Fifth IFIP/IEEE Int. Workshop on Distributed Systems : Operations & Management (DSOM'94)
October 10-12, 1994, Toulouse, France

Policies for Safety-Critical Systems: the Challenge of Formalisation

Jonathan D. Moffett[1] *and John A. McDermid*

Department of Computer Science, University of York
Heslington, York YO1 5DD, UK

{jdm, jam}@minster.york.ac.uk

Position Paper

1. INTRODUCTION

Policies are used to define the goals, responsibility and authority of stakeholders in some area of interest: in our case, complex safety-critical systems. Previous work, e.g. [Moffett 1993], has pointed out the advantages of making policies explicit and formalising[2] them, so that human managers can recognise and reason with them, perhaps with some automated support. The benefits of this for management tasks such as security administration and fault management in complex distributed systems have been recognised: they range from increased efficiency from the automation of simple repetitive management tasks, to making it possible to cope with the complexity and scale of very large systems.

This position paper describes the characteristics of safety policies in some safety-critical application areas; describes existing work in the formalisation of distributed system management policies, as a relevant starting point; and points out the areas which need to be tackled in order to provide tools and automated support for the management of safety-critical applications.

2. SAFETY-CRITICAL SYSTEMS

Safety-critical systems are characterised by the possibility of their malfunction causing injury to human beings, or damage to the environment. Examples include:

- Aircraft and other means of transport, where failures may result in injury or death to operators, passengers and the general public;
- Power stations (particularly nuclear) and chemical process plants, whose failure can result in immediate hazard to operators and consequential hazards to the general public as a result of explosions and/or emission of noxious substances into the environment of the plant;

---

[1] Supported by UK SERC grant GR/G49531.

[2] Not necessarily by a complete formal notation such as a formal specification language, but perhaps by introducing a controlled vocabulary of policy entities in the development and operation of the system.

- Medical care or operation systems, such as intensive care units and medical robots. The nature of the risk is slightly different in these systems, as the patients are typically in life-threatening situations in any case. The risk in using the system is that it will either fail to reduce the risk to life or actually harm the patient, but in any case this risk has to be weighed against the likely benefits of using the system;
- Information systems on whose accuracy human life or health depends, e.g. emergency services control systems, and medical diagnosis systems.

It will be noted that only the last of these are pure information systems, and even in these cases the information may not necessarily only be computer data: emergency control systems deal with a combination of data and voice communication; while some medical diagnostic systems are concerned with activities such as capturing information, such as abnormal cell counts on a microscope slide, as computer data. So policies about safety-critical systems have to take into account the various media in which the systems operate: mechanical, electrical, optical, voice, as well as humans interacting with computers.

A second characteristic, true of the first three of our examples, is that safety-critical systems are typically embedded real-time systems, in which computers are only a part of a more complex, typically electro-mechanical, system[3]. This has several consequences:

- The task of design is complicated by the fact that multi-disciplinary teams are involved in the development of the system. The need for an accurate common understanding of the terms used in system policies is therefore increased. A phrase such as "leave all settings at normal in the event of any component failure" may have completely different meanings to software and chemical engineers. Misunderstandings between these two disciplines have in fact led to accidents [Kletz 1983];
- Hazards can occur as a result of unexpected interactions between the various components of the system, as well as failures of individual components. Therefore unexpected behaviour must be allowed for.

The need to protect humans requires a very low rate of catastrophic failure in safety-critical systems. A mean time between catastrophic system failures of better than 10^9 hours (~10^5 years) is typically demanded for civil aircraft certification. One consequence of this is that this requirement cannot be met by individual components of the system, and therefore appropriate recovery procedures are needed in the event of component failures, and these will constitute part of the safety policy.

Further, almost all safety-critical systems are subject to certification; not only must they be safe, but they must be seen to be safe by being evaluated by an independent agent. Clearly the policies of the certification body may not coincide with those of the manufacturer and customer.

3. A MODEL OF POLICIES IN DISTRIBUTED SYSTEMS

Previous work has formalised the concept of policies in distributed system management, and we expound it here, in order to examine which aspects of it are applicable to safety-critical systems.

[3] See [Leveson 1991] for a general discussion of these.

There are several kinds of policy relevant to a system, including:

- Policies about how a system is to be developed, e.g. "there should be separation of duties between the developers and the independent verification and validation group";
- Policies about the characteristics which a system is required to have, e.g. "the system should be robust to any single hardware failure";
- Policies, represented and embedded within a system, about how a system is to be operated or controlled, e.g.

