5

Concepts and Application of Policy-Based Management

B. Alpers, H. Plansky
Siemens AG
Otto-Hahn-Ring 6, 81739 München, Germany
{Burkhard.Alpers,Herbert.Plansky}@zfe.siemens.de

Keywords
Domain, policy, policy hierarchy, policy formalisation

Abstract
Due to downsising, deregulation and tremendous growth, computing and telecommunications systems have become heterogeneous and complex environments with multiple players involved. For managing such systems more powerful methods than those used for network element management are needed. It must be possible to structure and partition management according to responsibilities, to provide managers with higher-level abstractions and to enable them to flexibly adapt the way of management to their specific needs. To fulfil these requirements, we introduce *domain-based management policies*. Policies allow the specification of management intention on different levels of abstraction. We discuss policy classification, formalisation and hierarchies. Then we present an architecture for policy enforcement services. Finally, we outline two application scenarios in the areas of distributed systems and telecommunication.

1. Introduction

Computing and telecommunication systems and services are of vital importance for enterprises as well as for organisations. Two current trends are leading to a rapid change in the structure of these systems and services: "downsising" and "deregulation". Downsising means the substitution of mainframe-systems by smaller networked systems leading to more flexible and extensible systems which form a complex and heterogeneous environment. Deregulation in the area of telecommunications results in a variety of new services, like Virtual Private Network (VPN) offered by independent service providers.
Management in such a heterogeneous, complex and dynamic multi-manager scenario requires efficient management methods which offer more functionality than the traditional network element-oriented management:

- It must be possible to *structure and partition* management responsibilities amongst several managers with different roles.
- Management systems must allow an *abstraction* of management to prevent the managers from becoming flooded with low-level network alarms and details.

A. S. Sethi et al. (eds.), *Integrated Network Management IV*

- Management systems should be *flexible* and reconfigurable to adapt to the introduction of new services, systems, applications or users.
- Management tasks should be largely *automated.*

The *domain-based policy concept* we introduce in this paper serves to fulfil these requirements. Policies enable the specification of management roles on a higher level of abstraction and hence deliver a powerful method for structuring management of large systems. Management policies need to be formalised without building them inflexibly into manager components .

In chapter 2 of this paper we elaborate the policy concept. After defining and classifying policies we discuss formalisation and hierarchies. Then, we relate our concepts to other work in this area. Chapter 3 deals with realisation aspects. We present an implementation architecture and describe an engineering model for implementing policies. In chapter 4 we present two application scenarios in the areas of distributed systems and telecommunication. Chapter 5 contains conclusions and describes future work.

The general domains and policy concepts described in this paper, the implementation architecture and the application to X.400 management are based on results of the IDSM (Integrated Distributed Systems Management, 6311) and SysMan (7026) ESPRIT projects.

2. Concepts of Policy-based Management

2.1 Principles and Classification of Policies

Management of networks, systems, and applications involves monitoring the activities and states of these systems, making management decisions, and performing management control actions. This management process, particularly the process of decision making is guided by *management policies*. Policies describe management activities rangeing from lower-level management operations to higher level objectives which are determined by economical or social requirements. In our approach we will use the concept of management policies to specify and formalise the *semantics* of management, whereas the state-of-the-art management techniques concentrate on the *syntax* of management, e.g. management protocols and information models, etc.

We define policy in general as *"information which influences the behaviour of managers and managed objects"*. A policy determines the management objectives, the roles, and the responsibilities between a subject set containing managers (human managers or applications) and a target set containing managed objects. Policy specifications can be interpreted by human managers. In order to automatically enforce management policies, they must be formalised and represented in the management system. Then, manager applications can be made responsible for the enforcement of policies, i.e. they have to interpret policies and to align their behaviour with the policies. This separation of policies and manager objects enables the dynamic change of policies without changing the managers.

The variety of management tasks results in a multitude of policies. There are many criteria for classifying policies (functional area, application area etc., see e.g. [Wies94]). The criteria we present in the sequel are particularly important for identifying generic classes which are interesting for many application areas. These are therefore very promising candidates for further formalisation, as discussed in the next section:

Criterion: Relationship between managers and managed objects

Managers have obligations and rights with respect to managed objects:

Authorisation policies define what management activities a manager is *allowed* to do in terms of the operations he is authorised to perform on a set of managed objects. An authorisation policy may be positive (permitting) or negative (forbidding).

