Cooperative Systems Design

Ian Sommerville, Richard Bentley, Tom Rodden and Peter Sawyer
Computing Department,
Lancaster University,
LANCASTER LA1 4YR.

Phone: +44-524-593795
Fax:+44-524-593608
E-mail: is@comp.lancs.ac.uk

Abstract

This paper discusses an innovative experiment where sociologists were actively involved in the requirements analysis for an interactive software system to support the work of air traffic controllers. Air traffic control is intrinsically cooperative and our work involved an analysis of that process from a social perspective and the development of a prototype user interface for air traffic controllers' interaction with a flight information system.

As part of the analysis process, sociologists were involved in ethnographic studies of work and discovered subtle and complex patterns of cooperation which we suspect would not have been discovered using structured methods for requirements analysis. From a software development perspective, we describe how the input from the sociologists was essential for understanding the real automation requirements, discuss the difficulties of inter-disciplinary cooperative working and suggest how social analysis can be integrated in the interactive systems design process.

Introduction

In 1979, the US General Accounting Office published a report which showed that more than half of the software systems which were commissioned and delivered were never successfully used [1]. This figure is probably extreme and perhaps out-of-date but it illustrates what is undoubtedly a general problem. A large percentage of delivered computer systems do not meet the needs of users of these systems. They become 'shelfware' or only a fraction of their functionality is utilised.

Part of the reason for this is that business environments are changing very quickly. Consequently, the requirements for a commissioned system are often out-of-date by the time that system is delivered. However, we believe that there is another reason. Our current methods of requirements analysis are incomplete and the analysis of the system requirements often misunderstands the true nature of the work which is to be supported by the computer system.

One approach to this problem has been to involve end-users in the system design process. This is sometimes called user-centred design [2] and, when users actually become full members of a design team, it is called participative design [3]. Users have a detailed understanding of their job and participate in the systems design process. They experiment with, comment on and evaluate proposed designs. They may themselves suggest appropriate user interface designs.

We recognise the value of involving users in the design process and the important contribution which they can make. However, individual user participation does not provide a complete picture of user needs and cannot guarantee that the system will be usable and acceptable to a more general community of system end-users. Users understand the details of their own jobs so their attention is focused on individual activities rather than on how these activities and the users themselves fit into an overall organisation of working practice. Different users may evolve individual ways and shortcuts of carrying out the same task so a particular user involved in the design process need not necessarily be representative of other system users [4].

The design process must also consider the *social situation* where the computer system is installed. As well as detailed activity descriptions, we need a broader picture where these activities and their inputs and outputs are set in a group and organisational context. We must understand how particular ways of working have evolved, how people interact in small and large groups, how the detailed execution of tasks is affected by organisational cultures and goals, etc. This broader view is particularly important where the tasks to be supported by the computer system are cooperative. When several users working together towards some shared goal, social and organisational factors govern the formation, structure and functioning of working groups.

This paper discusses an experiment where sociologists and software engineers worked together to carry out such a social analysis with a view to understanding the requirements for computer support. The deliberately ambiguous title of this paper reflects the nature of this experiment. We were concerned with the design of systems which were essentially cooperative in the sense that they were team-based. We were also concerned with working across disciplines so we were also participating in a cooperative design activity.

Our work has involved a study of air-traffic controllers and their working environment [5, 6]. In parallel with this study, we have developed a tool for command and control system prototyping which has been used to produce a number of experimental user interfaces to an air traffic control database. This system supports multi-screen manipulation of a shared database of flight information. The interface designs were informed by the results from the social analysis of the support required by controllers and have been evaluated by end-users. The details of this prototyping

system and the generated interfaces are not covered in this paper but are described in detail in a series of papers by Bentley *et al.* [7-9].

In the remainder of the paper, we first discuss the method of social analysis which was used and then describe the UK air traffic control process which establishes a background for the rest of the paper. We suggest how social analysis can fit into a requirements engineering process, discuss the problems of inter-disciplinary working and describe some results of our analysis which contradict, to some extent, the 'conventional wisdom' in systems design. Finally, we look back on this particular experiment and draw some conclusions about the nature of the cooperative systems design process.

Ethnographic analysis of work

The work of air traffic controllers was studied using an approach to social analysis called ethnography. Ethnography is a process which involves a social scientist spending an extended period of time with a group or society making detailed observations of its organisation, culture and practices. It is intended to discover the social character of groups and the group activities in their natural setting. Subsequent analysis of these observations reveals information about the structure, organisation and practices of the group which has been studied.

