How Responsibility Modelling Leads to Security Requirements

Ros Strens and John Dobson

Department of Computer Science
University of Newcastle upon Tyne
Newcastle NE1 7RU
United Kingdom

Abstract

When a technical system is placed in a social context organisational requirements arise in addition to the functional requirements on the system. Security is a good example of such an organisational requirement. A means of identifying these organisational requirements is needed and also a way of specifying them that is meaningful both to users and systems designers.

This paper proposes that the concept of responsibility fills both these needs. Responsibilities embody requirements in that the responsibility holder needs to do things, needs to know things and needs to record things for subsequent audit. These needs form the basis of a 'need-to-know' security policy. Furthermore a model of responsibilities describes the context within the organisational structure in which the requirements, including those related to security, arise.

KEYWORDS
Responsibility, obligation, requirements, enterprise modelling, security

1. Introduction

The first requirement that an organisation will have of a technical system is that it has the functionality necessary to serve the organisation's purposes. This defines the functional requirements on the system. Of equal importance is the need for the system to support those functions in a way which matches the structure, objectives and characteristics of the organisation. These requirements that come out of a technical system being placed in a social context are termed organisational requirements. We shall be particularly concerned in this paper with security as an organisational requirement.

The need to capture and deal with organisational requirements in the system design process has long been recognised, and a number of methods are now in existence to support the handling of such issues in IT systems design, but there is very little evidence that they are widely used. The ORDIT project, an Esprit II project investigating information technology and organisational change, has addressed these problems on the basis of socio-technical systems theory, with its premise that the system contains within it two sets of resources: technical and social (human) resources, and that these are so inter-related that any attempt to optimise only one of these sets of resources may adversely affect the other set so that the resultant utilisation is suboptimal. Design methods appropriate for technical systems cannot simply be applied to socio-technical ones, since equal consideration must be given to both human and technical issues if the design is to meet the real requirements of the organisation and be supportive of people in their work roles. Again, security systems will be considered as socio-technical systems instead of just as technical ones, recognising that security policies and models that have been developed in a purely technical context may not be applicable to the wider context we are here considering.

We therefore need a means by which system developers can recognise organisational requirements such as security properties and specify them in such a way that enables them to envisage and propose solutions to meet the achievement requirements. The problem here is twofold. Firstly there is the problem of how to capture the requirements, some of which may be apparent and easily ascertained, others may be more difficult to elicit, if, for example, they are implicit in the working practices, and others may only arise when design solutions are proposed.

The second problem is that the language of systems designers is suited to technical systems whereas the users' language is appropriate to the organisational context. What is needed is some set of boundary objects where these two worlds can meet. We are proposing that the concept of responsibility is one such boundary object, and that responsibilities may be regarded as the key to understanding requirements in implementable terms in that a responsibility has attributes that can be appreciated in both worlds although the language and implications differ.

We also see the concept of responsibility as being a means of solving our first problem, that of identifying requirements in the first place. We propose that an organisation can be viewed as a network of responsibilities that embody aspects of structure as well as function. The users' real requirements are manifest in the responsibilities they hold in that they have a need to know things and a need to do things for the proper fulfilment of their responsibilities, and a need for audit in order to show how they have fulfilled their responsibilities. A responsibility thereby implies requirements for information, requirements for action and requirements for the recording of history, and, by approaching these requirements through the responsibilities held by users by virtue of their work roles within the organisation, we not only capture the requirements but gain an understanding of the organisational context in which they arise. Note that we are assuming primarily a 'need-to-know' basis for security policies, though our ideas can accommodate other alternative bases for a security policy.

On the designer's side of the boundary, it is clear that the requirements for information and action can be translated into the data and functions that the IT system must provide. Thus the concept of responsibility as a boundary object between users and designers should lead to a better understanding by designers of what the technical system should achieve (rather than how it will do it which is purely within the domain of the designer), and its context of use within the socio-technical system.

The concept of responsibility is also a valuable boundary object between different types of model and between reality and models. By looking at all of the responsibilities held within a work role, we can unify different models of the organisation, such as a process-oriented horizontal view and a management or vertical view, into one responsibility based view.

In the next section the rationale underlying our assertion that responsibilities embody requirements and the context in which those requirements arise in terms of organisational structure is presented, and the final section briefly indicates how these ideas have been applied in the real world to produce a specification of user requirements for an integrated clinical workstation.

2. The concept of responsibility

In the paper delivered at the New Security Paradigms workshop last year, we argued for responsibility being a key issue for security. This section elaborates the notion of responsibility; a subsequent section will relate this elaboration to issues of security.