Obligation policies specify what management activities a manager *must* or must not do.

Criterion: Influence on the managed system
Policy can describe the activity of managers (*manager action policies*), or the desired behaviour of the influenced system resulting in *state achievement policies* or *state change restriction policies*:

- manager *action* policy: this is a (conditioned) sequence of actions to be performed by managers on managed objects;
- *state achievement* policy: this specifies the state of the system to be achieved in terms of existence of objects, attribute value ranges, relations; this could also include the optimisation of certain attributes to reach an optimal state;
- *state change restriction* policy: description of allowed or undesired state changes: this specifies state changes of the managed system which are allowed or forbidden.

Criterion: Abstraction-level of the policy
The concept of policies supports the abstraction of management. Policies with lower-level of abstraction are directly linked to the infrastructure of the managed system, whereas higher-level policies need further interpretation to lead to concrete management actions. Abstract policies can be used to form policy hierarchies, where an abstract policy is mapped on other less abstract policies (see 2.3).

2.2 Policy Formalisation

For analysing policies and automating their enforcement, policies must be formally represented. The complexity and variety of management policies makes it impossible to find just *one* "policy description language" covering all aspects of policy. In the IDSM project a generic policy object class is formalised which serves as a reference for defining more specific policy object classes. The generic policy object class describes a relation between manager objects and managed objects.
This object class was specified according to ISO's "Guidelines for the Definition of Managed Objects" (GDMO, see [GDMO92]) in the following way:

```
idsmPolicy    MANAGED OBJECT CLASS
              DERIVED FROM idsmManagedObject;
              CHARACTERISED BY
                    mandatoryPolicyPackage    PACKAGE
                    BEHAVIOUR idsmPolicyMandatoryAttributes BEHAVIOUR DEFINED AS
                    "This package contains the set of attributes which must be
                    specified for any policy object class";;
                    ATTRIBUTES
                    adminPolicyState    GET-REPLACE,
                    operationalState    GET,
                    subject             GET-REPLACE,
                    target              GET-REPLACE;;
              CONDITIONAL PACKAGES
                    globalConstraintsPackage  PRESENT IF ...
                    propagationModePackage    PRESENT IF ...
                    companionPoliciesPackage  PRESENT IF ...
```

In addition to the attributes for state management the `mandatoryPolicyPackage` contains attributes to determine the subject and the target of management. For these attributes we use domains which are arbitrary groupings of objects (see [DSOM94]). This way the subject and target scope can be easily specified without defining policy for each single object. The optional `globalConstraintsPackage` allows to impose restrictions on policies concerning the location of manager and managed objects or the time when the policy is valid. The `propagationModePackage` can be used to specify that from the domains used for scoping subject and target not only the direct members are considered but also the members of sub-domains (i.e. domains contained as members in other domains). In the `companionPolicyPackage` one can list associated policies which should be activated together with this policy (see next section on refinement).

This generic class serves as a starting point: Subclasses must be built in order to specify exactly what the managers grouped as "subject" should or are allowed to do with the managed objects grouped in the "target". This must be expressed in additional specific attributes. In the IDSM project we formalised authorisation policies and reporting policies (for exact specifications see [IDSM-D7]). The `idsmAuthorisationPolicy` class has additionally a package which allows the specification of permitted or forbidden management operations (Get, Set, Create, Delete, Action) including parameter values. As a specific obligation policy we defined the `idsmReportingPolicy` class which has a package where the events to be reported and the destinations can be specified.

