Co-realisation

Towards a principled synthesis of ethnomethodology and participatory design

Mark Hartswood[1], Rob Procter[1], Roger Slack[1], Alex Voß[1], Monika Büscher[2], Mark Rouncefield[3], Philippe Rouchy[4]

[1] School of Informatics, University of Edinburgh, mjh@inf.ed.ac.uk
[2] Department of Sociology, Lancaster University
[3] Department of Computing, Lancaster University
[4] Department of Human Work Science, Blekinge Institute of Technology

Abstract

This paper calls for a re-specification of IT systems design and development practice as co-realisation. Co-realisation is an orientation to technology production that develops out of a principled synthesis of ethnomethodology and participatory design. It moves the locus of design and development activities into workplace settings where technologies will be used. Through examples drawn from case studies of IT projects, we show how co-realisation, with its stress on design-in-use and the longitudinal involvement by IT professionals in the 'lived work' of users, helps to create uniquely adequate, accountable solutions to the problems of IT-organisational integration.

Keywords

Ethnomethodology, participatory design, design-in-use, co-realisation

Introduction

As IT systems become steadily more organisationally embedded, the challenge faced by IT designers and developers is to understand the social relations of the workplace and their implications for systems design. The search for methodological innovations and enhancements that might deliver this understanding has yielded some promising results. Of these, ethnomethodology and participatory design seem to us to have been the most valuable. Ethnomethodologically-informed ethnographic studies of work practices (e.g., Heath and Luff, 2000) have been used to inform IT systems design about the social character of work (e.g., Button, 2000). Participatory design (e.g., Greenbaum and Kyng, 1991), in contrast, has been instrumental in promoting the value of user expertise for design, and the cause of user involvement and control in IT projects.

Despite these important contributions, it seems to us that ethnomethodology – at least as it has been applied to IT systems design – and participatory design talk past each other, with the result that their full potential has not been realised. Both have been used, in effect, as 'patches' for more fundamental problems in IT design and development practice. In this paper, we call for a principled synthesis of ethnomethodology and participatory design, a radical re-specification of IT systems design and development practice as inter-subjectively constituted, lived experience. The essence of our proposal is that IT system design and development practices should be organised as a *co-realisation* by users and IT professionals. A fundamental aim of co-realisation is to break down boundaries both within technology production and between technology production and use (Suchman, this volume).

We begin by reviewing the contributions of ethnomethodology and participatory design to current IT systems design and development practice and their limitations as solutions for its deficiencies. We then set out what we mean by co-realisation and then briefly summarise our experiences so far of following co-realisation through in two different case studies. Finally, we discuss the implications of our experiences for co-realisation in practice, including its relevance to, and suitability for, large-scale IT projects.

Ethnomethodology and IT systems design

The question of how best to incorporate ethnomethodological analyses of work and technology into IT systems design and development processes remains a matter of ongoing debate (see, e.g., Button, 2000; Dourish and Button, 1998; Hughes *et al*., 2000). While such analyses have done much to explicate the social character of work and to explain why IT systems have often failed to achieve their goals, using ethnomethodology as an input into design presents an altogether different order of problem.

At its simplest, ethnomethodology provides – via ethnographies of workplaces – an informational input into the design process, making visible the 'real world' aspects of a workplace *as it exists*; focusing upon the specific and detailed organisation of activities which IT designers are concerned to understand, analyse and reconstruct. The use of ethnomethodological analysis to 'socially enrich' requirements capture is generally perceived as valuable in identifying the exceptions, contradictions and contingencies of work activities that do not necessarily figure in more formal representations of work. However, any attempt to use ethnomethodology as a tool for 'inventing the future' must inevitably fall foul of what Dourish and Button (1998) have called the "paradox of ethnomethodologically-informed design". The full implications of a new system for work practices cannot be grasped by studying the work as it is now, but will only be revealed in and through the system's subsequent use.

We argue that the solution to this paradox is to follow up Button's (2000) suggestion "... that ethnography can be trailed into the world of design in a harder fashion than our enthusiasm currently permits." (*op. cit.*, p. 330) In particular, this calls for creating a shared practice between users and IT professionals that is grounded in the lived experience of users. It is time, as Grudin (1990) has observed, for IT system designers to move beyond a narrowly conceived engineering mentality to attend to the

lived realities of being a user in an organisational setting.

Participatory design

The origins of what is known today as participatory design lie in the so-called 'Scandinavian' approach to systems design and its goal of increasing worker involvement in processes of technical change and innovation (Greenbaum and Kyng, 1991). Today, while its political agenda is no longer strongly emphasised, participatory design is widely used as a means to improve IT systems design by encouraging shared practice between users and designers. The problem as we see it is that participatory design does not follow this participation through to its logical – and necessary – conclusion. With few exceptions, the focus within participatory design projects seldom moves beyond the design phase or the construction of early prototypes, and onto development and use (Dittrich, 1998)

To facilitate shared practice, participatory design has built up an impressive variety of representations of technology. Kyng (1995), for example, describes the use of mock-ups and exploratory prototypes in design work, stressing the value of 'low tech' representations such as pencil and paper as ways of ensuring that they can genuinely become 'users' objects'. Building shared practice, however, is not only about designers getting to grips with users' needs, but also users grasping a sense of what technical work is, including what is, and what is not possible. "... end users have difficulties in understanding the space of possibilities and limitations for changing the system being designed, including difficulties in distinguishing the simple, the complex, and the impossible." (*op. cit.*, p. 49)

Participatory design attempts to avoid falling foul of its equivalent of the ethnomethodology paradox by envisioning how the new system will actually be used in practice. Here, a new form of representation is deployed, the scenario. As Kyng notes "... a scenario describes how the computer support being developed may improve upon the relevant work situations." (*op. cit.*, p. 53) However, reliance on scenarios raises problems, as Kyng acknowledges. "Particularly when the prototyping environment is the same

as the implementation environment ... maintaining a shared understanding of what is representational, what is coincidental, and what is actual becomes difficult." (*op. cit.*, p. 54-5)

The point is that while representations and scenarios are valuable tools for supporting shared practice, we should recognise their limitations (on this point see Bowers, 1991; Suchman, 1995). The system itself is more valuable than any representation – even prototypes – can be for communicating its scope and behaviour. Implementation – when users apply the system in their work – is the only way to find out how users actually make use of it. The issue is the failure of participatory design to take an interest in use, where "... the design practice and the designed-for practice are *not separated* in time or space." (Bowers, 1991, p. 347) Despite participatory design's championing of user expertise and control, at the very point where this becomes most valuable to design, and users have the opportunity to drive the process, the user-designer relationship is terminated. We must conclude that, despite its declared intentions, participatory design continues to privilege the role and expertise of IT professionals over that of users.

Contextual design

Before we describe co-realisation in detail, it would be helpful to outline the differences between co-realisation and superficially similar approaches. Contextual design (Beyer and Holtzblatt, 1998) has been advanced as a solution to the question of how to bring ethnography and participatory design together. Beyer and Holtzblatt define contextual design as: "A set of techniques to be used in a customer centred 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, quoted in Preece *et al*., 2002, p. 313). Beyer and Holtzblatt list the steps of contextual design as follows:

- Contextual inquiry – talking to people as they do their work
- Interpretation and modelling with cross-functional teams
- Consolidation of information gained through previous steps

- Visioning about work practices and the development of storyboards
- User environment design – using storyboards to develop 'a software floor plan (that drives) the user interface design'.

From Beyer and Holtzblatt's comments, it would appear that contextual design developed out of a concern with usability and the work of participatory designers such as Kyng and Ehn. It would also appear that one of the motivations was dissatisfaction with "all this qualitative stuff" (Holtzblatt quoted in Preece *et al.*, 2002, p. 314) in terms of how it came to be sidelined. It would seem that contextual design attempts to blend qualitative approaches (fieldwork) with more traditional process vocabularies and object-oriented software design techniques.

