1 Introduction: Dependability and Responsibility in Context

John Dobson, Ian Sommerville, and Guy Dewsbury

1.1 What this book is about

This book looks at socio-technical systems, that is systems which consist of a group of people working with some complex technology in order to achieve some common purpose. We shall be dealing in the main with the case of the technology being a computer system, though the ideas we shall present are applicable to other forms of technology, and we shall also discuss them in the context of a railway system. The main reason for looking at socio-technical systems is to explore the extent to which ideas of dependability, which have been developed for technical systems for some decades now, can be applied to socio-technical systems. It is no longer good enough simply to say that a failure was due to a computer error; in most cases there was a human error along the line too. We shall be looking at what sorts of thing can be said about error that applies to both computer error and human error. These ideas are not so much concerned with what causes errors, but how errors can be prevented or recovered from. It is this focus on prevention and recovery that led us to understand that there are indeed some concepts and structures that are common to the ways that errors are managed in both technical and human systems, though the actual causes may well be of very different kinds.

We now introduce two major concepts that this book is about: dependability and responsibility. Dependability is a term that has been used by computer engineers for three decades or more to mean that the computer system can be trusted to do what it is supposed to do, and our discussion of dependability in this introduction will closely follow the standard texts. However, as we will see, some of the thinking has to be revised somewhat if it is to be applied to the kind of socio-technical systems that are the subject of this book. We shall show that these revisions bring in human concepts of responsibility –roughly, that doing what you're supposed to do means discharging the responsibilities that you have been given or have assumed– and that this reinterpretation leads to ways of thinking about the relationship between people and computers in carrying out some socio-technical task to be performed by people and computers working together.

We shall then introduce the idea of modelling responsibilities, saying something about what we mean by a 'model'. We shall explain that making a model of responsibilities is a way of building a bridge between the social discourse of responsibilities for tasks and states of affairs in the world, and the architectural or engineering discourse of making an artifact that assists a person in performing those tasks or bringing about (or maintaining or preventing) those states of affairs.

Since this book is about the social aspects of a socio-technical system as well as the technical ones, it will discuss the use of ethnographic methods to discover how responsibilities actually lie in an organisation. This chapter therefore also contains a brief introduction to the use of ethnographic studies in understanding responsibilities.

These basic ideas –dependability, responsibility, modelling and ethnography– and the relationships between them are expanded later in the book, so although the reader may well be justified in thinking that this introduction raises questions that it does not answer and that there is more to be said, we hope the remainder of the book will go some way to saying more and answering at least some of these questions.

1.2 Dependability

As indicated earlier, the concept of dependability in computer systems has been around for more than three decades. We cannot do better in introducing the topic than presenting an extended summary of the standard texts (Avizienis, Laprie et al. 2004). Later on in this chapter we shall argue that the dependability of socio-technical systems encompasses more and sufficiently different things than a computer system alone and the original connotations of dependability summarised here need some extension.

The dependability of a computing system is the ability to deliver service that can justifiably be trusted. It is an integrative concept that encompasses such things as: availability (readiness for correct service); reliability (continuity of correct service); safety (absence of catastrophic consequences on the user(s) and the environment); confidentiality (absence of unauthorised disclosure of information); integrity (absence of improper system state alterations); maintainability (ability to undergo repairs and modifications). The service delivered by a system is its behaviour as it is perceived by its user(s); a user is another system (physical, human) that interacts with the former at a service interface. The function of a system is described by the system specification. Correct service is delivered when the service implements the system function. (This may or may not be the intended use of the system, since the system specification may incorrectly describe the intended use.)

A system failure is an event that occurs when the delivered service deviates from correct service, i.e. it is a transition from correct service to incorrect service. An error is that part of the system state that may cause a subsequent failure: a failure occurs when an error reaches the service interface and alters the service as delivered from the service as specified. A fault is the adjudged or hypothesised cause of an error. A fault is active when it produces an error, otherwise it is dormant.

A system consists of a set of interacting components, therefore the system state is the set of its component states. A fault originally causes an error within the state of one (or more) components, but system failure will not be recognised until the error becomes visible at one or more interfaces at which other system components interact with or observe the failed component. An error may be detected and its presence in the system indicated by an error message or error signal that originates within the system. Errors that are present but not detected are latent errors.

The relationship between faults, errors, and failures is summarised by the following figure, which gives the fundamental chain of threats to dependability. The arrows in this chain express a causality relationship between faults, errors and failures. Because a system contains a number of components, a failure in one component may cause (or be treated as) a fault in a larger component that contains the failing component. Also, several errors may be generated before a failure occurs.

