A Generic Approach to Policy Description in System Management

Cheh Goh
System Management Department
HP Laboratories Bristol
HPL-97-82
July, 1997

system management, policy, constraint-based, policy enforcement

The versatility of policy for use in system management depends a lot on the descriptive power of the policy language. An examination of existing approaches reveals a way of taking a generic constraint-based approach to describing policies. This unified approach takes the view that policies can be completely described according to the constraints of the system, and the categorisation and attributes of policy that previous works highlighted can be understood as convenient interfaces for the development and management of policy. We also introduce the concept of active and passive policy enforcement and explain how this classification helps us understand the underlying constraint-based approach.

A Generic Approach to Policy Description in System Management

Cheh Goh, cng@hpl.hp.com

System Management Department,
Hewlett Packard Laboratories,
Stoke Gifford, Bristol BS12 6QZ
United Kingdom

Abstract: The versatility of policy for use in system management depends a lot on the descriptive power of the policy language. An examination of existing approaches reveals a way of taking a generic constraint-based approach to describing policies. This unified approach takes the view that policies can be completely described according to the constraints of the system, and the categorisation and attributes of policy that previous works highlighted can be understood as convenient interfaces for the development and management of policy. We also introduce the concept of active and passive policy enforcement and explain how this classification helps us understand the underlying constraint-based approach.

Keywords: system management, constraint-based, policy, policy enforcement.

1. Introduction

In the recent development of information system management, use of policy is on the increase. [Sloman94, Neumair96, Guerroudji96]. This is mainly due to the recognition that in the real world, policies are used as an effective way to describe how a system should operate. In some areas such as system security management, discussion is carried out almost inevitably in terms of policies [Blaze96, Grimm96, Kühnhause95, Yialelis96]. In many of these papers, different notations are used to describe policies. The level at which these notations are used is often close to low level implementation. Instead of concentrating at that level of describing policy, we attempt to take a broader view and find a general approach which can, not only help the organisation manager according to her perspective where the policy is very unspecific, but also deal with the problem of policy interpretation and refinement, and assist the actual policy implementation at a low level.

In this paper, we will first clarify what we mean by policy in section 2 and introduce a new concept about its implementation in section 3. The relationship of policy with *overall objective* and *implementable* is explored in section 4 followed by a reflection of the motivation behind this new approach in view of previous works on policy categorisation. A discussion of the versatility of this approach is given in section 6, followed by the conclusion.

For the discussion in the rest of this paper, the definitions and terminology used in [Sloman94] are adopted. The object on which policies are applied is called the *subject,* and the objects affected by the activities related to the policy are known as the *targets*. We make the assumption that all objects will serve some purpose in a system at certain point of the policy evolution. For referencing purposes, we adopt as far as possible the object paradigm used in the DMTF Common Information Model (CIM) [DMTF-CIM]. Hence, an object is anything that has a definition and description of itself, and must possess *properties* which have primitive values of type such as integer, string and so on.

2. What is a policy

There are various ways of understanding what policy is. In an enterprise, there is always an *overall objective* for which a system exists, and is usually accompanied by a *policy*. We believe that a policy is about the constraints and preferences on the state, or on the state transition, of a system. It is a guide on the way to achieving the overall objective which itself is also represented by a desirable system state. For example, the overall objective of a man-

ager for a system could be "to ensure 95% system up time for the users." The policy for achieving this overall objective refers to the way this should be achieved, or the preferred approach to achieving it. A policy of preference, or imperative limitations, can be expressed as a set of constraints upon the usage of the limited resources available to the manager to accomplish her task. Two examples of possible policy are: "use fault-tolerant machines and hot standbys" and "system must be secure against attacks in the form of denial of service". Similarly, a policy for a system security refers to how the system should be made secure in its operation, and a policy for system usage refers to how the system should be used. All the "hows" can, again, be described as different sets of constraints. Later on in section 4, we will discuss more deeply the relationship between objective, policy and implementable.

When it comes to the implementation of a policy, it is a matter of making the system satisfy the constraints that describe the policy. This often involves a *state transition plan* to arrive at the final desirable state from an initial state. Note that a *state transition* can usually be regarded as synonymous to an *action* as is often referred to in the literature. We use the former to emphasis our view that the full description of an action must include also side effects, which is best described as part of the state transition of the system. Hence, it is more direct to consider policy in terms of constraints and states.