It is important to acknowledge that contextual design does do some things correctly: most important among these is the stress on understanding context as a sine qua non of IT systems design. While the traditional 'over the wall' design methods have often been set up as a 'straw man' to enable favourable comparisons with the latest methodological advances, it is true that traditional methods are light on context and that traditional 'requirements specification' takes little regard of the context as defined by anyone outside of management. Thus, contextual design is an evident improvement over traditional methodologies and we would applaud taking users into account in the building of any system. Keeping the eyes of designers focused on the context is important, but contextual design, we argue, falls into the trap of offering what Button (2000) refers to as 'scenic descriptions' of work and its context, as opposed to understanding them. Contextual design is at the mercy of its own process models in that they reify the world in terms of data to be used *for design*. Such models seem to us a means of disengagement with the world in favour of some trans-situational 'ontology'.

While contextual design and co-realisation both take the delivery of usable IT systems as central, the manner in which co-realisation proceeds is radically different. We contend that what contextual design does is best seen as reshaping work practice as opposed to affording it. Holtzblatt and Beyer (1998) claim that: "The challenge is to move the design team and the client together to invent ways to improve the

work. The result will be to define new ways of working and the software systems that support them." (*op. cit.*, p. 72)

A method that takes the creation of work affording artefacts seriously cannot treat work practice as a deficient system in the manner described above. The development of 'uniquely adequate' (Garfinkel, 1967) solutions can only proceed from a commitment to learn from the setting – not to change it, but to afford its practical actions – and no amount of statements to the effect that context is central can be taken seriously if one "develops the details of business process redesign" (Beyer and Holtzblatt, 1998, p. 73). It would be a mistake, therefore, to interpret co-realisation as re-jigging contextual design in some way or another. IT systems and the work practices they are intended to support need to co-evolve, but contextual design never gets to grips with the 'lived' reality of being a user of the new system. Yet, we argue, it is precisely this, as IT systems and artefacts penetrate more deeply into organisations and work settings, that IT design and development methodologies must strive to achieve. This is what co-realisation sets out to do.

What we aim to achieve through co-realisation is a re-specification of IT systems design and development practice, and the social relations that underpin it, in a manner consistent with concepts advanced both in ethnomethodology and participatory design.

Co-realisation

As IT systems and artefacts penetrate more and more into working lives, the 'design problem' is not so much concerned with the creation of new technical artefacts as it is with their effective configuration and integration with work practices. The key issue for a re-specified IT design and development practice is therefore not only 'design', but also 'use'.

Despite the significant benefits that ethnomethodologically-informed ethnography and participatory design have brought to IT design and development practice, and contextual design's attempt to bring them together, the fact remains that user requirements that can only be identified in the context of, and through, use, are being lost. Through processes such as 'learning by doing' and 'learning by

interacting', users are able to experiment, share and appropriate the innovations of others, mobilising their collective resources to evolve systems, to continue 'design-in-use' (Williams *et al.*, 2000) In these ways, users do cope to some degree with the shortcomings of conventional IT design and development practice, fixing or learning how to work around the deficiencies created through the reliance on *a priori* design (Procter and Williams, 1996).

Co-realisation seeks to leverage users' efforts at addressing these deficiencies by supporting design-in-use. As Suchman (this volume) observes, "This agenda requires crossing boundaries both within technology production and between technology production and use ... If technologies are to be made useful, practitioners of other forms of work must effectively take up the work of design ... that is appropriating the technology so as to incorporate it into an existing material environment and set of practices."

Co-realisation's aim is to look 'beyond design' (Procter and Williams, 1996) as an activity identified with a specific phase of the IT system lifecycle to the means by which design emerges and evolves as part of the ongoing struggle of making *this particular system work for these particular users, in this particular workplace and at this particular time*. As Trigg, Blomberg and Suchman (1999) write: "... co-development of CSCW technologies ... means more than engaging prospective users in the design of new computer systems to support their work. It requires that we as designers engage in the unfolding performance of their work as well, co-developing a complex alignment among organisational concerns, unfolding trajectories of action, and new technological possibilities." (*op. cit.*, p. 349)

Co-realisation's re-specification of IT design and development is a principled synthesis of ethnomethodology and participatory design as a shared, situated practice involving users and IT professionals. This is grounded in the lived experience of users as they grapple with the problems of applying IT, appropriating its functionalities and affordances into their work practices and relations. Co-realisation involves: attending to the evaluation of technologies; appreciating the benefit of active user participation; adapting to a particular organisational setting; the explicit connection of studies of work and system design; and commitment to a 'long-term engagement'. Co-realisation is in accord with the vision of Trigg et al. (1999) of the project of cooperative design: "... an approach to the creation of more useful and useable computer artefacts ... the combination of envisioning, building and use ... as we work our way through successive rounds of trial and discovery regarding all of the ways in which the world is different than we had imagined it to be." (*op. cit.*, p. 348-9)

When we talk about a principled synthesis, we should make one caveat. We do not intend to suggest that ethnomethodology or participatory design have the status of toolkits from which one can take this or that as one chooses. Our aim is, rather, to suggest that the insights of ethnomethodological workplace studies are consonant with the partnership elements of participatory design. We argue that taking the 'participation' of participatory design seriously calls for adopting an ethnomethodological mentality as opposed to some other 'professional' sociological mentality. We reject the latter because professional sociology's project of remedying members' accounts and practical actions is profoundly opposed to what we want to achieve. Put simply, one cannot build systems that support work practice if one seeks to re-specify that work practice in theoretically driven ways. It is only through becoming a member, to use Garfinkel's (1967) term, and being accountable, that IT professionals can deliver uniquely adequate, 'work-affording' artefacts. This is simply not possible under just any ethnographic rubric – this displays a false sense of some unified ethnographic terrain outside ethnomethodology – since becoming a member is only possible using the study policies of ethnomethodology.

In co-realisation, IT professionals attend to the specifics of the workplace rather than have representations of work brought to them, since "Working practice is lived experience, only partially representable." (Suchman, 1995, p. 60) Co-realisation therefore renders debates about how to bring ethnographic accounts of work back from the field and make them useful to the designer irrelevant: instead, the designer goes

into the field, to the user and to the work.

Participatory design is usually associated with 'taking sides' and the pursuit of a distinctive political agenda.[1] While this is not a defining principle of co-realisation, neither does co-realisation assume that design can be pursued unencumbered by the politics of the workplace, or, indeed, that IT professionals should not take sides. Co-realisation's approach is not to intervene in the politics of design as participatory design has tended to do with a pre-defined programme. Rather, co-realisation takes the view that politics is a members' matter to be worked through in context.

In conventional IT systems design and development practice, no matter how much attention IT professionals pay to user participation, maintaining a dialogue with users when the balance of technical work shifts from design to development is difficult, if not impossible. The locus of the designer's work moves away from the users' workplace, hampering or eliminating opportunities for informal interaction. Moreover, when users' work is unpredictable and demanding, attempts to arrange meetings are likely to be frustrated by the exigencies of unfolding events. Co-realisation's goal methodologically is to move from intermittent and over-formalised participation to a situation where informal interaction between users and IT professionals becomes a part of everyday experience and the basis for the constitution of a shared practice. (This does not mean that we have abandoned the meeting or workshop as a vehicle of user-IT facilitator communication, only that we have recognised their limitations as settings for interaction.)

Co-realisation's insistence on maintaining dialogue between users and IT professionals, and putting users in control, can only be afforded by IT professionals 'being there' in the workplace. As Suchman (this volume) states, this requires "… that system developers become responsible for locating themselves within the extended networks of sociomaterial relations and forms of work that constitute technical systems." It calls for IT professionals to shift the technical work of design and development into the users' workplace, if not completely, then at least routinely and over sustained periods of time.

Through *being there*, co-realisation's aim is to achieve a situation where users and IT professionals can spontaneously shift their attention between the different phases of the system/artefact lifecycle, even to the extent that these cease to exist as distinct and separable activities. Co-realisation means that IT professionals must help users *realise* their needs by playing the role of 'facilitator' in the broadest sense of the term.[2] This involves *inter alia* acting as design consultant, developer, technician, trouble-shooter and handy-person. To support user-led innovation and design-in-use effectively, IT facilitators need to be able to shift their efforts smoothly between these various tasks as circumstances dictate.

