An interest in the practical activities and interactions involved in the actual development of technology is pursued by Kathleen Jordan and Michael Lynch in their study of the construction and implementation of a new molecular biology technique. Although the technology addressed in this chapter is biotechnology, nevertheless the problems faced in exporting the technology off 'the drawing board' and out of the laboratory or workshop and into production are problems common to computer systems development, and the processes of implementation that Jordan and Lynch discuss are processes that are equally involved in the implementation of computer systems.

Jordan and Lynch are concerned with how an 'experimental tool changes into a commercial means of production'. They suggest that what is involved is what they call 'mainstreaming' the technology. They use this term in preference to a more common term 'diffusion of technology' that has been previously used to gloss processes of implementation. They do so in order to draw attention to the situated work through which an application is organised as, and made to be relevant for, a project, or an industry. Thus, mainstreaming 'is meant to draw attention more closely to the activities through which a new technology is promoted and shaped for adoption in different domains of activity, while the activities themselves undergo systematic transformations in the process'. Part of the implementation process can then be seen as the creation of organisational milieux in which the application is then promoted as relevant. Further implementation also involves the spinning off and dispersion of the technological techniques involved. Just a cursory speculation about the way in which personal computers have become dispersed throughout whole ranges of work, through the promotion and organisation of their local relevance for work previously not associated with computer systems, will testify to the general applicability of 'mainstreaming' as a way of capturing processes of implementation.

8

WORKING TOWARDS AGREEMENT

Wes Sharrock and Bob Anderson

In Chapter 1 some of the issues of basic sociological principle which differentiate the studies collected in it were examined. The general direction from which this chapter approaches its topic (and without much preliminary discussion of issues of strategy or otherwise necessary disclaimers about the modesty of its aims) is that of 'ethnomethodology'. As such, it does not see itself as contributing to a putative 'sociology of software' which derives its inspiration from recent 'strong programme'-influenced traditions in the 'sociology of knowledge'[1] but follows in a line of 'studies of work' carried out by Harold Garfinkel and his colleagues.[2] Measured against the intensely demanding standards which Garfinkel has recently been setting for those studies, particularly that of becoming proficient in the work being studied, the modesty of our current investigations – into the design and development of photocopying and printing equipment – should again be stated: there is no prospect that we might acquire skills relevant to designing and building the software or hardware that goes into such equipment. This means that four studies provide for ethnographically informed conjectures about the work activities of those under study. Those studies so far are of design and development work on three projects being carried out at a UK site of the development and manufacturing division of a multinational corporation, and they have been routine observational inquiries, involving us in frequenting scenes of work activity and attending to (and recording) what goes on.

Though there will be no *extensive* theoretical/methodological preliminaries there will be some brief consideration of them, intendedly sufficient to orientate readers unfamiliar with the sociological meaning of our approach to the substantive materials discussed in some detail below. Those materials are intended to display the activities of a small group of software engineers in the midst of a meeting during which they are trying to work out why two interfacing software systems do not appear to be interacting properly. The examination of these materials represents very

