and their readiness to have an impromptu discussion. This is not, of course, the first time that such discussions have taken place within the plant: while these may have been about other matters, Barbara can be reasonably expected to know who can be contacted and when, together with their competence and interest in the issue at hand. In sum, then, these discussions are predicated upon her knowledge of the working practices of plant members and their interests as well as her ability to deal with the practical politics of the setting.

Such ad-hoc meetings may not, of course, enable one to meet with all those with whom one would wish to speak. It is obvious that not all *Meister* will be present at her visit, yet her orientation to those who are there is one which enables her to

[39] With the assistance of the researcher, local IT staff are taking over responsibility for developing the code that produces PDF output from the database both for the basic printed copy of the daily reports and for longer-term analyses

resolve the issues she has come to discuss. It is a feature of this meeting that other workers who happen to visit the *Meister*s' office at the same time are able to take part in the discussion and are sometimes even explicitly invited to do so if they have a particular interest or something important to contribute. Such ad-hoc participation is normally brief as people are engaged in other activities. Nonetheless, the public nature of the meeting means that the fact that such a meeting is taking place is available to other members and can be oriented to in further interactions. It should be kept in mind that in addition to its ad-hoc nature, the meeting is located in a public area where it is likely that other workers (and not just *Meister*) might come in – it is not an invitation-only meeting. Anyone who is not taking part in the discussion can approach Barbara at any time to discuss their concerns and contributions. One should bear in mind that it is an open-ended consultation that can be taken up again at any time in the same ad-hoc ways. The results produced so far were also circulated as notes which were available for discussion in other arenas such as the daily 8 o'clock briefing.

### 5.6.5 Talking Design

The following fieldwork extracts are taken from the meeting taking place in the *Meister*s' office. Barbara begins the discussion by introducing the organisational reasons that lead to the current proposed architecture. She explains her department's interest to choose a set of technologies that allow the organisation to have more control over the system (*vis-à-vis* ITCo):

> ***Barbara**: ...but that always leads to a big fuss here and we would need this SQL server, that means that ITCo would always have to be involved and we didn't really want that and so we[...] have asked ITCo how they would do it. They have come up with the idea to realise it with a web server.[...]I find this a rather nice thing, we could possibly do shift books or other things with this later on and, most importantly, we can - when we have time - operate it ourselves on our own machine. We do the pages, we have the whole thing in our hands.*

Here, Barbara points to the expectable consequences ("…that always leads to a big fuss…") of one technical option versus the benefits for the organisation as a whole of the proposed one. She refers to past experience with IT systems development, taking for granted that her reference will be understood by the *Meister*, Mark and David, as

they share (to an extent) her knowledge about past experiences. She can reasonably make that assumption based on her past interactions with them, her knowledge of what systems they have used and may have seen in development and what they would know in general, as competent members who have worked in this plant for a number of years. It would be correct to say that these interactions are constitutive of and in turn informed by each other. As with Wieder's discussion of codes in the halfway house (Wieder 1974a), we find that there are maxims based on the natural history of the institution that inform and contribute to what is said and also to what does not need to be restated. An outsider would not possess these matter-of-course understandings upon which the conversations reported here clearly trade. The discussion turns towards other issues for a moment but Barbara leads it back to the question that motivated the consultation:

> ***Barbara**: Yeah, we're going to have to see that we can bring that into a form that we can make sense of it. How you want to enter this is the other thing. For a database thing it's always better if I have a page with entry fields and then you press a button somewhere and then the daily report comes out that looks like this.*
>
> ***Mark**: That is somehow formatted, right?*
>
> ***Barbara**: Yeah. Mark: Yeah - If that's ok with you.*
>
> ***Mark**: Yeah, sure*

For the purposes of the discussion between Barbara and the *Meister*, the issue of separating input functionality from the layout of the printed reports is settled and it does not reappear although, of course, any participant could bring it up again as such closure is revisable in the light of new implications being detected in the ongoing talk. The issue that prompted the visit to the *Meister*s' office turns out to be quite unproblematic. It is the *Meister* who start to drive the discussion at this point, raising two related issues:

> ***David**: That is, I'd say – the causes of problems, that one has them defined in a table or so?*
>
> ***Barbara**: let's continue here, I don't know if it makes sense to…*
>
> ***Mark**: (interrupting) I had in recent times - we are, I'd say, more precise with the daily reports….*

The intended discussion about the general architecture and client interface technology has effectively been used by the *Meister* to shift the topic to raise concerns they have. These concerns are related to detailed design issues but are very much relevant to the way the daily reports are produced and used. As Mark explains, they are related to the quality of the data that is recorded and to how it can be recorded in a standardised and convenient way (by having a list of potential sources of problems available to select from). This concern with the quality of the data collected appears throughout the entire conversation. However, we can see that Barbara is initially reluctant to discuss the design at this level of detail. After all, the visit was intended to be short, informal and about a specific issue. However, it turns out to take about two hours and it provides the bulk of the requirements that Barbara will eventually distil into a document to be used for design. Her initial resistance is not maintained after the next turn in which Mark gives a quite detailed account of how he starts to write an entry in the production breakdowns section of the report (remember that the reports themselves are available to the discussants as they are ready to hand in the *Meister*s' office):

> ***Mark**: I could imagine, I click on this here - time, zap, zap. Then, I'd say, a field, I wanted really to include this here, that I wanted to just tick a box. Well, let's say, line standstill, when I say I've definitely come to a stop. That are the various reasons that I have. I really always begin: what's happened, where, then a rough explanation if I know it - so that it's, I'd say, easier for others to understand. Right, like here for example: transport unit A-B, that was my problem source. Right? We've always had trouble to bring that into the right form. But, let's say, basically, we begin with: transport unit was the reason.*
>
> ***Barbara**: That means that in "problems" one first notes the station.*
>
> ***Mark**: Right.*

She turns from initial resistance to note taking, to engaging with the account that is given to her about how report entries about breakdowns are written. What is visible in the account that Mark gives is his orientation to Barbara as a member who can be expected to have an overall understanding of the affairs of the plant: the term "Übersetzer A-B" (transport unit A-B) is a local term used to denote a device that moves engines from one assembly line (section A) to another (section B). For Barbara, this term needs no explanation and the *Meister* can point to an example

from his own practice to illustrate his account, using an existing report as a resource to produce his account. His skilful use of examples from an existing report, his knowledge of what Barbara might need to learn to do the design and what she can reasonably be expected to know as a member in this setting is something worth noting here. He is certainly not someone who "doesn't know what they want" (an apparent common conception amongst designers). While he may himself not be able to undertake design in the same way as Barbara, her colleagues and the people of ITCo, he can very much orient to design as an activity that he can have some role in and contribute to. Note that the basis for this is not some special kind of personal quality but his ability to act in this way is based on the competencies of an ordinary member of this setting. His formulations do not contain readily formulated requirements – and this is in line with what Bowers and Pycock (1994) found – but we would argue that a statement that did contain an explicit statement of the "I want X" type would be noticeable in the discussion and Barbara would certainly ask where that requirement came from. Her expectation is not to be presented with readily formulated requirements but that work needs to be done to work up such formulations. We can see her doing this work in the extract above where she formulates the requirement for a field denoting the station in the plant where a problem occurs. This formulation serves both to produce an object of design (an input field of some kind) and to enable Mark to confirm that what he was saying was properly understood.

Barbara also brings in an understanding of the wider implications of the design decisions. Through her everyday work on IT systems in the plant, she's in contact with a wide range of people in different departments and is familiar to an extent with their concerns. When Mark mentions the need to design entries in the reports so that they are "easier for others to understand", he can assume that Barbara will be familiar with the role the reports play in the collaborative practices within the plant. The recipient designing (Sacks and Schegloff 1979) of report entries turns on members' familiarity with the ways that a smooth flow of production is maintained and how problems get reported and dealt with. Knowing about these features of the work is an important resource in designing the system so that it can effectively support collaboration. This is also illustrated by the following extract:

***Researcher**: At the end of the day one has to – what the people need for their analysis is all line standstills…*

***Barbara**: …failures…*

***Researcher**: … no matter if they have been caused by failure in a device or…*

***Barbara**: That's not quite right, no. There are different interests in these things. […] There is the interest of production management, Müller and his folks, they want to know what the productivity is and when I have line standstills. Of course, they are not interested in the cause. But there is also maintenance who also want to draw on this, they are interested in what failures there were. If I type in the transfer place that has come to a standstill because I didn't have any screws, then maintenance is not at all interested.*

Knowing about the different interests and concerns of different groups using the system or making use of the reports generated from it allows Barbara to see the system as more than just a management reporting tool but, rather, as one that will act as a resource in the work of various groups in the plant. She points to maintenance workers as a group of people who might wish to draw on the daily reports as a resource to monitor faults, to schedule repairs and replacements and to make their work organisationally accountable. Her mentioning the maintenance workers becomes relevant in the discussion as one of the *Meister* picks up this issue:

***David**: Normally, maintenance would have to write a daily report, too. So that one would be able to compare the reports about faults.*

***Barbara**: Who is going to do this extra work?*

***David**: A maintenance worker – a shift book – and then one could analyse that.[…]*

***Mark**: They do write into their shift book – by hand.*

***Barbara**: They write into the shift book sometime in the evening – that's what they also tell other people.*

***Mark**: Ok, let's leave the maintenance folks – that's for other people to think about.*

David is making a demand for the maintenance workers to produce similar accounts of events. However, the working practices of the maintenance workers differ in crucial ways from those of the *Meister*: maintenance workers are always on the

move, they don't necessarily return to their office after making a repair whereas the *Meister*'s work often does take place in their office. The writing of reports thus does not fit easily into the maintenance workers' ways of working. However, all participants in the discussion agree that they play an important role and that their input to reporting would add value. (It is also to an extent unavoidable as they will inevitably present their views at the 8am meeting.) The issue is deferred because it seems difficult to envisage exactly how maintenance workers might contribute their records of events and how this work might tie in with their other commitments. Further consultation would take place over the next few weeks which would address this issue.

The mentioning of the shift book as another artefact that plays a role in these activities is also important in that it flags another dimension of collaboration: information about the state of play needs to be made available to workers on subsequent shifts. This is the purpose the shift book normally serves. Its entries are not normally designed for anyone outwith the group of maintenance workers and they are written retrospectively rather than as a real-time account of events. The latter means that some details may be missing as they may well not be remembered by maintenance workers when they write the entries at the end of their shift, An example of such details would be the exact time of an event which is of less importance for maintenance workers as they are primarily concerned with fixing the problem. Other parties need the time to work up estimates of productivity lost or to establish correlations between different events. It is important to note here that the nature of maintenance work is understood by both the *Meister* and Barbara and that they make the practical decision to defer the discussion until other parties have had a chance to participate. As mentioned earlier, merely knowing about peoples' work does not provide a 'ticket' for acting on their behalf but it does provide a means to discuss possible issues so as to flag them and make them available for further discussion.

A final example and perhaps the best illustration of the value of Barbara's ready understanding of working practice and people's concerns is provided by an episode from a subsequent meeting between Barbara and David in which they discuss an initial implementation of the system. They test this largely functional prototype using

that day's production data and run into trouble at the point where David needs to select a station (a functional unit on the assembly line) to indicate the location of a fault. All stations are listed in a selection box:

> ***David**: where is the transport unit?*
>
> ***Barbara**: the transfer unit also has a station number – but don't ask me which[they collaboratively scan the list of possible entries] there was something just then […] further up[…] transfer unit, there it is.*
>
> ***David**: it's under "T"*
>
> ***Barbara**: well, "transport unit". I'm not sure how we're going to order this- I had it, this [the specification] is now how they are on the assembly line but someone has changed the ordering.*
>
> ***David**: Normally, the stations should be ordered as they are on the assembly line, I'd say.*
>
> ***Barbara**: Yes. That's exactly how I have entered them but someone has changed the sorting order.*

Barbara created the list of stations and handed this over to the ITCo programmers who have implemented the system. While she realised the importance of the ordering to match the physical layout of the plant, the programmers, who have a general understanding of how the plant works but only a limited understanding of peoples' working practices, have changed that ordering to facilitate a more general strategy for searching based on alphabetical ordering of the station names. In addition, this episode demonstrates that Barbara is aware that there are a number of different ways to refer to particular stations: by name as well as by a number of different numbering schemes which exist for historical reasons and are used by different parties in the plant. An important way of ordering the stations is by their physical location on the assembly line[40]. This form of representation is important in relation to the interactions between, for example, the *Meister* and the maintenance workers as it allows them to not only identify a station but at the same time to locate it. Station numbers representing locations are also a way for people working throughout the plant to communicate their current location, thus affording the effective scheduling

[40] Cf. Bødker and Grønbæk's (1991) discussion of prototyping in a dental practice and the issue of orienting representations of a patient's jaw in a prototype electronic patient record.

of mobile workers' activities. In asking a maintenance worker to attend to a problem, there are a number of factors that need to be considered, for example, their current workload, their skills but also their current whereabouts in a large space.

Understanding these details of production work and how it is collaboratively achieved is an important aspect of Barbara's skills. Making use of what she knows, she can attend to detailed working practices in drawing up requirements and designing systems, thus providing artefacts that are in "working order" (Button, ed., 1993).

### 5.6.6 Design *qua* Member

We have seen how Barbara, Mark and David each mobilise what they know about the plant and the working practices of the various people working there. In many cases, even when talking about the work of people not present, the problem is not so much to find out about that work but to relate it to the larger picture and the emerging understanding of what the system might look like and how it will figure in the evolving working practices. Where a decision cannot be reasonably made, issues are easily deferred as the interactions can be reasonably expected to continue after the meeting as people meet each other on a regular basis for a number of routine reasons. One such occasion are the eight o'clock meetings in which the daily reports are discussed by representatives of the various groups in the plant: the *Meister*, control room workers, plant managers, the plant systems management group, etc.

The point here is that the discussion can be conducted without the sense of urgency that is normally generated by the limited availability of participants within more traditional projects organized through a string of meetings. When Mark remarks that an issue is "for other people to think about", he takes for granted that participants in the meeting will know just who these people might be. He also knows that this deferring the matter does not mean that he is wasting his chance of voicing his opinion as all people involved (even the ITCo programmers) are normally available for him to speak to – by phone or in person as their offices are located within the plant.

The parties taking part in the design talk presented make use of a number of resources available to them qua members such as other members' understanding of

the divisions of labour in IT systems development and the history of previous development activities. Barbara is able to orient to this understanding to account for the reasons why a browser-based design is being offered by the plant systems management group. Because of her previous interactions with Mark and David, she can reasonably expect them to appreciate this motivation and to be willing to discuss the proposed design on that basis. She also knows that the *Meister* are also reasonably familiar with the general organisation of web-based applications (such as many of us use routinely these days when we book flights or order purchases over the Web). Mark's question "that is somehow formatted, right?" indicates his understanding of the difference between data entry and the way that the data may be presented in a printed report and also that the generation of the report is a separate step. Having heard this question as a confirmation that her point has been understood, Barbara can orient to this issue as one that needs no further discussion, unless occasioned by following interactions.

Mark and David make use of Barbara's presence and the informal nature of the meeting by raising concerns they have with the design. As they have experience with the development and use of the Excel sheet, they have very clear ideas about what kinds of problems the new system should address – issues they haven't been able to address using Excel and their knowledge of it. One such issue is that of standardising data entry. The new system is seen by them as a means to improve the quality of data entry. By presenting an account of how they do their work that is recognisably oriented to the design process, they give Barbara an opportunity to gather detailed data directly relevant to the design, thus enticing her to engage with the details of design that they are concerned with. Note how Mark combines presenting a specific example, making use of material ready to hand, with an account of how work might be done in any normal circumstances: "basically, we begin with […] the transport unit was the reason" – this statement can be translated directly into a requirement for design. While there are other candidate fields that could be placed in front of "station", there is no indication that the chosen order may be problematic. Note that the link between the practices discussed and the design of the user interface is not made explicit at this point and this is just one of many instances where things remain unsaid. This is perfectly acceptable for the current purposes – people act within the

"natural attitude" (Schütz 1973) and often take mutual understanding for granted unless such an assumption is called into question. We have already pointed to the paths open for participants to repair misunderstandings as and when they are discovered.

That is to say, following Schütz and Luckmann there is a 'reciprocity of perspectives' and a 'reciprocity of motives' (Schütz 1953, Schütz 1973, Schütz and Luckmann 1974) upon which Barbara and the *Meister* trade. Barbara and the *Meister* know about the workings of the plant and how things have come to be the way they are (although they may not share completely the same knowledge, but sufficient for the purposes at hand) and as part of the same organisation can assume that, to a degree, they share motives (e.g., that both parties orient to the realisation of efficient production and so forth). Barbara did not have, for example, to familiarise herself with the work of the *Meister* and the *Meister* did not have to employ advanced computing knowledge - each party trades on what they know about the other to get the design job done within the overall context of EngineCo as an organisation (on the concept of organisation and its articulation of and orientation to by members see Bittner 1974). Membership is, therefore, having these reciprocities and using them to inform one's practical reasoning in such interactions.

The same example shows how Mark and David can make assumptions of what Barbara should know as a fellow member about the various features of the plant and how production work is socially organised. Her understanding includes not only such features as the formal organisational structure and the subdivision of the plant into sections but also peoples' interests, the various dealings they have had with each other in the past, how people go about their work. She is aware of crucial differences in the patterning of work and can point out the relevance of this to the current project, representing concerns that other groups such as maintenance workers might reasonably have. Finally, in the last extract we see how she can make use of her knowledge as a member to fill in details of the design that have not been discussed. Such an understanding of details of work organisation and an awareness of their relevance is clearly not available to the ITCo programmer who works from a (relative) distance.

We have seen here how members in the plant make routine use of their members' knowledge to establish the social organisation of development work and to produce systems that are in working order. While they would probably not claim that they are 'doing' corealisation (and to suggest that they do would be to provide an ironicised account of what they do), their work exhibits many of its features. It is these practical arrangements of work on and with IT systems that corealisation seeks to build on.

# 6 Corealisation in Practice

This chapter discusses some aspects of my own work of corealisation in the production plant. My role was one that involved more than just developing systems but included such things as training workers, considering the operation of the resulting system, or mediating between workers and the IT department. The emphasis in the discussion will be on the collaborative nature of the activity and the fact that 'design' is not necessarily at the heart of the matter but rather one of the possible activities within an ongoing process of working with and working on IT. A lot of work involved discussions about the possible impact of particular arrangements, making small changes, working up solutions to problems that arise, and so on.

In the following, I will describe and discuss a number of IT-related activities and systems development efforts that I have been involved in during my stay at EngineCo, reflect on the ways in which this work was done and demonstrate how the systems or arrangements are each uniquely adequate and work affording. That is, despite their obvious limitations, they are tuned to the work they are supposed to support and their existence is accountable for in terms of the biography of the setting with which they are closely linked. That is, members of the setting are able to say, at any point, why things are the way they are, how they have come to be this way. Specifically, I will talk about the development of new practices around the distribution of various reports, the development of a shift book application and two visualisation systems.

## *6.1 Working with and on IT*

Control room workers maintained records about issues such as transports to and from the plant, special transports, scrap engines, missing parts, engines 'locked' at the logistics provider's, urgent engines, or errors at goods entrances. These records played an important part in the production of an orderly manufacturing process and also served to make activities accountable, observable-reportable. Record keeping was supported by a number of different artefacts such as paper forms, Microsoft Word templates, Microsoft Access databases, or the custom-built assembly control

host software that controls the overall operation of the plant. Some of the artefacts had been developed by members themselves while others had been developed by on-site IT staff. Of course, the arrangements for record keeping were open to potential revision at any time, depending on needs and resources. As a corealiser, I played a role in establishing requirements for revised record keeping practices which needed to be put in place because of new demands by senior management:

> **Extract from fieldnotes**: In the control room, I witness discussions about a new version of the Access database that is used to keep track of missing parts. People higher up the hierarchy now demand that they get a monthly digest of this data and Barbara is working on corresponding changes. This move also means that now entries cannot simply be deleted once the problem has gone away since questions may now come from up high and, probably, much later than before from the immediate colleagues. Thus, the old data is needed by control room workers to be able to refer back to.

The need to inform senior managers not only required a change in the presentation of the data but also that the data be permanently stored. It is a feature of control room work that a lot of information is temporary and discarded when it is no longer needed, for example, as soon as engines leave the plant. One might say that knowing when information can be discarded is part of what members know and use in their daily practices. Information in the control room is used for practical purposes, it serves the purposes of getting the job done, i.e., building engines. Increasingly, however, information is stored permanently to be available for later inspection and analysis. Additionally, there is a demand for reporting to management. Such reports are created as needs arise and as managers take an interest in various aspects of the state of affairs. Quite often, they are temporary measures that get *ad hoced* and are abandoned as soon as the situation that gave rise to them is over.