  P1 The Supervisor [and no other user] is authorised to change the trip settings;

  P2 The operator must initiate the plant shut-down procedure if the operating temperature exceeds its safe limit;

  P3 The system is required to initiate a backup job every Friday night.

  Note that the policy P2 does not specify whether the operator is human or automated.

3.1 Policy Hierarchies

Policies are typically first stated at a high level, and then progressively decomposed through intermediate policies down to a point where procedures, consisting of atomic actions, are specified which will, taken together, realise the high-level policy [Moffett, 1993]. Atomic actions are, in this context, computer system operations or human actions which do not require any further decomposition or explication. We refer to the combination of an agent, a modality (obligation or permission, or their negations) and a procedure consisting of atomic actions, as a low-level policy.

A brief example illustrates a policy hierarchy. There is a power plant, for which one of the high-level policies is:

P4 The manager of the power plant is to ensure its safe operation

One of its components is a boiler, which is known to be safe when its temperature is less than 100°C. It is potentially unsafe above 100°C. Therefore one of the intermediate policies which support policy P4 is:

P5 The manager is to ensure that the temperature of the boiler is never above 100°C.

A number of factors may affect the temperature of the boiler. They include:

- The Close_Valve operation which causes the temperature to increase;
- The Open_Valve operation which causes the temperature to decrease;
- Influences from the external environment;
- Faults in the system.

Clearly, in order to ensure the achievement of policy P5, a full analysis of the system and its environment is needed before defining procedures which will cover all feasible eventualities. However, for the purpose of this example we set out two of the low-level policies which will help to achieve P5:

P6 When Temp > 90°C (to allow for delay) and the valve is not Open, the Operator shall initiate Open_Valve

P7 All agents are prohibited from Close_Valve if Temp > 100° C

3.2 Management Action Policies

The model summarised here and elaborated in [Moffett, 1994] applies only to low-level embedded policies. Even for distributed systems management it is recognised that this provides a very partial view of policies and, as will be discussed below, this is doubly true for safety policies. However, it has acted as a starting point for the formalisation of policies, and offers some useful insights which are generally valid.

We define a subset of possible policies, called Management Action Policies, which consist of agents associated, by means of a mode, with goals on target objects, subject to a constraint. See figure 1.

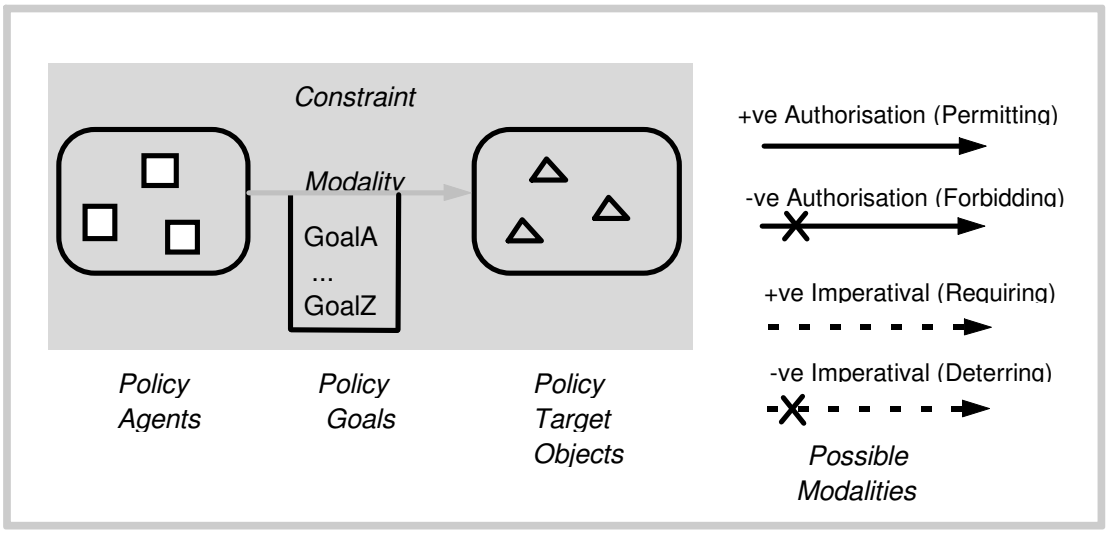

Figure 1 A Management Action Policy

The English word "policy" has a wide variety of meanings, and in the description below we deliberately apply a precise and restricted meaning to it.