the work that is involved in carrying through an engineering project. These preparatory remarks are needed to indicate that what follows is not intended to embody a (sociologically) theorised version of the nature of technological artifacts (be they hardware or software). Ethnomethodology has persistently been interested in the study of social phenomena as they can be found *prior* to the point at which they are subjected to reconceptualisation in terms of the (postulated) requirements of one or other of sociology's theoretical and methodological constructions. It has been interested, in a way that other sociologies have not been, in seeking to determine just what are, in the way of phenomena that, so to speak, 'anyone can find', the very phenomena that the other sociological strategies are (intendedly and indispensably) talking about. For example, amongst the proposed routes for a 'sociology of software' is one which aims to provide a 'labour process' analysis of it.[3] Our comments here are not critical of nor do they state any principled objections to the 'labour process' approach, but enable only the expression of a problem which 'the labour process' approach (or any of its direct rivals) present for us. The suitability of talking about a 'labour process' is dependent upon the invocation of a collection of pre-given *sociological* presuppositions *within which* it makes sense to talk of the work people are doing in such terms as the 'working of a labour process', with the claim being, of course, that the invocation of a vocabulary of 'labour process' provides the most correct and informative way to talk about the intended phenomena. We reiterate, the adoption, defence, and recommendation of those presuppositions is not at issue here, where the relevant question is: to talk about what? The phenomena which are to be written about in 'labour process' terms are ones which already possess a meaning, which are already assigned a definite sense within the world of daily life within which those activities are conducted. Talk about 'the labour process' is plainly intended to discuss in abstract ways whatever it is that employees are doing at work, those things which *make up their work*, the 'on the job' activities which are themselves observable and describable prior to and entirely independently of any invocation of the vocabulary of 'labour process' analysis or any competing sociological scheme. It is, therefore, *pre-theoretically* recognisable day-to-day work activities which a scheme like 'labour process analysis' proposes to discuss, it being the purpose of that (and cognate schemes) to reconstruct the character, and, most specifically and crucially, the meaning, of those activities.

The 'labour process' vocabulary (or any alternative to it) is designed to provide a re-description of activities which are pre-theoretically describable, but just what pre-theoretical identifiable activities are being redescribed and just how the adoption of the vocabulary of re-description is intended methodically to transform whatever prior meaning they might have is something that ethnomethodology finds that standard sociological

organised social activities *after* they have been transformed through theoretical/methodological re-description that the sociological literature devotes itself. It is to the identity that activities possess prior to and independently of sociological reconstrual that ethnomethodology by contrast devotes itself, taking the pre-theoretically identifiable day-to-day work activities as constituting the grounding phenomena of all sociological inquiry.

There is always some point in attempting to guard against gross, even crass, misinterpretation of arguments, especially when they are stated in the perfunctory way in which they have been here. What we have said perhaps runs a significant risk of being misunderstood, so as a precaution we should perhaps affirm in the strongest terms that we do not regard ethnomethodology as *an alternative* to 'labour process analysis' or any of the other potential sociological conceptions of software, as aspiring towards the production of an apparatus of theory and method which will enable the sociological re-conceiving of people's activities. Ethnomethodology's exercise is of an utterly different order to that conceived by 'labour process analysis' and its cognates.

Accordingly, therefore, we have no independently 'sociological' conception of hardware and software. The hardware and software are projected outcomes of the work in hand, and our actual investigative concern is, therefore, simply with whatever work is in hand – recognising, of course, that the work is hand is typically that which is directed towards the production of the projected outcome, that which comprises operations in the (elaborately concerted activity) of designing and developing (in the instance we shall be looking at) laser printing equipment. Most minimally, then, we will be simply reporting on some work – comprising an attempt to identify and resolve a problem in the software design which happens to get done on a particular engineering project. This particular episode from the work life under observation does not single itself out for any special reason, only for the reason that *any* episode might, that it provides an occasion to consider some of the multifarious circumstances with which those involved in software development have to contend and the ways in which these were practically 'managed'.[4] After all, this episode is one of those which any putative 'sociology of software' must inevitably claim to subsume. The fact that the episode involves a group working towards agreement does provide a modicum of thematic continuity between discussion of it and the debates which have been continuing for some years now within the 'sociology of knowledge' (which has in essential respects been equated, of late, with the sociology of science and technology).

Michael Lynch's adaptation of the notion of 'achieved agreement' from the work of Harvey Sacks[5] provides our guideline, as it replaces the more usual sociological concern with 'agreement' as something to be determined in abstraction from the overt concerns of society's members with

that of examining occasions on which those members are engaged in explicitly attempting to bring about agreement. The software developers engaged in the exchange we present, describe, and discuss are quite overtly attempting to arrive at an agreement-in-so-many-words and we emphasise the situated character of such a quest, detailing (some of) the circumstances which provide them with the necessity for agreement and with the conditions under which agreement must be sought and brought about.