The assumptions underlying ethnography are that a true understanding of a society can only be gained from within that society and that the social organisation depends on tacit knowledge and implicit practices which are not usually obvious to the casual observer. The knowledge and practices used may vary significantly depending on the individuals involved and the state of the system when they are initiated. Thus, they are not readily susceptible to a formalised task analysis which focuses on individual activities and largely takes for granted the significance of these activities in some broader setting. Ethnographers question the usefulness of task analysis as they argue that there is no such thing as a context-free task. Heath *et al.*[10] discuss task analysis in the context of a study of City of London dealing rooms where they state:

> "the concept of 'task' in task analyses is often restricted to work undertaken by an 'individual' and the particular representations utilised seem to impose a structuring which fails to account for the interactional and interleaved nature of activities in real world settings"

A characteristic of ethnography is that there are no pre-suppositions about the society being studied, there is no list of questions to be answered and (in principle at least), the ethnographer does not try and impose his or her value judgements on the practices which are observed. This is quite different from the approach adopted by analysis 'methods' be they structured methods [11], object-oriented methods [12] or

whatever. These methods deliberately limit the vocabulary of the analyst and force a particular analysis perspective.

We suggest that these structured methods are not well suited to the analysis of interactive systems, particularly where these systems must support cooperative activities. If work practices cannot be denoted using the notation and structures prescribed by these methods, they are usually left out of the analysis. As none of the methods make explicit provision for representing cooperation or even meaningful user interaction, this means that cooperative practices are, at best, simplified and, at worst, ignored, in method-based systems analysis.

In principle, many activities involve a set of procedural steps and, where these are cooperative activities, there is a prescribed division of labour. That is, it is defined who should be responsible for particular tasks and how they should cooperate with other workers. Sometimes, the working procedures are implicit rather than explicit but current trends in quality management are towards explicitly defining these procedures in so-called 'quality manuals'.

However, this procedural specification is often an over-simplification of the actual process involved and may give a misleading impression of the true nature of the work. Procedural process specifications, by their very nature, cannot define responses to all possible events especially as the nature, timing, and sequencing of these events are both unpredictable and dependent. Various studies [13-15] have shown that the actual practice of work often differs significantly from the prescribed working procedures. Anderson *et al*. [16] use the term 'working division of labour' to describe the situation where a team organises itself to carry out a task without being bound by job descriptions and job titles defined by the organisation. This working division of labour is not static. It is continually re-negotiated depending on circumstances, resource availability and priorities.

We are convinced that the emergence of this 'working division of labour' rather than the prescribed organisation is one important reason why the requirements for a software system are often such that the system does not meet the real needs of end-users. The system requirements are defined according to documented procedures and standards. Ethnographic analyses provide insights into the true nature of work and reveal system requirements to support actual rather than theoretical work practices.

Air-traffic control

To allow us to discuss the findings of our ethnographic studies, we first must explain, very briefly, the work of UK air traffic controllers responsible for *en route* (as distinct from take-off and landing) traffic control.

UK airspace is split into sectors with a team of five controllers responsible for managing each sector. Physically, controllers work at a 'control suite' which provides a

working area for two teams of controllers, radar screens and facilities for communications between aircraft and other suites. The radar is a 2-dimensional, real time representation of the sector's airspace. As well as displaying aircraft positions, it can also display sector boundaries, coast lines, major and minor airports, etc.

The radar shows what is happening in real-time but not what might be happening in a few minutes time. Controllers need to anticipate potential problems so, to supplement the radar, they use paper *flight progress strips* which carry information about expected and current flights being controlled, together with a record of controllers' instructions to the controlled aircraft.

An example of a flight strip is shown in Figure 1. The strip contains static information such as the flight number, aircraft identifier, radio beacon identifier, source and destination, and dynamic information such as time over beacon, heading, current height and requested height. When control decisions are made, these are communicated to the aircraft and recorded on the strip so that a record is maintained of controller actions.

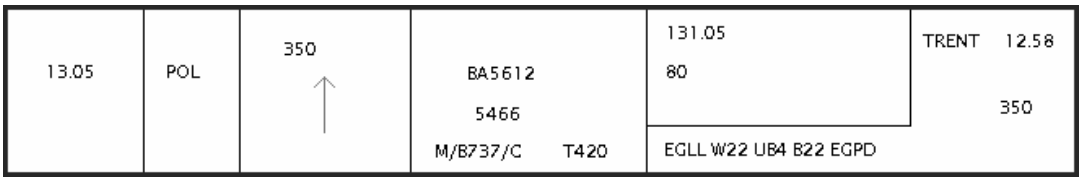

Figure 1 A flight strip

Flight strips are printed 40 minutes before an aircraft enters a controlled sector. As the flight time through a sector may be as little as 10 minutes, strips are replicated across different control suites. When a controller makes a decision, this may have to be discussed (by phone) with controllers in other sectors. Changes to the strips made by one controller are not immediately visible to other controllers and, if these changes affect the sector entry or exit heights or headings of an aircraft, discussions with the controllers of adjacent sectors may be required.

Flight strips are organised on a *flight progress board* where strips are aligned and organised in a rack according to the reporting points over which a flight will pass (Figure 2). To the accomplished controller this display is an 'at a glance' means of showing the flow of traffic through the sector and its characteristics such as aircraft type, airspeed, height, time of arrival at a beacon, etc.

