Integrating ethnography into the requirements engineering process

Ian Sommerville, Tom Rodden, Pete Sawyer, Richard Bentley and
Michael Twidale,
Computing Department,
Lancaster University,
LANCASTER LA1 4YR
UK
E-mail: is@comp.lancs.ac.uk

Abstract

This paper reports on experiences of an inter-disciplinary project involving software engineers and sociologists. The project is concerned with discovering the requirements for a user interface to a flight database which is used to provide real-time information to air traffic controllers. The sociologists are conducting an ethnographic analysis of the activity of air traffic control and this is being used to inform the development of a prototype system. The paper gives an overview of the project, discusses how sociologists can contribute to requirements engineering and suggests tool support which will allow ethnographic observations to be integrated into the requirements engineering process.

Understanding work through ethnography

Ethnography is a process which was originally developed by anthropologists to understand social mechanisms in 'primitive' societies. It involves an anthropologist spending an extended period of time (sometimes several years) living in a society and making detailed observations of its practices. Subsequent analysis of these observations revealed information about the structure, organisation and practices of these societies. A key 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.

These techniques have also been applied to an analysis of work where an ethnographer spends several months in a working environment observing and noting practices, cooperation and processes. The rationale for these studies is that actual work practices often differ quite markedly from the 'prescribed practices' set out in company manuals and handbooks. It is usually the case that a 'working division of labour' (Anderson *et al.*, 1989) evolves where a team organises itself to carry out a task irrespective of the 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.

The notion that there is a fixed process or procedure for most tasks which can be automated is an over-simplification. We believe that the existence 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 but don't take into account actual working practices.

Involving prospective end-users of a computer system in the requirements analysis does not solve this problem. We know from work in knowledge acquisition that experts find it very difficult to articulate their expertise. It is equally if not more difficult for end-users to describe the working division of labour which is, in fact, informal and dynamic. In other cases, the actual work practices may be quite contrary to organisational standards and the end-users of the technology will simply not admit that these practices go on.

The potential value of ethnography in deriving computer system requirements was first identified in seminal work by Suchman (1983, 1987). Subsequently, further studies concerned with air traffic control (Harper *et al.,* 1991), police database systems (Ackroyd *et al.,*1992) and underground railway control (Heath and Luff, 1991) have confirmed that an ethnographic study can give real insights into working processes which should be taken into account when deriving computer system requirements. Ethnography is distinct from 'traditional' systems analysis in that it focuses on the participants and their interactions in a system rather than the data, its structure and its processing. Furthermore, it is usually prolonged so the working division of labour can be identified.

The ethnographic approach is quite different from other 'soft' psychological approaches. Approaches to requirements based on user psychology are usually concerned with controlled experiments in an artificial setting to discover information about work practice. Ethnography must take place in a real setting. Suchman (1987) claims that her ethnographic studies of an intelligent help system for a photocopier were more effective in revealing problems and difficulties than any psychological experiment would have been.

The above ethnographic studies were not associated with a software development project concerned with automating some of the work processes. Therefore, although they produced insights into the organisation of work which are clearly of interest to systems designers, it is not clear how to translate their results into system requirements. In order to do so, there are a number of problems which must be addressed:

1. What are the problems which arise from inter-disciplinary working where the disciplines are as radically different as sociology and engineering?
2. Ethnography is normally very prolonged. How can it be organised so that it can contribute quickly to the requirements capture process?
3. How can the ethnographic record of the observations be organised so that system requirements can be identified amongst a great deal of anecdotal material?

In this paper, we are concerned with discussing the above problems of integrating ethnography with requirements engineering. After presenting a brief overview of our project, which is concerned with the automation of air-traffic control, we discuss insights which we have gained from the project and some of the problems of inter-disciplinary working. Finally, we explain how we are investigating the integration of ethnography into requirements capture through the use of a software tool which supports both formal and informal information capture.

Project overview and rationale

Our work, which follows on from previous studies of air traffic control (Harper *et al.,* 1991), is explicitly concerned with investigating how ethnographic analysis can be integrated with system prototyping to provide input for the requirements engineering process. Details of the prototyping system and the ethnographic studies are discussed in more detail elsewhere (Bentley *et al*., 1992; Sommerville *et al*., 1992a).