RECONSTRUCTING THE SOFTWARE

The following fragment of transcription provides a specific focus for our discussion; recording the exchange through which a small group of software engineers bring one phase of a protracted discussion to a close.

Jay: If we have 'out of paper' and 'duplex misfeed' is there any difference if the IOT or the ESS clears this
Mick: Before we do that although I agree that's the next logical step c'n we just test the understanding that what we have is a way of recovering from *out* of paper and a way of recovering from misfeed providing we do one or the other of these second columns.
Jay: Yeah's right. Is the conclusion then that in order to keep out of paper and misfeed operating in the *same* way in re in recovery in the same way in the ESS we only look at 'out of paper' when we attempt to feed from an empty tray
Stan: I don't know – I thought I heard from Sarah that the- now she understood where our out of paper was coming from you think that may be implementable
Sarah: No no I could (0.1)
Stan: because we would say we've got out of paper here
Jay: Hold on that that's correct but that does not keep 'out of paper' and 'misfeed' handled in the same way at all
Stan: That's right, true true. I
Sarah: ( )
Jay: My preference for that was that if we make a statement that we want to handle out of paper and misfeed in the same way in recovery then we do not not take out of paper when it goes empty only when we attempt to feed
Mick: that's why I asked the question
Jay: Yes
Mick: I don't know what we have agreed
Jay: I don't know if we're agreed, that was that was my understanding.
Sarah: do you want to handle both in the same way if you want this
Stan: I think it's totally up to the ESS to decide how they want to handle it
Sarah: ( )
Max: but this happens very rarely it being
Sarah: I agree
Max: more an issue of very rarely it it's relatively unimportant
Jay: I think that's a very good point so Max consequently I would think you'd like to recover in the same way so that the eh image manager maybe doesn't have to do one thing in one case and something else in another case eh
(0.5)
in any
Mick: Yep
Jay: in any event neither one of these two is the same as this
Sarah: Right
Mick: Agreed
Jay: Which everyone agrees to is so
Mick: So I think we can put a stake in the ground here and say that because out of paper is the *rare* event we should make that fall in line with the more common event which is misfeed and the only recoveries available to us from misfeed are ones which involve missed pitches. It really doesn't matter whether it's one missed pitch or we cycle out and cycle back up again as far as the communication is concerned. That's just an efficiency factor.
Jay: Having said that very nicely can you capture that and pass it on to Rick and Gordon so they'll understand why we're arriving where we are.

These transcribed remarks occur after some ninety minutes of discussion in a meeting that lasted some six hours and involved five software developers, mandated to tackle a problem with the interfacing of two software systems.

The problem was (for those involved) a problem produced by the distributed production of the software. Two of the participants (Jay and Sarah) were from one of the corporation's sites on the West Coast of the USA, the other three (Mick, Stan, and Max) were from the local UK site in England's south-east. The software was for a laser printer which was designed to support networked workstations; the UK software engineers were building the software which would govern the operations of the output terminal (typically spoken of as the IOT) that transferred the image

onto paper, whilst the US software operation was to develop an 'electronic sub-system' (typically called the ESS) that would generate the images and organise the printing jobs. Prototype machines had been built and a fleet of them were being put through tests at the UK site. At the time of this particular episode they were in a 'score test' phase. The machines were run under conditions intended to simulate routine use, with operators working their way through a series of prescribed 'jobs' and checking the machine's operations, recording all failures in considerable detail on a set of standardised forms. The testing involved the 'scoring' of identified problems against the various modules making up the machine (i.e. modules such as 'software', 'the stacker', 'the fuser', and so forth) and thus against the groups associated with the modules. The project itself was run on a 'management by problem solving' basis, which meant that all problems were catalogued on the computer and that project members were provided with 'problem lists' which effectively provided their work-load: they were required, that is, to work upon and solve the identified problems. Those problems were catalogued in terms of their seriousness and this identification provided, of course, a guide as to how tasks should be prioritised.

