Modelling real-world issues in dependable communications systems

J.E. Dobson
M.J. Martin

Abstract

Many system models of dependability have been proposed, and this paper is an attempt to provide yet another. Dissatisfaction with existing models arises from the fact that they deal solely with the computer component and ignore the roles played by people in the overall scheme of the system operating in its environment. It is argued that a carefully defined terminology which covers not only the computational aspects of dependability but also the human rules, roles and relations involved is required in order to clarify the various responsibilities in designing, analysing, and using dependable systems. Such a terminology is developed in conjunction with a set of models of a communications system, and it is shown how this naturally leads to a taxonomy of possible weaknesses and an analytical method of system assessment with respect to issues of dependability.

The work described in this paper will be presented in shortened form at the Safety and Security Symposium organised by the Centre for Software Reliability to be held in Glasgow in October 1986.

Series Editor: M. J. Elphick
© 1986 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

Modelling real-world issues in dependable communications systems.

Newcastle upon Tyne: University of Newcastle upon Tyne,
Computing Laboratory, 1986.

(University of Newcastle upon Tyne, Computing Laboratory,
Technical Report Series, no. 219.)

Added entries

MARTIN, Michael Joseph
UNIVERSITY OF NEWCASTLE UPON TYNE.

Abstract

Many system models of dependability have been proposed, and this paper is
an attempt to provide yet another. Dissatisfaction with existing models
arises from the fact that they deal solely with the computer component
and ignore the roles played by people in the overall scheme of the system
operating in its environment. It is argued that a carefully defined
terminology which covers not only the computational aspects of depend-
ability but also the human rules, roles, and relations involved is
required in order to clarify the various responsibilities in designing,
analysing, and using dependable systems. Such a terminology is developed
in conjunction with a set of models of a communications system, and it is
shown how this naturally leads to a taxonomy of possible weaknesses and
an analytical method of system assessment with respect to issues of
dependability.

About the author

Mr. J.E. Dobson is Principal Research Associate at the Computing Laboratory,
University of Newcastle upon Tyne.

Mr. M.J. Martin is working at MARI Advanced Microelectronics Ltd., in
Newcastle upon Tyne.

Suggested keywords

ENVIRONMENTS
RISK ANALYSIS
SAFETY
SECURITY
SYSTEM ANALYSIS
SYSTEM MODELLING

Suggested classmarks (primary classmark underlined)

Dewey (18th): 001.64404  658.47
U. D. C. 519.687  519.718
MODELLING REAL-WORLD ISSUES
IN DEPENDABLE COMMUNICATIONS SYSTEMS

J.E. Dobson
Computing Laboratory, University of Newcastle upon Tyne

M.J. Martin
MARI Advanced Microelectronics Ltd.

ABSTRACT

Many system models of dependability have been proposed, and this paper is an attempt to provide yet another. Dissatisfaction with existing models arises from the fact that they deal solely with the computer component and ignore the roles played by people in the overall scheme of the system operating in its environment. It is argued that a carefully defined terminology which covers not only the computational aspects of dependability but also the human rules, roles, and relations involved is required in order to clarify the various responsibilities in designing, analysing, and using dependable systems. Such a terminology is developed in conjunction with a set of models of a communications system, and it is shown how this naturally leads to a taxonomy of possible weaknesses and an analytical method of system assessment with respect to issues of dependability.

The work described in this paper will be presented in shortened form at the Safety and Security Symposium organised by the Centre for Software Reliability to be held in Glasgow in October 1986.

1. Introduction

1.1. The Concept of Dependability
The word "dependability" has been introduced by Laprie[1] as a generic concept subsuming such system characteristics as reliability, availability, safety, and security. The usefulness of such a neutral word is shown by the realisation that it leads to the discovery of useful analogies. For example, Dobson and Randell[2] advocate a new approach to secure system design based on considering security and reliability (concepts that are usually thought to be distinct) as different special cases of dependability. But as has been argued by a number of authors[3,4], concentration on formal aspects of dependability fails to capture many of the real problems of achieving dependability in practice. Many current formal models do not make allowance for
dependability factors related to the management of systems by more than one jurisdictional authority, for example. Nor do the models formally recognise the multiplicity of roles played by people in the overall scheme of the system operating in its environment.

As an illustration of the inadequacy of such models, consider a typical definition of a "safe" computing system as one that should never kill anyone nor cause injury or damage to its users or environment. Although this is of course a perfectly sensible definition at a rather vague level (who decides whether and how much damage has been caused?), it is not at all clear how it would translate into a formal specification for the behaviour of programs: nor is it obvious how to analyse a failure with regard to the specification. What the definition does indicate is the importance of human-related issues in the specification of a system and its environment. There are very similar issues too in the definition and analysis of systems that are intended to be "secure" in the sense of not revealing secrets entrusted to them. We can encompass both of these concepts by using the term "dependable" to mean the satisfaction of requirements such as safety and security which have connotations over and above mere conformance to a functional specification. Such connotations will include factors associated with the system itself and its relation to people[a], as in the definition of a safe computing system given above, and will also include factors which relate the people in the environment to each other, as in an electronic funds transfer application where a computer message can be a legal instrument which creates a contractual obligation between the parties concerned.