2.1. The responsibility relationship

So far we have spoken of responsibilities held by users as though they are a 'thing' that the user possesses. In fact the holding of a responsibility implies that there is also a giver of that responsibility and therefore the existence of a relationship between the holder and the giver of the responsibility. (From now on we shall refer to the 'people' involved as agents, since an agent can be any size of group from an individual to a department or even a whole organisation.) We therefore define responsibility as a relationship between two agents regarding a specific state of affairs, such that the holder of the responsibility is responsible to the giver of the responsibility, the responsibility principal (Figure 1).

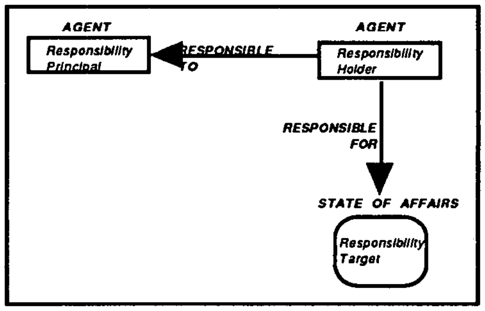

Figure 1. A Responsibility Relationship between Two Agents

The definition of a responsibility consists of:

a) who is responsible to whom;
b) the state of affairs for which the responsibility is held;
c) a list of obligations held by the responsibility holder (how the responsibility can be fulfilled);
d) the type of responsibility (these include accountability, blameworthiness, legal liability).

2.2. The relationship between responsibilities, obligations and activities

This brings us to the distinction between responsibilities, obligations and activities. We use these concepts in the sense that agents execute activities in order to discharge obligations imposed on them by virtue of the responsibilities they hold. These obligations are what the agents have to do and effectively describe their 'jobs' or roles. They are the link between their responsibilities and the activities they execute. Another way of describing this relationship is to say that responsibilities tell us why agents do something, obligations tell us what they do and activities are how they do it.

The distinction between responsibilities and obligations is apparent from the words we use: a responsibility is for a state of affairs, whereas an obligation is to do something that will change or maintain that state of affairs. Thus a set of obligations must be discharged in order to fulfil a responsibility. As such, obligations define how that particular responsibility can be fulfilled. For example a hospital doctor may have responsibility for the medical condition of certain patients. To fulfil this responsibility the doctor must discharge certain obligations such as to diagnose, treat, monitor and/or prescribe.

The distinction between obligations and activities is that obligations define what has to be done rather than how it is done. Activities are defined as operations that change or maintain the state of the system or affect the outside world. Role holders may (or may not) have a wide choice of activities that discharge the obligations they hold. Consider again the hospital doctor who has an obligation to make a diagnosis. According to circumstances he may choose one or more of several activities such as to examine the patient, order x-rays or do tests.

2.3. Creation of responsibility relationships: the delegation process

The responsibility relationship implies a structure as, for example, whether a particular responsibility held by a doctor is to the patients, to the employer or to other staff. These responsibility relationships are created when delegation takes place and obligations are transferred from one agent to another. This delegation process will frequently be implicit rather than explicit, and may be used to explain how the hierarchical organisational structure and distribution of responsibilities has come about over time. Our account of the delegation process is based on the view that, because a responsibility is a relationship between two agents, *responsibility holders* cannot independently transfer their responsibilities to other agents, but they can transfer their obligations. The result of this process is the establishment of new *responsibility* relationships between the pairs of agents involved. The original holder becomes the principal of the new responsibility relationship and the receiver of the obligation is the new responsibility holder. We will now examine this process in a little more detail, since a security policy must have the concepts to permit statements about what happens to capabilities for access to resources associated with obligations in the presence of delegation. It is possible, for example, that as a result of delegation of obligations, an undesirable set of capabilities ends up in the hands of the same roleholder; this would force the re-examination of the desirability of the delegation, and perhaps of the original division of responsibilities (which would have to be solved in the social system, of course, not the technical one).