There is a tension, then, between these two roles that records play in the control room, between the accomplishment of an orderly production process and the need to make work organisationally accountable. Quite often, control room workers can be seen to dispose of a piece of paper in a way that indicates some satisfaction with having finished some task and closed a case. Any piece of paper sitting on a desk in the control room indicates some job to be done or some issue to be taken into consideration, for example missing parts. An ‘untidy control room’ can thus be seen as an indication that the state of play is somewhat problematic, that control room

workers need to work hard to stay on top of the situation. Occasionally, however, the control room workers find that they have disposed of a piece of paper too quickly and they would try to find it in the trash or try to reconstruct the contents by, for example, turning to other workers in the plant who might still have records, who might remember something, or who might be working on the engines the note was about. Records have a temporary character mainly because, for the purposes of control room work, knowledge of the state of play can be re-established by referring to the actual situation in the plant. Also, any information recorded may become out of date as production moves on, so disposing of a piece of information can also serve as a way to ensure that out-of-date information is not inadvertently used.

Another tension arises in terms of the contents and the design of records. Records that are mainly used as resources in the control room will require different contents and a different layout than records that are shared with other members in the plant or even with the CEO of the company. For various reasons, records that leave the control room may need to be recipient designed (Sacks and Schegloff 1979) for others. This might be because others might not be able to make sense of information presented in a locally meaningful form or it might be because control room workers do not wish to disclose certain aspects of their work (e.g. workarounds or short-cuts).

In addition to issues of recipient design, the actual production and distribution of documents can also raise important issues which can be addressed in a number of ways:

> **Extract from fieldnotes**: It emerges that the problem of distributing the various nightly/weekly/monthly lists is quite a topical one. The list of missing parts needs to be distributed to more people than before. Some people want to receive it via Lotus notes, so the paper artefacts need to be faxed to a gateway machine. The fax machine can no longer store the distribution lists in its quick dial buffer and so the whole process of assembling and distributing the lists is made more cumbersome than it is already. Some of the lists need to be cut and pasted quite literally with scissors and glue since they consist of sub-lists that can only be printed off from the assembly control host. These assembled lists are then copied and faxed (possibly going through a transformation back into electronic form when people receive their faxes as email).

The distribution of the various lists had been an issue in control room work for quite a while but the longer distribution lists and the fact that the fax machine could not deal with them provided the occasion for people to reconsider their practices. There

was serious discontent with the fact that telephone numbers now needed to be typed in manually and thus the whole problem gained currency. Control room workers approached me to ask if I could help to provide a way by which the lists could be sent electronically, so I started collecting information and discussing the matter with various parties.

> **Extract from fieldnotes**: Asked Michael to give me a list of the reports that need to be sent and to include information about who gets them, if the format is crucial (e.g. they are forms that need to be further processed), where the data comes from etc. He opens one of the folders that contains various paper documents regarding control room work and gives me a document that contains a list of the lists and the recipients. He says that the document is out of date but that it would give me an overview of what we're talking about. I also mentioned that Steve had said that the form of the lists is not really crucial and that we could try to re-establish the format using the formatting options that Lotus Notes provides for eMail. Michael says that some of the lists need to be improved in terms of the layout anyway.

This extract shows that what is known about or available from the setting does not necessarily supply bounded requirements by itself – the distribution list is out of date, the format of the lists is 'not really crucial' and 'needs to be improved'. Rather, what has been uncovered are a series of open questions that have to be addressed in a practical way. Somehow requirements have to be elaborated from these initial understandings. One question then is this: will simply 'finding out more' (e.g. quizzing list recipients as to desired formats, discovering who will benefit from receiving the lists and so on) supply all the missing answers? This appears to be unlikely – it is more likely that there will be some aspects of the lists' organisation that hold no interest to members, other aspects that are contentious, and further aspects that are undecidable within the current state of play of the organisation.

Furthermore, the picture is one of what is done now, rather than of what a list collation and distribution system might look like if integrated. There are affordances and organisational issues associated with implementing the lists in a different technology that open questions and have implications not addressed or considered by the existing list distribution system (see below). Thus there are possibilities (and hence requirements to be specified) afforded by a new technology that cannot simply be read-off from the existing state of play.

The practical work of making design decisions involves making 'common sense' (or 'organisationally relevant') judgements about what parts of the distribution of lists are important, which are not, which need to be decided now, and which decisions can be postponed, whether further details of practice or opinions are required, or whether decisions can be made immediately on the basis of the designer's judgement. There will be decisions that will be made on an add-hoc basis and others where there will be lengthy discussion. There will be some decisions that will be made on a 'suck it and see' basis – where it is found to be difficult to find criteria for the decision in what is known about current practice and uncertainty about what the implications would be for future practice. Thus knowledge about a work setting may supply grist, but of equal importance is the working of the mill. In short, rather than extracting something from the setting that can be taken away and used for design, directly engaging with the setting provides additional opportunities for design.

As part of exploring possibilities for the distribution of the lists, I also talked to the control room manager (himself an IT professional):

> **From a design meeting with the control room manager**: It's really ridiculous the way it works now, first there is a loss in quality, the page that is printed looks fine and it then goes through the fax. […] In the control room they fight with the problem that they have so many distribution lists – at the moment the fax has only 16 memory spaces […] we need at least 25 […] I have talked to the company, there's one with 32 […] one spends 1500 Euro on a fax just to have this functionality and in Notes I can create that a hundred times.

The control room manager raised the issue of professional adequacy and how the documents produced using the current solution fell short of such standards. The quality of the document was degraded because it went through various steps of transformation, being printed, assembled, copied and then faxed to the recipient. Additionally, the process of sending out the list was quite cumbersome and was made more so by the fact that the fax machine in place could only store 16 telephone numbers. While a new fax would have helped to address the latter problem, limits were already visible where the new solution might have fallen short of expectations because the number of recipients might have increased even further. This, together with the costs involved and the quality problem led the control room manager to favour a solution based on the company's email system which would deliver lists via

email or to recipients' fax machines via an existing email-to-fax gateway. There were many ways the data could be incorporated into email messages. Sending the data as an attachment was a solution that immediately came to mind and, indeed, control room workers had tested this themselves but found that the email-to-fax gateway did not convert the attached data but would only print a paper clip as a representation of the attachment. So, the data needed to go into the message body itself. Again, there were many options. The data could be automatically sent to the Lotus Domino server by a program or it could be cut-n-pasted into the Lotus Notes client. As it was paramount to create a workable solution quickly, more 'elegant' candidates were dismissed as being too complex for the job at hand and also because they would have been more difficult to maintain. The researcher and the control room manager tried cutting-n-pasting a web page into a Lotus Notes client and then sending the resulting document via the email-to-fax gateway to the nearest fax machine. This quick test showed that the mechanism worked in principle and that the formatting of the web page seemed to be retained. So, they agreed that this was the way to go. This decision was made in the context of its feasibility not only in technical terms but also in terms of the adequacy of the solution to control room work.

> **From a design meeting with the control room manager**: People use this sometimes when they want to send something from the mainframe, they open their email and they have a terminal emulation […] and then they insert this into the email. […]I have to get this into Lotus Notes somehow, that can really be a simple solution. […] IT-wise it might be the least elegant solution.

Here, the control room manager established cutting-n-pasting as a candidate solution that built on what control room workers know and use in their daily activities. While the solution pursued was not the most elegant solution conceivable, it did represent a significant improvement over the existing practices. Also, more elaborate candidates posed questions of compatibility, quality, and maintainability. As Suchman *et al.* (2002, p. 171) observed, the building of practical IT solutions does not start from nowhere:

> Far from the de novo invention of a new device, configuring the prototype included identifying appropriate hardware and software, and acquiring the various pieces required through a variety of channels […] It included as well, and essentially, designing the computational glue that would connect them together into a coherent and working whole.

Corealisers work in a larger environment and need to take into consideration what we might call the biography of that environment. Sometimes compromises have to be made because the larger environment would not support solutions that would be 'ideal' in terms of the work being done. There might be cost considerations or the range of candidate solutions might be limited by which technologies local IT staff are able to support. Building uniquely work affording artefacts means taking members' practices seriously and this, crucially, includes the practices of IT professionals.

## 6.2 The Shift Book

Shift books are used to record relevant information about the production process and to communicate it to workers on subsequent shifts (cf. Kovalainen, Robinson and Auramäki 1998, Robinson, Kovalainen and Auramäki 2000). Control room workers developed an Excel sheet with a pre-defined format and a number of macros for creating a number of frequently used entries. Entries were created in the Excel sheet (see Figure 5) and printed off as the sheet[41] filled up or at the end of a shift. The printout was stored in a binder and the Excel sheet cleared of contents for the next shift to use. While these arrangements were well established and worked quite well, there were a few aspects that control room workers were unhappy with:

- Searching the paper record of shift book entries was cumbersome. Entries older than a couple of days were transferred to a larger folder stored in a cupboard, making them less accessible.
- It was difficult to keep important information in a place where it can easily be referenced. Only a single entry from a particular shift might be of importance for a longer period of time but it would normally have been moved to the archive together with the other entries of that day.
- There were no representations of relationships between shift book entries and between shift book entries and operational data. When workers dealt with production orders they needed to remember any shift book entries relating to

[41] A single A4 sheet was used.

them. This was a problem especially when workers returned from holidays and after weekends or holidays.

- Because of network file locking semantics, only one person could access the Excel file at a time even if they only wanted to read it. This led to coordination problems (about who had the shift book opened) and lack of awareness (when the shift book was not read for some time).

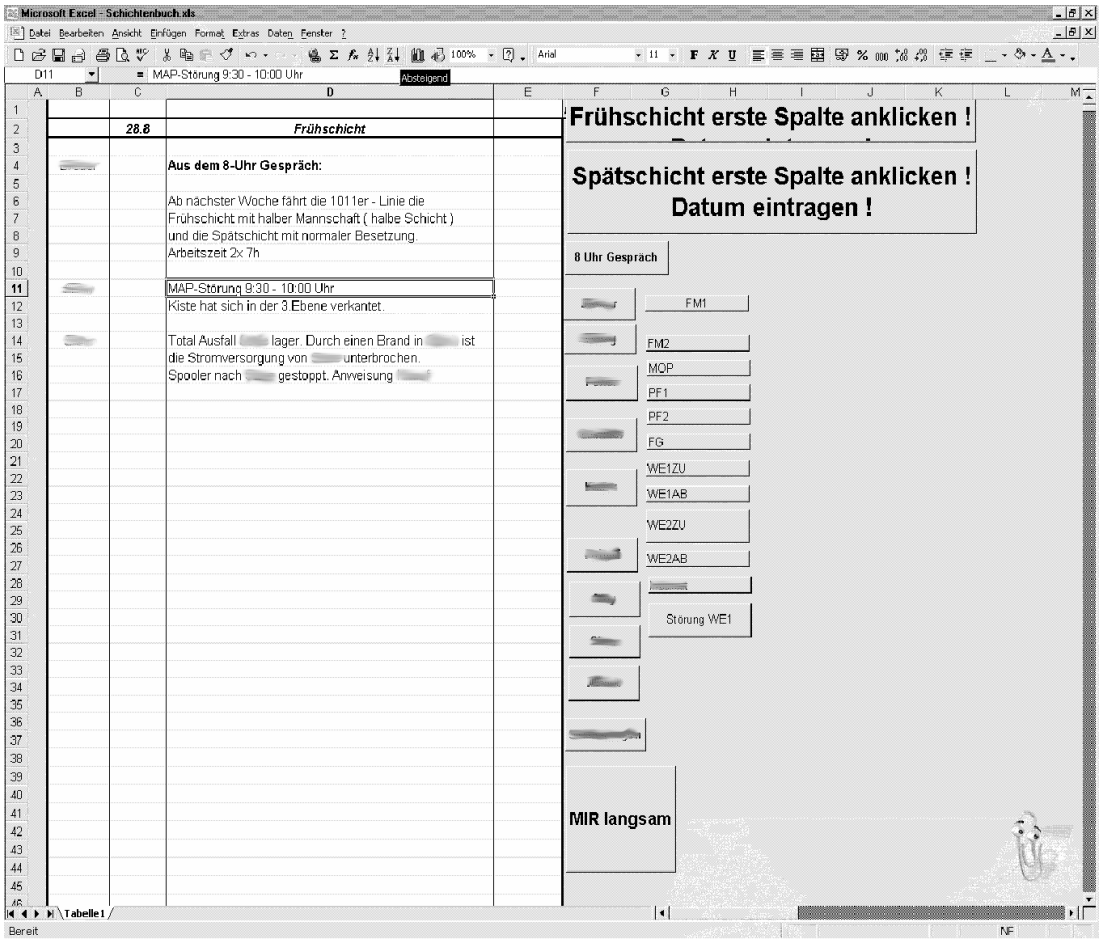


**Figure 5 The Excel version of the shift book. Buttons on the right hand side provide ways of inserting names and standard pieces of text quickly. They also act as reminders of entries that need to be made on a regular basis (e.g. about the 8 o'clock meeting).**

Because of these problems, it was decided that I should develop a new shift book application that would help to address these matters. In order to integrate the system with the existing IT infrastructure and to start development at a high level of functionality, I considered using the existing Lotus Notes / Domino system as a platform and started developing some demonstrators. However, I found that the

application programming interfaces of the Lotus Notes client in use at the time were insufficient to develop some aspects of the envisaged system. The main requirements for the shift book were as follows:

- The electronic shift book should support full-text searching and retrieval based on relevant categories as well as links between entries.
- It should be accessible by all control room workers, and coordination should be supported, especially with regard to the whether a particular entry had been 'dealt with'.
- We planned to link the shift book to other systems such as the assembly control host. This would have required an interface to some well-supported middleware infrastructure which was not available.
- It was intended to make the system customisable and programmable by end users. This required the use or design of a language that is simple to learn and is easy to extend with domain-specific constructs.
- As shift books were used in other parts of the plant, we wanted to develop the system in a way that would allow its wider uptake within the plant (especially by the maintenance workers).
- An important part of making information available in the shift book was to include screenshots of the assembly control host screens. The process of producing them involved the use of a screen capture tool (to capture and crop the image) and pasting into the Excel sheet. We wanted to make the process less cumbersome and allow for data available in the screen to be available in searches.

A review of these requirements as well as the problems involved in getting access to the Lotus Domino server led me to chose a different set of technologies to develop the shift book: the MySQL relational database management system and the Tcl/Tk language (with some extensions written in C++). A concern in making this choice was that EngineCo should be able to take over development of the application at the end of the project (or to get a third party in for this purpose), so the openness of the

technologies and tools used was of importance as was the development of a modular architecture.

Work on the shift book was done in the control room, thus making it accountable to control room workers and enabling them to see its progress and to comment on partial results. I would show them what worked but also what did not in order to give them a sense of the state of play and enable them to start seeing where the complexities of such a project lay. Occasionally, there was also a need to clarify issues that arose, for example, around the version of Internet Explorer that was available on the control room desktop PCs. A particular version was required for the print function to work, a problem I demonstrated on my laptop computer. Control room workers told me they were going to get new PCs which would have the latest version of Internet Explorer installed, so the issue would be resolved soon. An alternative solution would have been to change the code for the print function which would have made it more compatible with earlier versions of the web browser. The fact that the issue could be instead resolved through a simple enquiry demonstrates the situated nature of corealisation's work. Another issue regarding printing was brought up by one of the control room workers who wanted to be able to print off individual entries so that they could be taken into meetings. Until then, the system would only allow the current selection visible on the screen to be printed.

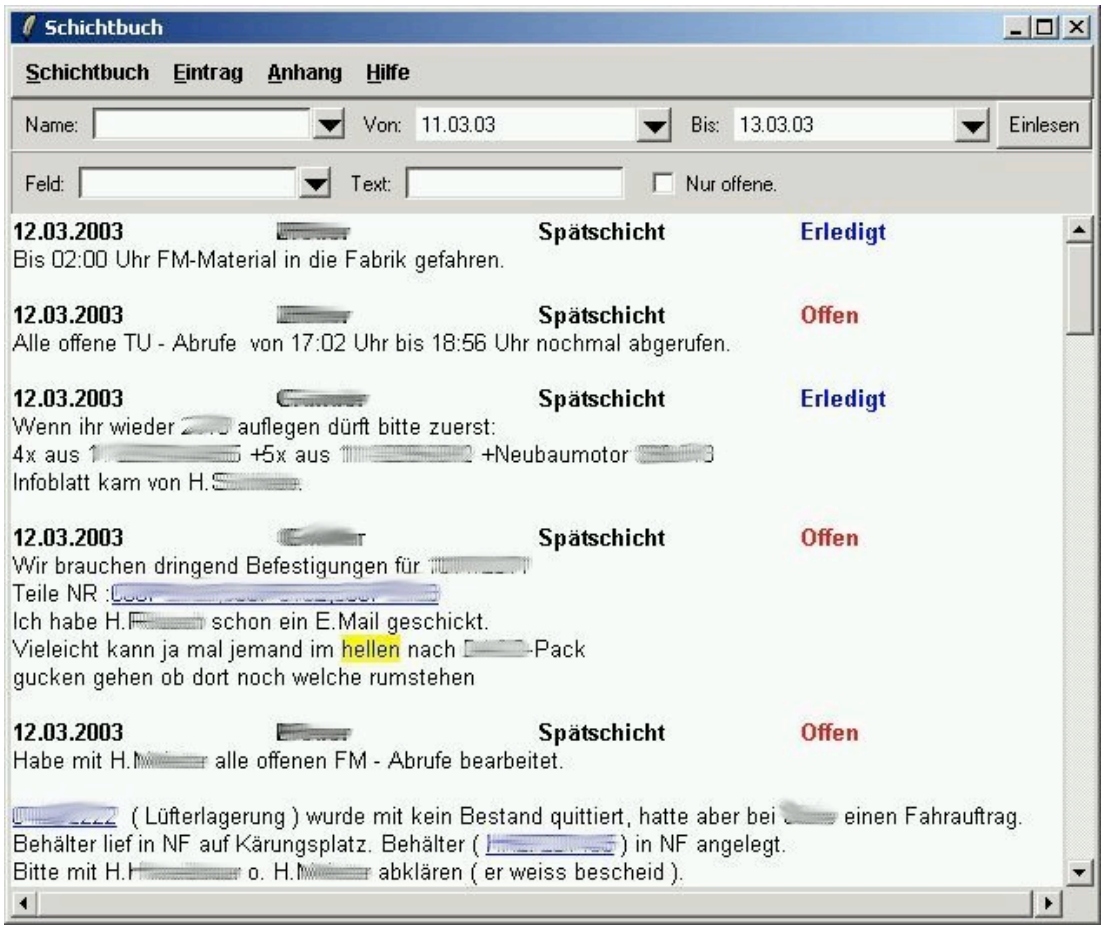


**Figure 6 The shift book application (list view, anonymised).**

A concern workers had was to be able to search the shift book for particular engine, order or part numbers. Two features were included in the system to facilitate this: text could be 'marked up' as representing engine numbers, order numbers or part numbers. The text would then be presented as a link. The idea was that this link would display information about the particular engine, order or part. Unfortunately, the link with the assembly control host could not be established during the project time, so this functionality was never completed. Another function of this markup was that it offered the potential to standardise the presentation of these numbers so that they would be easier to search for:

> ***Ralph***:*But you are still displaying a part number in different ways?[…]*
> ***Alex***: *How do you mean, differently?*
> ***Ralph***: *It is really eight digits… there are some that have eight digits, some with dashes, … [browses through shift book entries]*
> ***Alex***: *Yes, Peter has mentioned that today. That's why I said that perhaps I should install a new version*
> ***Ralph***: *… seven, eight, and then a space somewhere*

***Alex**: Yes, that should be standardised on entry, that this is brought into a single format.*

Ralph's remarks refer to the problem that control room workers enter numbers in a number of different formats, sometimes omitting digits, sometimes inserting spaces to make them more readable, sometimes dashes. This means that the numbers would not be found in a search using a different format. Related to this issue was the aim to make information imported from the assembly control host searchable. Whereas previously workers would create a screen shot of the required information which would then be inserted into the Excel sheet, now they can cut and paste text from the terminal emulation window and insert it into the shift book as a special text type which would render it in a monospaced font and using semigraphic characters as in the original screen. This is show in Figure 7, note the order number (MA-NR.), the engine numbers (PA-NR.), customer name, etc. which can all be used to retrieve the entry.