3.2.1 Goals

We regard a goal as a condition associated with an object, where the condition may be an event such as an operation occurring, or a state of the object. One of the most powerful features of goals, and therefore of policies also, is that they provide a means of refining down from high-level general goals to the more detailed lower-level goals which will achieve or solve the higher-level ones. [McDermid 1993].

It is clear that it is desirable to be able to express low-level policies in terms of a state to be maintained for an object, but except in very simple cases (where there is an operation "set state" for the attribute in question) it is hard to specify a general mechanism by which this is to be achieved. All policies that are expressed in terms of a state to be achieved must be refined into low-level action policies in order to be expressed in terms of operations.

3.2.2 Agents

We use the term agent to cover both human subjects or stakeholders in a system, and automated processes which represent them. The distinction between policies and goals, in which policies specify agents

associated with goals, provides a convenient way of ensuring that, in our description of policies, no goals are left "dangling" without associated agents, i.e. mechanisms or individuals responsible for ensuring that they are met. We believe that our insistence in involving agents in policies is also valid for safety policies, as a policy is of little value if there is no means of achieving it.

3.2.3 Modes of Policy

Another reason for defining policies to have both agents and goals is that the nature of the association between the two is very important. The relationship may be imperatival in nature, as when an agent has an obligation to achieve a goal. These policies can be implemented by a Situation Monitor, see [Moffett 1994]. Alternatively the relationship may be one of authority, as when the agent has the legitimate power to achieve it. These policies can be implemented by a Reference Monitor [Anderson 1972]. See figure 2. It is fundamental to our view of policies that no goal is achieved unless the agent has both the imperative and the power to achieve it.

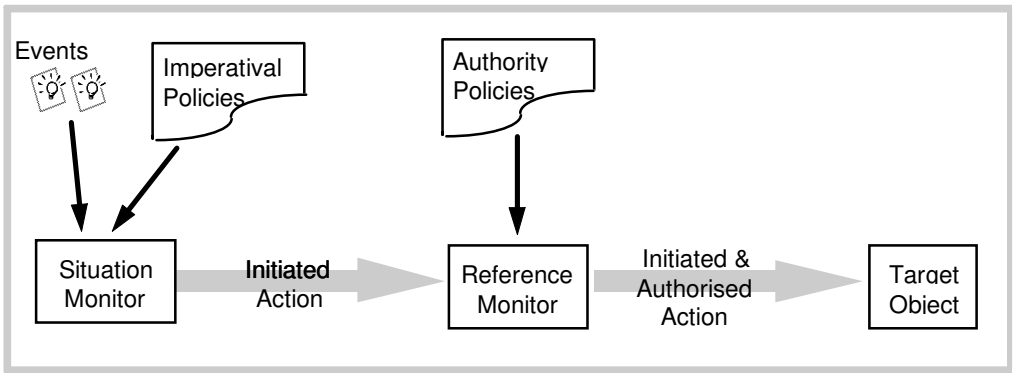

Figure 2 Imperatival and Authority Policies, and their Monitors

This simple view has to be modified somewhat to take into account some realistic safety policies, as will be seen below, but is a useful starting point for analysis of safety policies which explicitly or implicitly call for actions. For example, if there is an imperatival policy in operation such as "the operator shall shut down the plant when safe limits are exceeded", there needs to be a corresponding authority policy, expressed in the access control rules of the system, to ensure that he also has the authority to do so. The two kinds of policy are not necessarily in concert (see the example in section 4.2).

It is also useful to distinguish between positive and negative modes of the policy, as will be seen below (section **Error! Reference source not found.**).

4. SAFETY POLICIES

We will not attempt in this paper to categorise all possible safety policies, but to point out some aspects of them which give problems in formalisation.

4.1 State-based Policies

Safety policies are typically about avoiding hazardous states, i.e. they have the goal "state X must never be reached for object Y", for example policy P5 "the temperature of the boiler must never be above 100°C". This poses problems for their implementation, at two levels of difficulty:

- Within the computer system there is the possibility of the hardware or the operating system changing program variables contrary to the specification. There are of course similar problems in computer security, which are dealt with by introducing a trusted reference monitor which prevents insecure states being reached. An analogous "safety kernel" has been proposed for safety-critical systems [Rushby 1987];
- Worse, we have to take a wider view than computer systems. Unsafe states may be reached by a combination of physical events and failures, design errors and operating errors. We have to think much wider than computer systems in order to identify the possible causes of reaching unsafe states, and techniques such as Fault-Tree Analysis and Failure Modes and Effects Analysis [Vesely 1981; ARP 926] have been introduced for this purpose.