3. Policy Implementation

In the discussion of policy in system management, little attention has been given to policy enforcement in the literature so far. We want to introduce the concept that there are two different ways of implementing a policy: through passive enforcement and through active enforcement. Passive policy enforcement is carried out within a permissible system state space, in which checks will be carried out against the constraints under which state transition in the system may take place. In this case a policy represents a set of constraints that the system must keep consistent during its transitions. Active enforcement of a policy, on the other hand, implies that when a system arrives at a state S_i which violates the necessary constraints, a transition will take place to bring the system to S_{i+1}, which is considered to be desirable according to the policy. (See Figure 1.) Here a policy represents a set of constraints with which the system state must be made consistent.

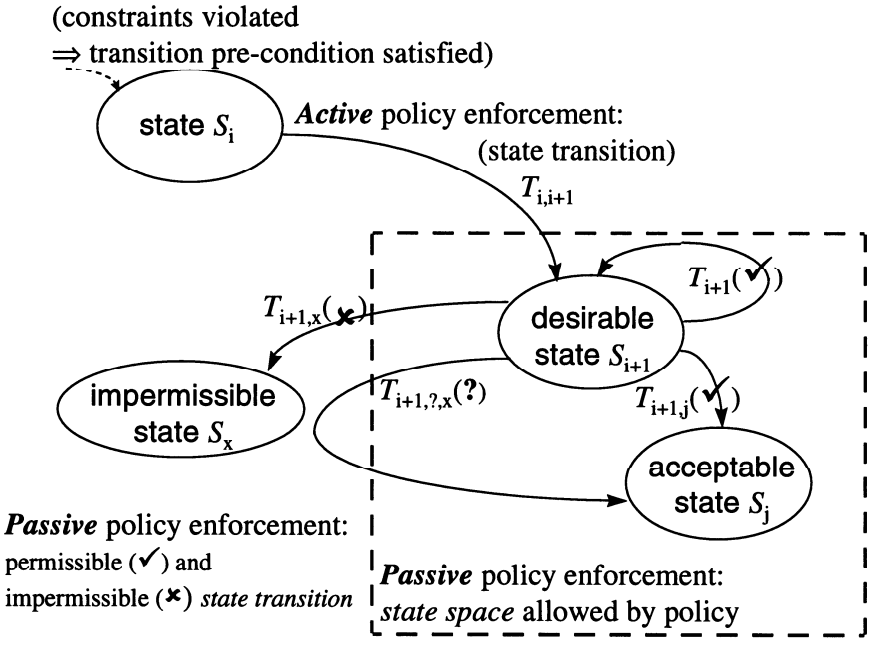

Figure 1 Active and passive policy enforcement in a system

As is widely recognised, a policy such as the example given in section 2 "system must be secure against denial of service attack" is not machine implementable. In order for this policy to be implemented, many iterations of refinement are needed to transformed it into configuration information in some hardware device or some soft-

ware application. Eventually, when the refinement process cannot go any further, the original policy has indeed been transformed into a set of constraints either for passive enforcement or for the system to transition into.

3.1 Passive policy enforcement

A policy which is expressed purely as a set of constraints without any transition plan can only be enforced passively. Using the example in Figure 1, we can identify the following applications of constraints in the system:

- A constraint set which describes the *permissible state space* for the system. The system may get out of the state space as a result of faults or external perturbation, but when the system policy has been enforced, it will not be allowed to transit out of that space legally.

  An example of this permissible state space is "passwords that reduce dictionary attack risk", which is a sequence of characters that do not form words found in dictionaries. A change of password from, say, "*pBSm5+*" to "*apple*" will be unacceptable, and is equivalent to the transition, such as $T_{i+1,x}$ from S_{i+1} to S_x in Figure 1. Note that a transition arrow may represent one or more steps (intermediate states).

- A constraint set which describes the *permissible state transition* relative to the permissible state space. Using Figure 1 as example, transition T_{i+1} from S_{i+1} back to S_{i+1} and transition $T_{i+1,j}$ from S_{i+1} to S_j both satisfy the constraints. Transition $T_{i+1,?,x}$ that leaves the permissible state space from S_{i+1} before returning to S_j and is depicted with a question mark, however, may or may not be allowed, depending on the actual policy. A constraint set which allows a transition that has some intermediate states outside the permissible state space is enforced by ensuring that all constraints are satisfied, not which state to transit to.