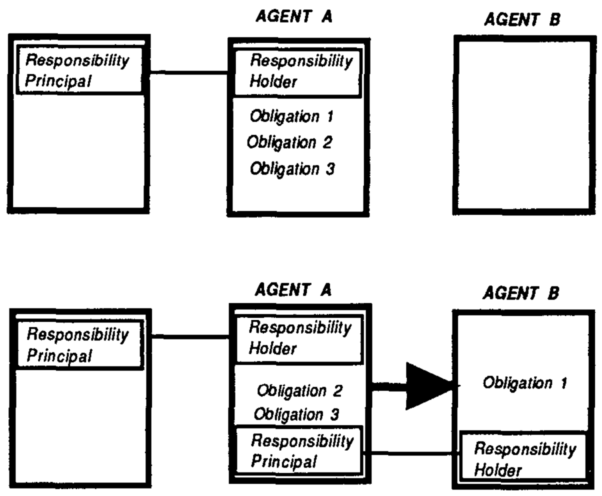

Figure 2. A responsibility relationship created by the transfer of an obligation

The top diagram in Figure 2 shows a situation where agent A is the holder of several obligations associated with a responsibility. If an obligation to do something is passed to agent B (lower diagram), agent A still retains the original responsibility *since this is not transferable*, and we will see in the next section how that responsibility can be fulfilled. Meanwhile agent B has acquired an obligation relating to the state of affairs for which agent A holds responsibility. Agent B also holds responsibility now for that same state of affairs, as well as agent A, because it will be affected when the obligation is discharged. However agent B's responsibility is to agent A who delegated the obligation; in other words a new responsibility relationship has been created between them.

An example of this process is where the first author of a book is responsible to a publisher for the production of a text. The first author retains this responsibility to the publisher even if the obligations to write individual chapters are transferred to other authors. The other authors then acquire responsibility for the writing of their respective chapters, but their responsibility is to the first author and not directly to the publisher.

A chain of responsibility relationships can thus be created as obligations are passed from one agent to another. Within each individual responsibility relationship both agents have a responsibility for the same state of affairs, although their obligations differ.

2.4. Functional and structural obligations

The obligations referred to above are functional in nature. They are what agents must do with respect to a state of affairs (e.g. execute activity), in order to fulfil their responsibilities regarding that state of affairs. These we term functional obligations.

We have seen however that when an agent delegates an obligation to another agent, the first agent still retains responsibility for the resulting state of affairs. To fulfil this responsibility the first agent must ensure that the transferred obligation is discharged satisfactorily by the other agent. The first agent thus acquires a new obligation to do whatever is appropriate with respect to the other agent in order to fulfil his responsibility, such as directing, supervising, monitoring and suchlike of the other agent. This other agent also acquires an obligation of a complementary nature to be directed, to be supervised or whatever. These we term structural obligations (Figure 3). For example if a director passes an obligation to a manager, the director acquires a structural obligation to direct the manager in the discharging of the transferred obligation, and the manager acquires an obligation to accept direction. Other examples of these structural obligations (e.g. to verify) occur in the context of auditability obligations. Again the structural obligations may be implicit in the hierarchical structure of the organisation rather than a result of explicit delegation.

To summarise, we have shown that everything that a responsibility holder must do, whether with respect to a state of affairs or to another agent, is represented by the functional and structural obligations held.

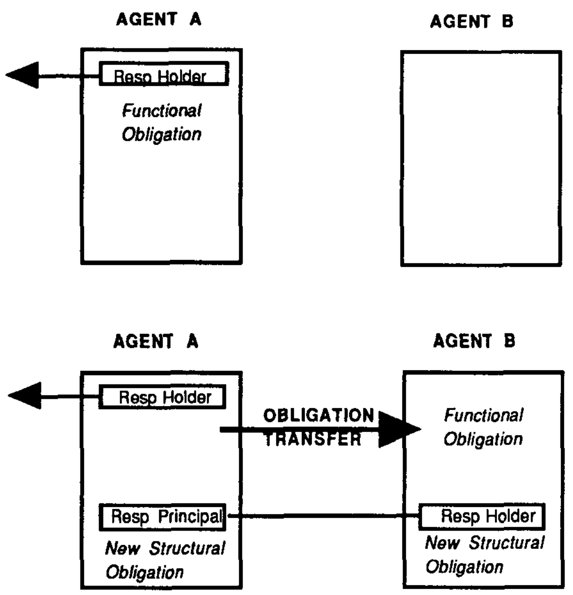

Figure 3. New structural obligations created by the transfer of an obligation

2.5. Responsibilities embody requirements

We have distinguished between functional (process) and structural (organisational) obligations solely to show how the distribution of function and the organisational structure are embodied in the network of responsibility relationships. From the point of view of defining requirements we only need to know what the agents need to do and the distinction between functional and structural obligations is unimportant.

Thus the obligations that a responsibility holder must discharge tell us what the responsibility holder needs to do, and this leads us directly to requirements on the IT system. These fall into two categories. Firstly some of the actual obligations (what the agent needs to do) can be transferred to the IT system and realised as functions on the system. These are therefore functional requirements on the IT system. Secondly the IT system may be used to support agents in discharging their obligations. One form of this support is meeting their information requirements, i.e. what the agents need to know. Another form of support is keeping a record of what has been done (the need for audit). In practice the 'need to do', 'need to know' and 'need for audit' lists are generated for each responsibility and are interpreted as functional and information requirements on the IT system.