One technique for avoiding or minimising these problems is the introduction of physically separate *safety systems* whose purpose is to monitor for unsafe states and take appropriate action, such as system shutdown, in the event of hazards being threatened. The shutdown option is, of course, not available in all systems; power plants can be shut down, at a cost, but an aircraft flight or a medical operation cannot simply be aborted and some positive action has to be taken to bring the proceedings to a safe conclusion.

4.2 Kinds of State

Safety is not an "all or nothing" condition. In any system there are three kinds of state.

Normal safe state

In the normal safe state the variables of the system are within acceptable limits, e.g. the boilers temperature is below 100°C. As another example, an aircraft is operating within its flight envelope, which provides an adequate margin of safety. Figure 3 shows a very simplified flight envelope model for an aircraft, relating its air speed and the angle of attack (i.e. to the air flow) of its wings. This relates the maximum angle of attack to the minimum air speed in order to remain within the flight envelope, because the slower the air speed the less the permitted angle, down to a point where the aircraft can go no slower without stalling.

Within this state, the policy imperatives, except for the imperative to stay in this state, are not usually explicitly about safety. They are concerned with other factors such as economy, efficiency and human comfort and convenience. Thus policies for economy might be imperatival, but the (overriding) safety policy would define limits of authority[4]

Abnormal, but safe, state

In the abnormal, but safe, state the variables of the system are outside normative limits, but it is still possible to control the system. This is illustrated by the grey area in figure 3. For the boiler example this state is not defined explicitly, but there is likely in practice to be a margin of error over 100°C . The policy imperatives in the abnormal state are changed to emphasise recovery back to a normal safe state, at the expense of the previous priorities.

---

[4] The aerospace industry makes explicit use of the term authority, e.g. Full Authority Digital Engine Control; the distinction between imperatival and authority policies goes some way towards modelling this concept.

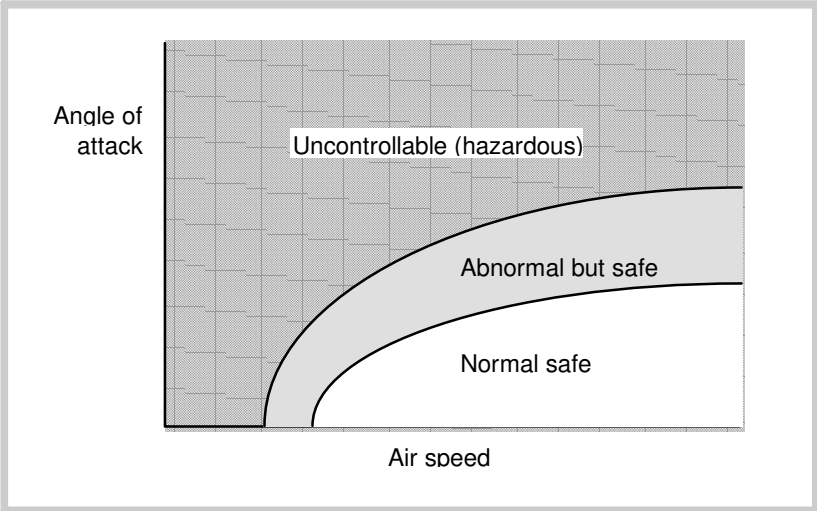

Figure 3 Flight Envelope

Thus any policy model which wishes to take into account recovery after policy violations must be able to displace its normal policies with recovery policies which temporarily override them. This observation applies as strongly to system management policies as to safety policies.

Uncontrollable state

There is a third situation, where the variables are so far from their normal values that normal (and even abnormal) control laws can no longer be assumed to apply and the system may be uncontrollable – the hatched area in figure 3. In a safety-critical system this will usually indicate a hazard.

In this area, automatic controls cannot be relied upon to make correct decisions, and experienced humans may be more likely to recover the situation. This is recognised in many systems by causing the automated system to recognise potentially hazardous states and make recommendations to the human operators but give control back to the humans. In practice, a major area of difficulty is in deciding an appropriate balance of authority between man and machine. Perhaps this is the most substantive issue for work on safety policies.