What we claim is new about our approach is the attempt we have made to provide a rigorous framework which allows for the expression of the activities and interests of people and organisations. We have also allowed for the uncomfortable but true fact that there are some people and organisations who see their interests as being best served not by the correct operation of the system but by its misbehaviour. It is of course true that (in principle at least) the intentions of the latter can be frustrated by the design and development of a system which is provably correct in all respects, including its applications; but such a system is a system of the future, and probably always will be. In any case, it is not our intention to develop or design or even specify such a system; rather, we are concerned with developing a model and a method which can be used to analyse a system (whether the system is provably correct or not) with a view to determining its vulnerabilities (if any), the external threats that can exploit those vulnerabilities, the consequences of an actual execution of a threat, the countermeasures that might be available, and the decisions concerning the relative costs and probabilities of the corresponding risks and assurances.

Such a programme is ambitious, and it is not claimed that the current paper solves all these very difficult problems. What is claimed is that in order to begin to attack these issues, a rigorous[b] model, or more properly set of models, must be developed.

[a] We are using "person" and "people" to include not only individuals but also organisations - legal personae in fact.

[b] by "rigorous" we mean "based on logic and mathematics". We distinguish this from "formal" meaning "specified in formal notation" (i.e. one with defined syntax). It is a mistake to confuse rigour and formality; they can be independent, though when used correctly they can reinforce one another. We hope in this paper to be rigorous without formality.
In addition to a computational model which allows formal reasoning and correctness proofs, the models must allow for the specification and analysis of real-world constraints and properties. There are many human factors which cannot be well expressed in current models designed only for reasoning about the correct operation of computer systems. Human error and incompetence is the biggest single threat to dependable systems, and analysis of the consequences thereof is the biggest single problem associated with that threat. We believe that our model must permit the expression not only of a failure in dependability but also of any liability which attaches to the failure.

Once the notion of liability has been admitted, one has left the realm of the computer component of a system. If a computer-controlled nuclear plant melts down, one does not reprimand (whatever may be left of) the computer. Instead, it is essential to recognise that there are a number of relevant parties with a variety of roles to play in the environment of a system, and it is to one or more of these parties that any liability would be attached. Such parties include not only the users of a system, but its designers, builders, owners, and would-be penetrators.

What is common to all these parties is intentionality (used in the philosophical sense that one can ascribe to them hopes, fears, beliefs, intentions, and so on). Although we have used philosophical analysis in an attempt to achieve rigour, we do not wish in this paper to analyse intentionality. We merely note that it is a major defect of many models that they have limitations because they deal solely with non-intentional components, and hence cannot be used for modelling real-world issues. In particular, such models fail to recognise that the major feature of the systems they attempt to analyse is that such systems are usually designed to allow the interested parties to engage in a market relationship—something is bought and something is sold, and sometimes something is stolen—and hence the parties have something which it is in their interest to protect and maybe conceal. In this paper we shall refer to that something as wealth, recognising that it is a multi-variate component of the internal state of the interested party about which little further can yet be said. In particular, it is not always simply monetary value. Reputation, for example, counts for a lot.

Finally, our search of the literature on safety and security[5,6] as applied to computer systems has revealed a not unexpected confusion of words. Some words are used to mean very different things by different authors. (For example, "integrity" has been used to mean both "no unauthorised amendment of records" and "no introduction of fake messages". That these are separate aspects of dependability can be seen by considering the countermeasures available. Encryption can protect against the first but will not necessarily prevent an attempt to capture and replay previous genuine but encrypted messages.) Sometimes different words are used to mean the same thing (e.g. "integrity" and "reliability" in certain cases). Some words appear to have no well-defined meaning at all ("privacy" for example[7]). We have tried to use words in a way that conforms to common usage as far as possible, but sometimes common usage is neither clear nor consistent. Our paper therefore presents a number of proposals for a uniform use of words, drawing clear distinctions where it is possible to do so, on the basis of our rigorous approach. There are enough difficult problems in this area without adding to them through confused terminology.
1.2. Justification of Proposed Approach

The main motivation behind the proposed approach is that it should allow the explicit treatment of interested parties and the way they view the system. The expressive power and accuracy of the sociological model is as important as the formal correctness of the computational model. By "sociological model" we mean the relations between the interested parties, the differing roles that a particular party can play, and the different views of the system associated with each role. One particular role that is of major interest is of course that of the system analyst. We are particularly concerned with the way in which the analyst might examine the system for potential weaknesses and provide assurance that actions and responsibilities following the exploitation of a weakness have also been analysed and defined as far as possible. We shall return in a later section to a discussion of the nature of this analysis and attempt to show how it can be performed in the terms of our models.