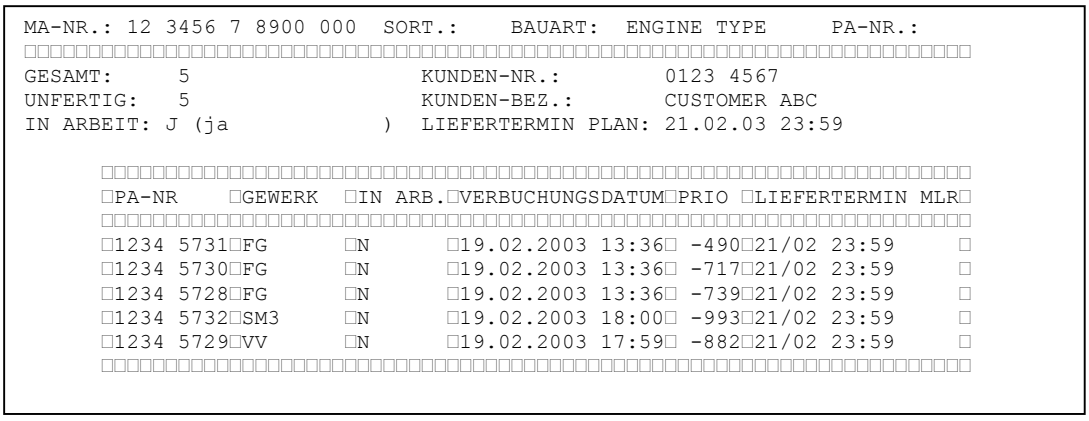


**Figure 7 Content from the assembly control host 'cut-n-pasted' into the shift book application as text, making it searchable and retaining its formatting.**

Initially, the shift book application had been used only by control room workers; others would consult the paper copy available in the control room instead of accessing the contents remotely via their networked PCs. However, the next fieldwork extract shows how this changed:

**Extract from fieldnotes**: The manager of the control room asks me to install the shift book application on his PC. After I've done the job I go the control room and tell Pete that his boss is now reading the shift book from his office. I say that I'm not sure if this is interesting information for them but that I wanted

> to tell them anyway, just in case. He says that the manager is reading the shift book anyway but that it's still good to know. He also informs the colleagues who are working on his shift. Will have to ask him if this has had any consequences for their use of the shift book. I know that some workers write their shift book entries at the end of the shift as opposed to as events unfold and they might have to change this practice?!

This points to the importance of the timing of information. It seems to be computer science's mantra that information should be accessible everywhere and all the time. However, as the above example illustrates, being specific about exactly when information should be made available might be quite important. So far, the control room manager had to go to the control room to read the paper copy of the shift book and he would do this at particular times during the day and control room workers would see him read the entries, so they would be aware of what he does or does not know. Similarly, one could imagine making the lists mentioned above available to interested parties via a web server but then again, control room workers would loose control over and awareness of who knows what at a particular time. At the moment, lists are compiled and sent out at the end of the late shift so that all interested parties would have them in time for the early morning briefing session taking place every day at eight o'clock where they are discussed.

## 6.3 Systemwatch

The visualisation system EngineCo uses (cf. section 5.3) was available only in a limited number of locations in the plant, so most workers were not able to see what the status of production was (the progress measured against the plan). Since the software package on which the visualisation system was based was no longer supported by the supplier, EngineCo did not have an easy way of making this information available to more people. I was therefore asked if I could try and find a way to make some information available through a web browser based interface. While writing the system that displays the data was relatively easy, finding a way to extract data from the legacy system was more complicated. A link was established with the help of a support technician who was maintaining the visualisation system and was able to extract data from it into XML files which would then be copied over to another server using a remote copy utility. Unfortunately, not all the information could be made available in this way as some was simply not available through the

legacy interfaces. However, the effort resulted in a visualisation of the most important parameters of production (part of it is shown in Figure 8) which is in routine use in the plant today.

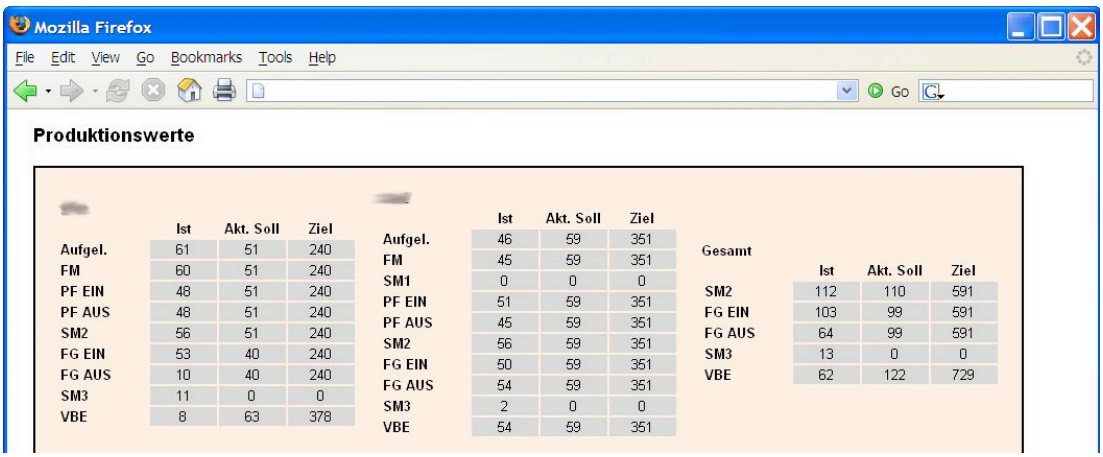


**Figure 8 The Systemwatch visualisation (only top part shown). It displays, for every unit, the number of engines produced, the target for the current point in time and the target for the whole shift.**

The example of the visualisation system shows how corealisation affords system development on the basis of what one might say are rather fragile arrangements. The service technician works for the original provider of the visualisation system and helps out at EngineCo only occasionally on the basis of a locally arranged contract between his boss and EngineCo. This arrangement allows EngineCo to continue using the visualisation system despite the fact that official support has ceased long ago[42]. Work on the system had to be fitted in with the support technician's other duties and significant negotiation was needed to reconcile his work on the project with the terms of the contract. Another aspect that was important was his knowledge of the system and the way it was configured – reengineering this would have been prohibitively expensive.

An interesting aspect of the visualisation system is the role it plays in the discussions within the organisation about productivity. Clearly, the system incorporates management concerns about productivity: it displays the current state of production in terms of engines built, and measures it against the overall target for the day as well

[42] There are plans to switch to a more up-to-date system but so far suppliers have not been able to demonstrate that they can transfer across the configuration of the system which is extensive.

as the expected progress at the current point in time. The target prescribed by management has increased significantly during the course of the study. These increases in demanded productivity went hand-in-hand with various reorganisations ranging from minor adjustments to major reconfigurations of the production plant (cf. section 5.4.2). One might therefore say that it is a tool for management to enforce their views about what level of productivity should be achieved. However, workers themselves are interested in monitoring their productivity. On the one hand, in as far as they are willing to comply with management demands, they do need to know when they are 'behind'. On the other, there are regularly situations when they are 'ahead' of schedule. These latter situations can arise for a variety of reasons but the main factor is the relative complexity of the engines currently built, their size as well as the degree to which they can be treated in routine ways. In addition, one has to consider the impact of breakdowns when formulating a new productivity target, i.e. the new standard has to allow for a certain degree of slack introduced by problem solving activities, so one should not orient to the maximum throughput observed as a measure for what might be a new target. Nevertheless, situations where production is 'ahead' of the current target are often picked up by management as an indicator that 'more is possible'. Consequently, workers have an interest in monitoring their productivity and taking appropriate action. This might consist in speeding up or slowing down their work or in complaining to control room workers about a 'bad mix' of engines scheduled for production.

It is important to note that these organisational politics were readily available as they form the background for a number of activities in the plant. The corealiser can draw on this knowledge and thereby make decisions about the adequacy of a particular course of action. This is not to say that they should aim to become arbiters in conflicts. However, an awareness of these issues is still important as it may impact on the feasibility of particular design decisions. Of course, should one *decide* to take sides or to try to facilitate, this knowledge is also of importance.

## 6.4 The Lift Station Case

Another effort related to the visualisation of processes and the issue of productivity in the plant was related to a particular part of the plant, the lift station where engines

are fed into an overhead carrier system and prepared for varnishing (see Figure 10). In this case, production workers complained that they were not getting enough deliveries of engines and were therefore unable to reach their target. They blamed the autonomous carrier system for this situation. I was asked by Ralph, a member of the plant systems management group (and former control room worker) to help produce a visualisation of the transports to the lift station on the basis of log files that were available. The graphical representation was intended to either refute the claims made by the production workers or to provide a handle on the problem that would allow it to be investigated.

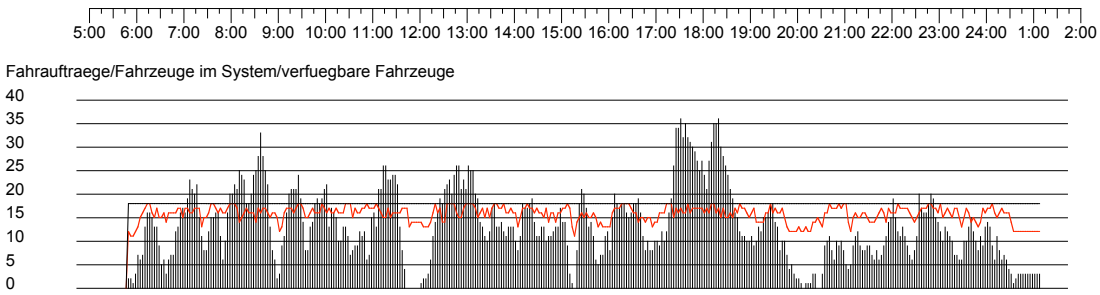


**Figure 9 The visualisation for the lift station, the graphs track the number of transport orders to the lift station against the number of available carriers (red line) and the total number of carriers in the system (the flat line).**

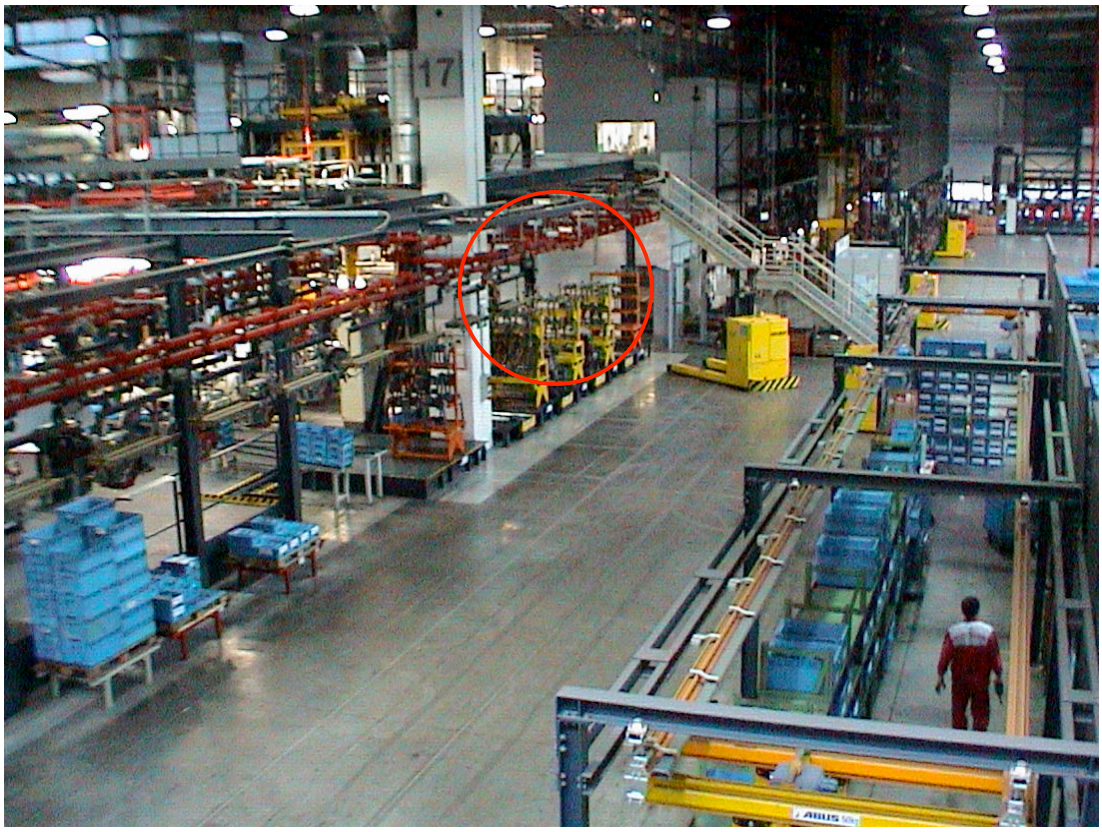

**Figure 10 The lift station which feeds engines into varnishing (in the red circle). Also visible are the yellow carriers which supply engines to it from stationary assembly (to the right) and**

As one can see in Figure 9, there are instances where there are significantly more transports scheduled than there are carriers to service them and therefore problems with supply can arise. However, such instances were quite infrequent and did not normally lead to situations where workers would be idle. Nevertheless, the problem was taken seriously and discussed in the 8 o'clock meeting. Ralph started looking for possible causes. The work of maximising the performance of the carrier system is a general theme in the plant which leads to a number of changes being made, ranging from relatively cheap and easy adjustments, for example changing the assignment of parts to storage places so they require fewer transports, to major changes in the physical structure of the plant (see 5.4.2).

The development of this visualisation required changes to the assembly control host to make log files of transport activities available for analysis. These modifications were made by ITCo programmers on Ralph's request and log files were made

available for download from an intranet page. I then wrote a program to parse these log files and convert the data into a number of diagrams in an SVG[43] image file (one of these can be seen in Figure 9). These could then be displayed in a web browser window or imported into a vector graphics program for further processing and inclusion in presentations or reports (e.g., for the 8 o'clock meeting).

The following extract is taken from a discussion between Ralph and myself:

> ***Alex**: which log file contains the information where an engine comes from that goes into the lift station?*
> ***Ralph**: log file 'FG' but that's ...*
> ***Alex**: no, it's not there*
> ***Ralph**: then you've got an old log file.*
> ***Alex**: ok, if that's been changed already*
> ***Ralph**: what date is this one? Yeah, ... since the ... 25th that's included, where the engine comes from*
> ***Alex**: so... can you give me an up to date one because then I have to change this because the format of the log file will have changed.*

We were involved in devising a strategy to parse the information in the log files of which there were a number of types. One would contain entries consisting of a list of entries containing the time of day of events such as 'transport ordered', 'transport initiated', 'carrier assigned', 'transport finished', etc. Another type would provide information about the status of carriers. While going through the individual fields, we realised that we had a problem in that a field was missing which would have given us information about which station in stationary assembly an engine came from (which would affect the duration of the transport). It turned out that I was using an old version of the log files which did not contain that field but that the change to include it had been made a few days ago and that therefore files in the new format were available.

The fieldwork extract shows how the ground for design is a shifting one and how people engage in practical coordination activities to align their work with that of others. The format of the log files changed a number of times as people explored what data was needed and, crucially, what was available. This involved various discussions within the plant systems management group and with ITCo staff. In each

[43] Scalable Vector Graphics, an XML-based graphics format.

iteration, a change request had to be made to the ITCo programmers and some time had to pass before new data was available before the next round of development could begin. Ensuring that the latest version was used and knowing the relevance of changes made were an important aspect of the work. Another important part was accounting for the log file's features, for example lines containing a sequence of dashes instead of the expected time of day entry. These would occur where an engine was left in the lifting station over night, so the day's log file would not contain entries showing its arrival. These entries were exceptions because normally workers would ensure that the station was empty at the end of a shift. Another feature found was that durations recorded could be negative. While this could have suggested a problem with the way the data was generated, Ralph had an alternative explanation to be found in the detailed arrangement of work in the lift station: workers would sometimes use the worker information system to retrieve an engine's data before that engine reached the workplace, that is, while it was still in transit from a buffer space. Interpreting the data and relating it to real-world events involved significant skill on Ralph's side, as a knowledge of both the working of the lift station, the practices of the workers there and of the carrier system was required to make practical sense of the data provided. This knowledge allowed him to distinguish 'interesting' features and those which could be safely ignored.

## 6.5 Ad-hoc Innovation

While at EngineCo I started working with XML (as part of the shift book development) and when I observed a control room worker browsing through an XML file using a web browser, I asked him what he was looking at. He explained that these were log files generated by the process of programming the control units for certain engines and that he was looking for error messages. As the browser did not support a formatted view but showed the 'raw' XML code, I offered to write a program that would convert the data to a more convenient format in HTML which would highlight error messages. The program was written within a day and deployed immediately. This gave rise to a number of activities around these particular XML files. A few days after the first program was written I was approached by a member of the plant systems management group who asked if I could help to identify all

engines affected by a particular problem. By modifying my initial script I was able to scan all available log files and produce data that showed which engines were affected. This data was passed on to the quality control department who assessed the situation and asked the control room workers to 'lock' the engines affected.

This is just one example of a number of ways in which 'being there' enabled me to observe situations where I could offer to apply my skills as an IT professional. Doing this in turn encouraged colleagues to approach me with problems as they became more aware of what was possible and how easy or difficult a particular development would be. The example given here demonstrates that the local availability of IT expertise can lead to unanticipated innovation which can be of immense benefit. In the case described the benefit of being able to hold engines in the plant and correct the problem before they were shipped to customers far outweighed the costs involved (and not just because I was funded on a research grant). Solutions to problems also accumulate as a stock of known-to-work candidate solutions and can potentially be traded and deployed more widely. It is a corealiser's job to exercise judgement as to whether the potential benefits justify the costs, that is, they should be concerned with the management of the essential reasonableness of requests which is predicated on their knowledge *qua* member.

# 7 Discussion

Having introduced corealisation as a radical respecification of IT design work and demonstrated the nature of its practices through a real-world study, I wish to now consider how it can help to deal with the issues identified in chapter 2 and how it builds on and extends the approaches described in chapter 3. In this I will refer back to the fieldwork material presented in chapters 5 and 6, highlighting how the practices of corealisation are tied to members' mundane competencies and are collaboratively achieved.

## *7.1 The Work of Corealisation*

The fieldwork material presented demonstrates how IT work is collaboratively achieved at EngineCo by members of the plant systems management group as well as the researcher. In the case of the daily reports system, we have seen illustrations of some of the ways in which Barbara, Mark and David make use of their knowledge as members both to manage the process of design and to reason about the design object (the daily reports system) and how it might fit into evolving working practices. Whilst these findings do not directly contradict those of Bowers and Pycock (1994) of gradients of resistance, I would like to draw attention to the fact that in their fieldwork we find no orientation to a context of design and use. The features of the interaction reported by them exemplify all the problems of lack of the common ground as members normally enjoy it. The character of design talk here differs from that presented by Bowers and Pycock in that the interaction revolves around the object of design in context as opposed to the object *per se*. This study provides an insight not only into the interactional features of design talk but also into the ways in which such interactions take place within a larger organisational context, how they are reflexively tied to the biography or 'natural history' of the setting.

Corealisation at first sight might seem to place impossible demands on the IT professional who is called to not only attend to all aspects of the technical work but also engage in some form of 'practical sociology' (Anderson 2000) whilst maintaining cordial relationships with all members of the setting. There are two important aspects in this: firstly, there is the question to what extent corealisation

really introduces additional work and how work gets allocated to individuals. I would argue that, by and large, corealisation renders work visible that was necessary but hidden in development work according to traditional methodologies. Secondly, I will suggest some strategies for how the work can be done effectively and efficiently, i.e. I will suggest some *candidate methods* for corealisation – without wanting to define a methodology of any kind as this would violate the unique adequacy requirement.

In the following I will discuss the work of corealisation as instantiated by the fieldwork material presented in the preceding chapters. My aim is to explore how the principles formulated in section 4.7 are translated into practice and what issues arise.

### 7.1.1 Breaking Down the Boundaries Between Design and Use

There are good practical reasons for breaking down the traditional boundaries between design and use. Bødker *et al.* (1987) suggest that what is needed at the beginning of a design project is a phase of "mutual learning" where workers learn about technical possibilities and IT workers learn about the work domain and setting. Of course, this can best be achieved through co-location in the workplace itself, where the phenomena are massively available as a resource for design and where technical possibilities can be explored *in situ* and in anger. The traditional divisions of labour in systems development are often carried over into participatory design projects in that the interaction takes place outside the users' normal place of work, thus making the resources of the workplace unavailable:

> It was widely held among those involved with Trillium that meetings to discuss [workers'] experiences with the technology were important. However, because such meetings were removed from the situation of use, discussion frequently shifted to talk about the details of particular implementation decisions. The situation might have been improved if the discussion had taken place in the [workers'] work environment where supporters could have observed use of the itemtype menus in relation to [workers'] work activities and where [workers] would have had experiential access to their own work practices. (Blomberg and Henderson 1990, p. 356)

One of corealisation's main principles is to take the work of design into the workplace, so that all the resources that are available there (and ready to hand) can be used. We have seen examples of this in the case of the daily reports system where

existing reports were used to discuss their current use and to explore what a new system might look like.

Corealisers are called on to acquire and then make use of a member's understanding of the setting. What is required is the development of a particular 'design sensibility' or orientation to design that is focused not on the computer but on human activities. Blomberg, Suchman and Trigg (1996, p. 260) provide a number of 'rules of thumb' which might characterise what could be understood by 'design sensibility':

> Look for "invisible" work. […] Find the knowledge work in what is characterized as routine, and the routines in what is characterized as knowledge work. […] Expect and encourage joint project design and definition with worksite participants. […] Assume that change is already and always in progress. Understand the politics of change and where you stand within them. […] Understand how extended contexts (e.g., institutional and global) constrain the scope of what can be accomplished in a given setting, and attempt to question or take advantage of those contexts as appropriate

It is this orientation to the social organisation of work and its reflexive relationship with technological arrangements that makes the work of corealisation possible. It is acquired through experience but is also teachable through demonstrations and ethnographic accounts of the detailed accomplishment of social order.