The emphasis in co-realisation is on tightly coupled, 'lightweight' design, construction and evaluation techniques. Co-realisation seeks to bring about a context for IT design and development work where, as Buscher, Mogensen and Shapiro (1996) have put it, "… effort shifts fairly smoothly between implementing or adjusting previously decided possibilities, picking up on the host of small problems that arise during work, coping with the unanticipated consequences of previous actions, talking to individuals …" (*op. cit.*, p. 155)

Of course, crossing the boundary between IT production and use requires being there to be formulated and organised in a way that is compatible with the performance of technical work. We have been exploring what this means as a practical matter and the prospects for the co-realisation of IT systems. Below, we present case studies of doing co-realisation in two quite different settings.

The toxicology ward

The first of our case studies of co-realisation is set in the busy toxicology ward of a large hospital (Hartswood *et al.,* 2000). The toxicology ward provides a specialised inpatient service that allows for joint medical and psychiatric assessment of patients following a suspected self-harm incident, the majority of which involve an overdose of prescribed or 'street' drugs. One of its functions is patient 'disposal', i.e., determining the need for further

psychiatric and social care, and referring patients on as appropriate to other services. This work is carried out by the psychiatric assessment team.

Psychiatric assessment team members work with a high admission rate and patients can re-present with further episodes of self-harm, sometimes over a time scale measured in days or even hours. Follow-up care arranged immediately following admission is perceived to be crucial for patients who are often in crisis, and the lack of continuity and repetition that results if details of the previous assessment are not immediately at hand is perceived to be unsatisfactory.

Becoming a member

The case study began with a six-month period of familiarisation with toxicology ward work practices through sustained observation of the setting. The aim was to build relationships and understanding, to become a competent member in the setting, i.e., to become acquainted with 'the way we do things around here'. This is, as we argued earlier, an essential predicate for taking on the role of IT facilitator.

In the manner of more conventional participatory design projects, project work began with a series of group meetings, supplemented when their schedules allowed, by meetings with individuals. These centred on the discussion of a series of potential IT applications, including a resource database of information about services and contact details of other services involved in patients' care, the use of speech recognition for producing discharge letters, and a minimal electronic patient record system that could be used to recall basic information about previous admissions. Technically, the plan was to make extensive use of various off-the-shelf, configurable components, 'information appliances' that could be easily and rapidly customised to create new systems and tools for evaluation, experimentation and use.

Members were interested in investigating the possibilities of using 'off the shelf' speech recognition system for overcoming problems in the generation of discharge letters. At that time, these were produced by members' taped dictation and subsequent audio transcription by

ward secretaries. Limited secretarial resources acted as a bottleneck in this process. Discharge letters serve a dual purpose – to inform primary care providers of the admission and outcomes, and as a record of admissions for ward staff. The interest in the speech recognition system extended to considering it as a 'front-end' technology for a variety of different applications, such as, for example, a medical records system. Over time, through repeated cycles of discussion, proposal and review, additional 'problems' or 'requirements' were identified. While space does not permit us to discuss the project *in toto* here, the project also looked at supporting recording practices, record keeping and contacting other professionals and organisations. For an account of these other project components, see Hartswood *et al*. (2000).

After discussion with psychiatric assessment team members, it was agreed to use the speech recognition system initially for the production of transfer letters, which are written when a patient is to be transferred to another hospital for continuing care, rather than being discharged. It was agreed because these *handwritten* letters represented a small and well-defined subset of the letters produced by the psychiatric assessment team. It was also anticipated that there could be problems in complying with the hospital medical records department's requirement that routine discharge letters be archived on the hospital's patient administration system. With transfer letters, this issue would not arise.

In the following sections, we use extracts from fieldwork data (in *italics*) to illustrate how the speech recognition system evolved in use as a co-realisation of members and the IT facilitator.[3]

Adoption

One lesson quickly drawn following the implementation of the speech recognition system is that psychiatric assessment team members viewed the facility as a resource to be drawn upon in a manner dictated by the contingent demands of the work, rather than a piece of technology under evaluation with prescribed limits set on its use. In the following example, the consultant psychiatrist, who had undertaken a training session so that the system

might recognise his voice, was keen to make use of the system prior to the technical facilitator's belief that the system was 'ready':

Immediately on arrival, a CP asks me whether "we are ready" to use the speech recognition system to do transfer letters. I express uncertainty and explain that we are not quite ready for this saying that have not sorted out how to do printouts on headed paper. CP says that he is not bothered or that this does not matter.

In the following example, the speech recognition system has been used to produce transfer letters arising from 'ward referrals' – a caseload usually held to be distinct from the self-harm patients seen in the toxicology ward and one dealt with in a different way.

PSHO1 asks PSHO2 if the disk he has placed in the computer was the one on the desk - says that he already has a letter on it. Says that it was for a patient on the wards – that it was a psycho geriatric transfer letter – wonders where it should go.
PSHO2: suggests that it should go to the ward secretary.
MH: "Do you keep your own records up there?"
PSHO1: "I don't know – I think they do."

In opening up this novel use of the system, the psychiatric assessment team member also produces the problem of what to do next – there being no organisationally established mechanisms for passing on the electronic version of the letter to the appropriate person. This illustrates how, within this organisational setting, the adoption of a system for a particular purpose often has wider consequences, which the IT facilitator may be called upon to play a role in identifying and formulating.

Affordances of the speech recognition system are discovered and drawn upon in a seen but unnoticed manner, with little surprise at their discovery. The system just happens to be at hand to fulfil a need that has emerged out of the work of the moment and is used to meet that need. There are obviously prescribed uses of the system given its technical and organisational maturity (i.e., uses that it is not ready for, given its current state). However, whereas in traditional IT systems design and development, uses are legislated and bounded, the point of co-realisation is to treat these boundaries as movable, given time, and as potentially generative of new design ideas as opposed to a 'closed road'.

Facilitation

Almost inevitably, the IT facilitator develops a more complete understanding of the deployed technologies, including the problems and pitfalls arising in use, than do psychiatric assessment team members. Not only does the IT facilitator assemble the technologies, the technologies are the main focus of his daily activity, and a focus for his observations of and interactions with psychiatric assessment team members. Furthermore, the IT facilitator will seek to keep up with relevant technological developments. So, while the technologies are a constant factor in the life of the IT facilitator, this is not necessarily the situation for psychiatric assessment team members. The degree of familiarity with the system will vary between its components and between psychiatric assessment team members.

This reveals the differing foci of the participants: letter composition is the main focus for the psychiatric assessment team member, whereas the relationship between the user and the technology is at the forefront of the IT facilitator's mind. So, while the psychiatric assessment team member is focused on the content of the letter, the main focus for the IT facilitator is the interaction with the technology. Members' interest in technology generally goes as far as using it to get the job done. While they might suggest how it can be used to get a job done, this is the limit of their *immediate* interest. In general (and so long as it operates properly) technology is, for members, a scenic feature of the workplace. The job of reflecting on such accounts and on facilitating work practice in general falls to the IT facilitator.

This focus enables the generation of a corpus of understanding comprising each member's specific experience with using the system that can feed into various forms of documentation, crib sheets, and advice to users in a dynamic and incremental way. Various sorts of documentation have been produced in this manner, for example, an instruction manual, notes and reminders attached to the computer

itself, and 'background image' placed on the desktop indicating what the various icons on the desktop are for and how they should be used to produce a letter using the speech recognition system.