Whether or not the project was keeping to schedule was a persistent concern in all areas of its work and one of the ways in which the prospect of a schedule slip could be determined was through examination of the rates at which problems were being accumulated and cleared, i.e. whether they were continuing to accumulate new problems faster than they were clearing them. In the score testing phase, measures of machine performance relative to the standards which were set for exit from that phase also provided a basis for judging how the project was going. This, the 'Mersey' project, was failing to achieve the required standards and consequently suffering 'schedule slips'. The measures of performance were showing a substantial proportion of its current problems were attributable to the two software systems and it was at this point that the meeting which our transcript records was called; it was assumed that it was inadequate integration of the two software systems which was causing a significant proportion of the problems.

There were difficulties in the relations between project management in the two sites, with each (unsurprisingly) tending to blame the other. The test operations which were generating the fault reports were being conducted at the UK site and information about these was being routinely relayed to the US along with requests for corrective actions, but it was felt, at the UK end, that the US site was not recognising the seriousness and urgency of the situation and was failing to do necessary work. It was further felt that this was because the US team were not giving credence to the information from the UK and were discounting the problem reports they were receiving. It was against this background of reciprocal suspicion that it was agreed that two software engineers could be released from their work

at the US site to visit the UK site specifically for the purpose of tackling problems of software integration.

Jay came over a week before the recorded meeting in order that he could obtain firsthand experience of the test operations and of the problems that were being identified there, as well as to talk directly and at length with those involved in developing the IOT software and managing its incorporation into the working machines – all this with a view to preparing a diagnosis of what the problem was. Sarah arrived in the UK the evening before the meeting and arrived on the premises a few minutes before the scheduled 9 a.m. start of the meeting. Sarah was the writer of the ESS software that involved the critical interfacing with the IOT software. The other participants in the meeting were Max, who was Sarah's UK counterpart, Stan, who was Max's software manager, and Mick, who was handling the 'systems problems', i.e. those that arose from assembling the machine's constituent modules into a working whole.

After his preparatory investigations Jay had provisionally identified a source of the integration problems, 'the clear stack command' and 'all its ramifications'. The printing operations involved the relaying from the ESS to the IOT of instructions for the formation of the image to be imprinted, and the system developed allowed the 'stacking' of a small number of commands prior to their execution. However, as everyone knows, the operations of printing equipment are hardly fault free and sheets of paper can fail to feed at the required moment: the machine can run out of paper or sheets can, in various ways, misfeed. Such failures require the abortion of the job in progress and the abandonment of instructions which have been stacked, and Jay's proposal was that it was the position in the sequence at which the command to clear the stack was given that was pivotal. The 'ramifications' of this had to do particularly with 'job recovery', with building software sequences which would enable the aborted printing operation to be resumed once the paper fault had been cleared. If a print job was to be aborted, how was this being done: was a 'hard shutdown' called with the machine's operations being immediately halted or would a 'soft' shutdown be used to allow the machine to continue operating long enough to eliminate sheets of paper that would be superflous to resuming the print job at the precise point at which it had been abandoned? And how was that resumption achieved? What complications did the fact that a print job involved 'duplex' printing (i.e. printing on both sides of the paper) have for recovery? In just what order were images transferred to paper to ensure that what was printed on both sides of the sheet was the correspondingly numbered pages?

These were matters which had been examined in the ninety minutes of talk which had preceded the point at which the transcript begins as its participants sought to determine whether Jay's diagnosis was correct. Their talk comprised, then, an attempt to reconstruct the operations of the

software that had already been written and was implemented in the test machines, this operation taking place through their talking through the software operations, accompanied by the use of a flip chart to make a running record of the ongoing talk. Participants in the project typically manifest an intensely detailed knowledge of the features, operations, and ways of 'the machine', in the sense both that they are familiar with innumerable matters about 'the machine' as it figures in designs and specifications and that they are equivalently aware of the characteristics and performances of the prototypes in operation, down to the history of individual machines. This knowledge is one which they typically carry around 'in their heads', and against this background it is not, therefore, surprising that the parties set out to reconstruct the organisation of the software by talking it through rather than consulting records of its design.

