Computing Laboratory

A Methodology for Analysing Human and Computer-related Issues in Secure Systems

J.E. Dobson

Technical Report Series
No 307 March, 1990
A Methodology for Analysing Human and Computer-related Issues in Secure Systems

J.E. Dobson

Abstract

The paper describes work being carried out at the Universities of Newcastle and York which is developing conceptual and logical models of non-functional requirements such as 'security', whose meaning derives from the organisational environment of a computer system. The work brings together two novel approaches to the problem of deciding whether or not a system is 'secure':

(i) a recently-developed framework based on enterprise modelling principles for evaluating whether a system is fit for its purpose in some given organisational context; and

(ii) a logic based on speech act theory for evaluating the conversation structures in which the term 'secure' is defined and ascribed as a property of a computing system.

This paper is to be presented at IFIP Sixth International Conference on Informational Security, Helsinki, May 1990.

Series Editor: M.J. Elphick

© 1990 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne, Computing Laboratory, Claremont Tower, Claremont Road, Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

DOBSON, John Edward

A methodology for analysing human and computer-related issues in secure systems. [By] J. Dobson

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Laboratory, 1990.

(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 307)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.
Computing Laboratory. Technical Report Series. 307

Abstract

The paper describes work being carried out at the Universities of Newcastle and York which is developing conceptual and logical models of non-functional requirements such as 'security', whose meaning derives from the organisational environment of a computer system. The work brings together two novel approaches to the problem of deciding whether or not a system is 'secure':

(i) a recently-developed framework based on enterprise modelling principles for evaluating whether a system is fit for its purpose in some given organisational context; and

(ii) a logic based on speech act theory for evaluating the conversation structures in which the term 'secure' is defined and ascribed as a property of a computing system.

About the author

Mr. Dobson is a Principal Research Associate at the University of Newcastle upon Tyne, with particular interest in a system approach to security. He has been active in the field since 1982 and is the author of more than a dozen papers.

Suggested keywords

ENTERPRISE MODELLING
FAULT-TOLERANCE
SECURITY
SPEECH ACT THEORY
SYSTEM MODELLING

Suggested classmarks (primary classmark underlined)

Dewey (18th): 658.47 001.64404
U.D.C. 519.718 519.687
A METHODOLOGY FOR ANALYSING
HUMAN AND COMPUTER-RELATED ISSUES
IN SECURE SYSTEMS.

JOHN DOBSON

Computing Laboratory
University of Newcastle upon Tyne
NEWCASTLE NE1 7RU
United Kingdom

ABSTRACT

The paper describes work being carried out at the Universities of Newcastle and York which is developing conceptual and logical models of non-functional requirements such as 'security', whose meaning derives from the organisational environment of a computer system. The work brings together two novel approaches to the problem of deciding whether or not a system is 'secure':

(i) a recently-developed framework based on enterprise modelling principles for evaluating whether a system is fit for its purpose in some given organisational context; and

(ii) a logic based on speech act theory for evaluating the conversation structures in which the term 'secure' is defined and ascribed as a property of a computing system.

1. INTRODUCTION

Over the years, researchers and practitioners alike have been grappling with a particularly intractable information systems problem: how to elicit, specify and structure user requirements, particularly those concerned with the so-called non-functional aspects of the system. Examples of such non-functional requirements are those concerned with the safety, security, reliability, availability, maintainability... of the system. Ultimately, of course, these non-functional requirements will be reflected in a specification either of some explicit functionality, or else lack of some explicit functionality, of the system; up to now, however, very little has been understood of the process of translating non-functional requirements into aspects of a (functional) specification.
Our claim is that this can only be done in the context of a sound and principled approach to capturing and analysing requirements, and in particular so-called "non-functional" or dependability requirements, so as to provide a high degree of assurance that the design of a computer system does indeed reflect the needs, wants and policies of the owners and users of the system. We have thus been approaching the problem of system specification not from below (i.e. by asking what is required of a specification language that an implementation can be shown to be in conformance with a formal specification) but from above, by asking what is required of a specification methodology that a specification in some formal language can be shown to be in conformance with the informally stated requirements.