2.6. Responsibility modelling within the ORDIT modelling framework

The concepts presented above form part of a modelling framework developed by the ORDIT project. This framework will now be described briefly to show how responsibility modelling fits into the broader field of enterprise modelling.

The Generic ORDIT Model: The core concept in the ORDIT way of looking at organisations is the agent entity. These are the primary manipulators of the state or structure of the system, but essentially they are the people in the socio-technical system, although it is possible for a machine to behave as an agent entity. An agent entity is not just a person but any size of group from an individual to a whole organisation. Other essential elements in modelling an organisation are actions and resources, where an action entity is an operation that changes or maintains the state of the system, and a resource entity is what enables the agent to do the action. An icon showing these entities and the relationships between them is shown in Figure 4.

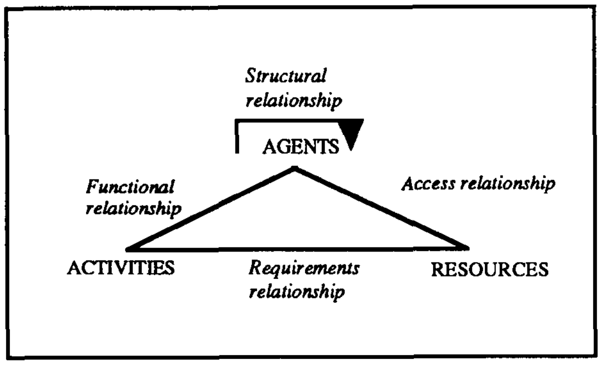

Figure 4. Icon depicting the Generic Concepts in ORDIT Models

We say that agent entities have functional relationships to action entities, since agents do the actions, and that they have access relationships to resource entities, while the action entities have requirements relationships to resource entities; i.e. agents must access resources that are used by actions performed by agents. We are particularly interested in organisational structure so structural relationships between agents are shown. These are basically responsibility relationships. Relationships between resources (resource schema) and between actions (interactions) are of less interest as these can be represented by data and process models respectively.

The ORDIT Modelling Framework: We have taken this generic model of an enterprise and from it developed three separate but inter-related models of the organisation. These models are of different aspects of the organisation based either on responsibilities, on obligations, or on activities. Each model includes the same three basic types of entity: agents, activities and resources, and also the relationships between them.

Figure 5 shows how the three models are related in that the vertical links join concepts of the same type. This scheme is based on the recognition of obligations, as we define them, as the link between responsibilities and actual activities. The model based on obligations is called an obligation model because the obligations held describe the holder's job or role. Similarly for the resource entity, capability tokens signifying capability to access resources are seen to be the link between rights or authorisations to access and the actual accessing of resources. For example a doctor must first be authorised to access the necessary parts of the IT system by an authorising agent before being able to obtain tokens such as an identifier and password that provide the capability to access. This allows access to the information resource.

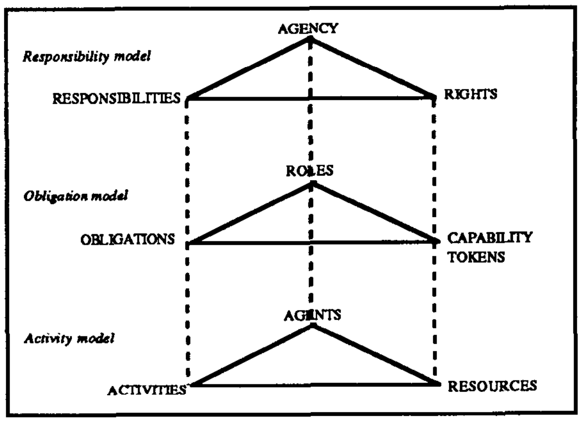

Figure 5. The Three ORDIT Models and how they are related.

3. Security requirements

Figure 5 can be considered as a framework for positioning security requirements. Security is often thought of as a way of binding together a particular set of capability tokens and resources according to the dictates of some security policy. What is often not stated, however, is how the security policy is derived, or any justification that a particular set of bindings compose together to achieve a particular security objective. The richer set of concepts, and the relations between them, exhibited in Figure 5 go some way to providing a language in which these arguments can be made.

We have used the approach outlined in this paper to develop a set of requirements, including security requirements, for an integrated clinical workstation for use in acute hospitals. These are open windows into extensive computer and communication services that provide a broad range of support to clinical staff in meeting their responsibilities at the point of care. The immediate objective of the part of the project cited here has been to capture the nature of the requirements of medical doctors, and ultimately of nursing and other staff who provide direct clinical care. The scope covers the problems of organising the process of medical care, of supporting the medical records, and of implementing computer support through carefully tailored user interfaces. Issues of security of access and confidentiality of information have throughout been of paramount concern.