Figure 1.1

$$\begin{tikzcd} \dots \arrow[r] & \text{fault} \arrow[r, "\text{activation}"] & \text{error} \arrow[r, "\text{propagation}"] & \text{failure} \arrow[r, "\text{causation}"] & \text{fault} \arrow[r] & \dots \end{tikzcd}$$

It is an important to understand that this chain can go from a system to a larger system of which it is a component, or to a separate system that it is interacting with, or to a system that it creates or sustains. It is therefore equally important to be clear about exactly which computer or socio-technical system is under consideration, particularly when the discussion concerns a fault in one system causing a failure in another system, which manifests itself as a fault in a third system; an example of this was given at the end of the preface where the failure was in a social system (the home), the error in a technical system (the technology procurement system) and the fault in another social system (the commissioning system). As already mentioned, however, determination of system boundaries is not always easy or even agreed, and the judgement as to the fault(s) that caused a particular failure will, especially in the case of socio-technical systems, often depend on the actual process of analysis that was used.

1.2.1.1 The Means to Attain Dependability

The development of a dependable computing system calls for the combined utilisation of a set of four techniques: *fault prevention*: how to prevent the occurrence or introduction of faults; *fault tolerance*: how to deliver correct service in the presence of faults; *fault removal*: how to reduce the number or severity of faults; *fault forecasting*: how to estimate the present number, the future incidence, and the likely consequences of faults.

Fault Prevention

Fault prevention is attained by quality control techniques employed during the design and manufacturing of hardware and software.

Fault Tolerance

Fault tolerance is intended to preserve the delivery of correct service in the presence of active faults. It is generally implemented by error detection and subsequent system recovery.

Error detection originates an error signal or message within the system. There exist two classes of error detection techniques: (i) concurrent error detection, which takes place during service delivery; and (ii) pre-emptive error detection, which takes place while service delivery is suspended or before it commences; it checks the system for latent errors and dormant faults.

Recovery transforms a system state that contains one or more errors and (possibly) faults into a state without detected errors and faults that can be activated again. Recovery consists of error handling and fault handling. Error handling eliminates errors from the system state. It may take two forms: (i) rollback, where the state transformation consists of returning the system back to a saved correct state that existed prior to error detection; (ii) rollforward, where the state without detected errors is a new state.

Fault handling prevents located faults from being activated again. Fault handling involves four steps: (i) fault diagnosis that identifies and records the cause(s) of error(s), in terms of both location and type, (ii) fault isolation that performs physical or logical exclusion of the faulty components from further participation in service delivery, i.e., it makes the fault dormant, (iii) system reconfiguration that either switches in spare components or reassigns tasks among non-failed components, (iv) system reinitialisation that checks, updates and records the new configuration and updates system tables and records. Usually, fault handling is followed by corrective maintenance that removes faults isolated by fault handling. The factor that distinguishes fault tolerance from maintenance is that maintenance requires the participation of an external agent.

Fault tolerance is a recursive concept: it is essential that the mechanisms that implement fault tolerance should be protected against the faults that might affect them.

Fault Removal

Fault removal is performed both during the development phase, and during the operational life of a system. Fault removal during the development phase of a system life-cycle consists of three steps: verification, diagnosis, correction. Verification is the process of checking whether the system adheres to given properties, termed the verification conditions. If it does not, the other two steps follow: diagnosing the fault(s) that prevented the verification conditions from being fulfilled, and then performing the necessary corrections.

Checking the specification is usually referred to as validation. Uncovered specification faults can happen at any stage of the development, either during the specification phase itself, or during subsequent phases when evidence is found that the system will not implement its function, or that the implementation cannot be achieved in a cost effective way.

Fault removal during the operational life of a system is corrective or preventive maintenance. Corrective maintenance is aimed to remove faults that have produced one or more errors and have been reported, while preventive maintenance is designed to uncover and remove faults before they can cause faults during normal operation.

Fault Forecasting

Fault forecasting is conducted by performing an evaluation of the system behaviour with respect to fault occurrence or activation. There are two sorts of evaluation: (1) qualitative evaluation, that aims to identify, classify, rank the failure modes, or the event combinations (component failures or environmental conditions) that would lead to system failures; (2) quantitative, or probabilistic, evaluation, that aims to evaluate in terms of probabilities the extent to which some of the attributes of dependability are satisfied; those attributes can then viewed as measures of dependability.

1.3 Responsibility and Role

The reason for starting from responsibility as a way of distinguishing the social from the technical in socio-technical systems is partly philosophical — that it is responsibilities that provide the basic structuring of society — and partly pragmatic because, as we hope to show, understanding responsibilities can lead to important considerations in the architecture of information systems that other current philosophical bases tend to ignore. Those who are interested in the philosophical concept of responsibility should read the next chapter and perhaps the book by Jonas (Jonas 1984) for a deep discussion of its importance in society and the interpretation of responsibility in the context of the relationship between society and technology. However, Jonas takes a view of responsibility which encompasses moral responsibility, which examines questions such as "What is the good and why should I attempt to commit to it and what form should such commitment take?", which –although of undoubted importance– are perhaps of less relevance to information systems design. A more general exposition of the relationship between philosophy and information systems can be found in (Hirschheim, Klein et al. 1995).