The key to the answer is to take a view of the system as seen from the enterprise. That is, the concerns of security must be seen not simply as the correct functioning of a piece of equipment with respect to a specification, but as the definition and execution of a set of organisational procedures to ensure that the system not only behaves correctly, but is specified, implemented, deployed, and perhaps ultimately decommissioned in accordance with approved security procedures. The procedures which surround the development and use of a security-critical system are as important a topic for detailed investigation as is the system per se. Equally important are the relevant obligations, roles and responsibilities defined in the organisation. The analysis of the roles, responsibilities and procedures lies within the province of the emerging discipline of enterprise modelling, and we are using enterprise modelling techniques as the basis of a security analysis of the system in its organisational environment.

2. PROJECTIONS OF A COMPUTER SYSTEM

Recent studies [1] have revealed that different information technology experts have different viewpoints about what are the crucial concerns that make up information processing. Further examination has revealed that five viewpoints are dominant and that although each viewpoint can be separately characterised, each viewpoint in some way or other acknowledges the concerns addressed in other viewpoints but with a lesser priority. These viewpoints, which we call projections, differ not in how much of the system they describe, but in what aspects of the system they emphasise. The five projections are termed the enterprise projection, the information projection, the computational projection, the engineering projection, and the technology projection. An information technology system can be described in any one of these projections, and the resulting descriptions, or models, reveal different facets of the system.

Designers used to working with one projection often have difficulty assessing the relevance of the concepts used in any of the other projections. It is important to realise, however, that a system has ultimately to be described from each viewpoint; all of the viewpoints are equally valid and it is a mistake to argue which is the most fundamental.
2.1. Enterprise Projection

The purpose of the enterprise projection is to explain and justify the role of a computer system within an organisation. An enterprise model describes the overall objectives of a system in terms of roles (for people), actions, goals and policies. It specifies the activities that take place within the organisation using the system, the roles that people play in the organisation, and the interactions between the organisation, the system, and the environment in which system and organisation are placed.

Enterprise models provide system designers with a description showing how and where the system is placed within an organisation. The interactions between the organisation and the system designers can be included in this projection so that the process of procurement, installation, maintenance and evolution is included as part of the system design.

Design decisions made using the enterprise projection concern what a system is to do and who it is doing it for, rather than how it does it. The design concepts defined in the enterprise projection allow the designer to develop a closed model which represents all the real world requirements, and in particular the so-called non-functional requirements such as security, which the designer is prepared to incorporate into the structure of the system.

2.2. Information Projection

The purpose of the information projection is the identification and location of information, and the description of the information processing activities. An information model describes the structure, flow, interpretation, value, timeliness and consistency of information held within a system. An important feature of systems that are distributed is that information can be collected, processed and presented in different places and at different times. Consequently, information models consider the system as a collection of information resources and information processes.

Designers use information models to explore the nature and role of information in the system. They take design decisions that are epistemological in nature: "Who knows what?" and "Where can information flow?". The people in the organisation may also figure in the information model so that the impact of a system on the organisation that is to use it can be properly analysed.

2.3. Computational Projection

The purpose of the computational projection is to show the organisation of an information technology system as a set of linked application programs. A computational model provides programmers with a description of a system that explains how the various application programs behave. This description is in terms of data representations, programming languages, system services and
program specifications. Computational models show how the system is structured for modularity and parallelism, and how components can fail independently and can be configured dynamically. Design decisions made in the computational projection concern the mapping of functional components described in behavioural terms onto application program, operating system and system infrastructure components.

2.4. Engineering Projection

The purpose of the engineering projection is to describe information technology systems in such a way that designers can reason about the performance of systems built to their designs. The engineering projection describes processing, memory and communications functions that can be used to support the requirements of programs structured according to the computational projection.

A designer uses an engineering model to make decisions that concern trade-offs between quality attributes such as performance, dependability and scaling. There is no discretion in the engineering projection over the sorts of information processing components that the system is to contain; design choices are restricted to the sorts of coupling between components and the management of computer and network resources to achieve desired quality attributes.