The methodology used accepts that the fundamental requirement is for users to have a solution to their problems. In other words requirements are the obverse of problems. The process therefore starts by making lists of problems and frustrations with current procedures and records, based on statements from potential medical users. These are couched in the language and concepts of the users.

Structuring the problems and transforming them into user requirements has been done by applying the concept of responsibility and related concepts shown in Figure 5, so that the user requirements can be expressed in terms familiar to the users while at the same time the expression reaches the edge of the kind of language used by systems developers.

The first step was to generate a list of eleven key responsibilities held by medical doctors; a need to know, a need to do and a need for audit list was then generated for each responsibility. Each item in the need to do list was then divided into two components: functions that could be transferred to the system and tasks for the management of the organisation.

Without going into details (which are in any case sensitive), one example of the process should indicate how the concepts were used. Consider the design of the security privileges to be afforded a consultant who has two separate sets of responsibilities: a National Health Service consultant and a private practice consultant. Whereas most of the time these two separate roles do not conflict, it is obviously desirable[1] to enforce separation of duties at the level of the computing system so that the private practice consultant cannot access NHS information, and vice versa. The approach we take to defining the boundaries of this separation is first to establish the separate responsibilities and associated rights (need to know, need to do, need for audit) for each agency. These responsibilities will be different since the two responsibility principals are different (NHS, the patient in person). The obligations associated with each responsibility are then examined and the need for capabilities assessed, bearing in mind that deriving capabilities from rights is part and parcel of, and structurally isomorphic to, the process of deriving obligations from responsibilities. Part of this process involves examining delegation in the way we have suggested (responsibilities cannot be delegated, but derived obligations can) and examining the consequences of the corresponding delegation of capabilities. It turns out, for example, that the delegation rules in NHS and private practice are different.

The mapping from obligations to activities is not always straightforward if the possibility and consequences of failure of activities is to be considered. In terms of the classical approach to fault tolerance, the obligation can be considered as the "acceptance test" and a variety of alternative activities might have to be considered. Each alternate will of course require its own set of resources and access modes and the way in which these derive from generalised capabilities will have to be considered.

4. Conclusions

A security policy must be capable of showing where security responsibilities lie — for example, who can authorise access to resources, who can validate claims of 'need to know', who can specify and operate prevention mechanisms. The concepts we have designed and shown in Figure 5 are an attempt to provide a modelling language for these kinds of policy concerns so that the issues of drawing security system boundaries can be discussed.

This issue of drawing system boundaries is not trivial. In a previous study we conducted for a patient records system for a small ward in a small hospital, one half of the time, and one third of the effort, spent in the requirements phase was simply finding out where the system boundaries lay. These proportions are not untypical. It might be difficult to prove, but our suspicion is that when a system seriously fails to fulfil its security objectives, the failure can more likely be traced to inappropriate or faulty boundary drawing than inappropriate or faulty (use of) security mechanisms. (Back door entries for system programmers are a good example of this.) In our experience, the main problem in system boundary drawing is a clear delineation or model of the space in which the boundaries are to be drawn. At the level of security mechanisms, this space is one containing activities, resources and agents (our activity model), but there are often real difficulties in relating this space to the space of an organisational security policy if the latter space is not well defined.

The aim of this paper has been to show how responsibility modelling can be used as a means of specifying security policy requirements on an IT system that is meaningful to both users and systems designers and in a way that solves these difficulties. We hope to have shown how obligations define what a responsibility holder must do, and how these can be divided into those that must be done by people and those that are transferable on to the IT system, thus creating functional requirements on that system. By listing what a responsibility holder needs to know and needs to record we can create lists of information requirements.

We also hope to have shown how organisational structure may be interpreted in terms of responsibility relationships, and therefore how a model of responsibilities and their associated obligations not only represents function but also the context within the organisational structure in which the responsibility is held. This has direct bearing on whether the responsibility holder holds the necessary authorisation and capability tokens to access the information resources required for performance in a role while respecting the constraints derived from a need-to-know security policy.

[1] at least in the case of elective treatment; the emergency situation is very different and it might not be at all desirable.

Acknowledgements

We gratefully acknowledge the contributions made by our colleagues in the ORDIT project and also to Stewart Orr, Project Manager of the Integrated Clinical Workstation Project to the ideas presented here. We also gratefully acknowledge the support of the University of Newcastle upon Tyne and the Esprit programme (Project 2301).