We will have more to say about responsibilities later on, but we want to introduce here a closely related concept, namely that of role. We treat a role as a collection of responsibilities that in some sense go together. A role can be a purely social role, such as parent, a partly social and partly professional role such as doctor, or a fully professional role such as office secretary. Such roles are defined by the sorts of responsibility that they entail and the spaces or domains in which they are given (or assumed) and discharged.

Many conceptualisations of role in information systems treat it as a set of capabilities, such as information that can be accessed, tasks that can be performed. But where do these capabilities come from? Who grants them? — The answer lies in the prior notion of responsibility. Many workarounds (where capabilities have for whatever reason been inappropriately denied) can be seen as an alternative error recovery strategy to fulfil responsibilities. For example (and a colleague has reported to us something like this), a hospital ward sister might not be allowed by the computer system to reserve a vacant bed for a patient known and expected to be coming in later, but she can deliberately fail to notify the system that the bed has become vacant. In doing so she is showing her responsibility to the incoming patient by ensuring that a bed is available.

In fact this example shows a very simple example of responsibility conflict in roles: the sister also has a responsibility to the hospital for efficiency in bed management which would require her to count accurately the number of vacant beds. These everyday conflicts of responsibility are probably ubiquitous in organisational life, and proper recognition, discussion and resolution of them has major implications for the process of producing a computer system architecture designed to support the work role in which the conflicts occur. This problem is well discussed in (Dahlbom and Mathiassen 1993).

Thus a role is a set of responsibilities possibly with a number of different authorities. For example, a doctor has responsibility to the patient, to the practice, to the National Health Service, to society as a whole. These can conflict, and there may be rules for conflict resolution: in certain circumstances, the duty of disclosure overrides the duty of patient confidentiality. In other cases, however, there may be no specific rules for conflict resolution and the resolution process may be individual (What do I think is right in the circumstances?) or legal (What should have been done? Was the decision in fact reasonable?). There may be guidelines to provide assistance in developing a way of resolving conflicts, but each practice is required to develop its own role definitions, though these are often left deliberately vague.

So we can characterise responsibility as a relationship between two roles regarding a specific state of affairs with respect to a particular mode such as bringing about, preventing, maintaining, accounting for and so on, such that the holder of the responsibility (the responsible) is responsible to the giver of the responsibility, the authority. The important point here is that responsibilities cannot be looked at on their own but must always be considered as a relationship between two roles. The states of affairs for which responsibilities are held may be at any level of granularity of the organisation. For example the responsibilities may be at a very high level such as for the financial soundness of the organisation, for the continuity of the services provided by the organisation, for safety, for security, for the accuracy of information and suchlike, or they may be at an individual level for a very specific state such as whether a particular door is closed, or whether a particular form is correctly filled in.

We now introduce an important distinction between causal responsibility, when an agent has an obligation to make something happen or prevent it from happening or to maintain a state, from consequential responsibility, when an agent is answerable when something happens or does not happen or a state is not maintained. These different responsibilities do not always rest on the same agent (the doctrine of 'ministerial responsibility' whereby elected politicians in charge of a department take responsibility for errors in their department even though they may be unaware of them). Consequential responsibility may be held to rest with an organisation as a whole whereas causal responsibility most usually can be traced to an individual or the fact that no particular individual at the time held the responsibility. Causal responsibility may sometimes be delegated, though some responsibility remains with the delegating agent (i.e. the responsibility for having chosen to delegate), whereas consequential responsibility is not normally capable of delegation, though it may sometimes be transferred.

Causal responsibility is the responsibility for doing something. This can be something explicit (A is responsible for locking the building; B is responsible for checking that the building is secure) or implicit (A is responsible for building security). In the latter case, it is assumed that A decides on what actions are appropriate. Thus it can be reformulated as A is responsible for deciding on what to do, then doing it. Although causal responsibility can be examined by methods such as task analysis, and asking questions about what information is needed and needs to be recorded in order to fulfil the responsibility, it is important to look at the whole set of responsibilities that define a role. It is also important to look at the nature of any shared or delegated responsibilities, and at conflicts of interest. Where a new information and communication technology (ICT) system is being procured, it is an important part of any requirements elicitation process to discover and examine all these things.

Shared causal responsibilities can be a form of fault tolerance but they have their own vulnerabilities too. If there is inadequate means of communication between the parties, or an inadequate protocol, there is the possibility of a particular action being taken twice or more, or not at all. Usually a properly performed vulnerability analysis is capable of revealing these possibilities, but such an analysis is often not done as part of a requirements exercise.