  An example of S_{i+1} is "user A has no access to file F". Suppose S_j is "user A has a copy of file F", and the permissible state space constraint is "no user can have root privileges". In order to go from S_{i+1} to S_j a possible state transition plan will be "change file mode for F to world readable", "user A copies file", "change file mode for F back to what it was". Another transition plan is "give user A root privileges", "user A copies file", "remove root privileges from user A". The second state transition plan temporarily violates the constraints space, but depending on the policy, it might be a legal move. As can be seen, policy enforcement here does not initiate any state transition; it only restricts state transition when one is taking place.

3.2 Active policy enforcement

Let us suppose that we have a policy which at the end of a series of refinement, yields a set of constraints with which the system state should be made consistent actively through machine implementation or by human action. At this point, a description of *implementables* will have emerged, in that the policy refinement process leads to the necessary context, or local initial-condition, for a state transition to take place, the completion of which will achieve the implementable that meets part or all of the requirements of the policy. A transition plan can now be determined to arrive at the desirable state S described by each and every one of the implementables.

The active enforcement of a policy is triggered when an event perturbs the system state, whereby the policy of maintaining a desirable system state leads to the activation of the necessary transition. This is abstractly represented as the transition from S_i to S_{i+1} in Figure 1.

Continuing with the previous security policy example, denial of service attack can take the form of a destroyed *passwd* file in the system. Since only a user with the root privilege may destroy the *passwd* file by corrupting its contents for example, the policy can be refined to include "users who are not the administrator may not have root privileges." Now, suppose S_i is "user A has root privileges", then the enforcement of the policy means that the system will actively take steps to move the system state to S_{i+1} where root privileges are removed if user A is not the administrator. The event that triggers the enforcement could be "the *passwd* file of the system is touched" and the enforcement is to scan *passwd* to remove root privilege from all ordinary users.

Naturally, during active policy enforcement, passive policy enforcement is implicit for the state transition process, even though the constraints may come from different policy sources within the same system. Conversely, during passive policy enforcement, the system state may be found, through monitoring for example, to correspond to a specific pre-condition, causing active policy enforcement to take place.

4. Objective, policy and implementables

In the previous sections, section 2 and section 3.2, attention has been given to distinguishing overall objective and implementable from policy. The operation of a system within a given context has an overall objective which is a description of what is to be achieved at a *high level.* It is often accompanied by policies, which are invariably described in natural language in a vague way at this stage, to help with its achievement. When a system has to move to a specific state as a result of active policy enforcement, it has to achieve an implementable at a *low level* within the same context. At this stage, the specific final state is fully described and often the transition plan to achieve the implementable is prescribed. Note that if an objective can be implemented by active policy enforcement directly, it can also be considered as an implementable at the same time. Clearly the achievement of all the implementables should collectively achieve the objective.

The transformation process from a high level policy to low level implementable through refinement is depicted in Figure 2. The process takes on a tree like structure. Starting from the top level policy $\mathcal{P}_0$, the interpretation and refinement process will take the policy to the leaf nodes. If a leaf node becomes an implementable that requires the necessary state transitions, such as $\eta_{\chi.1}$ in Figure 2 then this is the point where the active enforcement of the original policy is carried out. The state transition is accompanied by the constraints as described by the refined policy that immediately precedes the implementable. The constraints in fact describe the initial and final condition of that transition. If a leaf node is not accompanied by a transition plan, then the policy is to be enforced passively.