As mentioned earlier, we have tried to make our models rigorous and in principle capable of formal expression, and a formal representation of our models is in course of preparation. For the purposes of this paper, however, we have eschewed formality and have presented our definitions in what we hope is clear and correct English. It is crucially important to seek a clear definition of roles and their associated viewpoints and of dependability policies and rules, and to make a clear distinction between closely-related and oft-confused terms. Definitions and distinctions of this nature cannot be easily expressed in the low-level computation-oriented specification languages so widely used in computer science; something more abstract is required.

2. A Systems Approach

2.1. The Importance of Policies

Two concepts are fundamental to our model: policies and rules. A policy is an often unstated and ill-defined set of objectives that an organisation or individual appeals to as part of a decision-making procedure in cases where the topic is sufficiently important to preclude a mechanical method of choice. The reason for not explicitly stating a policy is not only that it might be difficult to capture except in the most general terms, but that it might involve secret or covert motives which the organisation or individual would not wish to have fully exposed. Our model recognises this and does not require a policy to be explicitly stated. However, a policy often results in rules of construction, composition, and behaviour and guidelines for conduct. In the absence of a definition of the policy, these rules and guidelines, though derived from the policy, can be taken as a (perhaps incomplete and inadequate) expression of it. Another way of putting this is to say that certain rules and guidelines are in conformance with a policy, and that certain other rules are not in conformance with the policy. When discussing this and similar issues in actual instances, it is important to be clear as to which policy is being referenced, since there may be a multiplicity of policies by virtue of the variety of organisations and roles, and rules that are in agreement with one policy might well not be in agreement with another. In such a case it might be necessary to define what resolution methods would be required to solve potential conflicts of interest.

Our approach is based on an architectural set of models of a dependable communication system. By "architectural" we mean that the set of models is designed to show
components and the relations between components rather than to act as an implementation specification. Of the many attributes of an architecture, the one that we are here most interested in is that an architecture does in fact incorporate a policy. Certain externally imposed constraints have to be respected and fundamental decisions have to be made on the basis of political considerations (which can, of course, override any technical ones).

We regard technical specifications and considerations as being made on the basis of rules (e.g. of construction, composition, or behaviour) which the system must observe or enforce; and we have defined a policy as being a predicate over such a set of rules. It might be possible to say such things as a rule is (or is not) conformant with a policy, or a set of rules is (or is not) a complete statement of the policy, and so on. We will give examples of possible sets of rules later on; for the moment the important thing is that although a policy need not be explicitly stated (since it may well embody commercial secrets), the rules must be, in order for determination of whether or not the predicate applies.

It might seem that a policy in the way we have defined it is redundant, and that we are just identifying a subset of rules that satisfy the predicate. But this is not so for two reasons. The first is that there are potentially many policies and an important issue is to model the possible conflicts and resolutions concerning rules that satisfy some policies and not others. The second reason is that policies in practice change and the fact that a rule satisfied a policy yesterday does not necessarily mean that it will do so today. We wish to model the notion of a rule being tested against a policy to see if under the circumstances the rule should be suspended or is no longer applicable.

We are stressing the importance of communications systems in our approach, since we are interested in an environment which contains "distributed computing systems" and we wish to model the nature of the communications between the various computers in the system. In particular, we recognise the non-instantaneous nature of the communications medium and the fact that it and the computers are capable of failing independently of each other. But the problems of dependability have a wider domain than mere computers, and our real interest is in "distributed systems", i.e. in systems which could include people and machines that are interacting with, and perhaps through, one or more computing subsystems and communications networks.

An important feature of such interaction is that it is very frequently provided through a set of services supplied by an independent operator (e.g. a PTT), and this gives rise to a set of constraints and problems that are to some extent outside the scope of the communicating parties. As a particular example in the case of communications networks, it should not be the responsibility of the network operator to enforce any dependability policy other than his own. Thus in order for dependability policies to be formulated and enforced over domains that include networks, all the rules under which the network operator provides his services must be explicit and available for public inspection. It is then the responsibility of the service user to determine whether or not those rules are in conformance with his own dependability policy.

An example of the kind of conflict that might arise from the application of differing security policies is as follows. It might well be that a particular organisation has a security policy that results in a rule to the effect that it never reveals its encryption/decryption algorithms and keys to anyone not employed by the
organisation. On the other hand, the PTT may have a policy of wishing to assist in national security and therefore applies the rule that it will carry encrypted traffic only if has previously been supplied with the decryption algorithm and keys (so that it could in principle check that the traffic is not subversive, for example). In this case, the user organisation has to choose between not using the PTT services for its encrypted traffic and changing (or compromising) its own security policy.