Problems with responsibilities often arise when the actions associated with a consequential responsibility or the activities for which a causal responsibility exist cut across organisational boundaries. These problems may arise because of differences in interpretation of responsibilities, because of differing priorities in time and resource allocation in different organisations, because of differences in competence, because of different organisational policies, etc.

By default, the holder of a consequential responsibility is causally responsible for the actions associated with that responsibility or for delegating that causal responsibility. Consequential responsibility does not require the use of resources to discharge the responsibility (resources, of course, are required for any associated causal responsibilities). Thus consequential responsibility leads to requirements too, but they may be more indirect. Information recording and audit trail processes can be important in determining how a particular responsibility was discharged. Shared consequential responsibilities are particularly difficult.

Although causal responsibility can be restated using dependability terms (e.g. the location of a fault, perhaps), consequential responsibility is something that cannot be restated, since invoking it after a failure is not necessarily part of a causal chain, or a repair, and the person responsible may not have had any agency over the failure. However, it is incumbent on holders of consequential responsibility to protect themselves by ensuring that everything possible is done to prevent or tolerate failure, and this may generate system requirements for dependability which might not at first sight be obvious from a purely functional or behavioural point of view.

In summary, roles and responsibilities are complex things, and simplistic models of them lead to inappropriate system architectures. Too often, a failure to perform a vulnerability analysis leads to a system which makes optimistic assumptions about they way that people have performed, or will perform, their duties, and about the effectiveness of their human communications.

1.4 Dependability and Responsibility

Section 1.2 has described the use of ‘dependability’ and related concepts as they are currently used in computer system design. This section looks at how these concepts could be applied to socio-technical systems.

Service

It would be easy to think of a human definition of service purely in behavioural terms — i.e. in terms of what people do. But this approach is too reductionist. It works for computers because the only thing that computers do is to behave; but the days of behaviourism as a complete explanation of human behaviour have long since gone. We therefore choose to take an understanding of service in socio-technical systems as *discharging responsibilities*, so as to include the motivation for people doing what they do. As the previous section has indicated, the concept of responsibility is more complex than a simple behavioural account can provide, and is examined in more detail in the next chapter. For the purposes of this introductory chapter, however, we want merely to indicate how the basic concepts of dependability can be reinterpreted in the context of human activity.

Specification

It is preferable in designing a system to have a correct and useful specification, but there may be none or it may be wrong. One difference between computer and socio-technical systems is that for a computer system, the specification is expressed purely in behavioural terms, whereas a specification of a human task may be in terms of goals and responsibilities, leaving it open about how they are to be achieved. Although goals can be reduced to behaviours for both computers and people, people are usually freer to take spot decisions in the moment which often involve dynamically changing the goal structures in response to event as they unfold in the world, arguing that maintaining their responsibilities requires such changes, and that specifications are dynamic things, subject to reinterpretation as its context changes. This is particularly the case in the presence of failure.

Dependability

In ordinary usage, 'dependable' as applied to a person does in fact mean that they do something they have undertaken to do, i.e. a responsibility they have been given or assumed. This is a useful starting point, but the dependability of socio-technical systems can usefully be extended by taking account of the terms 'fault, 'error' and 'failure' with roughly the same meanings as in the language of computer systems.

Fault

There are two distinct connotations of 'fault' in a human context: (i) blameworthiness: whose fault was it? and (ii) causal: faulty thinking, action or judgement leading to erroneous behaviour. These are often combined, so that the blame lies on the faulty thinker, actor or judge, though the law can assert otherwise — the driver of a train going though a red signal and causing an accident was at fault, but the blame may lie with the train operating company, perhaps because it failed to train the driver properly.

It is in order to distinguish between these two meanings of fault that we introduced two meanings of responsibility: (i) *causal* responsibility, which rests with the actor who performed the erroneous action, the actor being in some way faulty, and (ii) *consequential* responsibility, which rests with the actor who takes the blame, the actor being in some way liable. Determination of the actor with consequential responsibility may involve a judicial process.

To blame "the computer" is often merely to say that a computer was implicated, or to claim "whoever was to blame, it wasn't me". If we take 'the computer' to mean the whole computer system including its specifiers, designers, implementers, procurers and data providers, than blaming the computer is saying that the fault lies in there somewhere, but is also a refusal to analyse it any further.

Error

In the computer world, an error is a behaviour –a deviation from a specification– and is a link in a causal chain. In the human context, this carries over, an error being a behaviour. Erroneous or mistaken beliefs ("I thought the signal showed green") are analogous to faults in the computer account of dependability.