The detailing of the sequences had revealed a discrepancy between the two groups with respect to the place in which 'out of paper' should be signalled from the IOT to the ESS. The IOT software had been written to call the fact that a paper tray had emptied at the point at which the fact that the last piece of paper had been taken from it was detected, but the ESS software was written on the assumption that this message would be sent not as the tray became empty, but at the moment at which there was an attempt to take paper from it, when there was an attempt to feed. The location of this divergence provided an opportunity for discussion of the possibility that symmetrical procedures could be followed in 'job recovery' for cases in which the paper tray had 'gone empty' and those in which the paper had, for some other reason, failed to feed, and it is towards the end of that discussion that the excerpt from our transcription begins.

Jay is advocating that a symmetrical procedure can be employed for recovery from both 'out of paper' and 'misfeed', and in the opening remark (lines 1–2) is canvassing a possible objection to his own proposal, that there might be a consequential difference as to which of the two software systems clears this.

Mick (lines 3–7) offers partial agreement with Jay, agreement that Jay has proposed a proper next step in working through the problem in hand, but provides an alternative proposal as to what the meeting's proper next step is, which is to ascertain what has already been agreed to, and Mick offers a characterisation of what that agreement is. Mick thereby initiates a sequence in which clarification of what has already been agreed to is sought.

Mick's characterisation of the agreed-to understanding is that there is a standard sequence of operations which can be carried out in 'job recovery' from the two different situations 'out of paper' and 'misfeed' which is portrayed on one of the flip charts (i.e. 'in one or other of these second columns'. Jay (lines 8–12) confirms this and elaborates the conclusion that he sees it to imply: that they should agree a standard policy for the point at

which 'out of paper' is signalled, which is when there is an attempt to feed from an empty tray.

This leads Stan (lines 13–15) to present a possible objection, though it is one which stands to be confirmed or rejected by Sarah, for it turns upon the understanding of what she had previously said. Stan has understood her as saying that now she has grasped the discrepancy between the ESS and IOT systems with respect to the place of the 'out of paper' signal in their respective sequences she could write software which would manage the discrepancy. Sarah's (line 16) response (which appears to disconfirm Stan's understanding of what she has previously said and to begin a restatement of what she could do) is overlapped by Stan's continuation (line 17) of his prior turn in which he begins to specify how 'we', i.e. the IOT software, call 'out of paper'.

Jay agrees (lines 18–19) with Stan's specification of how the IOT proceeds, but suggests that this is effectively missing the point at hand, which is to handle the 'out of paper' and 'misfeed' situations in the same way, and the point is conceded by Stan (line 20). Jay then (lines 22–5) reiterates in so many words the proposal that he is making which occasions Mick's tying back (in line 26) to his call for 'testing the understanding' (lines 3–7), and it seems as though Mick is saying that he had understood that this was what Jay was proposing but was uncertain as to how far others had agreed to this. Jay (lines 29–30) confirms that this is what he was advocating but does not know how far others have gone along with him.

Sarah (lines 31–2) attempts another intervention, one which begins as though it is going to spell out some of the practical consequences of any decision to adopt Jay's proposal, but it formulates this as a question to the others – 'do you want to handle both in the same way' – rather than as something for Sarah to decide herself. Stan's, reply (lines 33–4) disclaims responsibility for deciding this, returning this to Sarah and Jay – 'I think it's totally up to the ESS to decide how they want to handle it' – and Sarah's response is inaudible on the tape, but at this point Max, the software writer, intervenes (lines 36 and 38) in a way which intimates that argument may be unnecessary as the matter of the relative frequency of 'out of paper' and 'misfeed' events resolves the problem, a point which earns agreement from Sarah and from Jay who (lines 39–42) finds further support for his own position on the grounds that it provides the less complicated solution and therefore, perhaps, one which is 'easier to write'. Jay's contribution then draws expressions of agreement from Sarah and Mick and Mick then ventures (line 49) that the matter is settled and sets out (lines 50–6) in so many words what has been agreed to but also tying that back to what has been earlier determined, namely that there is no way of avoiding 'missed pitches'. 'Pitches' are the units in which the machine's belt moves and there is a recurrent concern with minimising the number of these which are unnecessarily made, and Mick is now drawing out that, having agreed