4.3 Modes of Safety Policy

Distributed systems conform quite closely to the message-passing paradigm which underlies an object-based view of the system. One convenient implication of this is that in distributed system management it is possible to enforce a distinction between imperatival policies implemented by a Situation Monitor and authority policies implemented by a Reference Monitor (see section 3.2.3, above).

The situation is less clear in some embedded real-time systems. For example, policies about the pilot's authority to go outside the flight envelope vary from aeroplane to aeroplane. The A320 airbus conforms to our model of policy modes; there is an authority policy which prevents the pilot performing any action which takes the aeroplane outside its flight envelope. On the other hand, in the Boeing 777 the situation is less simple; the system will not absolutely prevent the pilot from going outside the flight envelope, but the controls will offer progressively greater resistance the further he goes beyond the envelope. This

cannot be regarded as a straightforward policy of either of the modes that are in our model; the model needs to be extended to take a situation such as this into account

4.4 Policies for Development

The acceptable rate of catastrophic failure for some safety-critical systems is very low indeed. A predicted failure rate of better than 1 in 10^9 hours cannot be validated by testing [Littlewood 1993], even using techniques for accelerated testing, in realistic time-scales. The only way in which a figure of this kind can be justified is by taking some "credit" for reliability from the high standards to which the product is developed. Thus policies for system development have to come into play when considering safety policies about the operational system. Examples of these are [DEFSTAN 00-55; DEFSTAN 00-56, RTCA DO-178B]. The issue of how much credit can be taken for conforming to high standards of development practice is controversial.

5. STATE-BASED POLICIES

A number of challenges in the formalisation of safety policies have been outlined in section 4. The most conspicuous way in which our model of action policies fails is its inability to provide an implementation of state-based policies, that is those policies whose goal is to achieve or avoid a state, rather than to carry out an action. We outline here a possible approach to this problem, which is under discussion, and which builds upon the existing model.

Since our model describes low-level policy actions, it is clear that state-based policies have to be regarded as higher-level policies, which need to be refined into a set of actions, e.g. the refinement of P5 (state-based) into P6 and P7 (action-based). The question is, whether this can be done in a manner which is susceptible of automation.

An analogy with database systems may be useful[5]. At the end of any transaction which updates a database, there is the possibility of checking a database integrity predicate before committing the transaction. If the predicate is violated, then a recovery action may be defined which returns the database to a state of integrity. Although a standard method for this is to roll-back the database to its state before the transaction was started, this may not always be desirable, e.g. because of a domino effect on nested transactions. So the recovery action may be one for forward recovery, i.e. carrying out a compensating transaction which does not roll-back the state but restores its integrity[6].

There is a clear analogy between entering an abnormal state in safety-critical systems and violating an integrity predicate in a database. There is usually no analogy to committing to a transaction, but the system integrity predicate could be sampled at a suitable rate. Then a pre-defined refinement of the policy hierarchy could be invoked to activate the recovery action policies which are appropriate when a particular state is reached. There is, of course, no possibility of rolling back real time, so the recovery has to be forwards.

---

[5] Suggested by Dirk Jonscher (personal communication).

[6] Note that this is an application of classical fault-tolerance concepts.

We therefore envisage an enhanced Situation Monitor which, as well as reacting to events as in the present model, also proactively samples state continually. There are two kinds of data which it must use: a) state-base policies which inform it of the state variables which it must monitor, e.g. policy P5; b) fragments of policy hierarchy which refine from the state-based policies to their corresponding low-level action policies, e.g. P5 refines into P6 and P7. Having identified the appropriate low-level policies, it triggers off the actions.

The additional functions, above our current model, given to the Situation Monitor, include:

- Regular sampling of state variables;
- Interpretation of policy hierarchy fragments;
- Alteration of the authorisation state of the system in cases, such as P7, where a new or altered authority policy is required.