Another way in which the breaking down of boundaries becomes important is in the social organisation of the development work itself. In the fieldwork material we can see how the work of corealisation is practically organised, how it is conducted as an orderly process. The main resources made use of here are not methodologies and the structuring devices they provide but rather a practical alignment of corealisation work with the organisation's other business. As we saw in the daily reports system example, Barbara makes use of her knowledge of who is available and when as well as what experience people have with various aspects of systems development (e.g. whether they were involved in building the original Excel-based reports) to arrange the visit to the Meisters' office. Because of the ready availability afforded by co-location, coordination of activities can be much more lightweight and opportunistic.

### 7.1.2 Membership and Hybrid Knowledge

While corealisation calls on IT professionals to become a member of the setting and draw on an understanding of the setting 'from within', attending to its biography,

there is the question of how one can initially familiarise oneself with a setting and how one maintains familiarity with it. As the aim of corealisation is to design, there is a need for corealisers to adopt an attitude that is different from the natural attitude adopted by other members of the setting. Corealisation calls for them to adopt ethnomethodology's analytic mentality which seeks to uncover the methods by which social order is achieved in and through people's ordinary actions in the here-and-now, i.e. in the setting at this point in time. It invites them to use this non-ironic stance to render the phenomenon of social order visible (so that the findings might be used to inform design decisions). The move made in ethnographies of rendering the familiar strange, noticeable and interesting is therefore one that will be of value in corealisation as well. Practically, this means that corealisers will engage in the same kinds of activities that ethnographers engage in, although their aim is not normally to produce ethnographic accounts for publication in scholarly journals.

I suggest that the work of Harper (2000) is of relevance to corealisation, who suggests ways for ethnographic observations to be organised (cf. section 3.1.1). In similar ways, there are practical ways of organising the work of corealisation. The fact that corealisers are not necessarily in the business of writing for academic publications does not mean that they are not involved in activities similar to those an ethnographer might engage in. Gathering 'fieldwork material', i.e. recoding activities, collecting documents or interviewing people can be part of the repertoire of a corealiser's methods. Consequently, the work of reviewing, sorting, compiling and transforming the various materials collected and notes made are also part of the work of corealisation. Methods that have been found to be useful in the work of ethnography for design are therefore of relevance. For example, Randall, Hughes and Shapiro (1994) found that the debriefing sessions they had with designers were actually helping them to work up the accounts of what was observed in more detail. Also, in the same way that writing a paper occasions the work of reviewing the material collected, reviewing one's experiences and seeing them from a particular angle, sharing 'stories' with fellow corealisers (who may or may not be working in the same setting) can provide a useful way of reflecting and drawing out implications for design. In this way, one may make use of the fact that one's work is part of a larger context which can provide opportunities to reflect on the process of corealising

by discussing the experience with other (IT) workers, potentially contrasting the view from one perspective with that from another.

While the problem of 'finding out' about the setting and the methods its members use is of particular interest at the beginning, when an IT professional first encounters the setting and needs to establish familiarity with it, there will also be a need for continuous reflection and reorientation as IT professionals will shift their attention according to shifting priorities, as the setting itself will change and new areas will become relevant while others will cease to be of importance, etc. Working in a 'living, breathing organisation' brings with it constant change and corealisers are asked to find practical ways of coping with this.

Of course, this learning process is not one way. In addition to the designers' learning about the practices of the workplace and the knowledges and skills workers have, users need to learn about IT design practice: about what is possible, what is easy or difficult to achieve. By bringing IT design and development activities into the users' context, corealisation supports user learning. In this way, users become better equipped to make suggestions that are informed by the realities of IT systems design and development, and by the capabilities of the technologies. It is the mutual understanding of respective rights and obligations and the establishment of reasonable expectations that is at the heart of the design *qua* member. Indeed, it is at the heart of the notion of membership itself.

I wish to explore this a bit further by borrowing the image of the 'Janus faces of design' from Bowers (1991)[44], who describes design as having two opposing faces: one face, that of 'ready made design', looks at the world in terms of clear structures, equivalences identifiable aims and objectives, 'best' solutions and so forth. Opposing it is the 'design in action' face that sees the wild contingencies of the world, the relativity of such terms as 'best', the conflicts and negotiation, their for-the-moment resolutions, the vagueness of similarities, etc. While the 'ready made design' face will sign the contract and go straight to work in the confidence that 'the right representation' will boil things down to their 'essence', the 'design in action' face is

[44] Who in turn borrowed from Latour.

in danger of being endlessly caught up in its bewilderment and inability to act in the face of the wild contingencies of the world.

Clearly, corealisers have to find a way to reconcile, for all practical purposes, these two extreme positions and come up with a way forward. Bowers suggests that the two faces represent institutionalised conventions and that designers display versions of both according to the task at hand (convincing customers to sign the deal vs. working out details of the design). While corealisers are faced with the wild contingencies of the world just like the 'design in action' face Bowers describes and may therefore find it difficult to come up with ways forward, they can also draw on the resources of the setting to help them overcome this problem.

It is perhaps important to restate the role that the notion of hybrid knowledge plays in the project of corealisation (see section 4.4). The problem that corealisers face is neither one of gaining complete knowledge (in some sense) of the setting, nor to become competent practitioners themselves. Becoming familiar with a setting in the same way that a member would (but not necessarily learning to do a particular job) is a feasible project as well as an ongoing, practical one. There is no need to worry about the wild contingencies faced as they can be faced one by one and in the company of others.

### 7.1.3 The Use of Surrogates

Whether one looks at participatory design and its problem of representation or at the use of ethnographies to inform design, one often finds that users get replaced by surrogates of some kind or other (either other users presumably speaking on their behalf or by the ethnographies produced from a study of the workplace). Even where users are allowed to speak for themselves, their actions are often captured in some form or other, be it in the users' accounts, video footage or ethnographers' notes. The activity that is at the heart of the matter is present only in represented form. While this is sometimes unavoidable for practical reasons, I wonder why we have spent so much time developing surrogates and methods for handling them without considering the use of the phenomenon itself in its massive availability as a resource for design. Suchman (2002, p. 94) speaks of "various sorts of surrogates, proxies, stand-in's for 'the user,' designed to allow the creation of usable technologies in the

absence of these other relations". While sharing her concern with the web of social relations between 'designers' and 'users', I would like to suggest that the loss of the phenomenon itself is equally problematic. This problem is a direct outcome of the separation of design from use in both space and time and it is this division that needs to be broken down as well as the barriers between 'users' and 'designers' as different professional groups.

Representations are valuable tools for supporting shared practice but one should not see them as the means *by which* design is achieved. They are practical tools that help in the process of working up a shared understanding of the 'design space', i.e. the relevant part of the lifeworld, the potential benefits an IT system might have, the potential problems and the space of candidate solutions. Scenarios in particular are useful in illuminating the potential futures but their limitations should not be ignored (Bowers 1991, Suchman 1995).

### 7.1.4 Prototypes, Mock-Ups and Working Configurations

In contrast to other approaches to prototyping, corealisation does not suffer from the problem of having only limited time (both in terms of the 'session' ending and in terms of the overall 'project') as it involves a longitudinal engagement with the setting where the parties engaging in the process are at least potentially available at any time. Like extreme programming, corealisation aims to provide value in the form of working (partial) systems early on in the process. This is not without its dangers, for as Kyng (1995) points out:

> Particularly when the prototyping environment is the same as the implementation environment and/or when versioning is applied, maintaining a shared understanding of what is representational, what is coincidental, and what is actual becomes difficult. (*ibid.*, p. 54)

This point, of course, applies equally well to corealisation with its emphasis on rough-and-ready initial solutions and refinement through experience with a working (but potentially partial and prototype-like) system. It will be necessary occasionally to remind people ('designers' as much as 'users') what the state of the system is, how it has come to be the way it is and what its envisaged trajectory is. On the other hand, a system in use is infinitely more valuable than any representation for communicating its scope and behaviour. Corealisation thus brings together the

system as standing on its own behalf and the opportunity to reflect on the current state of play through observation and informal interaction as well as more formalised practices.

At the same time, the context in which the design work is happening and the biography of the setting are important resources for corealisation. As Bødker writes:

> [It is] necessary to develop design strategies where the present computer use (experiences as well as software and hardware) constitutes the basis for the future change. (Bødker 1994, p. 17)

The aim is to foster an ongoing reflection on the state of play and possible courses of action that can be taken to develop working practices further and to maintain systems in good 'working order'. Corealisation takes seriously the 'hidden work' in IT (see section 2.4.1) in that it does not draw artificial boundaries between different tasks such as design, operation and maintenance but rather takes responsibility for the whole 'lifecycle' of a system.

The changing landscape of information technologies, the wide availability of modules and artefacts that are produced increasingly as generic components makes new approaches to building systems possible. Increasingly, practical systems can be built with little or no technical expertise through a 'pick-and-mix' approach (Brady, Tierney and Williams 1992). In a similar way, component architectures (Pfister and Szyperski 1996, Szyperski 1999, Hopkins 2000) have the potential to change the ways IT professionals go about designing systems and to improve their productivity significantly through re-use at a high level (compared to previous approaches such as subroutine reuse). At the same time, generic packaged systems have become common in many sectors (e.g., enterprise resource management, healthcare records, etc., cf. Pollock, Williams and Procter 2003).

Indeed, IT design and development is increasingly becoming configuration work (Fleck 1999) which is concerned with establishing a working ensemble of systems and practices from components acquired through procurement. This means that the character of IT work changes from design *ab initio* to making choices about which market offering to adopt and dealing with design decisions made elsewhere. The aim is to come up with workable solutions that establish a reasonable fit with local circumstances. Corealisation explicitly embraces these changes as they offer the

opportunity to lower the baseline costs of developing a candidate solution and, if the right basis is chosen, allow more effort to be spent on developing the aspects of the overall configuration that make it uniquely adequate. Components and packages can support prototyping by making high-level, generic functionality available quite easily. CSCW components in particular would be of immense value as they could help support informal working practices so often ignored in the design of vertical organisational systems (cf. Hardstone *et al.* 2004).

While the picture of technology supply has changed, so has the level of IT skills that non-IT professionals have. General computing skills are now widespread and popular desktop applications support the design of surprisingly sophisticated systems on a small scale. End-user computing is here and here to stay. I would argue that end-user computing and corealisation are ideal partners as the increased availability of IT expertise will encourage 'users' to experiment and learn more. At the same time, their efforts can be leveraged and taken to a new level if the systems they produce or configure can be integrated with wider IT infrastructures through corealisation. The two can play hand-in-hand, leading to more 'user'-drive innovation in IT development and use. While it has not been possible to explore this issue in much depth at EngineCo (as originally envisaged for the shift book application), the cooperation between myself and members of the plant systems management group, who are not formally trained IT professionals, suggest that corealisation can be a useful vehicle for enabling workers to get more directly involved in systems development. New developments in the IT industry like the move towards service oriented architectures promise to open up the arena even wider than traditional component technologies have.

### 7.1.5 The Practical Politics of Design

Various researchers have made the point that situations are often not characterised by what we might call capital-P politics between clearly defined societal groups but can involve complex lines of conflict and only partially consistent positions around a range of concerns which are seen within a much wider context than just a single workplace (Bødker 1994, Büscher *et al.* 2002, Shapiro 2005). At the same time, while IT projects often bring with them new potential for conflict, the responses they

meet are usually not ones of outright support or rejection as the potential outcomes are very much uncertain. In addition, even for individual parties, a number of different strategies – such as cooperation or confrontation – are often equally possible and views even in otherwise 'homogenous' groups may differ about which is the best to pursue.

Corealisation does not offer a political programme but through its analytic mentality provides an approach to dealing with the 'small-p politics' (Büscher *et al.* 2002), the practical politics found within the setting. In accordance with ethnomethodology's principles it does not treat these phenomena as something to be reified according to a political programme and subjected to a predefined remedy but rather suggests that the practical politics should be treated as a member's phenomenon. The challenges faced are often complex one where the lines of conflict are not clearly drawn and are often between peers as much between levels of the organisational hierarchy (cf. Kensing, Simonsen, and Bødker 1998a). What is required, then, is a general awareness of the possibility of conflict and practical measures where required. As Büscher *et al*. (2002) have put it:

> … there is in PD [and in corealisation] a *practical* politics relating to the sharing of responsibilities, the working up of trust and the sustaining of commitment, and these must be achieved in a situated manner within a constantly changing context. (*ibid.* p.183)

Managing the potential fragile alignment of technologies, components, working practices, people, etc. is part and parcel of the work of corealisation. The practical politics, as a member's phenomenon are recognisable and potentially solvable using members' methods. We have seen examples in the development of the daily reports system but perhaps most clearly in the case of the shift book: having the sense to realise that the installation of the shift book application outside the control room may lead to potential conflict and taking the practical action of letting control room workers know are both based on a member's understanding of the setting (see page 158) as is knowing that potential conflict can be avoided and that other courses of action, for example refusing access to the shift book, are not realistically feasible.

### 7.1.6 Opportunism Used Strategically

The closeness of design and other work activities, that 'being there' (Hartswood 2000) enables many opportunistic moves: anticipating requirements through observation of practice, using the tools at hand to envisage new ways of working, picking up on work-arounds and refactoring them so that the can become part of new systems and changed working practices. Corealisation provides the space for these opportunistic moves to occur more frequently, almost in a strategic manner in the sense of 'staying ahead of the game'.

Part of a corealiser's repertoire of methods is 'keeping an eye open' for such *ad-hoc* activities which allow new avenues to be explored. Some will turn out to be dead ends while others will spark significant interest and lead to useful innovation. The point is that, with corealisation, the opportunity exists to bring to bear IT skills in unanticipated ways as demonstrated by the example of the XML log files described in section 6.5.

## *7.2 Issues*

In this section I wish to consider some of the issues that systems development faces and how corealisation relates to them. In particular, I will discuss issues of flexibility, scale, organisational context and dependability.

### 7.2.1 Dealing with Shifting Grounds

Systems development has often struggled with the shifting ground it encounters: requirements that change and largely unpredictable contexts in which systems have to operate (cf. section 2.1). Recent methodologies such as extreme programming (Beck 1998, 2000) have started to develop methods which deal with these issues in a more proactive manner by making change part of the plan rather than treating it as a problem. In contrast to earlier approaches for iterative design, these 'agile' methods do not assume that there will be a steady approximation of an ultimate stable condition but acknowledge that the process will be more unpredictable and that each next design decision has to be taken in the light of circumstances.

Corealisation takes a similar stance and like extreme programming emphasises the need to deliver a working solution quickly which can be subsequently be modified in

the light of what is learned through its use in anger, in the real world setting. It also deals with changes in the context. However, in contrast to extreme programming, corealisation does not rely on intermediate representations to facilitate the process of adjusting the development process and the system to changing circumstances. Rather, it relies on corealisers' knowledge as members of the setting to handle these matters. The crucial difference is that the rich understanding of the setting and its various contingent features enables corealisers to *anticipate* change and to accommodate it in their design decisions. This enables them to avoid the effect of 'painting oneself into a corner'. We have seen an example of this in the case of the daily reports system where a core concern was that EngineCo should have the option of hosting the system on its own servers rather than buying in this service from ITCo. Another example can be found in the discussion of the reports distribution problem: to members it is obvious that a new fax machine will eventually not be sufficient as it still has a limited number of spaces for storing telephone numbers. By drawing on his knowledge of the working practices in the plant, the control room manager can make an 'educated guess' that this would be likely to become a problem soon, so the costs involved in buying a new fax would not be justified. This also shows how corealisation enables decisions to be made as to the allocation of resources according to what is known at the time that decision is made

It is important to stress that this does not mean that the decisions made will always turn out to be the 'right' ones. Things may still turn out different than expected but by drawing on their detailed knowledge of the setting and the state of play at a particular point in time, corealisers are able to exercise judgement as to where the uncertainties lie, what risks are reasonable to take and how they can be managed. As pointed out in section 6.1, some decisions will be made on a 'suck it and see' basis where uncertainty exists and people can make practical arrangements to limit the risks involved. Corealisation's commitment to a longitudinal engagement allows the case to be taken up again quickly and in a relatively unproblematic way as a matter of routine.

Foregoing the comfort of the illusion of stability will have its costs and of course there will be a need to manage the rate of change but these matters can be handled as a practical matter, that is, they become a matter of decisions taken in the light of

circumstances. Considerations can involve such issues as the urgency of a change, the availability of resources, other commitments and so on.

Of course, there are situations and systems that make stability a requirement. For example, some systems that produce legally binding results (e.g. in banking) or are otherwise regulated (e.g. healthcare) may require a strictly controlled change management process. However, I would contend that these circumstances do not preclude the use of a corealisation approach, indeed corealisation's unique adequacy requirement provides a handle on how these matters can be practically handled. Even in highly regulated systems there is often scope for 'areas of lesser rigidity', where systems can be made more malleable. In practice, it will likely be the case that the flexibility required is catered for outside the core systems (e.g. electronic patient record systems or the assembly control host system at EngineCo), through various working practices that allow a larger degree of flexibility (cf. Hardstone *et al.* 2004). Corealisation's analytic mentality provides a way to ensure that the detailed working practices are not ignored and 'designed out of' the system but that instead they can be taken seriously as properly constituting the organisation's work. It will enable corealisers to consider where systems need to support informality (*ibid.*) and where they need to enforce particular practices in order to create uniquely adequate assemblages of systems, artefacts and practices.

### 7.2.2 Designing for Collaboration

As mentioned above, the analytic mentality that underpins corealisation provides the grounds for consideration of detailed, situated working practices within the project of developing systems. Where more traditional approaches would enlist the help of social scientists or would call on workers to provide input for design, corealisation takes a radically different approach: by enabling corealisers to become immersed in the setting and thereby to observe the naturally accountable phenomena of work, it allows them to build up a familiarity with the setting that they can draw on to uncover details of working practices that are relevant to systems development. It trades on the massive availability of the phenomena and the members' ability to make sense of the social organisation of activities.

For example, Barbara's knowledge of what is involved in spotting breakdowns, reporting them, taking appropriate measures, etc. and how various parties are involved in these activities, enables her to see how the daily reports system is more than a management reporting tool. Rather, it supports a complex and detailed working division of labour involving the collaboration of a number of parties over varying periods of time and in various locations and roles. She can bring to bear this knowledge directly and relevance it to the evolving understanding of the requirements for the system by suggesting that maintenance workers will not be able to contribute to the reports in the same way as the *Meister* because their (i.e., the maintenance workers') work is mobile (see page 139).

Similarly, in my work on the shift book, I was able to draw on my experience gained during the initial period of fieldwork, which involved studying the use of the existing Excel application. I was able to formulate the requirement for a search function and to relate this to the issue of capturing information from other systems such as the assembly control host. The observation that control room workers provide information to fellow workers by pasting in a screenshot, seen in the light of the problem of searching let to the requirement that if should be possible to cut and paste data in a straightforward way and in its textual form (see page 158). Another example is provided by the issue of when workers write shift book entries. Having the sense to see that the fact that the control room manager could look at the shift book at any time and from his office rather than in the control room led me to consult with control room workers on this matter, resulting in changes to how some of them used the shift book.

### 7.2.3 Designing for Skilled Workers

Corealisation treats of members' knowledge and skills as a crucial resource in design. Through the direct involvement in the setting, it allows members to bring to bear their understanding of the setting in a direct, unmediated way. An example is the work on the lift station visualisation where Ralph was involved in the work of making sense of the log files and its various features and thereby directly contributed to the development of the software.

In addition, corealisation treats workers as competent practitioners in their field, who are interested in IT insofar as it can help them to get their work done. By making IT skills available at the point of use, it provides support for innovation driven by workers' interests. This is in contrast to much work that has focused on making systems 'idiot proof':

> A consequence of trying to make such a system is that an incredible amount of "intelligence" must go into its initial design and maintenance. Taken to the extreme, we have the prospect of artificially intelligent systems operated by morons – an absurd scenario. (Bannon 1991 p.29)

Bannon further points out that while it is a laudable aim to produce systems that are 'easy to use', this comes at a cost and may even be inappropriate for systems that get used by skilled workers every day. In the fieldwork examples, we see little concern with what might be called 'general usability' which is largely taken for granted. Rather, people are concerned with how systems will support them in their work and will enable them to bring to bear their knowledge and skills. We can see this in the example of the daily reports system where the arrangement of a list of items in a selection box becomes a matter for discussion (see page 141). The problem here is that the specific organisation of items in terms of local relevance has been replaced with a general arrangement; instead of ordering items by their location in the plant the programmer ordered them alphabetically.