Failure

It is in the concept of failure that the difference between computer and human contexts becomes most marked. The computer construal depends on two things: that a specification of correct behaviour exists and is unambiguous; and that a deviation from such a specification can be objectively determined, i.e. that all (competent) observers will agree on whether a behaviour is incorrect.

Neither of these is true in a human context. A specification may or may not exist, or not be specific enough to give useful design or operational guidance; and whether an individual has failed in a task is a mater of judgement on which different judges may legitimately differ (e.g. "Is George W. Bush a failure as US president?"). Failure is socially determined and the concept of 'failure' is of type 'judgement' and not of type 'behaviour'.

Judgement

The dependability definitions make a point of stressing that everything is relevant to the viewpoint being taken and the vital role that judgement plays –in identifying system boundaries, in recognising failures, in identifying their cause(s)_ whenever one has highly complex systems and situations. Although one does not need the presence of humans for such complexity to arise, or for such judgements to be needed, one may very well need humans to cope with failure in the system or in the system that is making the judgement.

Fault avoidance removal tolerance and forecasting

The fault-error-failure chain previously described does sometimes have a close correspondence in socio-technical systems, as do the concepts of fault avoidance, removal, tolerance and forecasting, though the techniques used are much less technical and prescribed. Fault avoidance is essentially ensuring that potential faults in a process never make it in to the implemented version. Though there is nothing in the social domain that correspond to the mathematical techniques that are deployed in the technical domain, it is possible to employ a systematic way of thinking about the design of a process, possibly with the assistance of models, which involves at every stage asking the question “On what am I relying on here and how do I know it is trustworthy?” In formal computer science, this is known as a rely-guarantee condition. Similarly there are techniques for fault removal that can be employed in the social domain — fire drills in an organisation is a simple example. Role playing and simulated exercises are commonly used in the emergency services and armed forces, where fault removal matters. Fault tolerance is widely used in practice — the use of alternatives (e.g. if one person is away, another deputises instead), independent checking (e.g. auditors), duplication (e.g. cheques that require two signatures), all are standard fault tolerance methods that recognise the possibility that people cannot always be relied on. Only fault forecasting seems to have little application in the social domain. Partly this is because the techniques are obviously more mathematical and partly because any forecasting has to be done on the basis of a repeatable or stochastically regular model. (By this is meant one whose uncertainty is predictable. For example, bias in a coin can be detected by tossing it a number of times and performing a statistical analysis on the results, but only if the bias is constant for each toss; it can’t be done if the bias changes randomly for some reason from toss to toss.) It is an open question whether and to what extent bias in people is sufficiently regular for forecasting to be reliable.

Vulnerability analysis

There are many tools notations and techniques concerning such things as system evaluation, safety cases, fault diagnosis and risk assessment that are used in the dependability community in the process of vulnerability analysis (“What can possibly go wrong, how likely is it and how serious would it be if it did?”), and some of these have influenced our methods of vulnerability analysis for responsibilities, which will explained in later chapters of this book. What is new is not the techniques that we use, but that we can demonstrate their usefulness in diagnosing such problems as missing or inappropriately assigned respo0nsibilities, confusion about roles and the allocation of responsibilities, and failures in the conversations that occur between actors who are sharing responsibilities.

1.5 Responsibilities in socio-technical systems

Achieving dependability in a socio-technical system isn’t just a matter of ensuring that the computer behaves correctly. Not only does this usually not happen, but there are often discrepancies between what it does do when working correctly (i.e. in accordance with its specification) and what users expect it to do. The fact that the specification can be wrong with respect to the expectations means that people will try workarounds to attempt to get the computer to assist in what they see as the discharge of their responsibilities.

In some cases this is because users take a broader view of what their responsibilities are than the view which has been inscribed in the computer system by its procurers and designers. This may be because of a poorly researched requirements phase or because those who commissioned the system had a different organisational agenda from those who use the system. Although requirements elicitation personnel are often encouraged to pay attention to the users, they cannot be blamed for paying more attention to those who pay them. This again is an issue of conflicting responsibilities. Most experienced requirements engineers who have engaged in socio-technical systems design will agree that the politics of the job is at least as important as the engineering, but calls for a different set of skills which cannot be taught but have to be learnt — something which is characteristic of conflict resolution.

Another characteristic of socio-technical systems is that the actors in them can and do dynamically change their goals and responsibilities in response to unfolding events in ways that could not have been foreseen at the time the original specification was drawn up. Although these changes are often implemented by those directly concerned with the operation of the system, they are sometimes associated with indirect stakeholders who do not interact directly with the system through a service interface but who nevertheless influence it and are themselves influenced by its existence. A common example is that it is important for the career of a politician who has fought for money for a new system that the system procured is judged to be successful, even if this means the success criteria have to be changed from those originally intended.