2.5. Technology Projection

The purpose of the technology projection is to describe the physical components, in terms of hardware and software, that make up the information technology system so that they may be related to the architectural concepts defined in the other projections. Technology models act as blueprints of systems during their construction and maintenance, showing how the architectural concepts have been implemented in practice. Design decisions made in the technology projection concern the choice of material used for building according to a design specification.

3. ENTERPRISE MODELLING

As we indicated earlier, we regard the enterprise projection as key for the definition of security and other non-functional requirements on a system, and in particular for the understanding of how those requirements derive from the roles and responsibilities of the requirements owners. We shall in this section introduce a language for modelling the way those roles and responsibilities are related to the enterprise as a whole; Section 4 will provide an example of the use of the modelling language.

The basic component of our modelling kit takes the form of an entity-relation schema defining three sorts of entities: activities, agents, and resources. It defines a set of relations between these entity types. The enterprise modelling process is expressed as a set of steps starting with the identification of the activities, then proceeds to identify all the associated resources and the nature of
the relations between the activities and the resources; it then identifies the agents who are identified in the activities and the relations between the agents and the resources.

The reason for starting with the activities, and in general seeing the whole thing from the viewpoint of the activities, is that we are primarily interested in ultimately producing a behavioural description of the system. We could have started by taking an agent-oriented view; this would be appropriate if, for example, we wished to explore the interactions between the system and the responsibilities of the people who would come into contact with it. Or again, we could have started with a resource-oriented view of the system; this would have resulted in our producing conceptual schemata for the information structures and data models of the system.

The result of this phase of our modelling method is a diagram and a natural language description of the entities and relations which explain what each activity in the system involves, what it does, and who has an interest in it. There are three main advantages that we claim for our modelling technique:

i) It forces the designer to consider agency: Who is responsible for what? What are the failure modes and who will handle them? How does the activity relate to the organisation?

ii) It provides a common format and set of concepts which will allow different views of organisations produced in different contexts to be represented, explored, and compared.

iii) It has the potential to support formal reasoning.

3.1. An Entity-Relation Model

We shall now offer an informal graphical notation for organising these models of an enterprise. The diagrams are not intended to stand alone, but provide a framework for description. These descriptions may be in natural language, or they may be further refined into some formal specification language, depending on the purpose for which they are to be used.

The basic entities are of three types:

**Agent**  This is a name attached to a set of responsibilities, e.g. customer fault manager. An agent must always be an appropriate answer to a Who? question. The symbol for an agent is a (rectangular) box.

**Activity**  An activity is to be distinguished from the doer of the activity. Thus an activity is an answer to a What? question, and takes the form of a verb, e.g. CustomerComplaintsHandling. The symbol for an activity is a circle.

**Resource**  Resources are answers to With? or ByMeansOf What? questions. The symbol for a resource is a parallelogram.

Between these three types of basic entity there are six kinds of relation:
Action-Action

Actions interact with each other. Such interactions are usually mediated by the exchange of resources, though direct interactions, such as interrupts, can also occur.

Action-Resource

The relation between an action and a resource is an access mode, such as read or write (for information resources) or provide or consume (for commodity resources).

Resource-Resource

The relation between resources is what in information technology terms is called the conceptual schema.

Agent-Resource

The relation between an agent and a resource is an access right, such as the right to create, to destroy, to allocate.

Agent-Action

The set of actions with which an agent has some relation constitute the functional role of that agent — see below.

Agent-Agent

The set of agents with which an agent has some relation constitute the structural role of that agent — see below.

Figure 1 shows the relations between the basic entities:
3.2 Role Relations between Agents

The basic principle of our representation is the aim to show relations between what we term structural roles and functional roles. A structural role defines the responsibilities laid upon the role-holder, the relationships of the role-holder with related roles (the 'role set') and the expectations placed upon the role by the role set. The structural role therefore defines the task responsibilities of the role-holder and the associated rights and obligations. This is different from the set of actions ('job description') that the role-holder is expected to perform. We shall term the latter the functional role. Thus by functional role we mean one which can be characterised by extensional properties such as access rights. In contrast structural roles are determined by the intensional concepts of responsibility and authority, typically deriving from a position in some organisation. Dependability ascriptions — e.g. the system conforms to government policy, the system can be safely deployed — are typically associated with (are the responsibility of) structural roles.