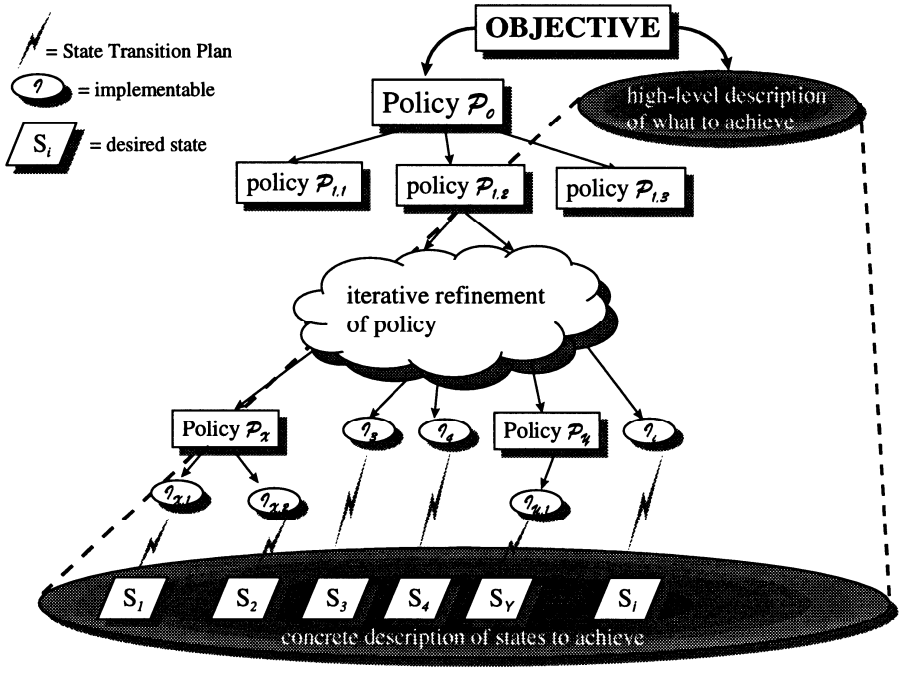

Figure 2 Objective, policy and implementable

Hence, where a policy can be actively enforced, the relationship between policy and implementable with respect to constraints is similar to that between wave and particle with respect to light. Before being observed, light behaves like wave and is probabilistic, but once light is observed it behaves like particle and is deterministic. Before implementables can be identified, a set of constraints is considered as a policy. Once implementables are available, the constraint set describes both the desirable final state of an implementable and the initial condition for the necessary state transition. They are interdependent and cannot be considered in isolation. If a policy can only be enforced passively, implying the absence of system state transition plan for the policy, there will be no transformation into implementables, and this wave-particle like dichotomy does not hold.

5. Policy Categorisation

If we examine the literature, policies have often been categorised into two types: obligation and authorisation [Sloman94, Marriott96], or imperatival and authority [Guerroudji95], or motivation and authorisation [Moffet-Sloman94, Koch95]. The key differentiation for these two broad categories is whether a policy entails some action, or concerns the granting of privileges only. This approach considers action as part of the policy description.

We take a slightly different view to this modality consideration and argue that policy can be generically treated as constraint description. Insofar as privilege granting, or authorisation, policy is concerned, obviously it is no more than a declaration of constraints in terms of the privileges and the objects to which the privileges are granted. In treating the category of motivation or obligation policies, we need to understand better what an action is.

As mentioned previously in section 2, we regard action as none other than the state transition to move from a state S_i to another state S_{i+1}. Such a transition has an initial condition C_i, so the policy in terms of constraints will form part of this initial condition. Motivational policy is actually the satisfying of all the constraints representing the initial condition, and the constraints governing the transition that leads to the final condition which satisfies the constrained state space. Of course, the concept of "obligation" as "something must happen" is weakened here, but logically this concept is irrelevant except in the realm of deontic logic which we do not include in our consideration. As pointed out in [Sloman94], the assumption of well behaved subjects does not always hold because no subject is fault free, and especially when we have to deal with human beings!

Given that in both cases the complete description of policy can be captured by specifying the relevant constraints, it is clear that the categorisation found in the literature mentioned above appears to be a convenient way to talk about policy without any real effect in the mechanism of dealing with policy. Despite its arbitrary nature, the concept of authorisation and motivation is useful to serve as a user friendly interface to ease policy writing and management.

A different categorisation of policies is given in [Guerroudji95]. Here, the concept of reactive policies which are invoked after the occurrence of an event, and permanent policies which are activated during the life cycle of the system for its maintenance are introduced. In relation to the actual enforcement of policies, this consideration appears to be just another view for policy management, and does not affect much else.