So dependability, in the sense of the system behaving as its users expect and supporting them in carrying out their responsibilities, is for socio-technical systems just as likely to be a political or organisational issue as a technical one. This cannot be avoided, and requirements engineers and system designers must be sensitive to it and have ways of dealing with it or accommodating it in the results of their practice. (Dahlbom and Mathiassen 1993)

This political aspect means that understanding responsibilities in an organisation and its relationship to dependability cannot be dealt with by a procedural method; each case must be approached in its own way depending on the nature of the organisation and the way that the people in it interpret their responsibilities. So although it probably does not make much sense to try to present a particular way of *doing* to achieve dependability of the socio-technical system, it is possible to present a particular way of *thinking* that will assist the requirements engineer or system designer in approaching their problem. We claim that the way of thinking we shall present is coherent because, as we hope to show, the concepts used apply equally well to thinking about the human aspects and thinking bout the computer support for the human aspects.

1.6 Using ethnographic studies to understand responsibilities

The inherent complexity of responsibilities in a large organisation and their negotiated, dynamic nature presents a major challenge to the responsibility modeller. How can we understand the real responsibilities in complex socio-technical settings? In this section, we discuss how ethnographic studies can be an effective approach to collecting information about responsibilities.

The conventional approach to understanding who is responsible for what in an organisation is to start with a formal organisation chart, identify the individuals associated with roles and interview them to discover the type and nature of their responsibilities. However, while such an approach can be a starting point, we believe that it suffers from serious flaws:

1. Organisation charts are usually formal, over-simplified views of complex organisations. The reality is inevitably more complex and messy.
2. Responsibilities are often implicit rather than explicit – people find it difficult to articulate what they really do.
3. Responsibilities are contingent and dynamic – people take on responsibilities depending on the particular tasks to be done. This is fundamental to the effective functioning of most organisations – it is only in a ‘work to rule’ culture, where people refuse to take on tasks outside of a narrow job description, that it is uncommon. Hence, again, articulation of these responsibilities is difficult.

An approach which we have used with some success to understand responsibilities in complex organisations is ethnography (Garfinkel 1967; Crabtree 2003) where an experienced social scientist spends a period of time observing the ways in which work is done, the dynamic division of labour in a particular setting and the ways in which the artefacts and the physical organisation of a setting influence the work carried out.

Ethnography is a method of data capture that works through the immersion of the researcher within the environment being studied, collecting detailed material (notes, documentation, recordings) on the ‘real-time real-world’ activities of those involved. Periods of immersion can range from intensive periods of a few days to weeks and months (more common in systems design studies), and even years. The style of ethnography that we have used is so-called ethnomethodological ethnography where the ethnographer does not attempt to fit these observations into some social theory. Rather, they are simply presented as an unbiased commonsense account of what goes on.

The primary product of most ethnographies is the development of a highly specific ‘rich’ description – a detailed narrative – of the work or activity in question, which may then be further *analysed* or *modelled* for various means, taking various approaches. In this case, this narrative forms the means through which we can develop an understanding of the real responsibilities in an organisation.

Our style of working involves periods of ethnographic observations that are inter-leaved with design work informed by these ethnographic studies and observations. Essentially, the ethnographer is debriefed by the members of the design team who then propose models based on this information. These are critiqued by the ethnographer on the basis of their knowledge, then further refined. Specific questions are identified and the next phase of the ethnographic study tries to answer these questions, as well as to gather additional information about the observed setting. Eventually, a model should be derived that the ethnographer feels is an accurate representation and this can then be taken back to the participants in the study for further comment.

Our motivation for exploring the use of ethnography to understand work was to better understand the real requirements of users of complex IT systems such as those used to support air traffic control, hospital patient administration, etc. Our contention was that ethnographic studies would reveal how people 'worked around' the problems with computer systems and developed coping behaviour when things went wrong. Over the last ten years or so, we have carried out ethnographies in a range of settings from air traffic control rooms, through financial institutions to steelworks (Sommerville, Bentley et al. 1992; Harper, Randall et al. 2000; Hughes, Martin et al. 2003; Martin and Rouncefield 2003).

A universal characteristic of all of the sites that we have studies is that one of the major problems that arise in the use of complex IT systems is that these systems often include an implicit model of responsibilities which:

(a) may not be configurable for each specific setting where the system is deployed;

(b) rarely if ever copes with the dynamic and contingent nature of responsibilities;

(c) does not recognise the critical distinction between causal and consequential responsibility and, hence, makes invalid assumptions about how work is actually carried out.

To illustrate the nature of responsibility as discovered through our ethnographic studies, we will here draw on an example from recent ethnographies that we have carried out in hospitals (Clarke, Hartswood et al. 2002; Clarke, Hartswood et al. 2002). For this example, we will highlight how commonly adopted approaches to information systems design can result in real problems for system users.