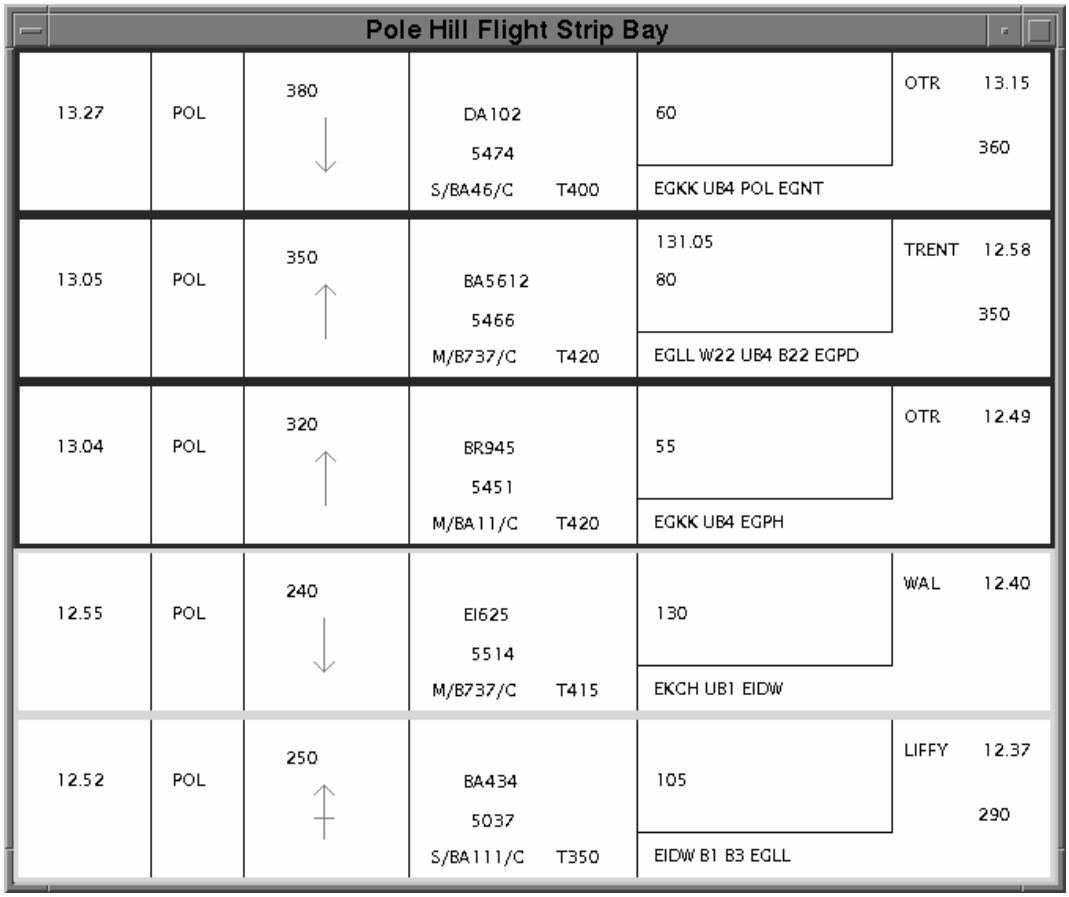

Figure 2 Flight strips organised by height

Flight strips are not simply a paper database which supplement the real-time information available from the radar. Hughes *et al.* [6] discuss how the strips are absolutely central to the process of air traffic control. Indeed, some controllers stated that 'working the strips' is the essence of the control process. The flight strips and flight progress boards are:

1. *An individual work site* Controllers physically manipulate the strips, placing them in the rack and re-ordering them as control decisions are made. This physical manipulation serves to remind the controller of the strips' contents and hence allows them to build and maintain a mental picture of the airspace which is being controlled.

2. *A planning tool* Strips are printed for controllers 40 minutes before an aircraft enters a controlled sector so the controller uses the strips to work out, in advance, control strategies for the sector.

3. *A team memory* The flight progress boards can be viewed by all team members so they serve as a repository of information for them. They do not just provide current state but also an immediate record of what control decisions have been made. The public availability of strips also means that controllers can maintain a constant check

on each others work, identify problems and, by looking at adjacent sectors, estimate future workload.

From a quick look at a flight progress board, controllers can assimilate information about the airspace and can check control strategies against the real-time information coming from the radar. Thus, strips act as an important safety device which facilitate communications between controllers.

Automating the interface

UK airspace is very congested and a significant traffic increase is likely over the next decade. The demands placed on controllers are increasing and the current system cannot cope with much more traffic. Automated assistance to controllers might allow new control strategies to be established which would allow more flights without compromising safety.

For example, a real-time flight database could provide the basis for automated warning systems such as a 'conflict alert' system. Aircraft might therefore be packed more tightly into the available airspace. Recording controller actions in real-time in a database should also reduce overhead in the control process by speeding up communications between controllers.