In Figure 2 we introduce the idea of enterprise object encapsulation. This is based around the agent-activity relationship. We bind together the set of structural roles of some particular agent and the set of activities (i.e. the functional roles) that are related to those structural roles. Resources are not bound in to these enterprise objects; rather, resources mediate the instrumental (i.e. observable, behavioural) interfaces between the enterprise objects. The structural roles that link the enterprise objects represent the obligations between them. The instrumental interfaces that link the enterprise objects represent the behaviours and messages that pass between them and that are mediated by the resources. These behaviours and messages represent the creation and discharge of the obligations that exist between the enterprise objects.
The interpretation of Figure 2 is as follows. There are two aspects to the client-server relationship, considered as a relationship between two agents.

The first aspect is the ServiceProvision/ServiceInvocation relationship. This we term the 'instrumental interface' since it is the interface between the agents across which things happen (i.e. the instrument). However, this instrument exists only by virtue of the second relationship between the agents, which is the Obligation relationship, i.e. a contract between a service provider and a service consumer.

In practice there will be more than one instrumental interface between the client and the server. For example, the provider/consumer relationship may well imply an obligation to invoice and pay bills for services rendered, a complaints procedure and so on. The fact that a service obligation may be associated with more than one instrumental interface has implications for the automation of some function. For example, if service provision by a human agent which is paid for at the point of service is automated, what happens to the charging mechanism? to the complaints procedure? and so on.

These are questions that have to be answered in terms of the information projection. It is the job of the language associated with this projection to represent the relation between the various instrumental interfaces that exist by virtue of a single obligation relation (e.g. that between provider/consumer).

Figure 3 shows what happens when we abstract away the resources that mediate between agents and simply show the relations that exist between them. We shall often use our diagrams in this form.
4. EXAMPLE: AN AUTHORISATION FUNCTION

This section examines the process of automation of functions within an enterprise. It is the first stage in the process of system design, and produces as output a set of descriptions against which a representation in the computation projection can be validated. Rather than showing in detail what these descriptions actually look like, we shall describe the process of generating them.

The example we shall examine attempts to describe an authorisation function for a secure network service and is designed to assist a discussion concerning what aspects of the authorisation function could be automated and what would be the implications, particularly the security implications, of such an automation process.

4.1. Enterprise Projection

The function that is the subject of this demonstration implements the policies of who is allowed to use what service from what service access point under what circumstances. (However, the fact that a particular service request is authorised does not guarantee that service will be delivered. There may be other functions in the system that may validly block service delivery.)

The definition of the authorisation policy involves a number of policy holders. The statement of the authorisation policy which will be given next implicitly identifies these policy holders. In practice, one first identifies the policy holders and extracts from each their policy statements; for this presentation, however, it has turned out easier to state the various policies and to annotate each policy statement with an attribution to the appropriate policy holder.
Figure 4

Authorisation Policy Objective

To allow the creation of any pattern of authorisations subject to controls and predicates over any combination of the following factors:

1) The claimed identity of the requesting agent (by 'requesting agent' we mean the agent requesting authorisation to use some function, not the agent requesting invocation of that function). This claimed identity may require the corroboration of some pre-arranged token.

The personal identity and the token are both optional and are independent of each other. One person may have many tokens, or tokens may be shared.

The policy identifies a registration agent who receives the request (and token, if present). It may also identify an authenticating agent if the token is to be authenticated; note that in this case what is authenticated is the combination of identity and token. This registration agent clearly needs some interface to the authorisation function — this is shown at A. The interface between the authentication agent and the authorisation function is shown at B.

2) The state of account of the requesting party.
A function might be authorised only if there are no outstanding debts from some previous service use of the requesting party. This implies an interface for the accounts agent, shown at C.

3) The requested service (or quality thereof).

This implies that a particular (quality of) service is potentially available. Such availability is the responsibility of the installing agent (D). A request for authorisation can be refused if the requested service is not available to be installed.

4) There is a set of requests, the authorisation for which will not be automated in principle. Such requests will always be referred to a human operator for a decision. This is called the online authorisation operator (E).