2.2. (System-Environment) as Basic Construct

The basic construct of our models is a system which exists in an environment of relevant parties. The distinction between these entities is that only the relevant parties are deemed to have intentionality; the system may exhibit behaviour and have internal state, but it has no volition (is non-intentional). At this level of abstraction, there is only one system but there may be any number of relevant parties. If we were to decompose the system we would find that it is recursively structured in that it is built out of smaller systems of the same structure (i.e. it is self-similar); but such an internal examination of a system reveals no new relevant parties. Thus all the relevant parties are visible at the outermost level. We do not attempt to decompose or say anything about the internal structure of the relevant parties. More will be said later about the composition and behaviour of the system; for the moment we wish to examine the relations between the relevant parties.

There are two kinds of relevant parties, distinguished by the role that they play and their relation to the system. On the one hand there are those that are related to the system and to each other by a shared context involving wealth which can potentially be changed by system events (examples are network operators, VAN suppliers, and end-users; also spies and competitors); and on the other hand there are those parties who are relevant but disinterested (examples are lawyers who negotiate over disputes, arbitrators, and liability insurers). Instances of the former kind of interested party have of course a significant part to play in our model, and because it is admitted that they could well have secret or selfish purposes to promote in addition to their overt role in the system, we shall call them axegrinders. Relevant parties that are not axegrinders will be termed disinterested parties: such parties do not have a direct relation with the system, but only an indirect one by virtue of their relations to axegrinders. Strictly speaking, disinterested parties are disinterested only with respect to the domain of discourse under consideration; they may well also be axegrinders having their own policies and rules with respect to some other system. With respect to any particular system, only axegrinders can have dependability policies and publish rules, since policies and rules reflect an interest in the system.

It will be important for an analysis of any particular system and environment to discover, state, and clarify the relations between the various axegrinders (e.g. Uses-Services-Provided-By, and Attempts-to-Steal-Secrets-From), between the axegrinders and the system, and between the axegrinders and the disinterested parties (e.g. one axegrinder may not wish to use the services of another if the former discovers that the latter enjoys no Is-Insured-By relation). The importance of the relations between the axegrinders and the system derives from the fact that although the system does not itself have wealth as a component of its internal state, the axegrinders do, and the value of such state variables can be changed (possibly drastically) as a result of system events.
Our fundamental picture of the relation between axe grinders and the system is that of the system acting as a communications medium between axe grinders. An axe grinder stimulates the system by creating and submitting a message, as a result of which the system responds by delivering messages to other axe grinders. In addition, there are certain external stimuli to the system that can be regarded as messages, albeit in a rather degenerate sense: an example is a bulldozer ploughing up and breaking a communications cable. We regard all intended stimuli as being associated with an axe grinder even if the connection is sometimes a little delayed or remote; thus accounting records for example can be regarded as deferred messages from the system builder. Unintended stimuli may or may not be associated with an axe grinder. (The importance of this association is that should some stimulus be erroneous in some sense, one may wish to ascribe blame to someone.)

A message is an important primary symbol in our model, and it has two related functions. Firstly, we have the notion of communication between axe grinders being mediated by messages; and secondly, we employ the notion of a stimulus/response model of system behaviour in which the form of the stimulus and the response is also that of messages. Note that although we have used language that implies that the communicating axe grinders are separate parties as if to a telephone conversation, we recognise and include the common case of a single axe grinder using the system to assist in the activity of axe grinding. An example would be a safety officer monitoring and controlling the state of a plant through a terminal. In this case we would distinguish the roles of the same axe grinder acting as sender and as receiver of messages.

3. Modelling Principles

Our modelling approach is to develop not a single model of a dependable system but a related set of models. Each model ignores detail irrelevant to its purpose and allows concentration on the set of features it is designed to exhibit. It is obviously important that the different models fit well together so that they can all be seen to be models of different aspects or facets of the same underlying entity. We shall now briefly describe our basic models of a dependable communications system.

There are in fact four models in our set: a model of system behaviour, a model of system composition, a model of messages, and a model of communication. Each of these will be described in a separate subsection, but all the models will be used for the same purpose, which is to provide the basis for a clear and well-defined terminology, to allow us to develop a taxonomy of types of dependability breach and to assist in the analysis of real systems with respect to vulnerabilities and the consequences that might arise should those vulnerabilities be exploited. Section 4 of this paper describes the basis of the taxonomy and section 5 provides an outline of the method of analysis that we envisage.