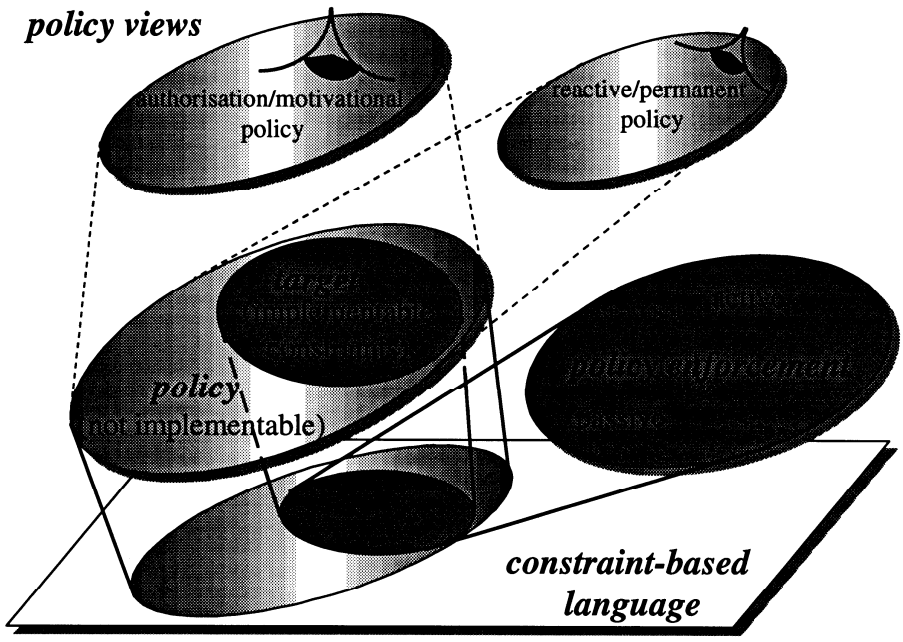

Figure 3 Relationships between the various aspects of policy

We want to argue, therefore, that a more generic way to deal with policies is to consider them purely from the point of view of *constraints*, principally at the level of underlying logical representation. The representation of constraint can be rule-based [Koch95] or otherwise [Marriott96], but the language must have the ability to fully capture all forms of constraints within the system being managed.. This separation of the actual mechanism that carries out the state transition, i.e., action, from the policy description means that at the implementation level, we are no longer tied up with the mechanism, allowing us to describe generic state transition behaviours and allow run time binding of behaviour to the attributes of the managed objects. In this way, we achieve the full potential of the flexibility in object technology. The relationships between the various aspects of policy is depicted according to the appropriate layers in Figure 3.

As can be seen in Figure 3, a constraint-based language to describe policy has the advantage of unifying the various aspects of policy. The view for policy writer and system operator/manager will be different, and there would be benefits to be gained by classification such as authorisation or motivational policy, or as reactive or permanent policy. Clearly further classification can be added on top of the layering shown in Figure 3. For example, authorisation policy itself may allow, at the user interface level, the differentiation of access control list, privilege granting list and so on. On the other hand, it is also possible to understand in theory the metamorphosis of policy into implementable with the same constraint-based representation. In policy implementation, if we have a reasoning engine which can work with the actual language that represents the constraints of the policy, then it can be used both during active and passive enforcement. Policy to be enforced passively requires managed objects to check with the system manager, but the same checking process can be made generic so that for policies which require active enforcement, identical approach may be used, hence achieving integration.

6. Discussion

When we regard policy as a description of constraints, these constraints must be applied to some objects known as the *subject* as mentioned in the introduction. For the purpose of discussion, we also define a policy $\mathcal{P}$ to be such that $\mathcal{P} = \mathcal{C}(\mathcal{S})$, the application of constraints represented by the constraint descriptor $\mathcal{C}$ to the subject $\mathcal{S}$.

6.1 Policy subject and target

Using the concept of *domain* as in [Twidle-Sloman94, Sloman94], the subject $\mathcal{S}$ is taken to be a collection of objects in the management domain to which the policy constraints will be applied. Notice that the notation used to describe policy does not explicitly indicate the domain and the targets. This is because the constraint descriptor $\mathcal{C}$ itself already includes domain either as part of the properties of the subject or as an associated object [DMTF-CIM] of the subject, and the object relationship in the association with the subject $\mathcal{S}$ will show who the targets are. It can be seen that the constraints descriptor $\mathcal{C}$ includes naturally policy scope expression [Sloman94] and the identification of the subject of the policy can be made precisely and straightforwardly.