The experience of practicing corealisation at EngineCo has demonstrated that workers (in the control room as well as on the shop floor) are often quite sophisticated in their use of information technologies and are interested in expanding their skills where they find that it may help them in their work. Ease of use is one dimension, power of expression another. Seen from this angle, the usefulness of the ubiquitous menu-driven, point-n-click and draw-n-drop interfaces becomes an interesting issue. While such interfaces are convenient for many tasks and relatively easy to learn as they do not require recall of commands, they make an important class of operations next to impossible: complex manipulations of large numbers of objects. While it is relatively easy to convert a group of image files to a different format using a command-line interface, doing the same using today's GUI interfaces would require repeating the same operation for each file. This has been seen at

EngineCo where people often struggle to cope with vast amounts of data which are not easily manipulated using the desktop computing tools available. This is where corealisation can provide help by providing scripts which can be taken up and modified by or in collaboration with workers. Systemwatch is a case in point as it has been developed jointly by Barbara and myself and is now maintained by her.

### 7.2.4 Orienting to the Organisational Context

Systems development methodologies serve not only to organise the work of developing systems but also to make it organisationally accountable (e.g. Button and Sharrock 1996, 1998). For example, requirements documents are used for a number of reasons (Jirotka and Wallen 2000), some of which may not be optional in the context of organisational arrangements.

> [Requirements specifications] must first inform system design and procurement, second, postulate the relationship between the system design and system performance when deployed, and third, do all this in a transparent and accountable way to a number of interested parties with different motivations and attitudes to innovation in the workplace (*ibid*., p. 243).

What Jirotka and Wallen point to is the fact that requirements specifications play an important organisational role that is not easily dispensed with. Systems development work needs to orient to the organisational context in which it takes place and needs to be made accountable. Corealisation will not be an exception to this if it is taken seriously and accepted as the organisation's work. In my fieldwork, there were situations where I had to account for my activities at EngineCo even though I was not a member of staff but a student funded on a research grant. The accounts I produced took the form of oral or written status reports to the control room manager. Usually, these reports were quite informal but on one occasion I was asked to produce a more formal report to be discussed with the plant manager who had to sanction my work at EngineCo. Corealisation, if employed in an organisational setting will need to work under the rules of that organisation and in this respect will need to account for its work and its achievements.

Other ways in which the work of corealisation needs to take into consideration are visible in the way in which Barbara reminds the Meister of the mobility aspect of maintenance work in their discussion of the daily reports system. One might say that

a member's knowledge of the setting is not limited to particular local surroundings but encompasses also knowledge of other work contexts as well as knowledge of wider organisational concerns. When corealising systems, various concerns will be relevant which will extend beyond the immediate context of the particular setting in which the work is conducted. This is particularly important where corealised systems are taken up on a wider basis. For example, what would it mean to make the shift book application available to maintenance workers or the *Meister*? In what ways does their work differ from that of control room workers? While a members' understanding may not provide exhaustive answers to these questions, it will at least provide some tentative ones and the sense to ask and explore further. Corealisation's treatment of systems development as producing evolving arrangements provides the basis for their treatment as a resource with which corealisers actively engage rather than a finished 'solution' that can be unproblematically adopted elsewhere.

There are a whole host of other concerns relating to the organisation context such as technological alignment and standardisation which I cannot discuss in depth in this thesis.

### 7.2.5 Scale and Costs

This raises the issue of scale and whether this approach can play a role in the development of complex systems with many users. After all, corealisation's aim of abolishing the boundaries between 'designer' and 'user' becomes hard to achieve when a large number of people with different concerns and commitments are involved. Scaling corealisation to this level by increasing the number of IT professional engaging in corealisation throughout the organisation would not be feasible because of the prohibitive costs involved.

There are a number of answers to this issue but first I want to discuss the question of resource usage. As corealisation orients to the whole lifecycle of a system and combines a number of activities into one, its costs should therefore not be compared to the core areas of systems design alone but to the overall costs of acquiring, operating and maintaining a system. This would include costs for initial development, training, operational support, evaluation as well as maintenance and development. In addition, costs need to be seen across the board, not just in terms of

costs associated with the system but also costs in user departments as well as the benefits a system provides. This may be difficult to do, especially where the benefits accrue elsewhere. However, in order to assess the effectiveness of corealisation (in any particular instance), one needs to look at the total cost of ownership *and* the total benefits of a system. With conventional design, many of the costs arise in 'use' and are therefore difficult to account for under existing regimes of cost-benefit analysis. For example, the workarounds to remedy the problems of many systems are part of organisational life and therefore conventionally hidden away from view. At the same time, the costs involved in doing corealisation in terms of the overhead of the designers familiarising themselves with a setting are mostly incurred only once, when they enounter settings they are not familiar with. There is no extraordinary effort involved in maintaining familiarity through membership but an initial effort may need to be expended to observe and learn about the setting in pretty much the same way an ethnographer would. The aim is to make this initial effort pay off over a longer period of time where maintaining familiarity with the setting and its biolgraphy becomes something that is part and parcel of the work of doing design.

Corealisation is of most benefit in relation to systems that are handled less effectively by traditional MIS strategies. Such systems are not necessary 'small' by any measure as they can be complex and they can have many users. The difference rather lies elsewhere, in the nature of the system and its relationship to working practice. Corealisation is not a means to introduce systems that are aimed to transform organisational working practices in pre-specified ways. Rather, it is an attempt to capture the specific, detailed working practices that are of importance in the use of systems of any scale and in this respect it can be a valuable adjunct to other design methods.

Finally, it is important to note that large-scale, vertical systems are often only made to work through the addition of a whole host of local configurations of off-the-shelf software, paper artefacts and other 'bits and pieces' that support local practices not catered for by them (Grønbæk, Kyng and Mogensen 1993). It is precisely at the boundary of these two types of systems that a lot of experience is brought to bear and that innovation happens. Corealisers are in a unique position to take up these innovations and feed them into larger-scale undertakings which may run along more

traditional lines of software engineering practice. If the vertical systems provide appropriate interfaces, corealisation can even provide integration between them and the locally managed configurations. (This idea was originally envisaged for the shift book application but not realised because of the lack of access to an interface to the assembly control host.)

One aspect that needs to be carefully considered in arranging working practices is the benefits of co-location with other IT staff. It would be ironic if the benefits of co-locating corealising partners were bought at the cost of IT professionals losing their contact with fellow IT staff, introducing problems of coordinating an overall development effort (cf. Herbsleb *et al.* 2000, Grinter 2003). A reasonable balance will need to be found within larger-scale organisations that allows IT professionals to have 'a foot in both camps'. Modern equipment such as laptops and mobile phones as well as CSCW and software engineering tools can play a role in this but mundane practical arrangements are also of importance, e.g. agreeing fixed times for certain activities and making sure these are protected from other demands and providing an allocated space close to the work setting designated to IT staff collaboration. Obviously, the arrangements will need to be tailored to the nature of the setting and the resources available.

A corealisation orientation to systems development brings with it not just 'local' knowledge but knowledge also about how local concerns are tied in with larger developments in the organisation and how they relate to other places and people. In the discussion of the daily reports system we saw how Barbara was able to draw on her knowledge of the wider divisions of labour, specifically of the role that maintenance personnel play, to inform design decisions. A corealisation perspective is not necessarily a 'local' one but one that is located, i.e. is a view from somewhere and while I would not claim any kind of objectivity associated with this perspective, it seems much better to me than the alternative of a view from nowhere (cf. Suchman 2002). It is where a particular *system* is build for a particular purpose and deployed within a context of organisational arrangements, constraints and contingencies that the value of information technologies is ultimately realised. Suchman, Trigg and Blomberg (2002, p. 164) suggest that:

> Making technologies is, in consequence, *a practice of configuring new alignments* between the social and the material that are both localized and able to travel, stable and reconfigurable, intelligibly familiar, and recognizably new. (emphasis in original).

Corealisers cannot be expected to 'be everywhere' but they can be expected to be familiar with the ways in which their efforts relate to larger organisational arrangements and developments. All this is not to say that corealisation is an answer to all questions and I happily concede that there are any number of situations where it is not a viable approach – however, that does not mean the idea is pointless, I believe this study has show the opposite.

### 7.2.6 Dependability

With respect to the issue of dependability, corealisation offers an extension of what can be understood by dependability. Rather than conceptualising it as an attribute of a system, it recognises that dependability in practice is an ongoing achievement. As Clarke *et al.* (2006) have put it:

> […] *dynamically responding in the best way* to problems as they arise and achieving dependable production […] involves far more than simply reconfiguring the system but attending to the complexities of collaborative working. (*ibid*. p. 141, emphasis in original)

As mentioned in section 2.2.3, complex organisational systems are inherently difficult to evaluate. They usually suffer from a whole range problems which can be through of as normal natural troubles. While these are not catastrophic and while there are usually a range of established practices for dealing with them, they are nevertheless important.

As said above, corealisation treats dependability as an ongoing concern and through its longitudinal engagement provides the resource making systems practically dependable. In real-world situations, people will to a greater or lesser extent be involved in making systems stable enough for the current purposes, that is, they are 'satisficing'. Yourdon has introduced the concept of "good enough software" (Yourdon 1995):

> In the best of all possible worlds, our users would like us to develop software instantly, at no cost, and with no defects. But that's not possible in today's world. In more and more application domains, we've been forced to accept that

> the reengineering slogan "faster, cheaper, better" really means "fast enough, cheap enough, good enough". (*ibid*., p. 79).

In a corealisation context, "good enough" takes on a different meaning. Rather than being a general capitulation in the face of high customer demands, it becomes the informed judgement of someone who is intimately familiar with the context of use and can foresee the potential implications of bugs remaining in the software. Work on the system can be prioritised accordingly so that practical dependability can be maximised given the time and resources available as well as other circumstances. An important aspect in this regard is the question of technology supply, that is, how to make the decision between buying or building parts of the system. Recent changes in the software market such as the availability of components and technologies for combining them in flexible ways are of importance here as they allow a 'pick and mix' approach to systems building (Brady, Tierney and Williams 1992).

Meyer (2003) introduces the idea of 'trusted components' as highly quality assured building blocks that would "make application development less dedicated to producing software from the ground up, and hence less dependent on the individual skills of project developers: the focus would shift towards composition, combination, mix-and-match" (*ibid.* p. 661). Meyer himself points to the fact that reuse also has its problems as it may permeate bugs or fail because of incompatibilities between components. His aim is to encourage the development of a market of 'trusted components' which are either proven to be correct against their specification or certified, i.e. his concern is with the supply side. While it may currently be difficult to assess the quality of components on offer, there is certainly a sizeable (and growing) market for components and tools – one need only look at vendor's websites or visit open source repositories such as sourceforge.net.

In the context of corealisation, the important question relates to the use of components versus developing equivalent functionality from scratch: "can I do better given the resources I have?" While there are many other aspects that will influence the choice between use of an existing component and development from scratch, it should be clear that in most real-world non-critical scenarios the economics of dependability will favour re-use. Despite the numerous deficiencies of both commercial and open source software, achieving the same functionality *and* the same

quality given the constraints of the undertaking is normally impossible. By basing a design on a stable foundation of known-to-work building blocks and expending the available effort on the corealisation of the overall system, corealisers will maximise the achievable quality of the system.

As said at the outset, corealisation is not a replacement but rather a complement for existing good practice in systems development. There are even some elements of such practice which are particularly interesting in this respect. For example, aspect-oriented programming (Kiczales *et al.* 1997) supports a clean separation of business logic from other concerns such as security concerns or transaction management. It has the potential to increase the dependability of systems while at the same time making the project of corealisation more manageable and cost effective.

# 8 Conclusions

The aim of this thesis was to respecify design as an ongoing, collaborative activity undertaken in context and by all members of the setting. Through the presentation of fieldwork material showing both the established working practices at EngineCo and the work that I did *qua* corealiser I have demonstrated the practical feasibility of this project. The fieldwork shows how people can make use of a member's understanding of a setting and its features to *directly* inform design decisions made in the context of ongoing work of systems development and use.

To design an IT system means to engage with members, their understandings, interests, collaborations and possible moves. One can of course ignore this and implement a system without engaging, one may even succeed in implementing a system in this way but the risks and costs associated are immense. Repeated failures of 'design from nowhere' (Suchman 2002) have demonstrated this. All too often, perfect solutions are designed for the wrong set of work problems (Ackoff 1974, Crabtree 1998). Both participatory design and ethnography for design (as developed within the CSCW community) are ways to try and address this problem and both are successful to an extent but, as I have argued, they are 'patches' to a more fundamental underlying problem: the separation of design from use. They aim to repair the fact that IT professionals do not have an ordinary member's understanding of the setting, its practices and working relations. Not only is achieving this aim problematic but it can also easily be seen as a costly and optional exercise under the regime of traditional *a-priori* design. Corealisation "sidesteps the problem with its insistence that the corealiser becomes a member of the work setting, with a member's expertise and thus able to reason about user requirements and how these might be met technologically" (Williams, personal communication).

It is precisely this understanding that allows IT professionals as members to attend to the collaborative dimensions of the setting and to collaboratively work up socio-material configurations that support these activities. Many technologies coming out of CSCW are orthogonal to the specific working practices found in places such as the plant described in this thesis and therefore support collaboration only in a fairly generic sense. In contrast, corealisation aims to support collaboration with greater

specificity, and to embed the system in the biography of the setting and therefore make it accountable. It is here, where the pragmatics of collaboration in design are foregrounded that we find that the concerns of CSCW and participatory design are consonant and that they can be synthesised in a fundamental way, leading to a radical respecification of what the work of design is about (cf. Hartswood *et al.* 2002).

My aim is to attend to the ways that design opportunities present themselves as situated, contexted matters. The IT professional who is not able to take advantage of direct immersion in the setting they design for seems to me to be at a disadvantage. IT system design is, or arguably should be, an activity that orients to pragmatic, practice-based interactions as the foundations of the social ordering of the design process. What might appear rational to a designer external to the setting may turn out to be problematic within the setting. Such matters are contextually defined, and what may appear to be settled may be revised or oriented to in different ways given the circumstances. In sum, the question is how designers could be aware of such situated matters if they are not part of the 'fabric' of the setting?

As stated earlier, the IT system development work reported here is somewhat different in character from the usual project orientation and focus of IT work – at this point it is useful to tease out some of the similarities and differences. In some ways, people are engaging in a 'project' – their objective is to deliver a technological configuration that meets a particular need, and in doing this they have a timescale to work to, they have to enlist and marshal resources, secure commitments and elaborate requirements. What is largely absent is the project's more formal organisational framework and management: project plans, workshops, structured meetings, prioritisation processes and so on. What is notable is the *ad hoc* character of many of the activities. One might suggest that this is in part because the work is on a 'manageable' scale, making it easy for people to manage the work without the assistance of formal tools. On the other hand, one might also argue that people's biographical familiarity with the plant enables a rapid engagement with the salient features of the work and because of this the problems she has to solve and the people she needs to deal with become readily apparent. One can also look to the nature of the work they are doing – it is not radically transformative, but builds incrementally on the existing practices and technologies.

We have seen how the role of membership is central to the design activity that takes place in our example. Orienting to what people know and use, i.e., how they get their work done, and knowing and using it oneself to get the design job done seems to be a key resource for the practical activity of doing design. The work of design is about making sense of collaboration in order to design artefacts to support it. At the same time it is collaborative work itself. For the IT professional who enters the "extended set of working relations" (Suchman 2002) and thus grounds the work of design in the biography of the setting, her everyday experiences become a resource for design much richer than any requirements document or ethnographic account of work.

Engagement has been a leitmotif in CSCW and participatory design (see, for example, Bødker *et al.* (1987), especially p. 263) and I would argue that membership is perhaps the most fundamental mode of engaging with those amongst and together with whom one does design. As noted in Hartswood *et al.* (2002), corealisation calls us to move away from doing the ethnography and then doing the design to a longer term engagement with the contingencies of the work site. Placing IT professionals at the heart of the workplace, enabling them to become conversant with 'where the action is' (Dourish 2001), so to speak, and who to talk to – in other words to make them competent members – is central to the approach to design advanced above. To be sure, 'ethnography for design' has its place, and I do not wish to deny this, but it seems that if one is concerned with what it means to be engaged in the pragmatics of collaboration in design, one has to make such a commitment to engage over time.

Corealisation is an attempt to take these matters seriously and to provide a principled synthesis of the concerns of ethnomethodology and design as practiced in the areas of computer supported cooperative work and participatory design. This thesis provides a contribution to the formulation of the corealisation approach by demonstrating how its principles can be instantiated by corealising systems in a complex organisational setting over a significant period of time.

Further studies will be needed to continue to explore the approach and its practical application in different settings. Additionally, there are avenues that have as yet to be pursued, for example the use of end-user programming environments which may give workers an even stronger role within corealisation. Other aspects to be explored

include the role that component technologies may play in leveraging what can be achieved. I hope that this thesis will provide a stepping stone towards a programme of research to further develop corealisation and to further its adoption by practitioners.

# 9 Bibliography


Abran, A., Moore, J. W., Bourque, P., Bupuis, R. (eds., 2004) *SWEBOK®: Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society. Available at: www.swebok.org (accessed 23.12.2005).

Ackoff, R. (1974) *Redesigning the future: a systems approach to societal problems*. Wiley.

Anderson, D.C. and Sharrock, W.W. (1983). Irony as a Methodological Theory: A Sketch of Four Sociological Variations. *Poetics Today* 4(3), pp. 565-579.

Anderson, R. and Sharrock, W. (1993). Can Organisations Afford Knowledge? *Computer Supported Cooperative Work*, vol. 1, pp. 143-161.

Anderson, R.J. (1994). Representations and Requirements: The Value of Ethnography in System Design. *Human-Computer Interaction*, vol. 9, pp. 151-182.

Anderson, R.J. (1997). Work, Ethnography, and System Design. Kent, A. and Williams, J.G. (eds.) *The Encyclopedia of Microcomputing*. New York: Marcel Dekker. pp. 159-183.

Anderson, R.J., Hughes, J.A. and Sharrock, W.W. (1989). *Working for Profit: The Social Organisation of Calculation in an Entrepreneurial Firm*. Aldershot: Avebury.

Anderson, B. [R.J.] (2000). Where the rubber hits the road: notes on the deployment problem in workplace studies. Luff, P., Hindmarsh, J. and Heath, C. (eds.) *Workplace Studies: Recovering Work Practice and Informing System Design*. Cambridge University Press.

Attewell, P. (1987). Big Brother and the Sweatshop: Computer Surveillance in the Automated Office. *Sociological Theory* 5(1). pp. 87-100.

Auramäki, E., Robinson, M., Aaltonen, A., Kovalainen, M., Liinamaa, A. and Tuuna-Väiskä, T. (1996). Paperwork at 78kph. *Proceedings of the ACM conference on computer supported cooperative work*. pp. 370-379.

Axtell, C.M., Waterson, P.E. and Clegg, C.W. (1997). Problems integrating user participation into software development. *International Journal of Human-Computer Studies*, vol. 47. pp. 323-345.

Bannon, L. and Schmidt, K. (1991). CSCW: Four Characters in Search of a Context. Bowers, J.M. and Benford, S.D. (eds.) *Studies in Computer Supported Cooperative Work: Theory, Practice and Design*. Amsterdam: North-Holland. pp. 3-16.

Bannon, L.(1991). From Human Factors to Human Actors: The Role of Psychology and Human-Computer Interaction Studies in Systems Design. Greenbaum, J. & Kyng, M. (eds.) *Design at work.: Cooperative Design of Computer Systems*. Hillsdale: Lawrence Erlbaum Associates, pp. 25-44.

Bansler, J.P. and Havn, E. (1991). The Nature of Software Work: Systems Development as a Labor Process. Van den Besselaar, P., Clement, A. and Järvinen, P. (eds.) *Information System, Work and Organization Design*. Elsevier Science Publishers B. V. (North-Holland). pp. 145-153.

Bansler, J. and Kraft, P. (1994). Privilege and Invisibility in the New Work Order: a Reply to Kyng. *Scandinavian Journal of Information Systems*, 6(1). pp. 97-106.

Barrett, R., Kandogan, E., Maglio, P.P., Haber, E.M., Takayama, L.A. and Prabaker, M. (2004). Field Studies of Computer System Administrators: Analysis of System Management Tools and Practices. *Proceedings of the ACM conference on Computer supported cooperative work*. pp. 388-395.

Bauer, F.L. (1973). Software and Software Engineering. *SIAM Review* 15(2). pp. 469-480.

Beck, K. (1998). Extreme Programming: A Humanistic Discipline of Software Development. Astesiano, E. (ed.) *Proceedings of the 1st International Conference on Fundamental Approaches to Software Engineering*, Lecture Notes in Computer Science 1382, Springer. pp. 1-6.

Beck, K. (2000) *Extreme Programming Explained: Embracing Change*. Addison Wesley.

Bergmann, J. R. (2005). Studies of Work. Rauner, F. (ed.) *Handbuch der Berufsbildungsforschung*. Bielefeld: W. Bertelsmann Verlag.