The example we will use concerns the administration of chemotherapy treatment of patients suffering from cancer. The cocktail of drugs which is administered to each patient is prescribed by the oncology consultant who has the (consequential) responsibility for treating that patient. The consultant must arrange for the appropriate drugs to be available for each chemotherapy session. The (causal) responsibility for the treatment session may, of course, be devolved to a more junior doctor. Maintaining patient records was, however, the responsibility of the nurse assisting with the chemotherapy.

In practice, hospital consultants are very busy people and they often forgot to order the required chemotherapy medication for particular patients. If this was not available, the treatment session had to be cancelled and re-scheduled – a distressing experience for patients who were often very ill. To avoid the problem, the clinic staff had devised a work-around – the day before a patient was due, a nurse checked if the required drugs had been ordered. If not, with the connivance of the staff in the hospital pharmacy, the nurse placed an unsigned order that was then replaced in the pharmacy with a signed prescription whenever the consultant was available.

Our studies have shown that such coping behaviour is extremely common in complex socio-technical systems and is fundamental to the dependability of these systems. In this situation, the essential problem was a problem of responsibility. There was an assumption that the consequential responsibility of the consultants for prescribing the medication was equated with the causal responsibility of physically drawing up a signed prescription.

Now consider what might happen if the process was automated. A secure system here would associate prescribing permission with a consultant and not with a nurse. It would be difficult for a nurse to repair the problem of forgotten orders. Workarounds, which are not strictly according to the rules, would be more difficult although, human ingenuity is such that they would surely be discovered.

1.7 Responsibility Modelling

An important part of this book deals with modelling. Over the years we have developed a way of modelling responsibilities, of which a detailed account is given in a previous book written by members of the DIRC project (Clarke, Hardstone et al. 2006) and is extended here. We shall not repeat or precis that in this introduction since it is dealt with in a later chapter, but look instead at what lies behind our approach to modelling.

The importance of our models lies in the processes in which they have a role to play. Again, we shall say more about these processes in later chapters, but it is worth explaining here that the background to our process-oriented use of models lies in the soft systems methodology (SSM) developed by Checkland (Checkland 1981; Checkland and Scholes 1990). In SSM, models are used as a way of facilitating discussion. One way it does this is by taking a normative model –that is, a model of what something should be like to count as an example of that thing– and comparing it with a descriptive model –that is, a model of a facet of reality– and using any discrepancies between them as a starting point for discussion on what things perhaps need to be changed and in what way they might be changed. Our responsibility models can be used in this way. Another way in which our models can be used is to act as a bridge between the social and the technical aspects of a socio-technical system design. Briefly, we describe responsibilities and the conditions (resources, competencies etc.) needed for their successful discharge. We can then say something about the resources required and the actions that have to be performed and use this information to build a workflow or similar model. A third way we can use our models is as a way of summarising a piece of ethnography. The reasons why it might be desirable to provide such a summary might be to focus a discussion, to explore options for new ways of working or the introduction of ICT, or to explore similarities between one ethnographically observed setting and another.

Finally, it is important to realise that our models are necessarily incomplete and not necessarily correct. There is often no point in decomposing responsibilities down to the finest level of granularity since at that levels responsibilities are often just assumed (anyone can choose to tell the office manager that more paperclips need to be ordered). Our models of responsibility are used to look at the allocation of and communication between major responsibilities and to focus discussion and attention. For this purpose, a model that is wrong is often just as useful than one that is right, since it forces the articulation of a proposed correction which may lead to a discussion of whether the proposed correction is in fact correct. Once the relevant actors have agreed on a model, it can be recorded as an agreed representation and the process of turning it into something fit for input into an engineering process can begin.

1.8 The Social, Socio-technical Systems and Responsibility

It is not an afterthought that we include three chapters in this book that emphasise an ethnographic stance on considering responsibility. We all have common everyday notions of what responsibility means, but these notions are often the cause of many systemic organisational failures. The assignment of responsibility and the way in which causal and consequential responsibility are understood and enacted directly effects outcomes in many situations.

A significant part of DIRC looked specifically at how people interacted with technology from a social perspective. This meant sociologists and other social scientists working with computer scientists in the development of software and computer systems. These ranged from medical and allied medical investigations (neonatal units, mammography, assistive technology, electronic patient records etc) through to more organisational settings, such as finance houses and factories. The use of social investigations is a strength of the research that was conducted and the way in which responsibility was enacted and deployed in these different locations was, of course, in different manners. People and organisations use responsibility in different ways to achieve differing goals, and for this reason an ethnographic approach is useful in uncovering what these goals are and how they change in response to the changing world.