Outcomes

The adoption of speech recognition has not been uniform amongst psychiatric assessment team members. Some (for example, those that had a strong accent which the system had particular difficulty recognising accurately) did not routinely make use of the speech recognition system from the beginning – preferring instead to continue writing transfer letters by hand. Many initially made use of the speech recognition system but then switched to typing transfer letters. A small number of psychiatric assessment team members persisted with the system. This emerging pattern of usage contrasted with our initial presumptions that psychiatric assessment team members would be resistant to data entry by using a computer keyboard. In the following extract, a psychiatric junior house officer comments on the relative merits of using the speech recognition system, typing and handwriting letters:

The PJHO had not used the speech recognition system yesterday, but had typed in the letter. Today, I asked him why he had not used the system. He says that he prefers typing to using the speech recognition system. I ask if he prefers typing to handwriting – "yes, definitely". Says that it "looks better – my handwriting isn't very good". Says also that it is "more courteous – you can't always get the information out (of a handwritten letter)" – "If I receive a letter I prefer it to be typed" – "It's nicer" – "It's easier to have the sentence in your mind and to type it than have a sentence in your mind and speak it" – "It would be easier if you could speak at your normal rate rather than at one word at a time"- "too much hassle" – "It took too much time – I did this letter in 20 minutes and it would have taken me more than 20 minutes to use the voice recognition." He says that he thinks that the speech recognition system is good – that in a few years time it would be good if it could dictate at the speed you can think of a sentence – it would be good to have computers without a keyboard.

While not losing sight of the vision of a 'computer without a keyboard', the psychiatric junior house officer spells out a number of practical reasons for his preference for typing – that it remains "faster" to type, that using the speech recognition system is "too much hassle". Here, his comments are concerned with the mechanics of letter production. In contrast, his comparison of typed and handwritten letters orientate to the letter's presentation – that a typed letter is more legible, "nicer", that he prefers to get a typed letter, that it is "more courteous". Thus, he attends to what is professionally expectable in an adequately produced transfer letter as a warrant for its production by typing rather than handwriting.

Through experiences of using the system, a pattern of use is established that complements the affordances of the system, the skills of the users, the demands of their professional status, and the contingencies and demands of the work. It would be a mistake to judge this co-realisation effort on the strength of how the speech recognition system itself is used. The system is but one component of a wider system that includes a word processor, a bespoke Word letter template, the transfer of disks to the ward secretaries – that is, an organisationally embedded and technically realised means of producing letters for various sorts of discharge outcomes. The goal of producing all letters using a speech recognition system has become a surrogate for the emergence of a system that enables the production of 'professionally adequate' letters in certain warrantable circumstances. The work that has gone into the production of a letter is often not apparent in the final polished version, enabling psychiatric assessment team members to appear "damn slick" as one member put it.

The decline in the use of the speech recognition system does not necessarily spell its demise within the project, it may yet find its place in the ensemble of components that have been developed, or are still under development. Work has recently started on an electronic medical records system for toxicology ward admissions, based around the psychiatric component of an existing paper-based record. In order to tie in with the paper records, and to avoid duplicating effort, a summary of the admission/assessment

details is printed. One consultant psychiatrist proposed a novel use of this summary, suggesting that it could be sent to the patient's doctor as a predicate to a more detailed discharge letter. While such a solution may appear interim as compared to the speech recognition system's affording immediate dictation of the discharge letter, the consultant psychiatrist saw the solution as adequate for all practical and professional purposes.

It remains to be seen how various possible configurations of technology and practice will actually play out – what shape professional adequacy will take on with the adoption of an interim discharge report. Another consultant psychiatrist, when informed of this idea, responded:

CP: "To use this as an interim thing and still dictate the letters."
MH: "Yes."
CP: "What about the speech recognition?" The CP suggests that this could be printed out and then the speech recognition be used to add a free-text component. MH suggests typing as well.
CP: "This is almost there", she says, regarding the current printout – i.e., that there wouldn't be much more to add.

The 'printout' is considered to be "almost there" and implying that the step to becoming a professionally adequate discharge communication would only be a small one. Thus, one important aspect of design-in-use is recognising and supporting the innovative processes of adoption and reconfiguration to ensure those functions meet the demands of professional adequacy.

We suggest that innovation in a co-realisation environment depends on recognising these dual (and not always distinct) processes of refinement and opportunistic use and then building upon them. With its attention to detail, an analytic approach like ethnomethodology becomes an invaluable tool for the IT facilitator to topicalise or foreground members' practices such that they may be used as a source of discussion about requirements for alternative system configurations.

Crossing organisational boundaries

When we think about projects such as this, it is all too easy to treat them as if they exist in isolation, yet this would be problematic in that to have the fullest use their technical outputs have to be integrated into the fabric of other IT systems and organisational routines. This is not simply a matter of making the right connections and having the systems talk with each other, it is about integrating the routines afforded by the new system into the working division of labour within the wider organisation. There is a sense in which the co-realised project has to use what is at hand to achieve this connection.

During the project, a psychiatric assessment team member asked the IT facilitator if the system could be used for writing urgent non-referral letters. The following fieldwork extract illustrates the problem:

A Senior House Officer asks if she could use the speech recognition system for a letter that she has to dictate. I ask if it is a transfer letter. She said that it was an urgent referral letter that she wanted to fax through to the organisation she had referred the patient to. I say that at the moment we are only doing actual transfer letters – that the office manager had spoken to me about this and the secretaries have been getting confused about how they should deal with non-transfer letters.

That the system could be used for the writing of these letters shows the need for the work of co-realisation, yet the point is that as far as the wider organisation was concerned, the system was being used to do things that had other precedents, thereby confusing the staff 'normally' charged with undertaking that work.

The problem the IT facilitator is referring to in the above example concerned the transfer of letters to the ward secretaries on floppy disk as a work-around to the hospital's requirements that all such letters should be archived on the hospital patient administration system. Now that an electronic version was available, the secretaries had themselves requested to receive the letters on disk so that they would not have to re-type them. However, the secretaries (through the office manager) had raised concerns about the quality of the letters (one letter in particular was practically illegible due to uncorrected speech recognition errors), how the letters were subsequently to be handled by them (should

copies be produced for the addressee, the GP, for the clinical notes, the toxicology ward records and so on?) and to what extent the assessment team members had done these tasks themselves. The secretaries pointed out that the system they have for printing out letters was slow, taking some minutes to produce each letter, creating a significant overhead for them if they were simply duplicating work already done.

The matter of the 'illegible letter' was dealt with: the letter was identified as an occasion where the saved to disk version did not include manual corrections. Part of the problem here was that secretaries had attempted to correct the letter themselves. A procedure was agreed upon for returning such letters to the relevant psychiatric assessment team members for correction if the problem subsequently arose. The office manager suggested a solution to the issue of deciding which copies of transfer letters are required: placing the disk in an envelope stamped with a grid with spaces for ticks to show whether copies for various recipients had been made, or were required, and passing this back to the secretaries.

Thus, the IT facilitator is concerned not only with the production of technology, but also with the work required to mesh the technology with the wider fabric of the organisation. It is only by being situated in the workplace that the IT facilitator can pick up such issues as a matter of routine and only through membership can the IT facilitator be ratified to take responsibility for organising and implementing a solution.

Here, we underline the importance of the social as well as the technical in and as a part of the work of co-realisation. In its widest sense, the work of co-realisation involves not only an appreciation of what is going on within the group of co-realisers, but within the wider framework of 'how we do things around here' – what people know and use. This is one reason for our insistence that co-realisation requires membership from the IT facilitator.

ENGINECo

EngineCo is a manufacturer of mass-produced, customised diesel engines. Work in the control room of EngineCo's manufacturing plant involves various tasks like monitoring the production process, adjusting parameters, translating between the production process and the work of various other plant staff (e.g., quality control), and being involved in continuous re-organisation and optimisation activities that are required to constantly match the plant's working to the exigencies of production (Voß *et al.,* 2001). Because of this mix of tasks, some of which require constant attention, there are few opportunities for control room workers to participate in IT systems design and development activities that are shaped along the more traditional lines of project work. Although the social relations in this setting are actually quite favourable, in that the company's IT staff are located on-site and communicate with control room workers on a regular basis, most of the design and development activities take place outside the control room and workers there do not play a role in them. The traditional break-off point between requirements analysis and design with all its attendant problems (e.g., a lack of responsiveness) is maintained.