Air traffic control in the UK is managed through two sites which control all domestic and international air traffic. Our studies were concerned with the London Air Traffic Control Centre (LATCC) where there are 8 radar suites each controlling 2 sectors of the airspace.

Each suite provides a working area for two teams of controllers (one team per sector), 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 and the positions of aircraft within it. It can display other information about a controlled sector including sector boundaries, coast lines, major and minor airports, etc. However, the radar only shows what is happening 'now' 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 controllers' instructions to the controlled aircraft.

An example of a flight strip (as replicated in our automated system prototype) is illustrated 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.

Figure 1 A flight strip

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. To the experienced and knowledgeable eye, different indicated ETA times and different heights may represent problems. For example, 'climbing through' or 'catching up' given the relative performances of aircraft. The way in which the controller organises the strips, for instance, according to arrival time over a reporting point, or according to flight level, or to possible conflict points, provides important information about the state of the sector.

Figure 2 Flight strips organised by height

Previous experiments in automating the UK system have been undertaken primarily from a technical viewpoint. They have not been acceptable to air traffic controllers and we believe that this is because these systems have not effectively supported the 'working division of labour' as adopted by controllers. Hopkin (1991) 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 is different. We are starting from the position that air traffic control is a subtle cooperative activity and we believe we must understand the nature of that cooperation in order to build effective computer support. We are particularly concerned with deriving the requirements for a system which allows direct manipulation of the flight database which holds flight plan details. This system must be integrated with a radar data processing system which provides real-time information on aircraft position, height and heading. To build an effective and usable system, we must incorporate support for the actual shared work practices and cooperative activities which characterise air-traffic control.

Our work involves a detailed, ethnographic study of controllers and their working environment (Sommerville *et al.,* 1992a). In parallel with this study, we are developing a tool for command and control system prototyping (Bentley *et al.,* 1992) which is then instantiated to produce a user interface to an air traffic control database. This interface supports both flight strips and radar displays taken from a shared database. This interface is being developed using the results from the ethnographic analyses of the support required by controllers. The project will shortly enter an evaluation phase where the developed interfaces will be assessed for usability by working air traffic controllers.

Insights from the ethnographic analysis

Our work has now reached a stage where we are generating system interfaces whose design has been informed by the ethnographic observations. We have found that the information provided by ethnography is essentially background information which has provided a deeper understanding of the application domain. The ethnography did not result in specific, detailed systems requirements; rather it provided pointers to appropriate design decisions.

From our observations we have become convinced that some ‘conventional’ assumptions made by systems designers may be invalid when cooperative systems are being developed. Examples of these assumptions are:

1. Computer systems should always automate tedious manual 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. User interface designers should always provide facilities for end-users to tailor interfaces to suit their own personal ways of working and personal preferences.

Our ethnographic observations revealed that the manual manipulation of flight strips and the manual re-ordering of the flight strip racks were significant activities. When a controller is using strips ordered by time of arrival to a beacon and new strips have to be added to the display, it would seem natural for these strips to be positioned automatically in the right place on the user’s display. However, the ethnographic work suggests that 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. This is an important safety device as it allows the early identification of potential problems. Automated positioning is undesirable because these problems may not become apparent until there is an explicit need to use the new strip.

A further example of manual intervention forcing data checking occurs when strips are first printed in the current system. These strips are fitted into different coloured holders by assistant controllers to distinguish between flights on different headings. In this action, 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.

The manual of air traffic control specifies that strip bays should normally be ordered according to the time of arrival of an aircraft over a beacon. This model is followed by the majority of controllers but our ethnographic observations revealed that some controllers have evolved an alternative way of working which uses a different strip ordering. Therefore, while we provide explicit facilities which allow a controller to order his or her display according to the manual, we also immediately switch off this automatic ordering as soon as a manual re-ordering operation is carried out.

To provide an analogue of this manual manipulation, the following requirements for the controller’s system have been derived:

1. Strips should not be positioned according to some default order when they are added to a display.

2. The system should not automate activities such as assigning a strip to a coloured holder because the manual activity forces a feasibility check of the strip information to be made.

3. There should be no automated re-ordering of the flight progress board when a strip is added, removed or edited. Support should be provided for many different possible orderings or ordering as specified by the controller.

It is a common belief amongst developers of user interfaces that user tailorability is essential. Indeed, Fisher and Girgensohn (1990) state:

> “End user tailorability 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”.