3.1. Model of System Composition

We need to begin this section with a preliminary remark on terminology. There are two aspects of building systems, which we distinguish by the terms construction and composition. Each of these activities may have its own sets of rules. System construction concerns the components themselves, of what kind they are and of what technological material they are made. An example of rules of construction is "It shall employ a 68020 processor and shall be programmed in Ada". Such instructions are
communications between the system specifier and the system builder but do not usually pass through the domain of the system itself; rather they form part of the context in which the system is built. **System composition** concerns the manner of connecting the components together and is topological in nature. An example of a system composition rule might be "The peripherals shall not interface to the memory directly but must go through an I/O processor". The difference between rules of construction and rules of composition is that adherence to the former is a requirement on the process of construction and is therefore a behavioural constraint on the system on which the construction is performed, whereas adherence to the latter is a compositional constraint on the constructed system itself.

It is important for our model that we recognise that a communications network is itself a system. Indeed, a network may be regarded as just an implementation mechanism or component used in the composition of systems. Although the dependability of the network can be considered in isolation from the overall system, including services and applications, of which it is a component, this is a somewhat limited exercise which fails to take proper account of the dependability of the overall system and the role of networks within it. The significant property of networking technology is that it makes it possible to put systems together to make bigger systems. Thus the problems of networks are those concerning systems built from systems, and the issues that our models should address are those of putting systems together. This will have a number of implications for the statement of the relations between the axe-grinders.

A system is composed of interconnected **components**, which can be given names. A component is a self-contained computational entity provided with **sockets**. No information may enter or leave a component except through one of its sockets. As seen from a component, its sockets are uni-directional (i.e. input or output). A component may either be **atomic**, meaning that it has no internal structure visible at this level of abstraction, or it may be recursively composed of a set of interconnected smaller components (i.e. of systems). Components are interconnected by linking their sockets (with the obvious constraint as to gender). In the terms of this model, a conventional uniprocessor system is typically viewed as an example of an atomic component; a simple network can be regarded as a component that provides lots of sockets but rather limited computational power.

In order to apply this notion to dependability, it is convenient in addition to associate labels with sockets so as to distinguish between different kinds. This will enable us to model the provision of rules concerning system composition (e.g. of the form "red-to-red and black-to-black", as used in the composition of secure systems). These composition rules should presumably be derived from the dependability policy of the system builder or of the system owner and may permit the ascription of the attribute "dependable" only if the system is constructed and composed according to the rules. (There may be other constraints of course, including those on behaviour.)

With this model of system composition, the behaviour of a component could be expressed as a predicate over the sequences of messages entering and leaving the sockets of a component. It is a fairly straightforward matter to extend this notion to the interconnection of components: for example, if we have two components D1 and D2 each with a single pair of sockets with message sequences **InD1, OutD1 (i = 1,2)** whose behaviour is described by predicates Pi(InD1, OutD1), then one form of description of the behaviour of D1+D2 (meaning the output of D1 is connected to the input of D2) is
P: there exists M such that P1(Ind1,M) and P2(M,OutD2)

where M represents the internal message sequence between the components.

3.2. Model of System Behaviour

Our basic premise is that the behaviour of a system or component can be described by a trace of the stimuli received by that system or component and the responses it exhibits. As previously described, stimuli and responses take the form of (possibly degenerate) messages, and responses from one subsystem or component can act as stimuli to another. Since stimuli and responses have form, this allows us to construct a trace space of all possible modes of behaviour of the system. This trace space is formed by the association of stimulus and response messages with the externally visible sockets of the system; thus we can in principle have a parallel set of traces of a system that has multiple sockets. At any instant in time, the current state of the system will be represented by a point in the trace space of behaviour up to that instant. We can also describe the behaviour of any system component in just the same way by considering the trace space of the messages entering and leaving the sockets of the component. In what follows, however, we shall ignore the recursive composition of the system and treat it as if it were atomic.

Within this trace space we can define boundaries around certain regions and characterise a boundary by the attributes of the behaviour of a system that lies entirely within it. Such a boundary we term a selector; thus we can have selectors for desired behaviour, for dependable behaviour, for erroneous behaviour, and so on. These selectors will be described by rules concerning the attributes of behaviour of a system constrained by the selector. These various selectors will of course overlap in interesting ways - we can describe the deliberate leaking of sensitive documents to the press in order to make a political point as behaviour that is outside the "secure" selector but within the "desired" selector. This example is a case of a more general point, that it is clearly possible to have a variety of possibly inconsistent[c] selectors arising from different sets of rules. A set of rules can be said to be complete if it yields a selector which is closed; otherwise it is incomplete. If we assume that there is a complete set of rules that describe a dependability selector, we are in a position to introduce some definitions of terms which are widely used, though less often widely used with a common precise meaning.