There have been several previous experiments with automating the user interface to the flight data base so that instructions issued by a controller are automatically captured. By and large, air traffic controllers have not been happy with these experimental systems. We believe that this is because these systems have not effectively supported the 'working division of labour' as adopted by controllers. Hopkin [17] identifies the problem:

> "One striking aspect of automation applied to Air Traffic Control systems is that most of the forms of automation for the controller to use, as distinct from those which sense or process or compile data automatically, are for one controller at a human-machine interface. They are aids to an individual controller's decisions, problem solving or predictions, yet they are being introduced into contexts where many of these functions have previously been performed by teams"

Our work was principally concerned with understanding the nature of cooperation between air traffic controllers and the impact of this cooperation on the requirements for support software. We were not concerned with broader managerial aspects of air traffic control nor with the problems of efficiently implementing large-scale, highly reliable flight database systems. In terms of software, we were interested in discovering appropriate user interface designs which would support the electronic display and manipulation of flight information and, at the same time, recognise and support the cooperative control process.

Requirements engineering with ethnography

This section describes the process which we used to exploit the ethnographic analysis in prototype development. For both sociologists and computer scientists this project was a trailblazing experience. Previous ethnographic studies such as those by Suchman [14] had suggested that ethnographic studies could reveal systems requirements. However, there was no previous work where ethnographic studies of work had been integrated with software system development.

Because there was no previous work and the lack of any existing theoretical framework for collaboration, we devised a development process which allowed system prototyping to take place in conjunction with on-going social studies. The model of development which we used and which we propose as a general model is shown in Figure 3.

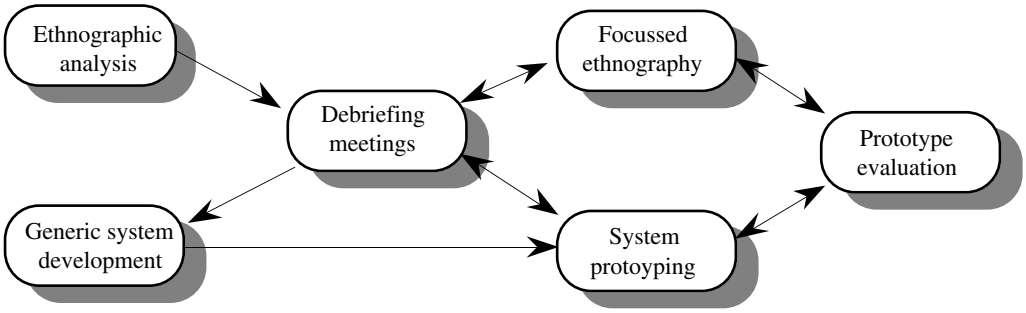

Figure 3 Integrating ethnography with system prototyping

The stages of this process are:

1. *Ethnographic analysis* This involved both the ethnographer and the software developers spending some time in the air traffic control centre gaining an overall understanding of the control process and the existing flight strip system. This was supplemented by reading manuals and procedural descriptions. This initial analysis was intended to provide sufficient information to allow software prototype development to get started.

2. *Generic system development* After some initial analysis, we started developing a generic system to represent and display flight information. In parallel with this, we initiated more detailed ethnographic studies. These studies fed back information regularly to the generic system development process. At this stage, the information flow was one-way, that is, the ethnographer provided information which improved our understanding of the control process. This stage lasted for a period of about three months.

3. *Debriefing meetings* During the debriefing meetings, system developers questioned the ethnographer about their observations. These debriefing exercises were central to the understanding of the work process. During the meetings, systems developers picked up the significance of particular work practices and questioned the sociologist for more details. These points were often aspects of the system which would pose particularly difficult automation problems. Usually, further ethnographic studies were required to resolve the issues raised by the systems developers.

4. *Focused ethnography* Once the generic system had been sufficiently developed, we were then in a position to put specific questions to the ethnographer about the control process and these questions focused the next period of ethnographic analysis. The results were fed-back through briefing meetings and a new set of questions posed for the next analysis period. These briefing meetings and the record of the ethnographic studies informed the development of a prototype system which was generated using the generic system which we had developed.

5. *Prototype evaluation* Once a demonstrable system had been developed, system evaluation began. The first evaluation stage involved the ethnographer using his knowledge of the control process and standing in as a controller. Later evaluations involved air traffic controllers experimenting with the system.

The next logical stage in the process is to refine the system then to evaluate it with end-users in conjunction with a more detailed overall systems analysis. In fact, before end-user evaluation, we completely re-implemented and generalised the underlying prototype development system to allow us to explore alternatives to strip-based interfaces.

The specific objective of our research was use ethnography to inform interactive systems design. It was never intended that we should combine our ethnographic analysis of air traffic control with a more conventional requirements analysis using, for example, some structured method. However, based on our experiences, we have formulated a model of how ethnography can be integrated with other approaches to requirements analysis as shown in Figure 4.