Our activities in this setting aim at making IT design and development work visible to, and accessible for, the control room workers and involving them in these activities as much as is feasible (Voß *et al.*, 2000). As in the toxicology ward, actual design and development work was preceded by a period of familiarisation with the setting through observation, interviews, etc. At the time of writing, the IT facilitator maintains a sustained presence in the control room (currently about four days a week) and works on a number of systems that are being co-realised for control room work, most prominently an electronic shift book application.

Co-realisation of these systems is occasioned by the everyday activities in the control room. In one instance, the IT facilitator observed a worker's use of Internet Explorer to browse XML-based log files generated by a particular plant system. Since there was no mechanism in place for formatting the file for display, the data was quite difficult to read. The IT facilitator became interested in this problem and offered to try to come up with a solution that would display the same data in the form of a table. Using a combination of off-the-shelf components, including an XML parser and

Internet Explorer, and some bespoke elements in the form of Tcl/Tk scripts written by the IT facilitator, a solution in the form of a 'log file browser' tool was created within a single day. Though the solution was far from perfect, it nevertheless allowed workers to look at the data in a much easier to read format.

But this is not the important point here. Far more important is that this quick-and-dirty solution occasioned a discussion (involving control room workers, the IT facilitator, and other IT professionals) about the general usefulness of such an application, possible extensions of it, of how this would mesh in with working practices and what the effort/benefit tradeoffs might look like.

The log files are routinely used to trace the trajectory of individual engines, or to trace occurrences of a particular error or problem situation. An extension of the log file browser tool was created over the course of the next days that allowed workers to search for occurrences of error codes and messages, or to find all engines that had been worked on, on a particular day. The design and development work took place within the control room, leading to many discussions about what the system should look like and how it would be worked with.

In and through doing the design and development work, various kinds of design choices became apparent and by being there, these could be explored in close cooperation with users. Importantly, possible tradeoffs and shortcuts were discussed and negotiated in context, and they were immediately put to the test by applying the system within the actual work setting. An example is the browser search function, which does not support the formulation of queries of arbitrary logical form, but is restricted to a simple conjunction of instances to look for (e.g., status = "not ok" and error code = "4003"). It was determined through situated discussion and "tinkering with the system" that generic logical operators were not immediately needed (although they were seen to be generally useful), and thus a temporary trade-off was made between development costs and immediate benefits. Another discussion of effort/benefit evolved around the question of how often workers would have to deal with those log files. One of the IT staff said that she

thought that it would be needed in the future since the system writing the logs was under constant development. This points to the fact that co-realisation is part of a wider context of IT systems development and use in the organisation, and that it is ideally suited to cover the issues that are not addressed by more formalised processes of IT design.

The following fieldwork extract illustrates how IT systems development influences – and is influenced by – working practices.

Control Room Worker (CRW): "I'm having trouble with your […] tool. It doesn't display everything anymore."
IT Facilitator (AV): "How do you mean? Anymore? It doesn't show you stuff that it used to show?"
CRW opens an example.
CRW: "It shows only entries until the 23rd March but I worked on this engine today, there should be new log entries."
AV: "Let's look at the log file."
AV opens the corresponding log file in a text editor and scrolls to the end of it.
AV: "Hm. There are no new entries in it."
AV opens the 'open file' dialog to check the path of the log file that was opened. He sees that the data they are working on is from a backup directory.
AV: "That's backup data. That explains why the new entries are not in there."
CRW: "That's funny. That would mean that no one's made a backup for two months."
[…]
CRW: "Did you change the directory because of the problem with the tool when the engine that you look at is being programmed at the same time?"
AV: "I can't remember changing this, I always had that funny feeling that the tool might create trouble again. Maybe [one of the IT staff] changed it. Of course, there was always the idea to try and overcome that basic problem by selectively copying only new data and putting everything into a database. Maybe it's time to take to take up this idea again if the […] work is ongoing."

Because use of the browser tool had created difficulties in the production process (production equipment failed because log files could not be written when they were open for

reading), one of the company's IT staff changed the script to work on a backup copy rather than on the operational data. However, she did not create a new backup (of 2 GB worth of data) and did not tell the control room worker or the IT facilitator. After this was established, the IT facilitator, together with the IT staff member, started working on a scheme to do incremental backups of the operational data.

Clearly, co-realisation may call for the coordination of the activities of various actors and this coordination may require structures such as those one might find in more conventional IT systems design and development projects. However, the example above also illustrates that some problems of coordination are hard to foresee and that these contingencies are subject to repair as members go about their daily activities. Co-realisation does not in itself address these issues – it is not a methodology but an orientation – but methods for achieving practical coordination may be employed in the process if, and when, the need arises.

The shift book and its evolution
Since the case study began, the main focus of the IT facilitator's work has been the development of an electronic shift book application. Its development is influenced not only by practices in the control room, but by a wider range of issues regarding how people work in the plant and how the various IT systems are operated. Since the shift book is employed during the course of production, as opposed to office hours, and because it contains crucial information that may affect the overall reliability of the plant, reliability is a major concern.

At a very early stage of the project, the IT facilitator decided on a client-server architecture for the shift book. The original implementation plan, negotiated between the IT facilitator and local IT staff, was to use Lotus Domino as a back-end server, since Domino provides a rich API with functionality close to what was envisaged – at that time – to be needed for this kind of application. Equally important was the ready availability of a working infrastructure, in terms of the actual server machine and software. During the course of the development of the

initial version of the shift book, however, two factors emerged that were to force a re-think of the system architecture. First was that the Domino server is not under local control and that the IT staff who maintain and support it are not available outside office hours. Second, as the shift book was co-realised, it became clear that functionality was needed that was not provided by the Domino package.

As the shift book architecture consisted of a number of layers, with only relatively weak coupling between the client and the server, the IT facilitator offered to implement an alternate version of the server ('back-end') access layer. In the choice of server, the need to host the shift book application on a well-supported package under local control became the crucial factor in the decision to adopt a relational database solution. A number of relational database packages were discussed and MySQL identified as a readily available one which would not rule out a future shift to a commercial database package, and for the relative ease with which it could be grafted onto the existing substrate of IT structures and practices. The implementation of the new back-end access layer took about three weeks and after its completion the question of the server was revisited. The conclusion reached was that the MySQL option should be persisted with.[4]

This example shows how IT systems design and development is inevitably tied to the contingencies of the workplace and its wider (IT) environment. In this case, it was important that the system architecture afforded some flexibility regarding the back-end server. By creating a layered architecture with clear boundaries, premature closure was avoided and the shift book evolved with the growing understanding of the implications of its development and envisaged use. The understanding of the reliability requirements, and of the potential practices the shift book would be expected to support, was built through a series of interactions and there is no closure yet. On the contrary, new potential uses of the shift book are being discussed and it is being re-factored to accommodate these. All along the way, the environment in which design and development takes place changes as new possibilities are explored (e.g., moving to a

different system architecture) and surrounding systems change, creating new opportunities and risks.

Because in the initial phase, estimates of the reliability of the new system are difficult to make, it was decided to use the old shift book application (a simple Excel sheet) and the new one in parallel, printing off entries after every shift (as was standard practice with the old system).[5]

At this point, the IT facilitator had to leave for three weeks. He left local IT staff with instructions on how to make backup copies of the data in the shift book and a small number of procedures for dealing with troubles he expected. He promised the control room workers a keg of beer should the shift book work without fault for the time he was away. While he was away, workers kept him updated on his prospects of loosing the bet and after two weeks of fault-free operation of the new shift book, they decided to stop using the old one as a backup. When the IT facilitator came back to the control room and learned about this he commented: *"You trust the system more than I do!"* One of the control room workers replied that *"once you have the day's entries on paper nothing can happen. The data might be lost for the search function but we didn't have that with the old application anyway."*

The control room workers decided that it was quite safe for them to stop using the double-entry backup mechanism because they knew that the risks they ran were acceptable: any old entries would be available on the printed copies. The worst case would be for them to lose a shift's worth of entries and these could be reconstructed easily from other records available on that shift. The control room workers thus demonstrated that they were willing and able to take on the responsibility of using the system for their practical purposes without slavishly relying on professional advice. The IT facilitator, of course, needs to take this into consideration when further developing the system; he is now committed to deliver a certain degree of reliability.