4.2. The Information Projection

This stage identifies the data structures, access modes and access rights, data types and resources that are visible at each interface between the various agents and the authorisation function. We shall concentrate on commenting on those aspects where there are security and integrity issues to be addressed by the system designers.

Requesting Agent

In the system we have described, which is typical of many manual systems, there is deliberately no direct interface between the requesting agent and the authorisation function. Instead, the requesting agent has some structural role relation, defined by the enterprise, with the registration agent.

Registration Agent

The registration agent needs to present to the authorisation function some subset of a claimed identity, a token (either pre-authenticated or with a request for authentication), and a request. The identity and request presented to the authorisation function must be the identity as claimed by the requesting agent and the request as presented. The token may or may not be invalidated to prevent reuse (e.g. it may or may not be returned to the requesting agent following the outcome of the authorisation request).

Authentication Agent

This agent deals solely with the authentication of a claimed identity and accompanying token. The authentication agent is usually trusted not to corrupt the token.
Accounts Agent

This agent deals with a confirmation of an authenticated identity and a request. In addition, this agent must be able to withdraw any existing authorisation on the basis of the state of the account, according to some internal policy. Withdrawn authorisations can be reinstated.

Police Agent

This agent is invoked primarily when the registration agent has reason for suspicion. The police agent must be able to access all the relevant evidence. In addition, this agent may withdraw authorisation for good reason based on internal policy. Withdrawn authorisations may be reinstated.

Online Authorisation Agent

This agent receives a referred authorisation request and makes a decision in accordance with some organisational policy.

Using Agent

The using agent has, by virtue of a successful registration, acquired the right to have an instrumental interface to a service access point, and, as a result of interactions at this interface, further requests for authorisation may be generated and submitted to the authorisation function.

Complaining Agent

The purpose of this relation is to establish liability in the context of the failure of a service request to receive the authorisation expected as of right by the user.

Diagnostic Agent

This agent receives and categorises an incoming complaint. In order to investigate a complaint, the diagnostic agent must be able to make various diagnostic queries on the authorisation function.

4.3. Computational Projection

The authorisation function has two classes of interface which are distinguished by the purpose and context of their use. The first of these includes all the use interfaces, which must be specified (in Z say, though this will not be done here). The interactions on these interfaces describe authorisation as a service component.

The second class of interface describes the control aspects. This provides the means for the operating enterprise to ensure that it meets its policy objectives relative to the authorisation function. Examples of the kind of control interface are monitoring, recording, creating audit trails, and other security-related functions. These controls must be present in order to meet the requirements of
all agents in the enterprise who need to interact with the authorisation function in order to discharge their obligations within the enterprise. Again, these control interfaces could be formally specified (and in fact have been, though details are omitted here).

For the purposes of this paper, the important point is not the details of the specifications, but the fact that the two interfaces have quite different purposes. Hence when a function is (partially) automated, the effect of and on these two types must be separately considered. In general, control interface functions are harder to automate correctly since the human purpose of these interfaces can be explained only by reference to the enterprise as a whole. For this reason, it is important that they all be identified correctly.

4.4. Summary of the Methodological Approach

The purpose of this section has been to argue that security policy issues can be traced back to relations between agents in an enterprise, and it is the purpose of a security policy to describe how the obligations associated with roles within an enterprise give rise to constraints on and control interfaces to some function within the organisation. Such a security policy, when formalised, will have to recognise that the subtleties of roles cannot be captured simply in terms of subject and objects and authorised accesses, but must allow for the expression of use and control interfaces in a role-dependent way. In particular, when a function is automated, the security policy must be able to describe what happens to the various control interfaces that are necessarily part of the automated function.

In general, control interfaces cannot be automated with the same ease as use interfaces, and the assumption must be that following automation, a new set of controls must be introduced. It is for this reason that the subject/object dichotomy is insufficiently expressive: that whether an object is a system object or an enterprise object both determines and is determined by the security policy by virtue of the controls that have to be enforced by that policy.