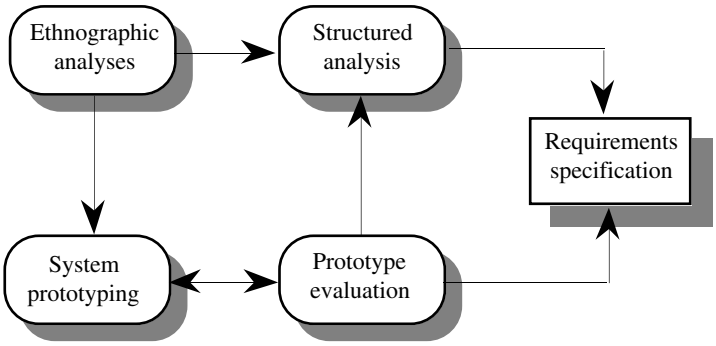

Figure 4 Requirements engineering with ethnography

We do not believe that ethnographic analysis, on its own, is sufficient to derive system requirements nor that the results of that analysis can act as a system specification. Ethnographic studies and system prototyping might come first in the requirements engineering process and may be followed by more detailed analysis, perhaps using a structured method, and prototype evaluation. The function of ethnography is to discover the essence of the work being studied without the constraint of conventional analysis methods. More conventional analysis has a place later in the process where detailed information about the work and the required system is required.

Other work on integrating ethnography into the requirements engineering process has been done by Goguen and Linde [18]. They propose initial studies to identify the areas where ethnography can be useful and then they study those 'problem areas' in more detail using ethnography. The rationale for this is that it is important to limit the time spent on expensive ethnography.

Our experience suggests that the value of ethnography is discovering problems which other approaches cannot reveal. Applying some structured approach (say) as an initial step in requirements analysis is liable to miss the key areas where ethnography can provide insights. However, the high cost of ethnography is indeed a real problem which limits its practical application and there is no doubt that further work is needed to integrate it with requirements analysis in a cost-effective way.

The ethnographer's role

Ethnographers are not systems analysts and their role in the process is simply to observe the work in progress, to note the characteristics of the work and to report information to the system designers. Their study is recorded in a set of field notes. These are inherently unstructured and cannot be used without help from the observer of the work.

It is common for ethnographers to 'go native' and to identify closely with the potential end-users of a computer system. They can therefore act as surrogate users when participating in the design process. Such surrogate users may, in fact, be more effective participants in this process for three reasons:

1. They do not have a vested interest in the success or failure of the system. Therefore, they can take a less biased perspective than real users who, commonly, have a particular position within an organisation to defend.
2. As a surrogate user, they can effectively integrate the opinions and work practices of a range of different users. They do not therefore simply present a single user's viewpoint.
3. They have more time to spend in the development process and, hopefully, more patience with designers who misunderstand the nature of the work and consequently make poor design decisions.

As well as participating in the system requirements analysis, sociologists might also take other roles in the overall systems engineering process.

1. During the evaluation phase, the ethnographer can again act as a surrogate user and participate in the initial system evaluation. This may help to find gross design errors to be found which can irritate end-users and cause them to reject a system without a thorough evaluation.
2. When a system is available for experiment, the sociologist can identify specific training and documentation requirements which will allow users to investigate the facilities of the system. In particular, where changes to established work processes are required, the ethnographer can pinpoint specific training needs and can help relate the new processes to the previous activities.

In our experiment, we had a lengthy evaluation involving the ethnographer as surrogate user and air traffic controllers who experimented with the system at different stages of development. The ethnographer did find some design errors but we found that they were particularly useful in communicating with the controllers participating in the system evaluation. They understood the control process and could explain system concepts to the controllers more readily than the system developers. We have not yet explored the second possible role where the ethnographer identifies training requirements.

Cooperative working in the software development process

Working across disciplines is always difficult. Different disciplines evolve their own vocabulary and methods and these may not mesh neatly with those of complementary disciplines. We came across problems of different research methodologies, communication and understanding the nature of sociology. The problems are summarised here with a more complete description given elsewhere [19].

As an illustration of some of these problems, we asked our colleagues in sociology to make some judgements about what were and what were not critical system

requirements. This caused them great difficulties. They argued that the essence of the system was in the integration of activities but our questions required them to consider isolated activities rather than the overall system. They argued that the system was not a set of loosely interacting individual activities but a coherent whole which was more than the sum of its parts.

Of course, this is inarguable. In practice, however, computer systems can't be built without structure and pragmatic decisions have to be made about what to support and what to leave out. We cannot avoid making decisions but need appropriate expertise so that decisions are better informed. Our sociologist collaborators now understand this and accept that they cannot opt out of making judgements about the process which they are studying.