6.2 Conflict detection

High level policies that are laid down in an organisation get implemented eventually after many iteration of refinement, potentially introducing, in the process, many opportunities for conflicts. It is well known that these problems exist because the description of policies is never complete due to the impossibility to capture all knowledge pertaining to any given subject, and also because the interpretation of policy by different parts of an organisation is a process that necessarily lacks rigour [Hinton94, Guerroudji95, Lupu97]. Policy conflict detection in system management is, therefore, a necessary aspect of using policy.

Consider two policies $\mathcal{P}_1$, $\mathcal{P}_2$ respectively represented as $\mathcal{C}_1(\mathcal{S}_c)$ and $\mathcal{C}_2(\mathcal{S}_c)$, where $\mathcal{S}_c$ is the common subject (shared set of object) for the policies, and the two constraint descriptors refer to two separate contexts. Let $s_1=\{\mathcal{S}_i\}\neq\phi$ be the set of states satisfying $\mathcal{C}_1(\mathcal{S}_c)$ and $s_2=\{\mathcal{S}_i\}\neq\phi$ be the set of states satisfying $\mathcal{C}_2(\mathcal{S}_c)$. The sets s_1 and s_2 describe the states that $\mathcal{S}_c$ may legally exist within the respective context. When the two contexts are considered together, a shared object may have properties that satisfy one set of constraints and yet represent a "threat" to another—failure to satisfy the other's constraints. When this happens in such a way that $s_1 \cap s_2 = \phi$, it means that it is impossible to find a state for subject $\mathcal{S}_c$ when both policies are enforced. A conflict between policy $\mathcal{P}_1$ and policy $\mathcal{P}_2$ is said to have occurred.

The failure to find a solution for constraint satisfaction problem (CSP) indicates that, if the constraint model is the working model of the system, then the actual state of the system must have departed from the working state, and fault is implied. Using failure in solving CSP as a way to carry out diagnosis of network and system problem is a technique that has recently become better explored [Sabin-96, Sabin-97]. In a similar way, the conflict of policies may be viewed as the failure to find a state that satisfies all the constraints declared in these policies. When policies are brought together, diagnosis techniques [Pell-95] can be applied to detect conflicts. The use of a constraint description language with the appropriate reasoning engine will potentially allow general detection of conflict among policies.

It must be pointed out, however, that with a given policy it is not always possible to find all the legal states, even though under certain condition the number of legal states may be guaranteed to be finite. Because of the lack of certainty in identifying all legal states, it is not always possible to conclusively prove that $s_1 \cap s_2 = \phi$, or otherwise. Hence, even with the use of a constraint-based approach to describe policy, conflicts cannot always be detected.

6.3 Role

The associations of an object and the related constraints together are able to specify the role [Sloman94] of that object. Different sets of associations and related constraints can be used to indicate the different roles for that object, and the relationship between these different sets, which are themselves objects, will govern the way an object is allowed, or otherwise, to take on one or more roles. The implementation of role in policy based management, therefore, inherently exists when the description language is fully constraint-based.

Associated with role is delegation, an important concept in management policy [Sloman94], and especially in security [Yialelis96]. From the point of view of constraint description, delegation is the establishment of the constraints to be transferred, and a relationship between the delegating object and the delegated object. For example, the role of a system administrator implies the granting of root privileges. Delegation of "the administrator role" from the normal administrator to a user who deputises for her means that the constraints associated with the role, "possession of root privilege", is passed on to the user for that period of delegation.

7. Conclusion and future work

Many low level notations have been used in dealing with policies, but they often lack the sort of generality that helps to specify policies in an integrated manner. We have suggested the use of constraint description language as the basis, and shown how policy-based management can be based on constraints. Constraints form the unifying factor for describing policies, and because of its versatility, can capture many important concepts in the discussion of policy. We have shown in section 6 some of these important aspects and discussed ways to manage them.