A vulnerability is a point in trace space (and which therefore describes a possible mode of behaviour of the system) which is outside the closed region defined by the dependable selector. A dependability breach is a system event represented by the occurrence of a transition from a point inside the dependability selector to a point outside. A threat is a stimulus that causes a dependability breach (or that could cause a possible security breach). A dependability lapse is an action (or inaction) of an axe-grinder that permits the possibility of a certain threat. The important point is that a lapse occurs when an axe-grinder (or employee or servant) fails to conform to a behavioural specification appropriate for a person.

[c] two selectors are inconsistent if they contain no common points. Actually, this is usually weakened in practice to "no common points which describe interesting behaviour".
Such a lapse need not immediately precede the threat. For example, a malfunctioning piece of equipment is not itself a lapse, though it may imply a lapse in the quality control of the equipment supplier. This definition does not require a lapse to be identified or even observed; all that is required is that a disinterested party could agree that an axe-grinder has failed and that the system has exhibited a breach as a result. Since a breach is a transition from a point inside a dependability selector to a possible point outside, we can define a **countermeasure** as a restriction on trace space. The effect of the restriction is that certain modes of behaviour or breaches which were possible before the countermeasure was imposed are now no longer so. Thus the countermeasure increases the cost of perpetrating the threat and/or decreases the consequential benefit of the breach to the perpetrator. Countermeasures may also be probabilistic in nature; that is, they may merely serve to reduce the likelihood, rather than prevent the occurrence, of a breach.

### 3.3. Model of Messages

The standard form of a message in our model contains three fields: sender, message text, recipient set. The sender is the name of the component of origin and the recipient set is the names of the destination components intended by the sender. At the outermost level, a system is a component with sockets. Messages arrive at the sockets from axe-grinders and are delivered from the sockets to axe-grinders. However, our model of a message insists that the sender and recipient fields name components and not axe-grinders. Thus in the case of these messages that are sent and received by axe-grinders, we have to associate a specialised component called a **portal** with the connection of an axe-grinder to a system. Portals correspond to such things as keyboards and screens, and are provided with only one socket which can connect only to a system socket. We can now regard messages to and from axe-grinders as naming the corresponding portals. For most purposes, however, the distinction between an axe-grinder and the corresponding portal(s) can be ignored – i.e. we can treat any portals as being part of the axe-grinder.

Message text is **uninterpreted** in our model of messaging; thus a message is simply abstract syntax. Questions of interpretation and semantics will be discussed in the next subsection, on the communication model. It is because of our separation between the syntax and semantics of communication that the sender and recipients of a message are taken to be components and do not identify axe-grinders. Acts of God can be modelled as messages from an unknown sender component.

Because a response from one component can act as a stimulus to another, we have the possibility for the modelling of message passing; and since a single stimulus can induce multiple responses from a component, we can model broadcasting or message replication.

### 3.4. Model of Communication

As previously mentioned, our model of communication deals with the interpretation of messages, and the fact that messages can be passed from one component to another. This leads to there being five kinds of entity involved in a communication:

generator, sender, transferrers, recipients, interpreters.
The generator and interpreters are by definition axe grinders who share a common understanding in the context of which the interpretation of the message text is performed. The generator maps the semantics onto a sequence of messages which are transmitted from the sender component (portal) to the recipient components (portals); at each of the latter another mapping takes place from the messages to their semantics. There are thus two interpretation functions performed: one at the generating end and one at the interpreting end. These interpretation functions are difficult to characterise, but they involve the notion of a context which may include all previous messages between the parties. A context may also include messages transmitted by means other than the system of interest. Interpretation functions may also involve wealth functions which change the internal state of the parties. How useful it is to develop a theory of wealth is a matter for further study, but the section on dependability breaches introduces some of the issues that such a theory would have to address.

The role of transferrers in the communication model is potentially an important one. They represent the abstraction of components performing a message switching function or mail server. Although they are deemed to have no associated interpretation function, i.e. they act only on the abstract syntax of the messages they pass, there are obviously correctness proofs of invariance that they have to satisfy, and possible dependability breaches that result from invariance not being observed will have to be investigated.

One important aspect of a communication in the context of dependability, and of security in particular, is that of clearance and authorisation. The interpretation function at a generator will yield a message which acts as an initial stimulus to the sender component (portal). This message may have an attribute called sensitivity assigned to it by its generator. The authorisation of an axe gringer in a particular context is the set of sensitivities of messages that can validly be generated or interpreted by the axe gringer in that context without causing a dependability breach. A clearance is a set of authorisations (and hence by implication a set of contexts) issued to or to be attributed to an axe gringer.

4. Dependability Breaches

In this section we shall define some terminology associated with dependability breaches and begin to develop a taxonomy of breaches based on our models.