One important facet of dependability that is often not given enough attention is usability. This is particularly true for systems and appliances used in home settings, and it is here that ethnographic methods are particularly useful. One example that the DIRC project investigated was the use of assistive technology in the home for the elderly and the disabled, where we found many examples of unsuitable configurations and devices. Technology that is unused because it is unsuitable for its intended use is undeniably a failure; but examining the causal chains in order to determine where the causal and consequential responsibilities lie can be very difficult because of the fluidity and unclarity of the boundaries of responsibility, particularly in domestic settings, where the agencies involved will include social services (local government), the national health service, the housing association in the case of sheltered accommodation, and the technology suppliers and installers. Each of these will have their own areas of responsibility, but even so the responsibility for usability may well fall between them all so that no-one has clearly identified ownership of it and agency to do something about it.

Anecdotal reports suggest that usability and other failures in multi-agency systems are very common. We suggest that one of the reasons why this is so is that processes for identifying and reconciling complex and conflicting responsibilities can be time-consuming and difficult to manage. We believe that the role of ethnographic enquiry and action research interventions in system development projects should not be seen as exclusively a means of enhancing 'requirements capture' and the work of system design and development. Rather there is a new role which now needs to be explored in the development of generic frameworks, techniques and guidelines which support the development of resources based on generic responsibilities and which takes into equal account the three domains of technology, governance and participation mentioned in the Preface. This role can act as the means of knowledge transfer from different domains (for example between public service agencies and private sector system suppliers). Through this kind of engagement socio-technical techniques can be deployed in the shaping of frameworks through which new technological opportunities are exploited rather than restricted to finding ways of matching social variables to system applications whose essential characteristics have already been determined. Our modelling has been developed to assist in this process, but it is the shaping process and the role of modelling within it that is important, not the actual syntax of the models, which can be agreed and changed at will.

Thus the kinds of responsibility modelling introduced in this book are intended not only to support the processes of system development but also the processes of pre-development such as the ones just mentioned which are concerned with the establishment of system boundaries and the identification of responsibilities that are either missing or inappropriately allocated, or outside the boundary but should perhaps be within it (and of course *vice versa*), or that are implicitly shared but without adequate communication for the sharing to be successful. It is by this criterion that we hope the book will be judged.

1.9 References

Avizienis, A., Laprie, J.-C., Randell, B. and Landwehr, C. (2004). "Basic Concepts and Taxonomy of Dependable and Secure Computing." IEEE Transactions on Dependable and Secure Computing *1*(*1*): 11-33.

Checkland, P. (1981). *Systems Thinking, Systems Practice*, J. Wiley & Sons.

Checkland, P. and Scholes, J. (1990). *Soft Systems Methodology in Action*. Chichester, John Wiley.

Clarke, K., Hardstone, G., Rouncefield, M. and Sommerville, I. (2006). *Trust in Technology: A Socio-technical Perspective*. Dordrecht, Springer.

Clarke, K., Hartswood, M., Procter, R., Rouncefield, M. and Slack, R. (2002). *Some Practical Problems of Integrating and Interpreting Information Technology in a Hospital Trust*. Proc. BCS Conf. on Healthcare, Harrogate.

Clarke, K., Hartswood, M., Procter, R., Rouncefield, M., Slack, R. and Williams, R. (2002). "Knife to Skin Time: Process Modelling and New Technology in Medical Work." Health Informatics Journal *8*(*1*): 41-44.

Crabtree, A. (2003). *A Practical Guide to Ethnography*. London, Springer-Verlag.

Dahlbom, B. and Mathiassen, L. (1993). *Computers in Context*. Cambridge, MA and Oxford, UK, NCC Blackwell.

Garfinkel, H. (1967). *Studies in Ethnomethodology*. Englewood Cliffs, NJ, Prentice Hall.

Harper, R., Randall, D. and Rouncefield, M. (2000). *Organisational Change in Retail Banking*. London, Routledge.

Hirschheim, R., Klein, H. K. and Lyytinen, K. (1995). *Information Systems Development and Data Modelling*. Cambridge, Cambridge University Press.

Hughes, J., Martin, D., Rouncefield, M., Procter, R., Slack, R. and Voss, A. (2003). *Dependable Red-hot Action*. Proc. 8th European Conference on CSCW.

Jonas, H. (1984). *The Imperative of Responsibility*. Chicago, University of Chicago Press.

Martin, D. and Rouncefield, M. (2003). "Making the Organisation Come Alive: Talking through and about the technology in remote banking." Human-Computer Interaction *18*(*1 & 2*): 111-148.

Sommerville, I., Bentley, R., Rodden, T., Sawyer, P., Hughes, J., Randell, D. and Shapiro, D. (1992). *Ethnographically-informed System Design for Air Traffic Control*. Proc CSCW92, Toronto, ACM Press.