5. SPEECH ACT THEORY

'Policy' is a particularly difficult word to give a dictionary definition for. It combines notions of action, desirable future states of affairs, constraints, and pious hopes. Similarly it is so widely used and in so many different contexts that a definition on the basis of use is likely to be equally difficult. There is, however, one specific sort of use of the word that is of importance to security concerns and which our methodology must be able to express. The example we have in mind is its use in a phrase such as 'Security Policy', where the intention is to be able to state what sorts of patterns of accesses to information are to be regarded as valid or legal, and what sorts of accesses are to be regarded as a violation of the security policy.

The view of security here being espoused is expressed by "Provided these procedures have been invoked and correctly executed, it is legal for this agent to
do this action”. Thus in order for a trusted subject to read a classified object, the subject must have been cleared through the appropriate social procedures, and have been given a token of clearance to present to the authorised guardian of the classified object. Similarly the guardian must be able to prove that it was in fact duly authorised, the object must have been classified by an agent authorised to classify, and so on. All of these matters are issues of security policy (as we are using the term). It is a major defect of most current models of security, such as that of Bell and LaPadula [2], that while they well express (one view of) security as a matter of who can do what to whom, they are powerless to express the social procedures by which the access rules come into being and which ascribe authority and security attributes to the subjects and objects of which they treat. In our terms, such models attempt to model security purely in terms of instrumental acts in an activity, since they do not have even the concept of the structural role relations between agents of which those acts are the instrument. But both are needed, and it is the relation between them that provides a more coherent and complete answer to the question What is a security policy?

The approach to examining the process of developing secure systems that we are adopting is to examine the structure of the exchanges or conversations in which the system specifications come into being, the non-functional requirements are captured, and the very term "secure" is defined and acquires its meaning. The reason for this approach is that it brings into the focus of a single framework a number of issues concerned with system dependability:

- protocol structures for ascribing meaning
- protocol structures for identification and authentication of communicating entities
- policies (e.g. security policies) over information storage, access and manipulation
- transaction and commitment protocols
- recovery and rollback protocols
- the relation between program and data structures on the one hand and user interface structures on the other.

The starting point which we have adopted is the work that has been done on using speech act theory [3][4] as a basis for system description and design. This approach shifts the focus of attention away from formal reasoning as the basic cognitive activities undertaken by the human user of the information system, to the social activity by which we generate the space of co-operative actions that the enterprise as a whole undertakes. In taking this perspective, one is not denying the obvious fact that people in their jobs do in fact collect information and reason about that information in the context of making decisions. Rather, one is taking a complementary view that shifts the emphasis to the actions by which groups originate and co-ordinate a network of interconnected activities and obligations, and considering the commitments that individuals impose on each other and on the system.

The following features of a speech act formalism are important:
1. The description of the purpose of the conversation.

Experience of the use of speech acts indicates that in order to determine whether a particular conversation is well-formed (with respect to its social function), it is necessary to be able to make certain distinctions, e.g. between requesting and responding to a request, between agreeing to an offer and making a counter-offer, between committing a transaction and aborting a transaction. All of these distinctions are distinctions of purpose. Purpose is described in terms of the so-called "illocutionary points" of the individual speech acts. It is an application-dependent decision as to what different sorts of illocutionary point there are.

2. The description of the conditions for successful conversation.

Any conversation can only be held on the basis of certain assumptions, which are described in the so-called "preparatory conditions" for a speech act. Of course the complete set of assumptions can never be stated in its entirety, since it is probably not finite, or only just so. However, the more important assumptions, which in general will be those about the assumed impossibility of certain failure modes, should be made clear.

3. The emphasis on describing the understandability, coherence and completeness of the conversation.

Understandability, completeness and coherence of a conversation are expressed in terms of the structuring of a conversation into speech act patterns, sequencing, reference to context, the expression of alternatives, control over dialogue structure, and conditions and constraints. The collection of the individual speech acts together with the description of the superimposed structure is called the conversation program. Representations of the individual speech acts in the program can be installed as data items in some database. The structure can then be represented as a set of relations over those data items, and can therefore (as is desirable) be kept separate from them.