We have defined a dependability breach as a transition from within a dependability selector to without. It follows that this was in response to a particular stimulus; the fact that such a stimulus could occur at all may well have been as a result of some previous dependability lapse. The difference between a lapse and a breach is similar to the difference between a fault and a failure in classical reliability theory.[8]

The standard definition of "failure" and "fault" is that a failure occurs when the behaviour of system or component first deviates from its specification, and that this deviation must be as a result of a fault either in a component or in the design or specification of the system. A fault in the design or specification can arise from an earlier lapse by an axe gringer (the system designer or specifier). Although we can ascribe faults and failures to components, it is not always so useful to ascribe lapses and breaches to components or systems. Indeed, we have deliberately chosen the words "lapse" and "breach" because we wish to emphasise the human aspects and consequences
of these failures. Thus although "breach" and "failure" may seem to describe the same phenomenon, that of the event of divergence from a specification, they have different connotations, and in fact have different denotations also. A failure is a relation between a behaviour and a specification, whereas a breach is a relation between a failure and one or more axe grinders. Similarly, a fault is a state of a component or design, and a lapse is a relation between a fault and one or more axe grinders.

We wish to incorporate a certain amount of flexibility in the way we model the tying of lapses and breaches to particular axe grinders. Our model therefore allows us to distinguish three relations:

i) an axe grinder (or set of axe grinders) is to be held accountable for the breach if the axe grinder(s) caused (or could be held in law to have caused) the stimulus that provoked the breach (i.e. the threat).

ii) an axe grinder (or set) is to be held responsible for the lapse if the existence of the possibility for the stimulus was as a result of (in)action by the axe grinder(s).

iii) an axe grinder (or set) is to be held liable for the consequences of the breach if any changes in abstract wealth to recover from or compensate for the breach are associated with the axe grinder(s).

A simple example may make things clearer. A night-time thief walks into a bank and steals some of its clients' money. The thief is accountable for the money that he stole; the bank's security officer is responsible for leaving wide open the doors (of both bank and safe presumably); and the bank is liable for the financial loss to its clients. The fault in the system was that the doors were left open; the failure was that the thief was not prevented from entering the bank because of the fault. Obviously more complex situations arise in the real world, and just which axe grinders (if any) fall into each of the sets may well be a matter for legal experts and other disinterested parties.

4.1. A Taxonomy of Breaches

We have found it instructive to use our models of messages and of communications to develop a taxonomy of different kinds of dependability breach. Our model of a message contains three fields: sender, text, recipients. Since each of these may independently be corrupt or falsified, we have the following categories of breach:

- forgery
  - RealSender/ApparentSender mismatch
- text_corruption
  - RealText/ApparentText mismatch

There are two cases of RealRecipient/ApparentRecipient mismatch:

- eavesdropping
  - RealRecipients > ApparentRecipients
- message_loss
  - RealRecipients < ApparentRecipients

In terms of the communications model, we have the following categories of breach associated with the generators and interpreters:

- impersonation
  - RealGenerator/ApparentGenerator mismatch
espionage  RealInterpreter/ApparentInterpreter mismatch
abuse    RealGenerator not authorised
betrayal  RealInterpreter not authorised
          (the betrayal is perpetrated by the generator)

In addition, there are a variety of breaches that can arise from the fact that interpretation functions are invoked by generator and interpreter. For example, there are a number of sins associated with the deliberate application of a false context to the interpretation function ("rewriting history"). Repudiation is a particular case in which an axe-grinder wishes to assert that a context contains a message trace less than that which it in fact contains. Fabrication is an attempt to assert that the message trace is greater than it actually is.

5. A Proposed Rational Decision Making Procedure

In this section, we propose a procedural method for performing an analysis of an allegedly dependable system. Ultimately we would wish to develop the prescriptive method to the point where tools could be developed in the environment of a decision support system in order to guide the analysis, check for consistency and completeness, and manage the data. Such tools could only be of clerical assistance since the task is one that calls for a considerable amount of human ingenuity and judgement. We shall therefore merely describe the process and provide reasons why we believe it to be a useful one.

The aim of the process is to assist the designer in performing a risk analysis. That is, the designer has a number of tradeoffs to make concerning the liability costs of a dependability breach and the real costs of providing countermeasures to prevent the breach. In extreme cases, such as military security, the tradeoff is easy because it is deemed that almost any liability cost is greater than any countermeasure cost; but in many other cases the decision is less clear cut, and the designer may be constrained by having only a limited budget. Design decisions will therefore have to be taken in order to make the best use of scarce resources and assessments made of potential risks arising from leaving certain weaknesses exposed.