Bertelsen, O.W. and Nielsen, C. (1999). Dynamics in Wastewater Treatment: A Framework for Understanding Formal Constructs in Complex Technical Settings. Bødker, S., Kyng, M. and Schmidt, K. (eds.) *Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work*. pp. 277-290.

Bertelsen, O.W. and Bødker, S. (2001). Cooperation in massively distributed information spaces. *Proceedings of the Seventh European Conference on Computer Supported Cooperative Work*, Bonn, Germany, 16-20 September 2001. Dordrecht: Kluwer Academic Publishers.

Beyer, H. and Holtzblatt, K. (1995). Apprenticing With the Customer. *Communications of the ACM*, 38(5). pp. 45-52.

Beyer, H. and Holtzblatt, K. (1997). *Contextual Design: Defining customer-centered systems*. Morgan Kaufmann.

Beyer, H. and Holtzblatt, K. (1999). Contextual Design. *interactions*, January and February 1999. pp. 32-42.

Beynon-Davies, P. (1995), Information systems failure and risk assessment: the case of the London Ambulance Service Computer Aided Despatch System. *European Conference on Information Systems*, vol. 4. pp. 171-184.

Beynon-Davies, P. (1999) Human Error and Information Systems Failure: The Case of the London Ambulance Service Computer-Aided Despatch System Project. *Interacting with Computers*, 11(6). pp. 699-720

Bittner, E. (1974). The Concept of Organisation. In Turner, R. (ed.) *Ethnomethodology: Selected Readings*. Harmondsworth: Penguin. Pp. 69-82.

Bjerknes, G., Ehn, P., Kyng, M. (eds., 1987). *Computers and Democracy: A Scandinavian Challenge*. Avebury.

Bjerknes, G. and Bratteteig, T. (1987) Florence in Wonderland: System development with nurses. In: Bjerknes, G., Ehn, P., Kyng, M. (eds.) *Computers and Democracy: A Scandinavian Challenge*. Avebury.

Bjerknes, G. and Kautz, K. (1991). Overview – A Key Concept in Computer Supported Cooperative Work. *Computergestützte Gruppenarbeit (CSCW): 1. Fachtagung, 20. September bis 2.Oktober, Bremen*. B.G. Teubner. pp. 153-169.

Blomberg, J. and Henderson, A. (1990). Reflections on Participatory Design: Lessons from the Trillium Experience. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering people*. pp. 353-359.

Blomberg *et al.* (1993) Ethnographic Field Methods and Their Relation to Design. In Schuler and Namioka (eds.) *Participatory Design: Principles and Practices*. Lawrence Erlbaum Associates. pp. 123-155.

Blomberg, J., Suchman, L. and Trigg, R. H. (1996). Reflections on a Work-Oriented Design Project. *Human Computer Interaction* 11(3), pp. 237-265.

Blythin, S., Hughes, J.A., Kristoffersen, S. and Rouncefield, M. (1997). Recognising 'success' and 'failure': evaluating groupware in a commercial context. *Proceedings of the international ACM SIGGROUP conference on Supporting group work: the integration challenge*. pp. 39-46.

Bødker, S., Ehn, P., Kyng, M., Kammersgaard, J. and Sundblad, Y. (1987). A Utopian Experience: On design of powerful computer-based tools for skilled graphic workers. In: Bjerknes, G., Ehn, P., Kyng, M. (eds.) *Computers and Democracy: A Scandinavian Challenge*. Avebury.

Bødker, S. and Grønbæk, K. (1991). Design in Action: From Prototyping by Demonstration to Cooperative Prototyping. In Greenbaum, J. and Kyng, M. (eds.) *Design at Work: Cooperative Design of Computer Systems*. Lawrence Erlbaum Associates. pp. 197-218.

Bødker, S. and Grønbæk, K. (1991a). Cooperative Prototyping Studies – users and designers envision a dental case record system. Bowers, J.M. and

Benford, S.D. (eds.) *Studies in Computer Supported Cooperative Work*. Elsevier Science Publishers B.V. (North-Holland). pp. 315-332.

Bødker, S. (1994). Creating conditions for participation: Conflicts and resources in systems design. Trigg, R., Anderson, S.I. and Dykstra-Erickson, E.A. (eds.) *Proceedings of the Participatory Design Conference*. pp. 13-20.

Bødker, K., Kensing, F. and Simonsen, J. (2004). *Participatory IT Design: Designing for Business and Workplace Realities*. MIT Press.

Bødker, S. (2000). Coordinating Technical Support Platforms. *Communications of the ACM*, 43(11), electronic supplement, article no. 5.

Boehm, B.W. (1988). A Spiral Model of Software Development and Enhancement. *IEEE Computer*, May 1988, pp. 61-72.

Boehm, B. and Turner, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, September/October 2005. pp. 30-39.

Bowers, J. (1991) The Janus Faces of Design. In Bowers, J. and Benford, S. (eds.) *Studies in Computer Supported Work*. Elsevier Science, pp. 333-349.

Bowers, J. and Pycock, J. (1994). Talking Through Design: Requirements and Resistance in Cooperative Prototyping. *Human Factors in Computing Systems (CHI'94)*, pp. 299-305.

Bowers, J. (1994). The Work to Make a Network Work: Studying CSCW in Action. *Proceedings of the ACM conference on Computer supported cooperative work*. pp. 287-298.

Bowers, J., Pycock, J., Rodden, T. and Dean, G. (1994). Running the Network: Supporting Co-operative Systems. *Information Technology & People* 7(2), pp. 7-28.

Bowers, J., Button, G. and Sharrock, W. (1995). Workflow from Within and Without, Marmolin, H., Sundblad, Y. and Schmidt, K. (eds.). *Proceedings*

*of the Fourth European Conference on Computer-Supported Cooperative Work*, Stockholm, Sweden, 10.-14. September 1995. pp. 51-66.

Brady, T., Tierney, M. and Williams, R. (1992). The Commodification of Industry Applications Software. *Industrial and Corporate Change* 1(3), pp. 489-514.

Braverman, H. (1974/1998). *Labor and Monopoly Capital: The Degradation of Work in the Twentieth Century. 25th Anniversary Edition*. New York: Monthly Review Press.

Brooks, F. P. Jr. (1975) *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley.

Brown, P.S., and Gould, J.D. (1987). An Experimental Study of People Creating Spreadsheets. *ACM Transactions on Office Information Systems* 5(3), pp. 258-272.

Brown, A. and Patterson, D. A. (2001). Embracing Failure: A Case for Recovery-Oriented Computing (ROC). *2001 High Performance Transaction Processing Symposium*, Asilomar, CA, October 2001. available at: http://roc.cs.berkeley.edu/papers/hpts01-draft.pdf (accessed 21.01.2006)

Burns, S. (1986). *An Ethnomethodological Case Study of Law Pedagogy in Civil Procedure*. Unpublished monograph, University of California, Los Angeles.

Büscher, M., Christensen, M., Grønbæk, K., Krogh, P., Mogensen, P., Shapiro, D. and Ørbæk, P. (2000). Collaborative Augmented Reality Environments: Integrating VR, Working Materials, and Distributed Work Spaces. *Proceedings of the 3rd International Conference on Collaborative Virtual Environment.* pp. 47-56.

Büscher, M., Gill, S., Mogensen, P. and Shapiro, D. (2001) Landscapes of Practice: Bricolage as a Method for Situated Design. *Computer Supported Cooperative Work* 10(1), pp. 1-28.

Büscher, M., Shapiro, D., Hartswood, M., Procter, R., Slack, R., Voß, A. and Mogensen, P. (2002). Promises, Premises and Risks: Sharing Responsibilities, Working Up Trust and Sustaining Commitment in Participatory Design Projects. Binder, T., Gregory, J., Wagner, I. (eds.) *Proceedings of the Participatory Design Conference*. pp. 183-192.

Büscher, M., Eriksen, M.A., Kristensen, J.F. and Mogensen, P.H. (2004). Ways of Grounding Imagination. *Proceedings of the Participatory Design Conference*, Toronto, Canada. pp. 193-203.

Button G. (ed., 1991). *Ethnomethodology and the human sciences*. Cambridge University Press.

Button, G.(ed., 1993). *Technology in Working Order: Studies of work, interaction, and technology*. London, Routledge.

Button, G. (1993a). An Organisational Account of the Question "Do Users Get What They Want?" *ACM SIGOIS Bulletin* 14(2), pp. 35-40.

Button, G. and Sharrock, W. (1994). Occasioned practices in the work of software engineers. Jirotka, M. and Goguen, J.A. (eds.) *Requirements Engineering: Social and Technical Issues*. Academic Press. pp. 217-240.

Button, G. and Sharrock, W. (1995). Practices in the work of ordering software development. In Firth (ed.) *The Discourse of Negotiation: Studies of Language in the Workplace*. Pergamon. pp. 159-180

Button, G. and Sharrock, W. (1995a). The Mundane Work of Writing and Reading Computer Programs. ten Have, P. and Psathas, G. (eds.) *Situated Order: Studies in the Social Organization of Talk and Embodied Activities*. pp. 231-258.

Button, G. and Sharrock, W. (1996). Project Work: The Organisation of Collaborative Design and Development in Software Engineering. *Computer Supported Cooperative Work*, Vol. 5, pp. 369-386.

Button, G. and Dourish, P. (1996). Technomethodology: Paradoxes and Possibilities. *Proceedings of the ACM Conference on Human Factors in Computing Systems*. pp. 19-26.

Button, G. and Sharrock, W. (1998). The Organizational Accountability of Technological Work. *Social Studies of Science* 28(1), pp. 73-102.

Button, G. and Sharrock, W. (2000). Design by problem-solving. Luff, P., Hindmarsh, J., and Heath, C. (eds.) *Workplace Studies: Recovering Work Practice and Informing System Design*. Cambridge University Press. pp. 46-67.

Button, G. (2006). Introduction: A New Perspective On The Dependability Of Software Systems. In: Clarke, K., Hardstone, G., Rouncefield, M. and Sommerville, I. (eds.) *Trust in Technology: A Socio-Technical Perspective*. Springer. pp. ix-xxv.

Card, S., Moran, T.P., Newell, A. (1983). *The psychology of human-computer interaction*. Lawrence Erlbaum Associates.

Carroll, J.M. and Campbell, R. L. (1986). Softening Up Hard Science: Reply to Newell and Card. *Human-Computer Interaction*, vol. 2. pp. 227-249.

Center for National Software Studies (2005). *Software 2015: A National Software Strategy to Ensure U.S. Security and Competitiveness. Report of the 2nd National Software Summit*, Washington D.C., May 2004. Available at: www.cnsoftware.org/NSS2Report (accessed 23.12.2005).

Charette, R.N. (2005). Why Software Fails. *IEEE Spektrum*. Sept. 2005. pp. 42-49.

Chin, G. Jr., Rosson, M.B. and Carroll, J.M. (1997). Participatory Analysis: Shared Development of Requirements from Scenarios. *Proceedings of the Conference on Human Factors in Computing Systems*. pp. 162-169.

Clarke, K., Hartswood, M., Procter, R., Rouncefield, M. and Slack, R. (2002). Minus nine beds: Some Practical Problems of Integrating and Interpreting Information Technology in a Hospital Trust. *Proceedings of the HC 2002 Conference: Current Perspectives in Healthcare Computing*, Harrogate, UK. pp. 205-211.

Clarke, K., Hughes, J., Martin, D., Rouncefield, M., Sommerville, I., Gurr, C., Hartswood, M., Procter, P., Slack, R. and Voss, A. (2003). Dependable Red

Hot Action. Kuutti, K., Karsten, E.H., Fitzpatrick, G., Dourish, P. and Schmidt, K. (eds.) *Proceedings of the Eighth European Conference on Computer Supported Cooperative Work*. pp. 61-80.

Clarke, K., Hartswood, M., Procter, R., Rouncefield, M. and Slack, R. (2003a). Trusting the Record. *Methods of Information in Medicine*, vol. 42, pp. 345-352.

Clarke, K. *et al.* (2006). Explicating Failure. In: Clarke, K., Hardstone, G., Rouncefield, M. and Sommerville, I. (eds.) *Trust in Technology: A Socio-Technical Perspective*. Springer. pp. 123-145.

Cleland, G. and MacKenzie, D. (1995). Inhibiting Factors, Market Structure and the Industrial Uptake of Formal Methods. *Proceedings of the Workshop on Industrial-Strength Formal Specification Techniques*. pp. 46-60.

Clement, A. (1991). Designing Without Designers: More Hidden Skill in Office Computerization. Eriksson, I.V., Kitchenham, B.A. and Tijdens, K.G. (eds.) *Women, Work and Computerization*. Elsevier Science Publishers B.V. (North-Holland).

Clement, A. and van den Besselaar, P. (1993). A Retrospective Look at PD Projects. *Communications of the ACM*, 36(4). pp. 29-37.

Crabtree, A. (1998). Ethnography in Participatory Design. In Chatfield, R., Kuhn, S., Muller, M. (eds.) *Proceedings of the Participatory Design Conference*. Seattle, WA. pp. 93-105.

Crabtree, A. (2001). The Practical Availability of Work-Practice. *Ethnographic Studies*, Issue 6, p. 7-16.

Crabtree, A. (2003). *Designing Collaborative Systems: A Practical Guide to Ethnography*. Springer.

Crabtree, A. (2004) Taking technomethodology seriously: hybrid change in the ethnomethodology-design relationship, *European Journal of Information Systems*, vol. 13, pp. 195-209.

Crabtree, A. (2004a). Design in the Absence of Practice: Breaching Experiments. *Proceedings of the Conference on Designing Interactive Systems (DIS)*. Cambridge, MA, USA. pp. 59-68

Cross, M. (2005). IT gurus attempt to win doctors' hearts and minds. *British Medical Journal*, vol. 330, February 2005, p. 276.

Curtis, B., Krasner, H. and Iscoe, N. (1988). A Field Study of the Software Design Process for Large Systems. *Communications of the ACM* 31(11), pp. 1268-1287.

Dijkstra, E. W. (1972) The humble programmer. *Communications of the ACM* 15(10): 859-866.

Dourish, P. and Button, G. (1998). On "Technomethodology": Foundational Relationships between Ethnomethodology and System Design. *Human-Computer Interaction*, 13(4), 395-432.

Dourish, P. (2001). *Where the Action Is: The Foundations of Embodied Interaction*. MIT Press.

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

Ehn, P. (1988a) Playing the Language-Games of Design and Use: on Skill and Participation. *ACM SIGOIS Bulletin* 9(2-3), pp. 142-157.

Ehn, P. and Kyng, M. (1991). Cardboard Computers: Mocking-it-up or Hands-on the Future. Greenbaum, J. and Kyng, M. (eds.) *Design at Work: Cooperative Design of Computer Systems*. Lawrence Erlbaum Publishers. pp. 169-195.

Ehn, P. (1993). Scandinavian Design: On Participation and Skill. Schuler, D. and Namioka, A. (eds.) *Participatory Design: Principles and Practices.* Lawrence Erlbaum Associates. pp. 41-77.

Erickson, T. (2000). Supporting interdisciplinary design: towards pattern languages for workplaces. Luff, P., Hindmarsh, J, and Heath, C. (eds.)

*Workplace Studies: Recovering Work Practice and Informing System Design*. Cambridge University Press. pp. 252-261.

Finkelstein, A. and Dowell J. (1996). A Comedy of Errors: the London Ambulance Service case study. *Proc. 8th International Workshop on Software Specification & Design IWSSD-8*, (IEEE CS Press), 1996, pp. 2-4.

Fitzpatrick, G.A. (1998). *The Locales Framework: Understanding and Designing for Cooperative Work*. PhD Dissertation, Department of Computer Science and Electrical Engineering, The University of Queensland.

Fleck, J. (1988) *Innofusion or diffusation: the nature of technological development in robotics*. Edinburgh PICT Working Paper No. 4, Edinburgh University, Edinburgh.

Fleck, J. (1993) Innofusion: Feedback in the innovation process. In Stowell, F. A. *et al.* (eds.). *Systems Science*, Plenum Press, pp.169-174.

Fleck, J. (1999). Learning by trying: the implementation of configurational technology. MacKenzie, D. and Wajcman, J. (eds.) *The Social Shaping of Technology*. 2nd edition. pp. 244-265.

Floyd, C. (1987). Outline of a Paradigm Change in Software Engineering. Bjerknes, G., Ehn, P., Kyng, M. (eds). *Computers and Democracy: A Scandinavian Challenge*. Aldershot: Avebury. pp. 191-210.

Floyd, C., Mehl, W-M., Reisin, F-M., Schmidt, G. and Wolf, G. (1989). Out of Scandinavia: Alternative Approaches to Software Design and System Development. *Human-Computer Interaction*, vol. 4, pp. 253-350.

Francis, D. and Hester, S. (2004). *An Invitation to Ethnomethodology: Language, Society and Interaction*. Sage Publications.

Friedman, A. and Cornford, D. (1987) Strategies for Meeting User Demands: An international perspective. In: Bjerknes, G., Ehn, P., Kyng, M. (eds.) *Computers and Democracy: A Scandinavian Challenge*. Avebury. pp. 137-162.

Friedman, A. and Cornford, D. (1989) *Computer Systems Development: History and Organisation*. Wiley, Chichester, 1989.

Gallie, D. (1996). New Technology and the Class Structure: The Blue-Collar/White-Collar Divide Revisited. *British Journal of Sociology* 47(3). pp. 447-473.

Gantt, M. and Nardi, B.A. (1992). Gardeners and Gurus: Patterns of Cooperation Among CAD Users. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. pp. 107-117.

Garfinkel, H. (1967). *Studies in Ethnomethodology*. Prentice-Hall.

Garfinkel, H. (ed., 1986). *Ethnomethodological Studies of Work*. Routledge & Kegan Paul.

Garfinkel, H. and Sacks, H. (1986). On Formal Structures of Practical Action. Garfinkel, H. (ed.) *Ethnomethodological Studies of Work*. Routledge & Kegan Paul. pp. 160-193.

Garfinkel, H. (1988). Evidence for Locally Produced, Naturally Accountable Phenomena of Order, Logic, Reason, Meaning, Method, etc. In and as of the Essential Quiddity of Immortal Ordinary Society, (I of IV): An Announcement of Studies. *Sociological Theory* 6(1), pp. 103-109.

Garfinkel, H. (1991). Respecification: evidence for locally produced, naturally accountable phenomena of order*, logic, reason, meaning, method, etc. in and as of the essential haecceity of immortal ordinary society, (I) – an announcement of studies. Button, G. (ed.) *Ethnomethodology and the Human Sciences*. Cambridge University Press. pp. 10-19.

Garfinkel, H. and Wieder, D.L. (1992). Two Incommensurable, Asymmetrically Alternate Technologies of Social Analysis. Chapter 10 in Watson, G. and Seiler, R.M. (eds.) *Text in Context: Contributions to Ethnomethodology*. Sage Publications.

Garfinkel. H. (1996). Ethnomethodology's Program. *Social Psychology Quarterly* 59(1), pp. 5-21.

Geertz, C. (1973). Thick Description: Toward an Interpretive Theory of Culture. In: *The Interpretation of Cultures: Selected Essays*. Basic Books.

Gibbs, W. W. (1994). Software's Chronic Crisis. *Scientific American*, September 1994. pp. 86-95.

Green, E., Owen, J. and Pain, D. (1993). *Gendered by Design? Information Technology and Office Systems*. Taylor & Francis.

Greenbaum, J. (1988). In Search of Cooperation: An Historical Analysis of Work Organization and Management Strategies. *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*. pp. 102-114.

Greenbaum, J. (1991). Toward Participatory Design: The Head and the Heart Revisited. Eriksson, I. V., Kitchenham, B. A. and Tijdens, K.G. (eds.), *Women, Work and Computerization*. Elsevier Science Publishers B. V. (North-Holland).

Greenbaum, J. and Kyng, M. (eds., 1991). *Design at Work: Cooperative Design of Computer Systems*. Lawrence Erlbaum Associates.

Greenbaum, J. and Kyng, M. (1991a) Introduction: Situated Design. In Greenbaum, J. and Kyng, M. (eds.) *Design at Work: Cooperative Design of Computer Systems*. Lawrence Erlbaum Associates.

Greenbaum, J. (1995). *Windows on the Workplace: Computers, Jobs, and the Organization of Office Work in the Late Twentieth Century*. Monthly Review Press.

Grinter, R.E. (2003). Recomposition: Coordinating a Web of Software Dependencies. *Computer Supported Cooperative Work*, vol. 12, pp. 297-327.

Grønbæk, K. (1989). Rapid Prototyping with Fourth Generation Systems - an Empirical Study. *Office: Technology and People* 5(2). pp. 105-125.

Grønbæk, K , Kyng, M. and Mogensen, P. (1993). CSCW Challenges: Cooperative Design in Engineering Projects. *Communications of the ACM* 36(4). pp. 67-77.

Grudin, J. (1988). Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces. *Proceedings of the ACM conference on Computer-supported cooperative work*. pp. 85-93.