Discussion

In both case studies, we found a multiplicity of ways in which new requirements can emerge.

We frequently observed that the recognition of defects and deficiencies arises from trying to use a system in the context of doing the work. When a member needs to 'get the job done' it is precisely then – when the options are foregrounded – that consideration will be given to the means of solving *this* problem, using *these* available resources. Particular artefacts and methods then become relevant to the members that were previously part of the unconsidered background of the workplace. The problem is made concrete and the contingencies associated with 'solving the problem' become recognisable. To this extent it is difficult to obtain details about requirements in the abstract in formal user/designer requirements prototyping exercises.

A tentative categorisation might be made as to the different ways requirements can emerge – be articulated or recognised as such – through the situated use of the implemented system: where defects or deficiencies emerge; where some aspect of the system is opportunistically used for some purpose other than for that which it was designed.

Where examples in the first category are associated with refinement of existing configurations, those in the second concern the possible emergence of novel configurations. One example of the latter occurred in the first case study when a psychiatric assessment team member was observed to copy a web page showing details of doctors in general practice and paste this into a Word document so that it could be printed out and given to a patient. The patient was not registered with a doctor and the psychiatric assessment team member was providing details to encourage the patient to do so. The IT facilitator asked the psychiatric assessment team member about this, resulting in the discovery that it was something the member had done on previous occasions and initiating a more general conversation about how printing might be better integrated across the application as a whole.

Membership

Experience to date reveals the role of the IT facilitator being reflexively tied to the ongoing process of dialogue with users. Thus so far, it includes aspects of 'operational support' and

'system maintenance' as well as system design and development. Through such interactions, we see the IT facilitator's role becoming redefined to include aspects of using the system, rather than simply designing and developing it. Our experience suggests that it is constructive for users to enlist and appropriate the IT facilitator's skills in these diverse ways, at least initially, or until users feel more comfortable and are able to be more self-sufficient.

The IT facilitator is required to reflect on how his or her role within the setting has evolved in relation to the goal of co-realising the technology that members need to become competent in using. Similarly, there is an onus on the IT facilitator to reflect on how expectations are produced and dealt with, and how this process might be managed to encourage, and help resolve, debates and differences of opinion about the system's requirements. Since many of the interactions between the IT facilitator and members are struck up spontaneously and opportunistically, there is a danger of the facilitator finding him or herself dealing with conflicts of opinion and interest. So far, instances of this have been few, but we may expect this to change as members come forward with more ideas. More formal interactions such as review meetings have a role to play here, but it must be the IT facilitator's responsibility at other times to articulate and make understandable the 'status quo', i.e., 'how things have come to be this way' when alternatives are proposed.

The interactions between IT facilitator and members range over many topics and serve multiple purposes; members make comments about the system, talk about what difficulties or troubles they have encountered; the IT facilitator seeks clarifications of remarks, informs members about new features and about features that are planned for implementation. Sometimes, talk moves onto issues of implementation as members try to gain an understanding of what is technically feasible, or the IT facilitator attempts to manage members' expectations of what is achievable in the short or longer term.

When we talk about the IT facilitator, we are not suggesting either that just anyone could do the job or that there is a need for special training.

The IT facilitator, as a member, has to have the commitment to listen and learn as much as suggest, and the consent space in which co-realisation takes place is contingent on that being the case, not only from the IT facilitator but also from other members. There is a sense of hybridity – of domain crossing – in taking this role but we would argue that there is a great deal of role and domain crossing to be done on all parts within co-realisation: in sum, the key attribute that such facilitators must possess is an ability to listen and learn in co-operation with fellow members.

Also, when we refer in following sections to the 'co-realisation team', it is important to note that we do not have an official ideology of how teams should be put together – the point is instead to find members who are interested in working with the IT facilitator as she or he learns about the setting. Members must then be prepared to work with (not for) the facilitator in order to develop work-affording artefacts. Members are required who will make a commitment to the project. Participation is not static; it shifts in and as a part of the working division of labour. Notions of who should and should not be involved are always preliminary: the unfolding project and participation are reflexively linked and worked through as thoroughgoingly practical matters.

The work of successful co-realisation inevitably entails building a shared practice between users and IT professionals that emerges from communication with the IT facilitator(s) and exploration of the technology itself. In the same way that the IT facilitator is unlikely to be able to undertake psychiatric examinations competently, it is also unlikely that, say, a consultant psychiatrist could take over development of the system. That said, moving design into the workplace affords a convergence of worlds centred on the production and use of technological artefacts.

Boundaries and bricolage

Co-realisation involves crossing the boundaries between the technical and non-technical. Observing these boundaries means that users generally receive very little support for their 'bricolage' work, i.e., effort spent in making 'the system' work. Co-realisation, in contrast,

foregrounds bricolage – the often *ad-hoc* and creative combination of materials at hand for a particular purpose (Buscher *et al*., 1996) – and leverages users' efforts. It also distributes the responsibilities for bricolage more evenly. The system becomes everyone's concern and the point is to work together to allow it to afford work as opposed to handing over all responsibility to users as happens when traditional boundaries between technology production and use are adhered to. New technologies are not bounded in this way. They can not be 'inserted' or 'slotted' into a dynamic and complex socio-technical system, but are, rather, themselves dynamic and open in a way that requires their being 'grafted' into an existing (changing) socio-technical substrate, becoming a part of its dynamic – in positive, but also potentially negative ways. Co-realisation is a way of acknowledging the risks and costs of this process, it so-to-speak takes the 'bull by its horns'. The process of facilitation-realisation is collaborative: the facilitator/bricoleur is able to show how to use the system while the members, having this support, are able to envisage more fully ways to integrate it into their everyday work tasks.

Commitment, risk and accountability

Relationships of mutual support and commitment are, therefore, an important component of co-realisation. When we look at a system in use it is obviously not perfect, there are drawbacks and sub-optimal elements, yet it is the role of the whole co-realisation team to discover and work around these so as to develop a technology that 'works' in the sense of becoming a working part of a stable and satisfactory state of a new working culture. In fact, it is only in and through use that we *can* discover drawbacks and sub-optimality – in other words, these are contexted matters. Within the co-realisation team they are also accountable matters. It is through the IT facilitator's continued presence in the workplace that problematic aspects of the system's use can be seen to be addressed – either through 'tweaking' the technology or through changing ways of working (even if only in a promissory way). Knowledge of technological potentials and risks, and knowledge of local practices begin to interpenetrate and show their interdependencies.

Through close proximity to the work being supported, the IT facilitator comes to appreciate the conditions and pressures of a particular workplace, and is thereby concerned that the technical interventions made should not pose a significant risk on those terms. In order to work in this way, the organisational exigencies of doing the work have to be a part of the facilitator's job. When, for example, a member of the psychiatric assessment team asks how a particular part of the system can or may be used, the IT facilitator's response must perforce embody some of the competencies of doing the work in this organisational context.

Our experience is that, as users gain familiarity with the system, they begin to request modifications to, or expansions of, the system to articulate more closely with aspects of their work. Our argument is that the competencies of users need to be considered over time as they develop and become more sophisticated in system use. The point is not simply that experienced users provide 'better' feedback, but that as users acquire certain competencies in using a given system, a range of design possibilities can emerge. As users become 'experienced' they develop new ways of using the system that in turn generate ideas for its further development. Rather than users simply adapting themselves to the new system, co-realisation stresses a change not only in the user, but also in their use of the system as a set of work practices evolve through use. Furthermore, we would argue that through this process users gain more general IT competencies and become better able to judge *inter alia* what is possible and what is not, what is simple and what takes time.

However, unlike the work of members, which is largely visible (or for reasons of accountability is often rendered visible), IT work remains somewhat opaque. This is undesirable if we want users to gain an understanding of the technology and of technical work (insofar as it impacts on their work and the development of work affording artefacts in their workplaces), and so be empowered. Thus, the IT facilitator must explore ways and opportunities to actively engage users – to explain what it is he or she is doing – in order to make his or her work understandable by, and accountable to, others.