Problems of communication are normal in any inter-disciplinary collaboration as each discipline has its own specialised vocabulary. Where a science or engineering discipline is collaborating with a social science the communication problems may be compounded because the same word may have different meanings in each discipline and the specialised meanings may themselves be different from the 'normal' dictionary definition of the term.

As an illustration of this problem, we produced a simple mathematical specification of some abstract data types representing aircraft and stated that this defined the 'semantics' of these entities. What we were saying was that an abstraction of the meaning of the entities which we needed for this project was set out by the mathematics.

Our collaborating sociologists found hard to understand the concept of a mathematical specification of semantics. Their view was that the 'semantics' of an entity was not simply dependent on the entity but also on the observer of the entity and the context of observation. After a great deal of mutual incomprehension, it became clear that we were each using the term 'semantics' in a different and discipline-specific way and we eventually came to see each other's point of view.

These problems of language contribute, to some extent, to the problems which we both had in understanding the nature of the different disciplines. As software developers, we were used to a hierarchic model of knowledge. When learning a new subject, we start with basic information then read progressively more advanced material.

Applying this approach to the understanding of sociology was not successful because sociological knowledge is not hierarchically structured. There are many diverse and distinct areas such as the sociology of work, the sociology of religion, etc. which, on the surface at least, appear to have little in common. Furthermore, there are several 'schools of thought' in many of these areas based on different theoretical frameworks.

We found it impossible to reconcile these different areas and 'schools of thought' in sociology to build a general model of the discipline.

Comparable problems are inevitable in any inter-disciplinary cooperation. The project team must recognise these problems and address them explicitly. Otherwise, they will simply result in confusion, misunderstanding and mutual hostility. Our collaboration has shown that the problems can be solved and we believe that both disciplines have been enriched by our mutual desire to tackle them.

Countering the conventional wisdom

Ethnographic analysis does not deliver a list of specific, detailed systems requirements; rather it provides pointers to appropriate design decisions and highlights key human activities which should not be 'designed out of' a process with automated support. It is not appropriate to discuss all of the insights which we gained from the ethnographic analysis. Rather, we focus here on insights which, to us as software developers, were not immediately obvious and which contradict, to some extent, what might be called the 'conventional wisdom' of systems development.

The principal general insights which we gained from the ethnographic analysis of air traffic control can be summarised as follows:

1. Controllers have a very strong safety culture and they analyse any automation in terms of its impact on safety. They don't want the computer taking actions which are invisible and therefore difficult to validate. This means that information should not be concealed 'behind' the interface and that the interface should always allow controllers to carry out their own checks. While automated checking may be of value, controllers will reject systems which do not allow them to verify these automated checks. In short, they don't really trust computers (and who can blame them!).
2. The seemingly repetitive manual activities of strip bay re-ordering are important as they allow controllers to build a mental picture of the controlled airspace. When new strips are positioned in a rack, the activity of moving strips around is a metaphor for moving the aircraft around in the airspace.
3. Although some communication between controllers is explicit (and should be explicitly supported), most communication is implicit and relies on a shared knowledge of the work and the shared safety culture where mistakes must be detected and corrected.

The implications of these observations contradict, to some extent, what might be termed 'conventional wisdom' in interactive systems design. For example, most designers would agree with the following guidelines:

1. Humans should do what humans are best at and computers should take over repetitive tasks. Computer systems should therefore automate tasks which involve comparisons of similar information and ordering of records in a data store. Therefore, the computer system not the human operator should be responsible for sorting information and maintaining the sort order when new information is added to the system.
2. Humans have different ways of working and therefore like to have a personalised interface with a computer system. User interface designers should therefore always provide facilities for end-users to tailor interfaces to suit their own ways of working and individual preferences.

However, as discussed above, 'working the strips' has the effect of allowing controllers to build a mental picture of the controlled airspace. The controller's action of manually placing a strip in the appropriate position in a bay focuses attention on that strip and to related strips. Automated positioning based on some aircraft attribute would be easy to implement. However, it is undesirable because it forces controllers to check the strips explicitly to see the effect of changes to their mental airspace model. We know that relatively minor changes are easy to miss but in an air traffic control situation, missing such changes may be catastrophic.

A further example of manual intervention forcing a safety check occurs when strips are first printed in the current system. Strips are fitted into different coloured holders by assistant controllers to distinguish between flights on different headings. The assistants read the strip information and detect irregularities such as known flight numbers flying to the wrong airport, an arrival time which is inconsistent with the aircraft type, etc. They either correct these errors themselves or draw the controllers' attention to them.

It is a common belief amongst developers of user interfaces that end-user tailorability is essential. Indeed, Fisher and Girgensohn [20] state:

> "End-user modifiability is not a luxury, but a necessity in cases where systems do not fit a particular task, a particular style of working or a personal sense of aesthetics".