Grudin, J. (1990). The Computer Reaches Out: the Historical Continuity of Interface Design. *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*. pp. 261-268.

Grudin, J. (1994). Groupware and Social Dynamics: Eight Challenges for Developers. *Communications of the ACM* 37(1), pp. 93-105.

Hardstone, G., Hartswood, M., Procter, R., Slack, R., Voss, A., Rees, G. (2004). Supporting Informality: Team Working and Integrated Care Records. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. pp. 142-151.

Harper, R.H.R. and Hughes, J.A. (1993). 'What a f-ing system! Send 'em all to the same place and then expect us to stop 'em hitting'. Button, G. (ed.) *Technology in Working Order: Studies of work, interaction, and technology*. London, Routledge. pp. 127-144.

Harper, R.H.R. (1995). Radicalism, Beliefs and Hidden Agendas. *Computer Supported Cooperative Work*, vol. 3, pp. 43-46.

Harper, R.H.R. (2000). The Organisation in Ethnography: A Discussion of Ethnographic Fieldwork Programs in CSCW. *Computer Supported Cooperative Work*, vol. 9. pp. 239-264.

Hartswood, M., Procter, R., Rouncefield, M., Sharpe, M. (2000). Being There and Doing IT in the Workplace: A Case Study of a Co-Development Approach in Healthcare. In Cherkasky, T. Greenbaum, J., Mambery, P. (eds.) *Proceedings of the Participatory Design Conference*. New York, November 28th - December 1st, 2000. pp. 96-105.

Hartswood, M., Procter, R., Slack, R., Voß, A., Büscher, M., Rouncefield, M. and Rouchy, P. (2002) Co-realisation: Towards a Principled Synthesis of Ethnomethodology and Participatory Design. *Scandinavian Journal of Information Systems*, 14(2), pp. 9-30.

Hartswood, M., Procter, R., Rouncefield, M., Slack, R., Soutter, J. and Voss, A. (2003). 'Repairing' the Machine: A case study of evaluating computer-aided detection tools in breast screening. *Proceedings of the 8th European Conference on Computer Supported Cooperative Work*. pp. 375-394.

Hartswood, M., Procter, R., Rouncefield, M. and Slack, R. (2003a). Making a Case in Medical Work: Implications for the Electronic Medical Record. *Computer Supported Cooperative Work*, vol. 12, pp. 241-266.

Heath, C. and Luff, P. (1992). Collaboration and Control: Crisis Management and Mutimedia Technology in London Underground Line Control Rooms. *Computer Supported Cooperative* Work, vol. 1, pp. 69-94.

Heath, C., Jirotka, M., Luff, P., and Hindmarsh, J. (1995). Unpacking Collaboration: the Interactional Organisation of Trading in a City Dealing Room. *Computer Supported Cooperative Work*, Vol. 3, 147-165.

Heath, C., Hindmarsh, J and Luff, P. (1999). Interaction in Isolation: The Dislocated World of the London Underground Train Driver. *Sociology* 33(3). pp. 555-575.

Heath, C. and Luff, P. (2000). *Technology in Action*. Cambridge University Press.

Hemmings, T., Crabtree, A., Rodden, T., Clarke, K. and Rouncefield, M. (2002). Probing the Probes. Binder, T., Gregory, J. and Wagner, I. (eds.) *Proceedings of the Participatory Design Conference*. pp. 42-50.

Henderson, A. and Kyng, M. (1991) There's No Place Like Home: Continuing Design in Use. In Greenbaum, J. and Kyng, M. (eds.) *Design at Work: Co-operative Design of Computer Systems*. Lawrence Erlbaum. pp. 219-240.

Hendy, J. *et al.* (2005). Challenges to implementing the national programme for information technology (NPfIT): a qualitative study. *British Medical Journal*, vol. 331. pp. 331-336.

Herbsleb, J.D., Mockus, A., Finholt, T.A. and Grinter, R.E. (2000). Distance, Dependencies, and Delay in a Global Collaboration. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. pp. 319-328.

Heritage, J. (1984). *Garfinkel and Ethnomethodology*. Polity Press.

Holtzblatt, K. (2005). Getting Started on a Contextual Project. *Personal and Ubiquitous Computing*, vol. 9. pp. 227-237.

Hopkins, J. (2000) Component Primer. *Communications of the ACM* 43(10)., pp. 27-30.

Hughes, J. A., Randall, D. and Shapiro, D. (1992). Faltering from Ethnography to Design. *Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work*, pp. 115-122.

Hughes, J.A., Randall, D. and Shapiro, D. (1993). From ethnographic record to system design: some experiences from the field. *Computer Supported Cooperative Work* 1(3) pp. 123-141.

Hughes, J., King, V., Rodden, T. and Andersen, H. (1994). Moving Out from the Control Room: Ethnography in System Design. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. pp. 429-439.

Hughes, J., King, V, Rodden, T and Andersen, H. (1995). The Role of Ethnography in Interactive Systems Design. *Interactions*, April 1995.

Hughes, J., O'Brien, J., Rodden, T. and Rouncefield, M. (2000). Ethnography, communication and support for design. Luff, P., Hindmarsh, J. and Heath, C. (eds.) *Workplace Studies: Recovering Work Practice and Informing System Design*. Cambridge University Press. pp. 187-214.

Jirotka, M., Gilbert, N. and Luff, P. (1992). On the Social Organisation of Organisations. *Computer Supported Cooperative Work*, vol. 1, pp. 95-118.

Jirotka, M. and Goguen, J.A. (eds., 1994). *Requirements Engineering: Social and Technical Issues*. Academic Press.

Jirotka, M. and Wallen, L. (2000). Analysing the workplace and user requirements: challenges for the development of methods for requirements engineering. Luff, P., Hindmarsh, J. and Heath, C. (eds.) *Workplace Studies: Recovering Work Practice and Informing System Design*. Cambridge University Press. pp. 242-251.

Jones, C.B. (2003). The Early Search for Tractable Ways of Reasoning about Programs. *IEEE Annals of the History of Computing* 25(2). pp. 26-49.

Karasti, H. (2001). Bridging Work Practice and System Design: Integrating Systemic Analysis, Appreciative Intervention and Practitioner Participation. *Computer Supported Cooperative Work*, vol. 10, pp. 211-246.

Kensing, F. and Blomberg, J. (1998). Participatory Design: Issues and Concerns. *Computer Supported Cooperative Work*, vol. 7, pp. 167-185.

Kensing, F., Simonsen, J. and Bødker, K. (1998). MUST: A Method for Participatory Design. *Human-Computer Interaction*, vol. 13, pp. 167-198.

Kensing, F., Simonsen, J. and Bødker, K. (1998a). Participatory Design at a Radio Station. *Computer Supported Cooperative Work*, vol. 7, pp. 243-271.

Kephart, J.O. and Chess, D.M. (2003). The Vision of Autonomic Computing. *IEEE Computer*, January 2003, pp. 41-50.

Kephart, J.O. (2005). Research Challenges of Autonomic Computing. *Proceedings of the 27th International Conference on Software Engineering*. pp. 15-22.

Kiczales, G. *et al.* (1997). Aspect-Oriented Programming. *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*. *Lecture Notes in Computer Science* 1241. Springer Verlag. pp. 220-242.

Kling, R. (1991). Cooperation, Coordination and Control in Computer-Supported Work. *Communications of the ACM*, Dec. 1991. pp. 83-88.

Kovalainen, M., Robinson, M. and Auramäki, E. (1998). Diaries at Work. *Proceedings of the ACM conference on computer-supported cooperative work.* pp. 49-58.

Kraft, P. (1979). The Industrialization of Computer Programming: From Programming to "Software Production". In Zimbalist, A. (ed.) *Case Studies in the Labour Process*. Monthly Review Press. pp. 1-17.

Kraft, S. and Dubnoff, S. (1986). Job Content, Fragmentation and Control in Computer Software Work. *Industrial Relations* 25(2) Spring.

Kraft, P. and Bansler, J. (1992). The Collective Resource Approach: The Scandinavian Experience. Muller, M. J., Kuhn, S. and Maskill, J.A. (eds.) *Proceedings of the Participatory Design Conference*. pp. 127-135.

Kyng, M. and Mathiassen, L. (1982). Systems Development and Trade Union Activities. Bjørn-Andersen, N. *et al.* (ed.). *Information Society, for richer, for poorer*. Amsterdam: North-Holland.

Kyng, M. (1994). Collective Resources Meets Puritanism. *Scandinavian Journal of Information Systems*, 6(1). pp.85-95.

Kyng, M. (1995). Making Representations Work. *Special issue of Communications of the ACM* 38(9), pp. 46-55

Laprie, J-C. (1995). Dependable Computing: Concepts, Limits, Challenges. *Invited paper to FTCS-25, the 25th IEEE International Symposium on Fault-Tolerant Computing*. June 27-30.

Lehman, M. M. (1998). Software's Future: Managing Evolution. *IEEE Software*. Jan.-Feb. 1998. pp. 40-44.

Leveson, N. and Turner, C. S. (1993). An investigation of the Therac-25 accidents. *IEEE Computer*, 26(7), pp. 18-41.

Lieberman, K. (2003). *Dialectical Practice in Tibetan Philosophical Culture, An Ethnomethodological Inquiry into Formal Reasoning*. Rowman and Littlefield.

Liker, J.K., Haddad, C.J., Karlin, J. (1999). Perspectives on Technology and Work Organization. *Annual Review of Sociology* 25. pp. 575-596.

Livingston, E. (1986). *The Ethnomethodological Foundations of Mathematics*. London: Routledge, Kegan & Paul.

Livingston, E. (1987). *Making Sense of Ethnomethodology*. London, New York: Routledge & Kegan Paul.

Luff, P., Heath, C. and Greatbatch, D. (1994). Work, interaction and technology: The naturalistic analysis of human conduct and requirements

analysis. Jirotka, M. and Goguen, J. (eds.) *Requirements Engineering: Social and Technical Issues*. Academic Press. pp. 259-288.

Lynch, M. (1985). *Art and artefact in laboratory science: A study of shop work and shop talk in a research laboratory*. Routledge & Kegan Paul.

Lynch, M. (1991). Method: measurement – ordinary and scientific measurement as ethnomethodological phenomena. Button, G. (ed.) *Ethnomethodology and the Human Sciences*. Cambridge University Press. pp. 77-108.

Lynch, M. (1993). *Scientific Practice and Ordinary Action: Ethnomethodology and social studies of science*. Cambridge University Press.

Macbeth, D. (2001). On "Reflexivity" in Qualitative Research: Two Readings, and a Third. *Qualitative Inquiry* 7(1), pp. 35-68.

Mackay, W. (1990). Patterns of sharing customizable software. *Proceedings of the Conference on Computer Supported Cooperative Work*. pp. 209-221.

MacKenzie, D. (1996). Computer-Related Accidental Death. Chapter 9 in MacKenzie, D. *Knowing Machines: Essays on Technical Change*. MIT Press.

Martin, D., Rodden, T., Rouncefield, M., Sommerville, I. and Viller, S. (2001). Finding Patterns in the Fieldwork. *Proceedings of the European Conference on Computer Supported Cooperative Work*. Kluwer. pp. 39-58.

Martin, D. and Sommerville, I. (2004). Patterns of Cooperative Interaction: Linking Ethnomethodology and Design. *ACM Transactions on Computer-Human Interaction* 11(1) pp. 59-89.

Mason, G. (2000). Production Supervisors in Britain, Germany and the United States: Back from the dead again? *Work, Employment and Society*, Vol. 14 No. 4, pp. 625-645

Maynard, D. W. and Clayman, S. E. (1991). The Diversity of Ethnomethodology. *Annual Review of Sociology* 17:385-418.

Meyer, B. (2003). The Grand Challenge of Trusted Components. *Proceedings of the 25th International Conference on Software Engineering*. pp. 660-667.

Mitev, N. N. (1996). More than a failure? The computerized reservation systems at French Railways. *Information Technology & People* 9(4), pp.8-19.

Mogensen, P. and Trigg, R. (1992). Using Artifacts as triggers for participatory analysis. In Muller, M. J., Kuhn, S. and Meskill, J.A. *PDC'92: Proceedings of the Participatory Design Conference*. Palo Alto, CA, USA.

Muller, M. J., Wildman, D. M., White, E. A. (1993). Taxonomy of PD Practices: A Brief Practitioner's Guide. *Communications of the ACM* 36(4) pp. 26-28.

Muller, M.J. (1999). *Catalogue of Scenario-Based Methods and Methodologies*. IBM Watson Research Centre Technical Report no. 99-06. available from http://research.ibm.com/cambridge.

Nandhakumar, J. and Avison, D.E. (1999). The fiction of methodological development: a field study of information systems development. *Information Technology & People* 12(2), pp. 176-191.

Nardi, B.A. and Miller, J.R. (1990). An Ethnographic Study of Distributed Problem Solving in Spreadsheet Development. *Proceedings of the 1990 ACM conference on computer-supported cooperative work*, Los Angeles, October 7.-10. 1990. pp. 197-208.

Nardi, B.A. and Miller, J.R (1991). Twinkling lights and nested loops: distributed problem solving and spreadsheet development. *International Journal of Man-Machine Studies*, vol. 34, pp. 161-184.

Nardi, B.A. (1993). *A Small Matter of Programming: Perspectives on End User Computing*. MIT Press.

National Institute for Standards and Technology (2002). *The Economic Impacts of Inadequate Infrastructure for Software Testing*. NIST Planning Report 02-3. Available at: www.nist.gov/director/prog-ofc/report02-3.pdf (accessed 23.12.2005).

Naur, P. and Randell, B. (eds., 1969). *Software Engineering: Report of a conference sponsored by the NATO Science Committee*, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO. Available at: homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF (accessed 23.12.2005).

Nerur, S., Mahapatra, R. and Mangalaraj, G. (2005). Challenges of Migrating to Agile Methodologies. *Communications of the ACM* 48(5), pp. 73-78.

Neumann, P.G. (1995). *Computer Related Risks*. ACM Press and Addison Wesley.

Newell, A. and Card, S.K. (1985). The Prospects for Psychological Science in Human-Computer Interaction. *Human-Computer Interaction*, vol. 1. pp. 209-242.

Newell, A. and Card, S. (1986). Straightening Out Softening Up: Response to Carroll and Campbell. *Human-Computer Interaction*, vol. 2. pp. 251-267.

Olerup, A. (1989). Socio-Technical Design of Computer-Assisted Work: A Discussion of the ETHICS and Tavistock Approaches. *Scandinavian Journal of Information Systems*, vol. 1. pp. 43-71.

Ortner, S. (1997). Introduction. *Representations*, no. 59, Special Issue: The Fate of "Culture": Geertz and Beyond. pp. 1-13.

Panko, R.R. and Halverson, R.P. Jr. (1994). Individual and Group Spreadsheet Design: Patterns of Errors. *Proceedings of the 27th Annual Hawaii International Conference on System Sciences*. pp. 4-10.

Panko, R.R. (1996). Hitting the Wall: Errors in Developing and Debugging a "Simple" Spreadsheet Model. *Proceedings of the 29th Annual Hawaii International Conference on System Science*. pp. 356-363.

Parsons, T. (1937). The *Structure of Social Action*. McGraw-Hill.

Patterson, D.A. *et al.* (2002). *Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies*. UC Berkeley Computer Science Technical Report UCB//CSD-02-1175, March 15, 2002.

available at: http://roc.cs.berkeley.edu/papers/ROC_TR02-1175.pdf (accessed 21.01.2006).

Perrow, C. (1984). *Normal Accidents: Living With High-Risk Technologies*. Basic Books.

Pfister, C. and Szyperski, C. (1996). Why Objects Are Not Enough. *Proceedings of the First International Component Users Conference*, Munich, Germany.

Piore, M.J. and Sabel, C.F. (1984). *The Second Industrial Divide: Possibilities for Prosperity*. New York: Basic Books.

Plowman, L., Rogers, Y. and Ramage, M. (1995). What Are Workplace Studies For? *Proceedings of European Conference on Computer Supported Cooperative Work*, Stockholm, Sweden, pp. 309-324.

Pollner, M. (1974). Sociological and Common-Sense Models of the Labelling Process. In Turner, R. (ed.) *Ethnomethodology: Selected Readings*. Penguin Education.

Pollner, M. (1991). Left of Ethnomethodology: The Rise and Decline of Radical Reflexivity. *American Sociological Review* 56(3), pp. 370-380.

Pollock, N., Williams, R. and Procter, R. (2003). Fitting Standard Software Packages to Non-standard Organizations: The 'Biography' of an Enterprise-wide System. *Technology Analysis & Strategic Management* 15(3), pp. 317-332.

Preece, J., Rogers, Y. and Sharp, H. (2002). *Interaction Design: beyond human-computer interaction*. John Wiley & Sons.

Procter, R. and Williams, R. (1996). Beyond design: Social learning and computer-supported cooperative work. In Shapiro, D., Taubner, M. and Traunmüller, R. (eds.) *The Design of Computer-Supported Cooperative Work and Groupware Systems*, Elsevier Science, pp. 445-464.

Procter, R. and Williams, R. (1996a) Social Learning and Innovations in Multimedia-based CSCW. *ACM SIGOIS Bulletin* 17(3). pp. 73-76.

Randall, D., Hughes, J. and Shapiro, D. (1994). Steps toward a partnership: Ethnography and system design. Jirotka, M. and Goguen, J.A. *Requirements Engineering: Social and Technical Issues*. Academic Press. pp. 241-258.

Randall, D., Rouncefield, M., Hughes, J.A. (1995). Chalk and Cheese: BPR and ethnomethodologically informed ethnography in CSCW. Marmolin, H., Sundblad, Y. and Schmidt, K. (eds.) *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work*. pp. 325-340.

Randell, B. and Buxton, J.N (eds., 1970). *Software Engineering Techniques: Report of a conference sponsored by the NATO Science Committee*, Rome, Italy, 27-31 Oct. 1969, Brussels, Scientific Affairs Division, NATO. Available at: homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1969.PDF (accessed 23.12.2005).

Robinson, M., Kovalainen, M. and Auramäki, E. (2000). Diary as Dialogue in Papermill Process Control. *Communications of the ACM* 43(1), pp. 65-70.

Rogers, Y. (1992). Ghosts in the Network: Distributed Troubleshooting in a Shared Working Environment. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. pp. 346-355.

Rönkkö, K., Dittrich, Y. and Randall, D. (2005). When Plans do not Work Out: How Plans are Used in Software Development Projects. *Computer Supported Cooperative Work*, vol. 14, pp. 433-468.

Rouncefield, M., Hughes, J.A., Rodden, T. and Viller, S. (1994). Working with "Constant Interruption": CSCW and the Small Office. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. pp. 275-286.

Rouncefield, M.F. (2002). *'Business as Usual': An Ethnography of Everyday (Bank) Work*. PhD Thesis, Department of Sociology, University of Lancaster.

Roy, D. (1954). Efficiency and "the Fix": Informal Intergroup Relations in a Piecework Machine Shop. *American Journal of Sociology*, Vol. 60, No. 3 (Nov.), pp. 255-266.

Royal Academy of Engineering (2004). *The Challenges of Complex IT Projects: the report of a working group from The Royal Academy of Engineering and The British Computer Society*. London: The Royal Academy of Engineering, ISBN 1-903496-15-2.

Royce, W.W. (1987 [1970]). Managing the development of large software systems: concepts and techniques. *Proceedings of the 9th International Conference on Software Engineering*. pp. 328-338. (originally published in Proc. IEEE WESCON 1970, pp. 1-9).

Russo, N.L. and Stolterman, E. (2000). Exploring the assumptions underlying information systems methodologies: Their impact on past, present and future ISM research. *Information Technology & People* 13(4), pp. 313-327.

Sacks, H. and Schegloff, E.A. (1979). Two preferences in the organization of reference to persons in conversation and their interaction. Psathas, G. (ed.) Everyday language: Studies in ethnomethodology. New York: Irvington. pp. 15-21.

Sacks, H. (1984). Notes on methodology. Atkinson, J.M. and Heritage, J. (eds.) *Structures of Social Action: Studies in Conversation Analysis*. Cambridge University Press.

Schenkein, J. (1974). Sketch of an Analytic Mentality for the Study of Conversational Interaction. Schenkein, J. (ed.) *Studies in the Organization of Conversational Interaction*. Academic Press. pp. 1-6.

Schmidt, K. and Bannon, L. (1992). Taking CSCW Seriously: Supporting Articulation Work. *Computer Supported Cooperative Work*, vol. 1, pp. 7-40.

Schmidt, K. (1997). Of maps and scripts: The status of formal constructs in cooperative work. *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*. pp. 138-147.

Schmidt, K. (2000). The critical role of workplace studies in CSCW. Luff, P., Hindmarsh, J. and Heath, C. (eds.) *Workplace Studies: Recovering Work*

*Practice and Informing System Design*. Cambridge University Press. pp. 141-149.

Schuler, D. and Namioka, A. (eds., 1993) *Participatory Design: Principles and Practices*. Lawrence Erlbaum Associates.