Finally, co-realisation avoids the polarisation of outcomes of technological interventions into either 'successes' or 'failures'. In contrast, we draw a picture of a process predicated on situated, practical reasoning involving finding utility in planful assemblages of technology where ambitions, work practices and explorations of technological limitations and affordances jostle together, and are reflexively reshaped in order to accommodate one another.

Conclusions and further work

Co-realisation calls for a re-specification of IT design and development as a principled synthesis of ethnomethodology and participatory design. A system which embodies workplace specific knowledge and which has IT professionals responding to the practical exigencies of living with the system is likely to produce a more elegant solution to the problems of living with IT. Put most simply, co-realisation advocates taking engagement with users seriously, asking IT professionals to capitalise on the mundane and to 'stick around' and see what happens.

Inevitably, co-realisation presents a challenge to conventional presumptions about IT system design and development practice, and the division of labour. In particular:

- The division of labour within the organisation in making decisions about systems and routines, and
- The temporal and organisational division between technical experts and organisational users.

We believe that this calls for significant changes in the training of IT professionals, system managers and the wider workforce. We are not suggesting, however, that all IT professionals must train as ethnomethodologists (although it would be useful to see design courses foster some appreciation of its potentialities). What co-realisation does require is that IT professionals learn to attend to the mundane features of the workplace, to the seen but unnoticed ways that work goes on, and to what people there know and use to get that work done. By doing this, we argue, there would be a number of fundamental pay offs.

First, IT professionals would be able to appreciate the environment in which they work and in which their artefacts will find a place – this would lead to artefacts which support the work tasks for which they were designed in a enhanced manner. In other words, an understanding of the setting and work practices in which the artefact is situated would enhance the situated use of the artefact.

Second, there is the issue of technologies as 'configuring the user' (Woolgar, 1991). Through an acquaintance with the lived work of using systems for work tasks, the IT facilitator will be able produce an artefact that is as much configured by the workplace as it configures it. Linked to this is the idea of co-ownership of knowledge. In the case studies described above, neither the IT facilitator nor the users were sole 'owners' of the knowledge embodied in the evolving artefacts, rather the sense was one of an evolving co-ownership (for a discussion of the notion of claims to 'owning' knowledge see Sharrock, 1974). Such relations embody the understanding that no one expertise is of itself sufficient to develop the system. Working around divisions of labour and knowledge in this way elicits co-operation and ensures that work practice is reflected in design and development processes.

Third, it is in and through such an enhanced, long-term engagement that IT professionals become *accountable*. Co-realisation asks that IT professionals become more committed to the moral order of the workplace. This is the ethnomethodological character of co-realisation and it turns on the notion of membership. That is to say, the IT professional has to capitalise on what people know and use, not in the manner of 'professional' sociology, but in the way that members come to be vulgarly competent and thereby to know 'what goes on around here' and doing so, as Garfinkel (1967) says, " … "from within" actual settings, as ongoing accomplishments of those settings." (*op. cit.*, p. viii) The culture of design as a relatively isolated process (*inter alia* organisationally, temporally and spatially) must be replaced by accountable design, which for co-realisation means enabling the unfolding implications of technology for the workplace in which it is located.

IT professionals who undertake such a process will produce uniquely adequate artefacts that

support the work of users because they have become immersed in the lived work of those users. Co-realisation advocates that IT professionals must 'become users' in the sense that they have the knowledge and commitment to embed artefacts in workplace specific settings over time. IT professionals have to show commitment to users in that they stay and facilitate the unanticipated uses of the system, they stay and assist in embedding work arounds into the unfolding technology so that these become part of the system, as opposed to troubles. IT professionals committed to co-realisation do not hand over a black box to users and expect them to cope with its vagaries, their task is instead to act as intermediaries or brokers (see Williams *et al*., 2000) between the users and the system in terms of developing the artefact over time. It is only then that systems will support the workplaces in which they are located and artefacts afford users the opportunity to work with technology that is really 'in working order'.

This is not to say that the IT facilitator needs to go looking for some arcane or potentially out-of-the-ordinary solution: our experience is that the solution is often at hand within the setting itself – hence our earlier reference to bricolage – and that the IT facilitator can realise solutions based on technologies that are readily at hand and which do not of themselves require any training for members to use. Co-realisation means that people who know how to use mundane or banal artefacts to afford work are likely to be present, as with the example of the formatting issues around log files discussed above.

We are continuing to explore the prospects for co-realisation as the case studies unfold. For example, we expect the demands made upon the IT facilitator to escalate as different modes of facilitation: design consultant, technician, trouble-shooter and handyman are increasingly called into play. This is a demanding combination of roles and raises issues of skill repertoires and the possibilities of over-loading.

More significantly, while many may agree with co-realisation's aims, it might be argued that the nature of IT projects makes co-realisation impractical in all but a few situations. It might be said that while our case studies demonstrate that co-realisation is feasible for small, self-contained projects, we have not shown this to be the case for large-scale projects and large-scale systems; in other words, that we are simply proposing a way of 'tinkering at the margins' of IT systems and infrastructures.

This raises the question of co-realisation's relationship to other approaches to IT systems design and development. Co-realisation is an orientation to socio-technical systems design and development, it does specify general principles of how design and development should be done, such as membership and accountability, but it does not specify a particular set of methods to be used. There is, then, the question to what degree co-realisation is consonant with various systems design and development practices, to what degree software engineering methods can be integrated in co-realisation and to what degree the orientation can be taken up within more traditional system design and development methodologies.

In its call for a long engagement with the setting, the artefacts created and the ways in which they are used, co-realisation is indeed incommensurable with any methodologies that, in specifying a strictly phased approach to design, commit the fallacy of demanding that a system be comprehensively specified *a priori*, i.e., before it is actually implemented and used. It would thus seem that co-realisation is not applicable to contexts where the existence of a complete system is a precondition for any work in the setting (such as a production management system). Co-realisation builds systems gradually, assuming that even early versions will be used in production, so that experience can be gained to guide further development. However, once a system exists and needs to be appropriated and evolved in response to what people have learned using it, co-realisation is applicable, taking over where *a priori* design breaks off and leaves users to fend for themselves.

It is clearly important to test co-realisation on a larger scale, though what constitutes a 'large' system is an interesting question (and it remains open as to whether any particular methodology can claim to have 'solved' the problem of building such systems). Certainly, there are a number of dimensions to scale, i.e., large user

groups, multiple workplaces and technically large-scale, complex systems. Certainly, there are scale issues that our case studies do not touch upon. However, in both the speech recognition system and the shift book application, we have had to deal with a number of scale issues that do not have anything to do with scale in terms of lines of code or size of the project team, but which emerge from the extent of the larger socio-technical setting in which the systems are embedded. So, while the speech recognition system and the shift book application may not seem to be instances of particularly 'large' systems, they are far from being 'toy examples'.

There are a number of IT design and development practices with which co-realisation is consonant, and which can be applied as part of a co-realisation effort. Examples include thorough documentation, rigorous testing, code reviews, modularisation (in particular, in terms of separation of concerns and of infrastructure and application), source code control, etc. Co-realisation can make use of most of the supporting (computer based) tools that have been developed to facilitate systems development, including CASE tools. Since co-realisation needs to be concerned with the operational support for implemented systems, other practices would need to be employed as well, such as strategies and tools for configuration management and backup.

Many issues of scale boil down to the question of how IT systems design and development effort should be managed. In its pure form, co-realisation calls for the locus and control of IT systems design and development to be pushed out from the 'centre' and into the workplace. We concede that this is unlikely to fit with the management requirements of large teams and large-scale projects, but we believe that co-realisation can be adapted to the needs of IT design and development in-the-large without compromising its essence. We note with interest that so-called 'agile' software development methods (which bear interesting resemblances to co-realisation[6]) have also been dismissed as unsuitable for large-scale projects (e.g., Turk *et al.,* 2002), but that agile method practitioners dispute this (e.g., Cockburn, 2000).