While this may be true 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 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 and this rapid assimilation is hindered if even slight differences in strip representations are supported. The requirements identified through ethnography are therefore ‘negative’ requirements. The system must not provide tailorability which will affect immediate understanding of the representation.

Inter-disciplinary working

Working across disciplines is always difficult. Each discipline has evolved its own vocabulary and methods and they may not mesh neatly with those of complementary disciplines. It is hard enough for engineers from different engineering disciplines to work effectively together; it is perhaps an order of magnitude more difficult to bring social scientists into the engineering process. The fact that we have done so with some success is to a large extent due to the willingness of our colleagues in sociology to adapt their working practice and to learn about the engineering process.

The major problems we have discovered can be classified under three headings:

1. *Communication* we had to learn something about each other’s language.
2. *Methodology* we had to learn about each other’s ways of working.
3. *Comprehension* we had to understand the principles underlying each other’s discipline.

These problems are discussed in outline below and, in more detail, by Sommerville *et al.* (1992b).

Communication

Problems of communication are normal in any inter-disciplinary collaboration as each discipline has its own specialised vocabulary. We believe the problems are greater in cases where a science or engineering discipline is collaborating with a social science. A fundamental problem is that each discipline uses normal English words as jargon terms. Practitioners in each discipline had no difficulties in deciding, from the context, when the term usage was in its specialized meaning or when it was in its ‘generally accepted’ meaning. However, realizing that there was a different specialized meaning was a problem for both disciplines.

As an illustration of this problem, we produced a very simple mathematical specification of some abstract data types and stated that this defined the ‘semantics’ of these entities. In essence, 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 the notion of a mathematical specification of semantics to be very alien. They argued that, since meanings were socially negotiated it was impossible to capture their sense by mapping them onto a mathematical system.

When talking of ‘semantics’, we meant the definition of the behaviour of a single entity; the mathematical specification allowed us to communicate this unambiguously to other developers. The sociologists 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 completely different and discipline-specific way and we eventually came to see each other’s point of view.

Methodology

The research methodologies in software engineering and in sociology are completely different. Software engineering is largely concerned with demonstrating the feasibility of concepts by building systems. Abstraction is a key part of this process - the software engineer is always examining the problem at hand to discover abstractions and to produce general rather than specific solutions. Confirmation of theoretical predictions or, indeed, the discovery of theories by analysis of experimental results is not currently a significant research methodology.

The sociology research methods of relevance to our work are based on observation. The sociologist looks at a society to determine its practices, and behaviours. The sociologist may try to fit these observations into an existing theoretical framework or may derive a new theory from them. For ethnographers, all detail is potentially significant and they are reluctant to make abstractions in case this hides potentially important system details.

Accepting that the different methodologies of each discipline are equally valid and accepting the need to reconcile them is extremely important. We have reached this stage of acceptance but have not yet found an effective way of merging our ways of working so that the sociological research can be used in a *systematic* way in the system development process. Working with system developers is making new demands on sociologists as they have an explicit service role in the process. It is likely that ethnography and ethnographic record structures will have to evolve (and perhaps become more subjective) if we are to make systematic links between the ethnographic studies and the system requirements.

Comprehension

Both disciplines have problems in understanding what the other discipline is about (that is, what do sociologists/software engineers actually do). We found this a particular problem as we were used to a hierarchic model of knowledge. When learning a new subject, our normal approach is to tackle the problem by starting with elementary texts then reading progressively more advanced material.

Applying this approach to the understanding of sociology was not successful because knowledge in that discipline is not hierarchically structured. Rather 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.

Furthermore, there is no single notion of what sociologists ‘do’. Software engineers have different specialities but have the shared objective of (somehow) building better computer systems. There does not appear to us to be a comparable broad objective which links practitioners in sociology.

Sociologists undoubtedly have comparable problems in understanding the systems development process. A particular difficulty (perhaps resulting from the poor usability of many personal computer systems) is that the sociologists have no conception of how long a particular system change might take. What appears an immensely complex task to them (e.g. changing the colours on a display) can be accomplished in minutes yet apparently simple system changes, such as adding a new operation to a system, may take several days or even weeks to implement.