Schütz, A. (1953). Common-Sense and Scientific Interpretation of Human Action. *Philosophy and Phenomenological Research*, 14(1), pp. 1-38.

Schütz, A. (1973). *Collected Papers I: The Problem of Social Reality*. Den Haag: Martinus Nijhoff.

Schütz, A. and Luckmann, T. (1974) *Structures of the Lifeworld*. London: Heinemann.

Sewell, G. (1998). The Discipline of Teams: The Control of Team-Based Industrial Work Through Electronic and Peer Surveillance. Administrative Quarterly 43(2). pp. 397-428.

Shapiro, D. (1994). The Limits of Ethnography: Combining Social Sciences for CSCW. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. pp. 417-428.

Shapiro, D. (2005). Participatory Design: the will to succeed. Bertelsen, O., Bouvin, N.O., Krogh, P.G., Kyng, M. (eds.) *Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility*, Aarhus, Denmark, August 20 - 24, 2005. pp. 29-30.

Sharma, S. and Rai, A. (2000). CASE Deployment in IS Organizations. *Communications of the ACM* 43(1), pp. 80-88.

Sharrock, W.W. (1974). On Owning Knowledge. Turner, R. (ed.) *Ethnomethodology: Selected Readings*. Penguin. pp. 45-53.

Sharrock, W.W. and Anderson, R.J. (1982). On the Demise of the Native: Some Observations on and a Proposal for Ethnography. *Human Studies*, vol. 5, pp. 119-135.

Sharrock, W. and Anderson, R. (1986). *The Ethnomethodologists*. Ellis Horwood and Tavistock Publications.

Sharrock, W. and Anderson, R. (1991) Epistemology: professional scepticism Button, G. *Ethnomethodology and the Human Sciences*. Routledge.

Sharrock, W. and Anderson, R. (1993). Working Towards Agreement. Button, G. (ed.) *Technology in Working Order: Studies of work, interaction and technology*. Routledge. pp. 149-161.

Sharrock, W. and Anderson, R. (1994). The User as a Scenic Feature of the Design Space. *Design Studies*, vol. 15 (1), pp. 5-18. Also Rank Xerox EuroPARC Technical Report EPC-91-105, 1991.

Sharrock, W. (2001). Fundamentals of Ethnomethodology. Chapter 19 in Ritzer, G. and Smart, B. (eds.) *Handbook of Social Theory*. Sage Publications.

Sharrock, W. and Button, G. (2003). Plans and Situated Action Ten Years On. *The Journal of the Learning Sciences* 12(2), pp. 259-264.

Simonsen, J. and Kensing, F. (1997). Using Ethnography In Contextual Design. *Communications of the ACM*, 40(7). pp. 82-88.

Slack, R. (1995). *Varieties of Sociological Reflexivity*. Unpublished PhD Thesis. Victoria University of Manchester.

Slack, R. (1997). *Information Systems Design – Whose Problem is it Anyway?* Unpublished manuscript.

Smith David, J., Schuff, D. and St. Louis, R. (2002). Managing Your IT Total Cost of Ownership. *Communications of the ACM* 45(1), pp. 101-106.

Sommerville, I. (2001) *Software Engineering*, 6th edition. Addison Wesley.

Sørensen, K.H. (1996). *Learning Technology, constructing culture, Socio-technical change as social learning*. STS working paper, no. 18/96, University of Trondheim: Centre for Technology and Society.

Stewart, J. and Williams, R. (2005). The wrong trousers? Beyond the design fallacy: social learning and the user. Howcroft, D. and Trauth, E.M. (eds.) *Handbook Of Critical Information Systems Research: Theory and Application*. Edward Elgar. pp. 195-221.

Suchman, L. (1983). Office Procedure as Practical Action: Models of Work and System Design. *ACM Transactions on Office Information Systems* 1(4), pp. 320-328.

Suchman, L. A. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge University Press.

Suchman, L. (1993). Technologies of Accountability: Of lizards and aeroplanes. Button, G. (ed.) *Technology in Working Order: Studies of work, interaction, and technology*. London, Routledge. pp. 113-126.

Suchman, L. (1994). Do Categories Have Politics? The language/action perspective reconsidered. *Computer Supported Cooperative Work*, vol. 2, pp. 177-190.

Suchman, L. (1995). Speech Acts and Voices: Response to Winograd *et al*. *Computer Supported Cooperative Work*, vol. 3, pp. 85-95.

Suchman, L. (1995a). Making Work Visible. *Communications of the ACM* 38(9). pp. 56-64.

Suchman, L. (2000). Making a case: 'knowledge' and 'routine' work in document production. Luff, P., Hindmarsh, J., Heath, C. (eds.) *Workplace Studies: Recovering Work Practice and Informing System Design*. Cambridge University Press.

Suchman, L., Trigg, R. and Blomberg, J. (2002). Working artefacts: ethnomethods of the prototype. *British Journal of Sociology* 53(2). pp. 163-179.

Suchman, L. (2002). Located accountabilities in technology production. *Scandinavian Journal of Information Systems*, Vol. 14, No 2, pp. 91-105.

Suchman, L. (2003). Writing and Reading: A Response to Comments on Plans and Situated Actions. *The Journal of the Learning Sciences* 12(2), pp. 299-306.

Sudnow, D. (1978). *Ways of the Hand*. Harvard University Press.

Sundblad, Y., Lind, T. and Rudling, J. (2002). IT product requirements and certification from the users' perspective. *Proceedings of the 6th International Scientific Conference on Work With Display Units, WWDU*. pp. 490-492.

Swartz, J.A. (1996). Airport 95: Automated Baggage System? *ACM Software Engineering Notes* 21(2), pp. 79-83.

Szyperski (1999). *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley / ACM Press.

Taylor, M. J., Maynihan, E. P. and Wood-Harper, A. T. (1998). End-user computing and information systems methodologies. *Information Systems Journal* 8, pp. 85-96.

ten Have, P. (2002) The Notion of Membership is the Heart of the Matter: On the Role of Membership Knowledge in Ethnomethodological Inquiry. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research* [On-line Journal], 3(3). Available at: http://www.qualitative-research.net /fqs/fqs-eng.html (Accessed 18.10.2005)

Törpel, B. (2005). Participatory Design: A multi-voiced effort. Bertelsen, O., Bouvin, N.O., Krogh, P.G., Kyng, M. (eds.) *Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility*, Aarhus, Denmark, August 20 - 24, 2005. pp. 177-181.

Trigg, R.H. and Bødker, S. (1994). From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. pp. 45-54.

Trigg, R.H., Blomberg, J. and Suchman, L. (1999) Moving document collections online: The evolution of a shared repository. Bødker, S., Kyng, M., Schmidt, K. (eds.) *Proceedings of the 6th European Conference on Computer-Supported Cooperative Work*. pp. 331-350.

Trigg, R.H. (2000). From Sandbox to "Fundbox": Weaving participatory design into the fabric of a busy non-profit. Cherkasky, T., Greenbaum, J.,

Mambrey, P. and Pors, J.K. (eds.) *Proceedings of the Participatory Design Conference*. pp. 174-183.

Turk, D., France, R. and Rumpe, B. (2002). Limitations of Agile Software Processes. *Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, XP2002*, May 26-30, Alghero, Italy, pp. 43-46.

Voß, A., Slack, R., Procter, R., Williams, R., Hartswood, M. and Rouncefield, M. (2002). Dependability as Ordinary Action. Anderson, Bologna and Felici (eds.) *Computer Safety, Reliability and Security: Proceedings of the 21st International Conference, SAFECOMP 2002*. Sept. 2002. pages 32-43.

Walldius, Å., Sundblad, Y., Bengtsson, L., Lind, T. and Sandblad, B. (2004). User-driven quality certification of workplace software, the UsersAward experience. *Proceedings of the Work with Computing Systems Conference*. pp. 150-155.

Walldius, Å, Sundblad, Y. and Borning, A. (2005). A First Analysis of the UsersAward Programme from a Value Sensitive Design Perspective. Bertelsen, O., Bouvin, N.O., Krogh, P.G., Kyng, M. (eds.) *Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility*, Aarhus, Denmark, August 20 - 24, 2005. pp.199-202.

Wastell, D.G. (1996). The fetish of technique: methodology as a social defence. *Information Systems Journal*, vol. 6, pp. 25-40.

Wieder, D. L. (1974). On Meaning By Rule. In Douglas, J. D. (ed.) *Understanding Everyday Life*. Routledge & Kegan Paul.

Wieder, D. L. (1974a). Telling the Code. Chapter 13 in Turner, R. *Ethnomethodology: Selected Readings*. Penguin.

Wild, Martin and Herges, Sascha (2000). *Total Cost of Ownership (TCO) – Ein Überblick*. Arbeitspapiere WI Nr. 1/2000, Lehrstuhl für Allg. BWL und Wirtschaftsinformatik, Universität Mainz.

Williams, M. G. and Begg, V. (1993). *Translation between Software Designers and Users*. Communications of the ACM 36(4), pp. 102-103.

Williams, R., Stewart, J. and Slack, R. (2005). *Social Learning in Technological Innovation: Experimenting with Information and Communication Technologies*. Edward Elgar.

Winograd, T. (1994). Categories, Disciplines, and Social Coordination. *Computer Supported Cooperative Work*, vol. 2, pp. 191-197.

Wittgenstein, L. (1984) *Tractatus logico-philosophicus Werkausgabe Band 1: Tractatus logico-philosophicus, Tagesbücher 1914-1916, Philosophische Untersuchungen*. Suhrkamp Taschebuch Wissenschaft 501. Erste Auflage 1984.

Womack, J.P., Jones, D.T. and Roos, D. (1991) *The Machine that Changed the World*. New York: Harper Perennial.

Woolgar, S. (1991). Configuring the user: the case of usability trials. Law, J. (ed.) *A Sociology of Monsters: Essays on Power, Technology and Domination*. Routledge. pp. 58-100.

Woolgar, S. (1994). Rethinking requirements analysis: Some implications of recent research into producer- consumer relationships in IT development. Jirotka, M. and Goguen, J. (eds.) *Requirements Engineering: Social and Technical Issues*. Academic Press. pp. 201-216.

Wynn, E. (1991). Taking Practice Seriously. Greenbaum, J. And Kyng M. *Design at Work: Cooperative Design of Computer Systems*. Lawrence Erlbaum Associates. pp.45-64.

Whyte, W.F. (1973 [1943]). *Street Corner Society: The Social Structure of an Italian Slum*, 2nd edition. The University of Chicago Press.

Yourdon, E. and Constantine, L.C. (1979). *Structured Design: Fundamentals of a Discipline of Computer Program and System Design*. Prentice Hall.

Yourdon, E. (1995). When Good Enough Software Is Best. *IEEE Software*, May 1995, pp. 79-81.

Zemanek, H. (1972). Some philosophical aspects of information processing. Zemanek, H. (ed.). *The Skyline of Information Processing*. Proceedings of the Tenth Aniversary Celebration of IFIP. North-Holland. pp. 93-140.

Zemanek, H. (1975). The Human Being and the Automaton: Selected Aspects of a Philosophy of Information Processing. Mumford, E. And Sackman, H. (eds.) *Human Choice and Computers*. North-Holland Publishing Company. pp. 3-30.

# Appendix 1: Publications

The following lists the publications that have arisen from work on this thesis:

## 9.1 Papers in Journals

1. Marina Jirotka, Rob Procter, Mark Hartswood, Roger Slack, Andrew Simpson, Catelijne Coopmans, Chris Hinds and Alex Voss. Collaboration and Trust in Healthcare Innovation: The eDiaMoND Case Study. *Computer Supported Cooperative Work*, vol. 14, 2005. pp. 369-398.
2. M. Hartswood, R. Procter, P. Rouchy, M. Rouncefield, R. Slack, A. Voss. Working IT Out in Medical Practice: IT Systems Design and Development as Co-Production. *Methods of Information in Medicine*, 2003.
3. M. Hartswood, R. Procter, M. Rouncefield, R. Slack, A. Voss, R. Williams. Building Information Systems as Universalised Locals. Journal of *Knowledge, Technology and Policy*, 14(3), Fall, 2002.
4. Mark Hartswood, Rob Procter, Roger Slack, Alex Voß, Monika Buscher, Mark Rouncefield, Philippe Rouchy. Co-realisation: Towards a Principled Synthesis of Ethnomethodology and Participatory Design. *Scandinavian Journal of Information Systems*, 14(2), 2002.

## 9.2 Papers in International Conference and Workshop Proceedings

5. M. Hartswood, M. Jirotka, R. Procter, R. Slack, A. Voss, S. Lloyd. Working IT out in e-Science: Experiences of requirements capture in a HealthGrid project. *Proceedings of HealthGrid* 2005. pp. 198-209.
6. G. Hartstone, M. Hartswood, R. Procter, R. Slack, A. Voss, G. Rees. Supporting Informality: Team Working and Integrated Care Records. *Proceedings of CSCW'04*. 2004. pages 142--151
7. D. N. Bateman, A. M. Good, R. Procter, R. Slack, M. Hartswood, A. Voß. The Work of Advice Giving in a Poisons Information Setting: an Analysis of Telephone Requests for Information to a Poisons Information Centre. Proceedings of the British Toxicology Society Annual Congress, Heriot Watt University and Neurotox 2003, University of Nottingham, UK. Published in: *Toxicology*, 202(1-2). 2004.
8. Karen Clarke, John Hughes, Dave Martin, Mark Rouncefield, Ian Sommerville, Corin Gurr, Mark Hartswood, Rob Procter, Roger Slack, Alex Voß. Dependable Red Hot Action. Kari Kuutti, Eija Helena Karsten, Geraldine Fitzpatrick, Paul Dourish, Kjeld Schmidt (eds.) ECSCW 2003: *Proceedings of the Eighth European Conference on Computer Supported Cooperative Work*. September, 2003. pages 61–80
9. Mark Hartswood, Rob Procter, Mark Rouncefield, Roger Slack, James Soutter, Alex Voss. 'Repairing' the Machine: A Case Study of the Evaluation

of Computer-Aided Detection Tools in Breast Screening. Kari Kuutti, Eija Helena Karsten, Geraldine Fitzpatrick, Paul Dourish, Kjeld Schmidt (eds.) *ECSCW 2003: Proceedings of the Eighth European Conference on Computer Supported Cooperative Work*. September, 2003. pages 375–394

10. Alexander Voß, Rob Procter, Mark Hartswood, Roger Slack, James Soutter, Robin Williams, Mark Rouncefield. End-User Participation in Standards Making? Taking the Practical Work of Standards Selection, Configuration and Use Seriously. Tineke M. Egyedi, Ken Krechmer, Kai Jakobs (eds.) *Proceedings of the 3rd IEEE Conference on Standardization and Innovation in Information Technology*. October 22-24, 2003. pages 261–269
11. Alexander Voß, Rob Procter, Roger Slack, Mark Hartswood, Robin Williams, Mark Rouncefield. Accomplishing 'Just-in-Time' Production. C. Johnson (ed.) *Proceedings of the 21st European Annual Conference on Human Decision Making and Control*. July, 2002.
12. Mark Hartswood, Rob Procter, Roger Slack, James Soutter, Alex Voß, Mark Rouncefield. The Benefits of a Long Engagement: From Contextual Design to The Co-realisation of Work Affording Artefacts. *Proceedings of NordiCHI*. 2002.
13. Monika Büscher, Dan Shapiro, Mark Hartswood, Rob Procter, Roger Slack, Alex Voß, Preben Mogensen. Promises, Premises and Risks: Sharing Responsibilities, Working Up Trust and Sustaining Commitment in Participatory Design Projects. T. Binder, J. Gregory, I. Wagner (eds.) *PDC'2002 Proceedings of the Participatory Design Conference*. June, 2002. pages 183--192
14. Alexander Voß, Roger Slack, Rob Procter, Robin Williams, Mark Hartswood, Mark Rouncefield. Dependability as Ordinary Action. Stuart Anderson, Sandro Bologna, Massimo Felici (eds.) *Computer Safety, Reliability and Security: Proceedings of the 21st International Conference, SAFECOMP 2002*. September, 2002. pages 32--43
15. M. Hartswood, A. Voss, R. Procter, R. Williams. Pick-n-Mix Approaches to Technology Supply: XML as a standard glue for systems development.. T. Egyedi, K. Dittrich (eds.) *Proceedings of the 6th EURAS Workshop on Standards, Compatibility and Infrastructure Development*. June 26th-29th, 2001. pages 117--135
16. Alexander Voß, Rob Procter, Roger Slack, Mark Hartswood, Robin Williams, Mark Rouncefield. Production Management and Ordinary Action: an investigation of situated, resourceful action in production planning and control. J. Levine (ed.) *Proceedings of the 20th UK Planning and Scheduling SIG Workshop*. December 13th-14th, 2001. pages 230--243
17. Alexander Voß, Rob Procter, Robin Williams. Innovation in Use: Interleaving day-to-day operation and systems development. T. Cherkasky, J. Greenbaum, P. Mambery (eds.) *Proceedings of the Participatory Design Conference*. 28 November to 1 December, 2000.

## 9.3 Book Chapters

18. Clarke, K., Martin, D., Rouncefield, M., Sommerville, I., Voß, A., Gurr, C., Procter, R., Slack, R. and Hartswood, M. Explicating Failure. In: Clarke, K., Hardstone, G., Rouncefield, M. and Sommerville, I. (eds.) *Trust in Technology: A Socio-Technical Perspective*. Springer, 2006. pp. 123-146.
19. Clarke, K., Hughes, J., Martin, D., Rouncefield, M., Sommerville, I., Voß, A., Procter, R., Slack, R. and Hartswood, M. 'Its About Time': Temporal Features of Dependability. In: Clarke, K., Hardstone, G., Rouncefield, M. and Sommerville, I. (eds.) *Trust in Technology: A Socio-Technical Perspective*. Springer, 2006. pp. 105-122.

## 9.4 Miscellaneous Publications

20. R. Williams, R. Bunduchi, M. Gerst, I. Graham, N. Pollock, R. Procter, A. Voß. Understanding the Evolution of Standards: Alignment and Reconfiguration in Standards Development and Implementation Arenas. European Association for the Study of Science and Technology (EASST) Conference. August, 2004.
21. Mark Hartswood, Rob Procter, Roger Slack, James Soutter, Alex Voss, Mark Rouncefield. Embedding the User: The Organisational Dimensions of End User Development. Workshop on End-user development: Current experiences and challenges. 2003.
22. Alexander Voss, Roger Slack, Mark Hartswood, Rob Procter, James Soutter. Moving Beyond "Ethnographies for Design": is there a role for ethnomethodology in IT systems design?. Presented at the 2003 IIEMCA Manchester Conference "Producing Local Order". July, 2003.
23. Roger Slack, Mark Hartswood, Rob Procter, Alex Voss. Instructed actions and lived work: cases of professional vision. Presented at the 2003 IIEMCA Manchester Conference "Producing Local Order". July, 2003.
24. M. Hartswood, R. Procter, R. Slack, J. Soutter, A. Voss, M. Rouncefield. Healthcare Technology and Professional Vision. International Conference on the Management of Healthcare and Medical Technology. 2003.
25. Mark Hartswood, Rob Procter, Roger Slack, Alexander Voß, John A. Hughes, Karen Clarke, Mark Rouncefield. "Cunning Plans": Some Notes on Plans, Procedures and CSCW. *Requirenautics Quarterly: The Newsletter of the BCS Requirements Engineering SG*, January, 2002. pages 12–18
26. Mark Hartswood, Rob Procter, Roger S. Slack, Alex Voß, Mark Rouncefield. Information Systems and Workplace Studies: Observing the Contingencies of 'Just-in-Time' Production. Workshop on "Interpretive" Approaches to Information Systems and Computing Research. July, 2002.
27. Mark Hartswood, Rob Procter, Roger S. Slack, Alex Voß, Mark Rouncefield. The work of co-realization. Workshop on "Interpretive" Approaches to Information Systems and Computing Research. July, 2002.

28. Alexander Voß, Rob Procter, Roger Slack, James Soutter, Mark Hartswood, Robin Williams, Mark Rouncefield. "It's All a Big Clockwork": Achieving Dependable Systems in Practice. November, 2002

29. M. Hartswood, R. Procter, M. Rouncefield, R. Slack, A. Voß, R. Williams. "It's All a Big Clockwork": Standards Selection and Appropriation in Manufacturing. European Association for the Study of Science and Technology (EASST) Conference. June, 2002.

30. A. Voß, H. Jørgensen, T. Herrmann, R. Procter. Structure and Process: the interaction of routine and informed action. Workshop at ECSCW'01. 2001.

31. A. Voß, R. Procter, R. Williams. "Being There, Doing IT": from User-centred to User-led Development. Poster presented at Mensch & Computer 2001. March 5th-8th, 2001.

32. A. Voß, R. Procter, R. Williams. A Social Learning Perspective on Processes of Change. Workshop on Beyond Workflow Management: Supporting Dynamic Organisational Processes, ACM Conference on Computer Supported Cooperative Work. November, 2000.