4. Simultaneous analysis of communication and organisational tasks.

Speech act theory makes the distinction between the instrumental act of asking a database for a person's salary in order to write out a monthly pay cheque and the speech act of asking a database whether a given password is valid in order to establish a person's identity. The difference is that the latter act, unlike the former, is concerned with the creation of a commitment that binds the future actions of the requesting program; it is about the establishment of trust rather than the mere transmission of information. The reason why this distinction is important is that it may well influence design decisions concerning the privacy, security, integrity and reliability constraints on the means of handling and transmitting the message; for example, different channels with differing dependability characteristics may well have to be used for instrumental acts and some speech acts.

5. The ability to take a more complete view of a policy statement.
The reason for examining speech acts at all is that they, and they alone, establish the basis on which obligations (accountability, responsibility) are established. It is these obligations that are fundamental to a statement of policy.

One of the more interesting uses of the kind of conceptual model of conversations we have introduced is that it makes it possible to perform an analysis of the various failure modes of a security policy. A systematic approach to the analysis of conversation structures which, it is claimed, is capable of determining and classifying all the possible failure modes of this kind of structure is outlined in previous work [5].

6. TECHNICAL AIMS OF THE BAINBRIDGE PROJECT

This section describes the Bainbridge research project jointly being carried out by the Universities of Newcastle and York into the issues that we have been discussing. As stressed in the introduction, particular attention is being paid to problems that arise in secure systems. The kinds of problems that are being addressed are not those of functional correctness with respect to a specification, since this is a very active research area in which a lot of progress has already been made. Rather, the project is addressing the problem of establishing a specification of non-functional requirements arising from the fact that the system designer has a set of obligations which arise from the nature of security and from the consequences of failure in security. These obligations have to be discharged before the system can benefit from any correctness claims and so be deemed to be secure. It is therefore just as important to investigate how these obligations come into existence and what counts as a discharge as it is to perform whatever is necessary in order to actually discharge them.

Examples of such obligations are

- **Composition.** Assuming that the security of an individual component can be predicated, what — if anything — can be said about the security of two components of known characteristics joined together? What sorts of composition rule are plausible and on what basis can composition rules be agreed?

- **Validation.** What assurance is there that any guarantees that can be given by the system designer actually correlate with what the problem owner perceives to be security?

- **Demonstration.** The system designer has got to be able to show that the best engineering techniques were deployed, that all reasonable (what counts as reasonable and how is this established?) fault hypotheses were examined and the system shown to be secure in their presence, that an appropriate set of vulnerability, treat, countermeasure, risk and sensitivity analyses were performed, and so on.

- **Certification.** A certification process may well require a proof of design correctness using some formal verification technology. What set of social
negotiations establish the basis for agreeing what constitutes a proof? By what token do we trust the verification tools?

These are difficult problems. One of the main reasons for the difficulty is that the philosophical logic of the problems is not well understood. It is not clear, but worthy of investigation, to what extent these problems can be cast into a first order logic and can therefore be made amenable to computational support in the design process. This understanding is important since it is needed in order to create a development environment which is suited for the creation of secure software, and also in order to identify loci for tools to populate such an environment. The development of an environment is not, of course, one of our objectives — it is much too ambitious for our project. It is, however, very much an area in which we hope to make substantial progress towards producing a map of the problem space. Some preliminary investigations in this area have been reported in our previous work [6] [7].

We can summarise our starting position by claiming that "secure" is ascribed as a bivalent property of a system as a result of two related processes:

(i) an investigation into the structure and behaviour of the system according to the framework we have briefly presented;

(ii) an investigation into the conversations about the system through which its desired features are specified and the term "secure" is defined in terms of those specified features.

It is important to realise that the framework described in Section 2 is more than just a series of decreasingly abstract levels of description of a mechanical system or component; it can equally well be used as a framework for describing the procedures and roles contained in an enterprise. In particular, we are using it to create a way of modelling enterprises that require, specify, develop, certify, deploy and decommission security-critical systems. The motivation for this modelling is that the ascription of 'secure' as a property of a system can be justified only if the relevant roles and procedures have been examined at least as closely as the system itself. We are thus using speech act theory to analyse the organisational procedures and produce a descriptive logic of such conversations which can be used to justify the belief that the system is indeed as dependable as it is claimed to be, not in a narrowly technological sense, but in terms of the roles and responsibilities in the enterprise both of those who created it and those that it has to support.