We have also clarified the conceptual subtlety of constraints in relation to policy and policy implementation, and established the relationship between objective, policy and implementable. In addition, we proposed that the typing of policy lies in distinguishing whether policies are enforced passively or actively, in contrast to policy classification found in the literature with the suggestion that they served largely as a way to provide different views of the same underlying constraint-based nature of policy, and that such classification is most useful in helping the system administrator to set up policies and manage them.

We have discussed many concepts about policy-based system management in this paper and have suggested, in a limited way, possible implementation approaches. The biggest challenge in future work will be in terms of policy enforcement in actual systems. We intend to use some of the technologies, including specification language, mentioned in [Pell-95] as a starting point to bring policy-based management into the real world as concrete implementation. Further on, research is required to enable the aspects of policy discussed in section 6 so that they too can be realised as part of system management.

8. References

[Blaze96] Blaze, M., Feigenbaum, J., Lacy, J., *Decentralized Trust Management*, Proceeding. IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 1996, pp 164-173

[DMTF-CIM] Desktop Management Task Force, *Common Information Model (CIM) Version 1.0 (Draft)*, December 18, 1996

[Grimm96] Grimm, R., Hetschold, T., *Security policies in OSI-management experiences from the DeTeBerkom project BMSec*, Computer Networks and ISDN Systems vol. 28, 1996, pp499-511

[Guerroudji95] Guerroudji, N., *Logical Model for Administration Policy Specification*, Proceeding. ACM 33rd Annual Southeast conf., Clemson, SC, USA, March 1995, pp227-232.

[Guerroudji96] Guerroudji, N., *A Management Policies Framework*, Proceeding IEEE 2nd International Workshop on Systems Management, Toronto, Canada, June 1996, pp 40-46

[Hinton94] Hinton, H. M., Lee, E. S., *The Compatibility of Policies*, Proceeding 2nd ACM Conference on Computer and Communications Security Nov 1994, Fairfax, Virginia, USA, pp258-269

[Koch95] Koch, T., Kra"mer, B., Rohde, G., *On a Rule Based Management Architecture*, Proceeding IEEE 2nd International Workshop on Services in Distributed and Networked Environments, Whistler, BC, Canada, June 1995, pp68-75.

[Kühnhause95] Kühnhause, W. E., *A Paradigm for User-Defined Security Policies*, Proceeding 14th Symposium on Reliable Distributed Systems, Bad Neuenahr, Germany, September 1995, pp135-144

[Lupu97] Lupu, E., Sloman, M., *Conflict Analysis for Management Policies*, IFIP/IEEE Integrated Management Conference, San Diego, CA, USA, May 1997

[Marriott96] Marriott, D., Sloman, M., *Management Policy Service for Distributed Systems*, IEEE 3rd International Workshop on Services in Distributed and Networked Environments (SDNE'96), Macau, June 1996.

[Moffet-Sloman94] Moffect, J.D., *Network and Distributed Systems Management*, editor Sloman, Addison-Wesley, 1994, chapter 17.

[Neumair96] Neumair, B.; Wies, R.; *Case study: applying management policies to manage distributed queuing systems*, Distributed System Engineering, 1996, pp96-103

[Pell-95] Pell, A.R., Eshgi, K., Moreau, J-J, Towers, S.T., Integrated Network Management IV, editors Sethi and Raynaud, Chapman & Hall, London 1995, pp95-105.

[Sabin-96] Sabin, M., Freuder, E.C., *Automated construction of Constraint-Based Diagnostician*, 7th International Workshop on Principles of Diagnosis, Val Morin, Canada, October 1996.

[Sabin-97] Sabin, M., Russell, R.D., Freuder, E.C., *Generating Diagnostic Tools for Network Fault Management*, 5th IFIP/IEEE International Symposium on Integrated Network Management, Dan Diego, CA, USA, May 1997.

[Sloman94] Sloman, M., *Policy Driven Management for Distributed Systems*, Journal of Network and Systems Management, vol. 2 part 4, 1994, pp333-60.

[Twidle-Sloman94] Twidle K., *Network and Distributed Systems Management*, editor Sloman, Addison-Wesley, 1994, chapter 16.

[Yialelis96] Yialelis, N., Sloman, M., *A Security Framework Supporting Domain Based Access Control in Distributed Systems*, ISOC Symposium on Network and Distributed Systems Security (SNDSS96), San Diego, CA, USA, February 1996.