Tool support for ethnographic requirements capture

An ethnographic record of work practice is inherently unstructured. It consists of observations of work processes made over an extended period of time. Inevitably, there is a significant amount of duplication and the information collected ranges from specific observations of particular activities to anecdotes and ‘war stories’ told by workers to the ethnographer. The nature of this information is such that it is quite impossible to fit these observations into structured requirements analysis methods whether they be functionally-oriented or object-oriented.

Clearly, however, the ethnographic record must be structured in some way so that it can be usefully used by engineers who did not actually participate in the studies of work. Furthermore, if ethnography is to be useful, there must be forward and backward traceability from the ethnographic record to a more structured formulation of the system requirements. From our practical experience, we do not believe that a methodological approach is likely to be successful in addressing this problem. Rather, we are convinced that an effective way to provide this structure is through software tool support. The tool must integrate the requirements of the ethnographer for informal information capture with support for a more structured approach to requirements expression which allows the requirements to be partitioned and analysed.

Our initial experiments in this area involved the use of the Hypercard system on the Apple Macintosh. While our colleagues in Sociology found this system easy-to-use, it has significant limitations in that it is difficult to show relationships between items and for end-users to extend information collection formats. Furthermore, it is unsuitable as a basis for supporting more structured requirements expression. We are now experimenting with a tool known as the Designer's Notepad (Haddley and Sommerville, 1990) which was originally developed to support the initial phases of the systems design process.

The characteristics of the Designer's Notepad which make it suitable for integrating support for ethnographic analysis with more structured requirements formulation include:

1. A fast, easy-to-use interface which supports the creation of directed graphs. These may be created at any number of levels with simple navigation from one level to another.
2. Untyped entities and relations. This is critically important in the early stages of an analysis where it is unrealistic to fit an entity into a type hierarchy.
3. Extensive annotation facilities which allow system entities and relations to be annotated with structured and unstructured text.
4. Support for multiple versions of system designs. Alternative formulations of a system structure can be established and discussed.
5. Method and type definition facilities which allow the types and rules of a structured method to be defined.
6. Post-creation type attribution where entities may be typed after they have been named and described. These facilities also allow the type of an entity to be changed. For example, an entity which is a data store (say) can be changed to a process entity. The type checking facilities in the DNP trap any inconsistencies which may arise during type changes.
7. Report generation facilities which generate a report on a particular set of requirements or designs.
8. Structured and unstructured information associated with a system entity can be maintained in parallel and it is simple and straightforward to move from an unstructured to a structured system view.

When the system is used for recording the ethnographic analysis, its initial use is typically unstructured where entities are identified and annotated with free text describing some observations (Figure 3).

Figure 3 Unstructured entity annotation

Figure 3 shows an initial view of the process of tearing a new strip off the printer and adding it to a rack. The strip is taken from the printer by an assistant controller (wingman) who checks the strip and may repair errors and generate a new strip. When satisfied with the strip, the assistant fits it into a holder and passes the strip to the radar controller.

The next stage in the process involves adding structure to the annotations whereby *as well as the free text* we define and use structured forms for data collection. The free text is not discarded and is always available for reference and future use (Figure 4). The availability of all representations means that we can trace requrements decisions to their source. The informal information capture provides (at least in part) rationale for system requirements.

Figure 4 Structured entity annotation

In figure 4, we illustrate a process description form where processes are described in a structured way. The inputs and outputs are identified along with the processing steps. In this instance, the process of strip preparation is described.

The addition of structure continues until, finally, we might end up with a data-flow diagram (for example) describing some particular process (Figure 5).

Figure 5 Data-flow diagram of flight strip creation

The data flow diagram is an even more structured representation of the process with activity sequencing identified. Notice that this need not be identical to earlier representations - these have to be interpreted by the requirements engineer to produce the structured description. In this case, the entities and relations which were originally identified cannot readily be mapped onto the types in a data-flow diagram. Therefore, we have opened this design at a deeper level (using the DNP's sub-design facility). It is possible, of course, to view both the structured representation and the source from which it was derived at the same time.

The focus on tool support helps us to address the problem of prolonged ethnography. In essence, the ethnographic studies generate 'nuggets' of useful information at unpredictable intervals. In conventional ethnography, these are made explicit in an analysis phase where the ethnographic record is analysed after the field studies have been completed.