Air traffic controllers have certainly evolved individual styles of working so it would appear that an ATC system fits the above definition. However, while tailorability may be a requirement for applications designed for single-user use, we do not think it valid for cooperative systems where there must be a shared representation with a common understanding of the syntax and semantics of that representation. Our work with air traffic controllers has shown the importance of such a common representation for implicit communication.

Much of the work of controllers requires 'at a glance' observations of strips and flight progress boards. A supervisor or chief controller, for example, will simply walk

around the suite and will assist more junior controllers if any potential problems or difficulties are observed. This can only be effective if all controllers can rapidly assimilate flight strip information. This rapid assimilation is hindered if even slight differences in strip representations are supported. The system must not allow tailorability which will affect immediate understanding of the representation.

All of the above does not mean, of course, that we have to adopt a 'replication' approach when building automated systems and simply mimic current manual activities. Rather, the ethnographic analysis allowed us to understand the real needs of controllers and to make design decisions to support these needs. For example, in our prototype of the strip system, the following decisions were made:

1. Strips are not positioned according to some default order when they are added to a display. Rather, the controller must manually move a strip image to the appropriate place in the display. The computer, of course, handles the problem of making room for the strip. This is very close to the manual system so, in this case, we have replicated the manual activity.
2. We automatically assign a coloured border to a strip based on the aircraft heading but we also incorporate some automated database consistency checks on the flight information. The activity of 'handing a strip over' from an assistant to a controller for inclusion in a flight progress board has been retained as a manual activity. In this way, we have partially automated this activity, incorporated computer-based checking yet retain human intervention where an additional check can be made.
3. We do not allow any user interface tailoring except the ability to switch from full-sized to summarised strips. This is necessary to make use of limited screen display space. Our prototype system has extensive user interface tailoring capabilities but these are restricted to use during the interface design phase.

Results and Conclusions

The end-result of this project was a set of prototype user interfaces to a shared flight database system. This was a limited experiment (we did not, for example, integrate a radar simulator with the system) but it was sufficient for evaluation with air traffic controllers[21]. We were generally gratified that the air traffic controllers reacted positively to the system and were satisfied that the ethnography had captured the critical implicit activities which are part of air traffic control.

As well as a strip-based interface, our development strategy which was based on using a prototype development system to generate interfaces allowed us to explore, with the controllers, alternative interfaces where strips and radar information were shown on an integrated display. The ethnographic study was also important here as it

allowed us to generalise from specific observations and incorporate these general facilities in the prototype development system.

As far as the specific technical objectives of the project were concerned, the work was successful. Our experiences of cooperative systems design has also led to the more general conclusions listed below. Of course, all of the observations here are from the perspective of software developers; a comparable account written by sociologists might have a rather different slant.

1. For the class of systems which includes air traffic control systems, namely cooperative systems where a team develops its own working practices within a broad overall framework with the aim of achieving a shared objective, ethnography can be effective in discovering the subtleties of the process and the implicit and explicit communication between team members.

2. The classes of system where ethnography is useful have still to be established. Our study and other studies by, for example, Ackroyd *et al.* [13] and Heath and Luff [15] etc. have focused on teamwork in command and control situations. Other work which is currently underway is concerned with observing the more loosely structured activity of system design. While interesting insights into the design process have emerged, the ethnography has been less effective in the requirements analysis for cooperative CASE tools.

3. The place of ethnography is early in the development process to identify the essentials of the system. Ethnography highlights these key characteristics and may also identify negative requirements i.e. things the system must not do as their inclusion would force changes to critical work practices.

4. The requirements identified through ethnography are often constraints of the system design rather than functional requirements. That is, ethnography may identify the significance of some action or series of actions but cannot usually suggest how this should be supported when the system is automated.

5. The unstructured nature of ethnography is both a strength and a weakness. It is a strength because it allows the unbiased analysis of systems. It is a weakness, because it is very time consuming to understand the ethnographer's field notes and these notes can only be interpreted by the ethnographer who carried out the work study. If ethnography is to be used routinely as part of the requirements engineering process, there is a pressing need for tools which help structure the ethnographer's notes and make them understandable to a more general audience. We are currently investigating the use of a locally developed information management system for this purpose[22].

6. Ethnography is both time consuming and expensive. It cannot therefore be used in its current form for systems which have to be brought to market quickly. It can be used in very large bespoke system development with longer delivery schedules and lifetimes. Interesting further research might investigate whether or not ethnography can be made more structured and more rapidly applicable without losing the current strengths of the approach.

7. Our evaluation experience was inconclusive as to whether or not the ethnographer can act as an effective surrogate user in system evaluation studies. In our evaluation, the ethnographer identified a number of relatively minor problems. However, major defects were still present in the system which were discovered when the system was evaluated by controllers. We believe that the reason for this is that the ethnographer observes the control process but is not a trained controller. There are detailed technical aspects of the process (as distinct from social aspects) which the ethnographer is unable to absorb simply by observation.