A general template for co-realisation in-the-large might involve allocating some team members to IT facilitator roles and locating them within selected workplaces. This, of course, will require more attention be paid to coordination and communication within the team. Here, rotating team members between roles may help to ameliorate some problems. As a final point on methodology, we would observe that a general solution to the problem of scale is to adopt a framework for project management and coordination that allows for methodologies to be matched with project attributes, balancing, in Cockburn's (2000) phrase, "lightness with sufficiency". Questions of how co-realisation might be implemented in other projects and other settings must be worked out in ways that acknowledge the specifics of those projects and settings.

In EngineCo, our exploration of scale issues is currently oriented to developing mutual understanding and trust between the IT facilitator and EngineCo's IT staff. (This signals the emergence of yet another role for the IT facilitator: the capacity to act as an intermediary between the world of 'technology use' and the world of conventional 'technology design'.) In the healthcare case study, the installation of a new hospital information system will provide an opportunity to identify strategies for integrating localised, co-realisation efforts with large-scale organisational IT infrastructure development and management policies.

Finally, we observe that there are a number of factors that lead us to believe that our advocacy of co-realisation is timely. First, is the changing technical landscape of IT systems and artefacts – including the growing market for commodified, packaged solutions. Many 'user-level' technologies are now available in the form of generic components, opening up the possibilities for solutions that can be customised, configured and evolved on a 'pick and mix' basis. Given the right choice of technologies, the scope for IT systems design and development work as bricolage can be significantly increased.

Second, is the shift over the last ten years in large-scale systems procurement strategy towards commercial off-the-shelf (COTS) software packages created by designers for "... unknown populations of prospective users"

(Suchman, 1995, p. 33). Organisations face new challenges to select, assemble and configure solutions that are appropriate to their needs, and to reconfigure them as those needs change. COTS solutions are merely indicative of design issues *postponed*, not resolved. The problem is to ensure that the generic models of work embedded in COTS solutions are evolved in locally meaningful ways, as Suchman (1995) points out. We would argue that to achieve these aims, the practices of co-realisation described above are, or will, become necessary. In this way, COTS packages can actually be made to 'work'. So-called enterprise resource planning (ERP) Systems, for example, do not come out of the box ready to use in any organisational context or setting: thus we are led back to our agenda of co-realisation. Appropriation is inevitable and, we argue, co-realisation is the way to do it.[7]

To be sure, systems of any scale may be built without co-realisation – but, as we have argued, the work that co-realisation does is inevitable at some point and a lot of the problems can be avoided if IT systems design and development is built around the principles of co-realisation. That is not to say that co-realisation is a magic bullet, but that it provides a means to access what people know and use and how it impacts on IT systems; surely this understanding is worthy of consideration up front.

Acknowledgements

Notes

1. Though this seems to have been progressively muted as participatory design has become more mainstream.
2. Henceforth, we will use this term when referring to IT professionals engaged in co-realisation.
3. The following abbreviations are used throughout this case study to refer to members: CP: Consultant Psychiatrist; PSHO: Psychiatric Senior House Officer; PJHO: Psychiatric Junior House Officer; MH: The IT facilitator.
4. Since this conclusion was reached, it has become clear that MySQL is reliable enough to justify its continued use.
5. For reasons beyond the scope of this paper, a strategy of simply backing up the data was not adequate.
6. For example, both argue that the functionality delivered should only be what is needed; that time to delivery should be as short as possible; that functionality should accrete over time and track work practice.
7. Mechanisms to feed experience of 'configuration as (re-)design' back to COTS vendors are generally poor. This is a problem that needs to be addressed, but is beyond the scope of this paper.

References

Beyer, H. and Holtzblatt, K. *Contextual Design: Defining Customer-Centred Systems*, Morgan Kaufmann Publisher's, 1998.

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

Buscher, M., Mogensen, P., and Shapiro, D. "Bricolage as a Software Culture," in *Proceedings of the COSTA4 Workshop on Software Cultures,* I. Wagner (ed.), Technical University of Vienna, Vienna, December 1996.

Button, G. "The ethnographic tradition and design," *Design Studies* (21:4), 2000, pp. 319-332.

Cockburn, C. *Balancing Lightness with Sufficiency*, 2000. Available at http://members.aol.com/acockburn/papers/barelysufficient.htm Accessed 19 September 2002.

Dittrich, Y. *Developing a Language for Participation*, Research Report 18/98, Department of Computer Science and Business

Administration, University of Karlskrona/Ronneby. ISSN 1103-1581, 1998.

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

Garfinkel, H. *Studies in Ethnomethodology*, Englewood Cliffs, New Jersey. Prentice Hall, 1967.

Greenbaum, J. and Kyng, M. (eds.) *Design at Work: Cooperative Design of Computer Systems*. Lawrence Erlbaum Associates, Hillsdale, N.J, 1991.

Grudin, J. "The Computer Reaches Out," in *Proceedings of CHI'90*, Seattle WA, ACM Press, 1990, pp. 261-268.

Hartswood, M., Procter, R., Rouncefield, M. and Sharpe, M. "Being There and Doing IT in the Workplace: A Case Study of a Co-Development Approach in Healthcare" in *Proceedings of the Participatory Design Conference*, T. Cherkasky, J. Greenbaum, P. Mambrey, and J.K. Pors (eds.), CPSR, New York, 2000, pp. 96-105.

Heath, C. and Luff, P. *Technology in Action*, Cambridge University Press, 2000.

Hughes, J., O'Brien, J., Rodden, T. and Rouncefield, M. "Ethnography, Communication and Support for Design," in *Workplace Studies: Recovering Work Practice and Informing Systems Design*, C. Heath and P. Luff (eds.), Cambridge University Press, 2000, pp. 187-214.

Kyng, M. "Making Representations Work," Special issue of *CACM*, (38:9), 1995, pp. 46-55.

Preece, J., Rogers, Y. and Sharp, H. *Interaction Design*, Wiley, New York, 2002.

Procter, R. and Williams, R. "Beyond Design: Social Learning and CSCW – Some Lessons from Innovation Studies," in *The Design of CSCW and Groupware Systems*, D. Shapiro, M. Tauber and R. Traunmüller (eds.), Elsevier Science, 1996, pp. 445-463.

Sharrock, W. "On owning knowledge," in *Ethnomethodology*, R. Turner (ed.), 1974.

Suchman, L. "Representations of Work," Editorial, special issue of *CACM* (38:9), 1995, pp. 33-34.

Suchman, L. "Making Work Visible," Special issue of *CACM* (38:9), 1995, pp. 56-64.

Suchman, L. "Located Accountabilities in Technology Production" in *Scandinavian Journal of Information Systems,* this volume.

Trigg, R., Blomberg, J. and Suchman, L. "Moving document collections online: The evolution of a shared repository," in *Proceedings of ECSCW'99*, S. Bødker, M. Kyng, and K. Schmidt (eds.), Dordrecht. Kluwer, 1999, pp. 331-350.

Turk, D., France, R. and Rumpe, B. "Limitations of Agile Software Processes," in *Proceedings of Third International Conference on eXtreme Programming and Agile Processes in Software Engineering*, Italy, May 2002.

Voß, A., Procter, R. and Williams, R. "Innovation in Use: Interleaving day-to-day operation and systems development," in *Proceedings of the Participatory Design Conference*, T. Cherkasky, J. Greenbaum, P. Mambrey and J.K. Pors (eds.), CPSR, New York, 2000, pp. 192-201.

Voß, A., Procter, R., Slack, R., Hartswood, M., Williams, R. and Rouncefield, M. "Production Management as Ordinary Action: An Investigation of Situated, Resourceful Action," in *Production Planning and Control. Proceedings of 20th UK Planning and Scheduling (SIG) Workshop*, J. Levine (ed.), Edinburgh, December 2001, pp. 230-243.

Williams, R., Slack, R. and Stewart, J. *Social Learning in Multimedia, Final Report to European Commission*, DGXII TSER, University of Edinburgh, 2000.

Woolgar, S. "Configuring the user: the case of usability trials." in *A Sociology of Monsters*, J. Law, (ed.), Routledge, London, 1991, pp. 58-100.