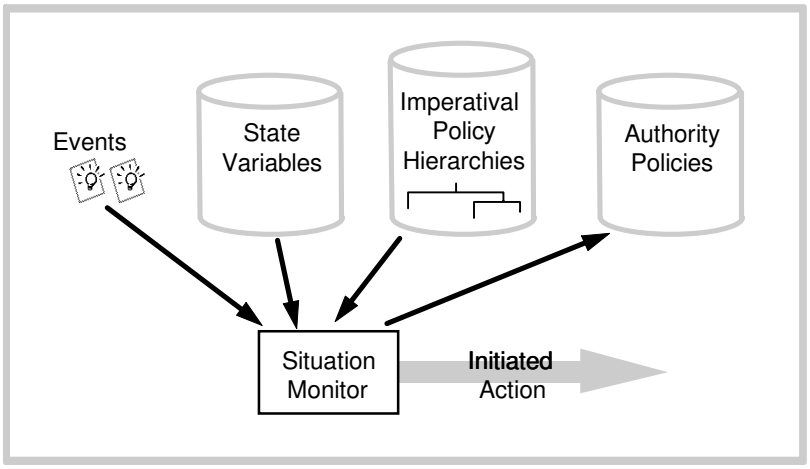

Figure 4 Situation Monitor with Extended Functions

The extended architecture of the Situation Monitor is illustrated in Figure 4. Implicitly, many existing safety-critical systems work in this way. The value of modelling policies is that it makes key safety issues explicit, thus giving the ability to analyse and reason about them. This is particularly important because policies can become obscure in complex systems, and low-level decisions, e.g. changing the order in which two actions are scheduled, can have a major, but difficult to discern, impact on high-level policies. However, it remains to be seen whether or not direct mechanisation of this model is feasible in a real-time situation.

6. CONCLUSIONS

This summary does not claim to have covered all aspects of safety policy, but has done enough to show that the policy model originally developed for distributed system management has some value as a starting point when considering safety policies, but requires considerable extension.

On the other hand, a number of aspects of safety policies go beyond the current model

- Safety policies are typically about states, not actions;
- The distinction between safe, abnormal and uncontrollable states has to be recognised;

- The clear-cut distinction between two modes of policy – imperatival and authority– does not always apply;
- Development policies have a contribution to make to safety.

In section 5 we outlined a possible approach to the realisation of state-based policies. Work on this, and other aspects of the formal representation of safety policies, is in its formative stages, but is seen as an important component in our understanding of how to build safety-critical systems.

REFERENCES

Anderson, J.P. (1972). Computer Security Technology Planning Study, ESD-TR-73-51, Vol. I, AD-758 206, ESD/AFSC, Hanscom AFB, Bedford, MA, October 1972.

ARP 926. *Design Analysis Procedures for Failure Modes, Effects and Criticality Analysis (FMECA)*. Aerospace Recommended Practice, Society of Automotive Engineers, Detroit, USA. 1967.

DEFSTAN 00-55 (1991). *The Procurement of Safety Critical Software in Defence Equipment: Part 1 Requirements & Part 2: Guidance*. UK Ministry of Defence, April 1991.

DEFSTAN 00-56 (1991). *Hazard Analysis and Safety Classification of the Computer and Programmable Elements of Defence Equipment*. UK Ministry of Defence, April 1991.

Kletz T. (1983). *Hazop and Hazan: Identifying and Assessing Process Industry Hazards*. Institute of Chemical Engineers, Rugby CV21 3HQ, UK.

Leveson, N.G. (1991). Software Safety in Embedded Computer Systems. *Communications of the ACM* **34**(2): 34-46.

Littlewood, B. & Strigini, L. (1993). Validation of ultrahigh dependability for software-based systems. *Communications of the ACM* **36**(11): 69-80.

McDermid, J.A., Wilson, S.P. & Fenelon, P. (1993). *ASAM II Concepts: Principles Underlying the Construction of Safety Cases Using SAM*. Dept of Computer Science, University of York, ASAMII-REQ-1, November 1993.

Moffett, J.D. & Sloman, M. S. (1993). Policy Hierarchies for Distributed Systems Management. *IEEE Journal on Selected Areas in Communication* **11**(9): 1404-1414.

Moffett, J.D., Jonscher, D. & McDermid, J.A. (1994). *The Policy Obstacle Course: A Framework for Policies Embedded within Distributed Computer Systems*. Dept of Computer Science, University of York, SCHEMA/York/93/1, revised 15 July 1994.

RTCA DO-178B (1993). *Software Considerations in Airborne Systems and Equipment Certification*. Radio Technical Commission for Aeronautics, Washington, D.C. USA. Also issued as EUROCAE ED-12B by EUROCAE Secretariat General, Paris, France

Rushby, J. (1987). Kernels for Safety? *Safe and Secure Computing Systems*. Blackwell Scientific. 310-320.

Vesely, W.E. (1981). *Fault Tree Handbook*. Washington DC, USA, US Nuclear Regulatory Commission.