to handle recovery in a symmetrical way and having agreed to handle it according to the procedures for recovery from misfeed, then they are also accepting that there will be 'missed pitches'.

THE RELATIONSHIP BETWEEN THE SOCIAL AND THE TECHNICAL

This sketch of the situation of the software problem within the engineering project together with some cursory observations on even a brief episode from a relatively protracted meeting may seem altogether too much (irrelevant) detail, though its presentation is not, in fact, without purpose. The provision of even a modest amount of detail shows how the work of engineering (in this instance, software engineering) is replete with social organisational considerations.

Against the background of recent work in the sociology of science and technology where the relation of 'the social' to 'the technical' is problematic, not to say controversial, the pervading of the detail of an episode of 'technical work' by social organisational consideration is hardly an irrelevant fact. The social organisational matters which we have identified are not, it should be emphasised, ones which we, as sociologists, have decided are relevant to an understanding of technical work, but are ones which the participants in that work pointed out to us or drew to each other's attention as relevant to the understanding and further conduct of their work activities. The connection between 'the social' and 'the technical' was *made for us* by those we studied in the course of and as part of the carrying out of their work activities, and it is, therefore, as phenomena of 'the day's work' that we consider these relationships.

The fact that there was a problem was, for those involved, one of the explicable and predictable troubles of their work. It was a product of the division of labour on the project, of work-load and work flow. With two groups independently writing software (which would, in operation, require detailed communication between and close coordination of the two systems) but without close reciprocal monitoring of the detailed designs, the likelihood was that somewhere along the line there would be unnoticed discrepancies between the two, which would manifest only when the two softwares were put into test. The division of labour was, further, the accountable source of the problem's character, which was not that there was a discrepancy between the two softwares, but that this discrepancy was persisting and producing a disproportionate and expanding share of the project's outstanding problems. The accountable source of this was divergent priorities between work sites. The major component of the UK site's current work was the running of the 'score test' and the solution of the problems that was generating, with part of that work being the recording and relaying of data on the problems scored against the US software.

Resolving the problems arising in testing was relatively low priority for the US operation but the UK designers were dependent on decisions about how the ESS was to be designed for the solution of some of their own problems, for in important respects the ESS was the dominant party, with the IOT group's own design decisions being dependent on those about how the ESS software was to be organised. This meant that, insofar as matters in the design of the ESS were unresolved, the IOT's problems could not be definitively worked out either. The integration problem was, further, in competition for attention: the work required for its solution would be additional to that scheduled for, and the recognition that the problem would not go away and would need to be addressed was accompanied by the recognition that it would not get addressed until it was 'worth' tackling and could be justified.

In this setting tasks are not entered into to see what it will take to do them. 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. The carrying out of tasks is, then, typically a matter of trying to achieve them on the basis of estimated resourcing which may or may not prove adequate to their eventual completion. This *ad hoc* group assembled to tackle the software integration problem confronts the problem as one which they are confident they can solve; that there *will* be a solution is taken for granted, and the question is whether they will have 'long enough' to achieve it. There was the remainder of the working week at their disposal before the visitors returned to home base and to other work, and the target they had set themselves was that of finding ways of achieving 'a significant reduction' in the number of problems resulting from the interaction of the software systems. At the outset of their deliberations they could not say how long it was going to take them to work through the problem but they reminded themselves that they not only had to work out what the problem was/might be, but had to rewrite the software and try out the new version on some test machines to confirm that their diagnosis had indeed located the cause. They should bear in mind that there was some urgency to what they were doing, how much they needed to do in the time available, but should also recognise that the urgency was in the end to be subordinated to the need to ensure that they had an adequate solution: the risk of going for a solution just to get one was to be avoided.