6.1. Conversation Structures

The speech-act-based theory of conversation structures that we have been developing specifies the following:

- a dependability definition system: a formal language for describing states of the world, a language for describing operators in the world, a set of component construction inferences, a specification of dependability-relevant structures. Semantics for the formal languages are under construction, following the lines suggested by Barth and Krabbe [8].
a conversation structuring language: definition of speech act operators (e.g. define, commit, abort). What are their effects? When are they applicable? How can they be realised?

a justified belief system: what sorts of conversations justify what sorts of beliefs? How can we be sure that two parties to a conversation leave it with the same understanding of what the conversation has achieved? What counts as counter-evidence for a belief and how can we be sure such counter-evidence is neither repudiated nor fabricated?

6.2. Fault-Tolerant Conversation Structures

One of the major technical features of our project is its concentration on the application and use of fault-tolerance. We need not rehearse here the importance of fault-tolerance in the design of a secure system; what we wish to stress is the importance of fault-tolerance in the procedures by which a system is specified, developed and certified, and within which a system is used. Issues of design diversity, comparison and adjudication are equally important here also. For example, if the security-critical features of a system may be manually overridden (e.g. during testing), what steps must be taken to ensure that such overriding is properly authorised and monitored, and what procedures should be adopted in the presumably unlikely (but unfortunately possible!) event of a security breach occurring during such testing? Unless questions such as these have been carefully considered, and can be shown to have been carefully considered, is it rational to characterise the system as 'secure'?

The question of the rationality of the ascription of security to a system is an important one that we have been addressing. The rationality of a statement, or belief, or ascription is a property of that statement with respect to a certain logic. An argument, or proof, can be rational or valid in one logic but not in another. So although there is probably little dispute as to the kind of classical logic appropriate for proving certain properties of a security-critical system kernel, there could well be dispute about the kind of logic appropriate for convincing a certifying authority that a proven system is in fact secure. Consider for example the following scenario:

(i) An evaluator requires that software to be certified must be mathematically verified using only a certified theorem prover.

(ii) It is convincingly shown that the only available certified theorem prover is also capable of proving the following

\[ \text{theorem:} \text{FALSE}; \]

and hence is mathematically suspect.

The obvious question that now arises is the following: Under what circumstances is it reasonable to invoke the meta-rule (escape clause)
(iii) If the software developer, design authority, customer authority, and certified evaluator all agree, then (i) and (ii) are deemed not to be in conflict.

Now this is an issue that can be decided only in the context of the enterprise. Our enterprise modelling methodology is attempting to argue for, develop and utilise a logic in which such a decision could be characterised as rational.

Thus a particularly important class of conversations that is being investigated is the class of faulty (or potentially faulty) ones — that is, conversations in which, or perhaps as a result of which, two parties each believe a common understanding has been achieved or commitment made, but this belief is in fact false (perhaps as a result of misunderstanding or because of contradictory unspoken assumptions). The reason why we hold faulty conversations to be important is that in many cases of failure of a secure system, the mechanical part of the system is in fact correct with respect to its specification, perhaps because of the correct engineering deployment of fault-avoidance, fault-removal and fault-tolerant mechanisms and techniques. What was faulty and caused the failure was a conversation structure, perhaps one involving mechanical-human interfacing or between two human operators in some formal but not mechanically-mediated dialogue.

But the correct application of fault avoidance, fault removal, and fault tolerance to such conversation structures is not yet known. It is not obvious that the current application of these techniques to a system extends to their application to a conversation about a system. (Their mechanical application depends crucially on the existence of a precise structural or behavioural specification of the fault-free system. It is not clear that a precise structural or behavioural specification of a fault-free conversation is a coherent concept.) This is a problem that will receive a lot of attention. It is possible that, as a result, deeper and more generally applicable notions of faults, failures, fault avoidance, fault removal and fault tolerance will result.

ACKNOWLEDGEMENTS

Many thanks are due to Mike Martin, John McDermid, Phillip Morris and Brian Randell for discussion and comment. Work on security at the University of Newcastle upon Tyne is supported by RSRE Malvern, and I am grateful to Ruaridh MacDonald for his continuing support.

References