There are four stages in our proposed procedure for making rational decisions concerning risks, which we call Vulnerability Analysis, Consequence Analysis, Countermeasure Analysis, and Risk Analysis. We shall describe each of them in turn, but before doing so, we wish to make a couple of general remarks relevant to all the analyses. The first point is that the analysis will be useful only to the extent that all the relations both between the axe-grinders themselves and between the axe-grinders and the system have been clarified. In general, this may not be completely possible, partly because of difficulties in identifying all the axe-grinders (e.g. the spies and embezzlers) and partly because some of the relations will be rather subtle; consequently some careful interpretation of the results will be necessary. The second point is to reiterate the importance of the rigorously defined terminology in the performance of the analyses, which should make the task easier (but not necessarily possible).
5.1. Vulnerability Analysis

The vulnerability analysis takes a description of the system in the following form:

- the actual composition of the system in terms of the interconnection of its components and sockets;
- a description of the expected behaviour of each component, in terms of the input and output messages over its sockets;
- the rules of system composition;
- any constraints on system behaviour;
- a set of communication patterns or scripts;

together with a threat list which constrains the scope of the vulnerability analysis, e.g. by excluding certain classes of threat from its scope.

The method of analysis is similar to the kind of semi-formal requirements analysis exemplified by the CORE method[9], in which a set of information flow diagrams is created and a set of rules is associated with each of the paths or flows. The various sets of rules imposed by the relevant axe-grinders provide the requirements on the system as described. Essentially a set of heuristics is applied to identify unusual features and possible conflicts and inconsistencies both in the rules and in the flows.

The output of the vulnerability analysis should be a description of potential system weaknesses expressed in terms of the taxonomy of vulnerabilities that we have indicated.

5.2. Consequence Analysis

One use of the list of potential system weaknesses is as a basis for the analysis of the consequences of any exploitation of the weaknesses. The system model described above will be required in order to follow through the propagation of a breach through the system.

Such an analysis is obviously dependent on assumptions concerning the context and likelihood of the exploitation, and those assumptions will be fed into the analysis. The analysis will necessarily involve human judgement since only the axe-grinder will have any knowledge of the possible chain of consequential liabilities that might follow from a breach and how such consequences will affect the distribution of wealth. The output from the analysis should be an actuarial time-dependent cost model of the consequences of each weakness being exploited together with an estimate of the probability of the cost penalty being invoked.

5.3. Countermeasure Analysis

A second use for the list of potential system weaknesses is in conjunction with a list of possible countermeasures. There will obviously be a cost function associated with each countermeasure, and it may also be necessary to specify an efficiency measure which may well vary with the cost. Analysis will be necessary in order to determine the effects of various combinations of countermeasure, since they may either reinforce one another, or interact in ways that prevent their full effectiveness. The output from the analysis should be actual costs and probabilities of effectiveness of optimal configurations of countermeasure.
5.4. Risk Analysis
The final analysis in the decision support suite is the comparison of various scenarios
developed with the Consequence and Countermeasure Analyses. Each comparison is
between a liability cost model and a countermeasure cost effectiveness model. Its
purpose is to allow the analyst or designer to quantify the assurances given by the
protective measures and the risks associated with waiving those measures, and thereby
to provide a sound basis for design decisions to be taken on dependability issues.

6. Future Directions
We have attempted in this paper to define some concepts and terms which we believe to
be required for a rigorous statement of real-world issues in dependable communications systems, and we have argued that a system designer or analyst needs to have a
precise vocabulary in order to undertake the task of making rational decisions concern-
ing the strengths and weaknesses of the dependable system. As we indicated ear-
lier, a logical basis is required for the formalisation of the ideas developed in this
paper and it is intended to examine the use of Z for this purpose. Possibly a more sig-
nificant task is to develop further the decision-making procedure outlined in section
5 using CORE as a starting base.

Our conclusion from the literature that we have read is that, although the sub-
cultures of computer safety and security are simply special cases of dependability,
they have quite a lot to gain from earlier lessons learned by the software reliability
community concerning the importance of standard terms and definitions. But such
standard terms and definitions must be extended to include things that take place out-
side the boundary of the computer system. It is not sufficient to regard a real problem
as solved when we have a formal solution to an inadequate abstract model of the real
problem. It is important to understand the differences between our models and reality.
The systems which we try to analyse must be examined not only in terms of our comfort-
able models but also from the point of view of the uncomfortable realities of the out-
side world.

Acknowledgements
It is a pleasure to acknowledge the contribution of Brian Randell, who was exposed to
an earlier version of these ideas and not only told us what we had done wrong, but also
sometimes told us what we should have done had we done it right. Our stimuli have also
provoked responses from Tom Anderson and Simon Wiseman, and to them too we are
grateful. John Rushby has made unwitting contributions to our thoughts through his
typically controversial remarks. The work at MARI is supported under the RACE initia-
tive of the Commission of the European Communities, and we would like to thank members
of the SPRINT Consortium (Project 1026) for their contributions to our thoughts. The
work at the University of Newcastle upon Tyne is supported by the Royal Signals and
Radar Establishment, Malvern.
References