That their deliberations were legitimately someone else's business was taken for granted by them. The *ad hoc* grouping involved collaborative work

across two departments and the parties had their respective managers to satisfy, to show that assembling the group had been worthwhile and to convince that whatever solutions they might contrive were acceptable. Thus, Jay's remark which closes the excerpt from the transcript identifies Rick and Gordon, his managers, as ones who will need to be convinced of their proposals. However, not all interests in and demands on that work are necessarily legitimate. In Jay's view the design featured serious inadequacies, and there were pressures on the group to look for solutions which would compensate for those design deficiencies, but such requirements should be rejected. It was not this group's work to take on and solve problems built into the basic design, but to work for the best solution regardless of what others would like them to do.

The work on the problem in important respects consisted in the management of mutual understanding. The presumed source of the problem was in discrepant understandings between the two software writing operations, most specifically amongst those involved in writing the instructions that provided for the communication of their respective system with the other. The parties were all intensely, though variously, familiar with the design of the software and the performance of the machines, and the meeting proceeded – as already mentioned – through a round-the-table reconstruction of the organisation of the software, providing a collaborative specification of what they understood the software was doing or should be doing, with the result that uncertainties about each other's design were manifested and eventually (as discussed) a discrepancy in respect to the place of the 'out of paper' signal in the respective sequences was revealed. In respect of the solution of the problem the task then became that of defining a common policy, and the transcript records a part of that task, the formulation of an agreement to employ the symmetrical procedure in job recovery. The excerpt further shows the importance attached (as it was throughout the meeting) to ensuring that matters were explicitly agreed and that what was agreed to was understood in common ways.

CONCLUSION

The purpose of this sketchy exposition of a few moments in the life of an engineering project is to make the point that the claim that 'technical work' and the 'production of technology' are socially organised need not in itself be a contentious one, for its substantiation involves nothing more than drawing attention to things which are, for those engaged in such work, the most ordinary and natural features of carrying out their work, matters which they recognise as the routine (or not-all-quite-so-routine) exigencies of carrying through even a small, though currently critical, operation on the design and development of a piece of technology. In part, ethno-

methodology's idea is to bring the abstractions of sociological thought 'back down to earth' and this we have here sought to do by offering a reminder that 'technical work' viewed from the point of getting it done involves the determination of such matters as how much work there is to do, specifically what things are required to do the work that is to be done, how long it will take, how many must be involved, how much time is available, how those involved are to combine their activities to carry the work through, and how they are to ensure that their activities will remain coordinated and synchronised over its course, what is to be done in various eventualities, who will make the judgement as to whether the work has been done satisfactorily and what it will take to satisfy them.

NOTES

1 For a review of such 'strong programme'-influenced initiatives toward a 'sociology of software', see Murray and Woolgar (1991).
2 See, for example, Garfinkel (1986).
3 The 'labour process' analysis derives from Marxian accounts of the organisation of work under capitalism, receiving recent impetus from Braverman's (1974) work. For the application of this approach to computing technology, see Friedman and Cornford (1989).
4 It may be important to add that we make no 'extrinsic' claims about the typicality of the doings we observed, are not concerned to advance our argument's cause on the grounds that it exemplifies things which are 'common', 'widespread', or 'typical' in software development. Our interest in these materials is not that they portray something which regularly happens but only in the fact that they did happen. Any concern with their putative generality is, for us, confined to that which is 'intrinsic' to the activities themselves, to the extent to which it matters, for those carrying out the activities, whether this situation is (say) a standard, familiar one or not.
5 Cf. the chapter 'Two notions of agreement' in Lynch (1985a: Chapter 6).