A more general conclusion of our work is that a social science perspective can be generally valuable in interactive systems development. An explicit social analysis may expose process complexities which are not likely to be discovered in a more structured analysis. Of equal importance, however, is an awareness of social and organisational factors which often govern whether or not a system is acceptable to its users. Ethnographic analysis may not always be possible but a sensitivity to these social and organisational factors on the part of software engineers is likely to result in improved systems.

Drawing conclusions from a single experiment is inherently unreliable so the above observations should be considered as tentative rather than definitive. Nevertheless, we are convinced that social analysis should become part of the requirements engineering process for systems which involve cooperative working and we are equally convinced that such analysis can provide valuable insights into the true nature of the requirements for a broad class of interactive systems. We found the cooperative inter-disciplinary collaboration involved to be both challenging and rewarding.

Acknowledgements

The work reported here was funded by the SERC/MRC/ESRC Joint Council Initiative in Cognitive Science/HCI under contract SPG8931598. Thanks are due to the air traffic controllers who provided insights into their work and to our colleagues John Hughes, Dan Shapiro and Dave Randall from the Sociology Department at Lancaster University for their patience and adaptability when faced with our demands.

References

1. GAO, *Contracting for Computer Software Development — Serious Problems Require Management Attention to Avoid Wasting Additional Millions*. 1979, US General Accounting Office:

2. Norman, D.A. and S.W. Draper, *User-centered System Design*, 1986, Hillsdale, N.J: Lawrence Erlbaum.

3. Kyng, M. 'Designing for a Dollar a Day', in *Proc. CSCW'88.*, 1988. Portland, Oregon, 178-88.

4. Downton, A., 'Evaluation techniques for human-computer systems design', in *Engineering the Human-Computer Interface,* A. Downton, Editor. 1991, McGraw-Hill: London.

5. Bentley, R., *et al.* 'Ethnographically-informed Systems Design for Air Traffic Control', in *Proc. CSCW'92*, 1992. Toronto, Canada, 123—129.

6. Hughes, J.A., D. Randall, and D. Shapiro. 'Faltering from Ethnography to Design', in *Proc. CSCW'92*, 1992. Toronto, Canada, 115-122.

7. Bentley, R., T. Rodden, P. Sawyer, and I. Sommerville. 'A Prototyping Environment for Dynamic Data Visualisation', in *Proc. 5th IFIP Working Conference on User Interfaces*, 1992. Ellivouri, Finland,

8. Bentley, R., T. Rodden, P. Sawyer, and I. Sommerville. 'An architecture for tailoring cooperative multi-user displays', in *Proc. CSCW'92*, 1992. Toronto, Canada, 187—94.

9. Bentley, R., T.A. Rodden, P. Sawyer, and I. Sommerville, 'Architectural Support for Cooperative Multi-user Interfaces', *IEEE Computer*, 1994. 27(3): To appear.

10. Heath, C., M. Jirotka, P. Luff, and J. Hindmarsh. 'Unpacking collaboration: the interactional organisation of trading in a city dealing room', in *Proc. ECSCW'93*, 1993. Milan,

11. DeMarco, T., *Structured Analysis and System Specification*, 1978, New York: Yourdon Press.

12. Coad, P. and E. Yourdon, *Object-oriented Analysis*, 1990, Englewood Cliffs, NJ: Prentice-Hall.

13. Ackroyd, S., R. Harper, J.A. Hughes, and D. Shapiro, *Information Technology and Practical Police Work*, 1992, Milton Keynes: Open University Press.

14. Suchman, L., *Plans and Situated Actions*, 1987, Cambridge: Cambridge University Press.

15. Heath, C. and P. Luff. 'Collaborative Activity and Technological Design: Task coordination in th London Underground control room', in *Proc. ECSCW'91*, 1991. Amsterdam,

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

17. Hopkin, V.D., 'The Impact of Automation on Air Traffic Control Systems', in *Automation and Systems Issues in Air Traffic Control,* J.A. Wise, V.D. Hopkin, andM.L. Smith, Editor. 1991, Springer-Verlag: Berlin.

18. Goguen, J. and C. Linde. 'Techniques for Requirements Elicitation', in *Proc. RE'93*, 1993. San Diego, CA., 152-64.

19. Sommerville, I., T.A. Rodden, P. Sawyer, and R. Bentley. 'Sociologists can be Surprisingly Useful in Interactive Systems Design', in *Proc. HCI'92*, 1992. York, UK,

20. Fischer, G. and A. Girgensohn. 'End-user Modifiability in Design Environments', in *Proc. CHI'90*, 1990. Seattle, USA,

21. Bentley, R., D. Randall, and M. Twidale. 'Situated Evaluation of Cooperative Systems', in *Proc. CSCW'94*, 1994. Chapel Hill, N.C., Submitted for publication.

22. Twidale, M., T.A. Rodden, and I. Sommerville. 'The Designer's Notepad: Supporting and Understanding Cooperative Design', in *Proc. ECSCW'93*, 1993. Milan,