We have adopted a working practice whereby the ethnographer does not work on site for long, uninterrupted periods but returns regularly to report on the progress of the work. Interim records can be entered into the Designer's Notepad. Entering these into the DNP focuses the ethnographer's attention on the records and often suggests useful structuring which can take place. The records then become immediately available to the software requirements engineers who are using the tool concurrently to develop a more structured requirements specification.

Conclusions

We believe that a 'conventional', technically-oriented approach to requirements capture and analysis is inherently incomplete for many classes of software system. In systems where there are multiple end-users, cooperating either explicitly or implicitly, we believe that there is always a dynamic and informal working division of labour which is unlikely to conform to formal organisational structures or job descriptions. If a software system is to be successfully used, it must support actual rather than formal work.

Some general conclusions of our work are:

1. Software engineers and sociologists can work together effectively. However, there is a wide gulf between these disciplines and entrenched philosophical positions will probably ensure that that gulf cannot always be bridged. Effective inter-disciplinary cooperation requires a great deal of flexibility on both sides and requires both sides to question their own assumptions and working methods.

2. Some conventional principles which are normally thought of as 'good design' may be inappropriate for some types of system. Manual intervention and manipulation of information may be essential implicit methods of communication and cooperation.

3. Observational methods such as ethnography can play an important role in informing systems requirements capture and analysis. However, there is unlikely to be a clear and simple correspondence between an observational record and a systems requirement document. The extent of the contribution of observational methods to the system specification process has still to be demonstrated.

4. It is extremely difficult to fit ethnographic observations into structured methods of requirements analysis as these methods factor out an immense amount of (useful) information which is collected during the ethnographic studies. The most effective way to integrate ethnography with structured methods is to provide tool support which allows all relevant information to be collected and which supports the generation of structured views corresponding to whatever structured method is thought to be appropriate.

Our work with air traffic controllers has now reached the stage where initial evaluations of the system are about to begin. Further work is also underway where an ethnographer is studying the systems design process itself. This is a much less structured process and we look forward to the insights for CASE tool requirements which the study might generate.

Acknowledgements

The work described here has been funded by the UK Joint Research Council Initiative in Cognitive Science and HCI. Thanks are due to our collaborators in the Department of sociology namely John Hughes, Dan Shapiro, Dave Randall and Val Kyng.

References

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

Bentley, R., Sawyer, P., Rodden, T. and Sommerville, I., 1992, ‘A Prototyping Environment for Dynamic Data Visualisation’, Paper to be presented at *5th IFIP Working Conference on User Interfaces*, Ellivuori, Finland, 10th-14th August, 1992.

Fischer, G. and Girgensohn, A., 1990, ‘End-User Modifiability in Design Environments’, *Proc. CHI '90*, (Seattle, Washington), ACM Press, 183-191.

Haddley, N. and Sommerville, I., 1990, ‘Integrated Support for Systems Design’, *IEE/BCS Software Engineering Journal*, 5 (6), 331-338.

Harper, R., Hughes, J. and Shapiro, D, 1991, ‘Harmonious Working and CSCW: Computer Technology and Air Traffic Control’, in J.M. Bowers and S.D. Benford (eds): *Studies in Computer Supported Cooperative Work. Theory, Practice and Design*, North-Holland, Amsterdam.

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

Heath, C. and Luff, P., 1991, ‘Collaborative Activity and Technological Design: Task coordination in the London Underground Control Rooms’, in *Proc. ECSCW'91* (Amsterdam, Sept 25- Sept 27), Kluwer, 1991.

Hopkin, V.D., 1991, ‘The Impact of automation on Air Traffic Control Systems’, in. J.A. Wise, V.D. Hopkin and M.L. Smith (eds.), *Automation and Systems Issues in Air Traffic Control,* Berlin: Springer-Verlag.

Sommerville, I., Bentley, R., Rodden, T., Sawyer, P., Hughes, J., Shapiro, D. and Randall, D., 1992a, ‘Ethnographically-informed systems design for air traffic control’. *To appear in Proc. CSCW’92*, Toronto, November 1992.

Sommerville, I., Rodden, T.A., Sawyer, P., and Bentley, R., 1992b, ‘Sociologists can be Surprisingly Useful in Interactive Systems Design’, *To appear in Proc. HCI’92*, York, September 1992.

Suchman, L., 1983, ‘Office procedures as practical action’, *ACM Transactions on Office Information Systems,* 1, 320-328.

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