Whereas the formalisation of authorisation policies is relatively straight forward, for obligation policies this is far more complex since what a manager is responsible for can vary considerably. This is usually fixed in job descriptions or functional specifications of automated managers. In order to abstract from concrete tasks one has to look for generic patterns in such descriptions. As a starting point we use the policy classes we identified in the last section according to the classification criterion "influence on the system". In order to formalise these classes one needs an underlying model of the system to be managed. ISO provides a description language for such a model with its GDMO. Thus, actions, states and - to some extent - state changes can be formalised wrt. such a model. The more semantics such a model covers the more fine-grained can the description of management obligation be: If the model covers only a few control variables, a management specification can only be very coarse. In other words: The management description can be only as detailed as the object model it relates to. Having a model, the following ways of formalising the "influence classes" are conceivable:

- *manager actions*: a scripting language could be used to formalise such (potentially conditioned) sequences of action. The examples in [Wies94] and [Moffett93] can be considered as written in a pseudo scripting language.
- *system state*: starting point for a state description language are to be found in the literature on monitoring distributed systems (see [Mansouri93]). In the DOMAINS project, language constructs for specifying management goals were investigated (see [Becker94]). These goals include the description of desired states.
- *state changes*: Note first that for capturing the dynamics of the system to be managed a very rich model is necessary. Since in GDMO the dynamic behaviour cannot be specified formally (except for notifications), this description language is not powerful enough. In [Bean93] a petri-net model is suggested for modelling the dynamics. Control can then be specified by disabling controllable transitions. If such a model exists, an exact specification of the desired behaviour is possible.

There is obviously a trade-off between model complexity and the power of model-based management formalisation. Therefore, there will likely be different models and hence multiple model-dependent management formalisations.

2.3 Policy Hierarchy and Refinement

A management policy is often linked with other policies. This relationships can be peer-to-peer interactions with possible conflict situations or hierarchical dependencies. A policy hierarchy promises additional abstraction and automation of management activities: Ideally, the operator defines and changes higher-level policies whereby these changes propagate automatically to lower-level policies. This automation can only be reached, if the relations and dependencies between policies are formalised. An example for management using policy hierarchies can be found in the telecommunication environment. In TMN (Telecommunication Management Network) management consists of several layers: the lowest level, the network element management layer, deals with network elements, the middle layer deals with network management, the upper layer with service management. In the service layer an obligation

policy deals with end-to-end communication links and quality-of-service (QoS) parameters and could be: "The bandwidth of communication links should not be more than 10% under the prescribed value". The target domain of this policy consists of the communication link managed objects; the subject domain comprises the service managers. This policy would be translated into lower-level policies on the network and network element management layer.
Policy refinement can be performed in several ways (see also [Moffett93]):

- The obligation of a policy can be refined by mapping it onto sub-policies; e.g. in Fig. 2.1., the availability policy is translated into test and monitoring policies. The task of finding appropriate sub-policies requires knowledge about the system, the configuration, etc. and cannot be automated in general.
- Delegation of management tasks: The target set, i.e. the objects to which the policy is applied, can be split into targets of other policies with the same obligation, but with different subject sets.

Since a policy consists of policy attributes, their values must also be translated to attribute values of lower level policies. Figure 2.1 shows an example where a certain availability of the system should be reached. The availability is guaranteed by an availability policy which is translated to a *test policy* and a *monitoring policy* on the system resources (here: storage disks). The several degrees of availability result in several test and monitoring policy attribute values.

availability (av) policy	90%< av <92% of scheduled operating time	92% < av < 94% of scheduled operating time	94% < av < 96% of scheduled operating time
test policy	every 3 h short test of all important components, every 24 h extended test	every 30 min. short test of important components, every 6 h extended tests of all components	every 5 min. short test of all important components, every 30 min. extended tests of all components
monitoring policy	alarm if disks are used to 90 %	alarm if disks are used to 80 %	alarm if disks are used to 70 %

Figure 2.1 Example for the translation of policy attributes

As we have seen in section 2.2, policies are formalised by policy objects and must be interpreted by managers. A policy hierarchy can be used to check the adherence of managers to policies. A way to detect policy violation is objective-driven monitoring [Mazumdar91], where monitoring and reporting policies are derived from higher-level policy obligations.

2.4 Comparison with other work

The concepts of domains and policies have recently received considerable attention in research and standardisation.

ISO

ISO considers these concepts in two drafts: The System Management Overview (Proposed Draft Amendment [IS0-10040]) defines the basic concepts, a specific Systems Management Function (Committee Draft 10164-19: [ISO-10164-19]) describes objects, relations and services for the management of domains and policies. In ISO a domain is a group of managed objects, which is determined by a grouping criterion, whereas in our approach a domain is an explicit enumeration of domain members. These two approaches can be combined allowing two kinds of domains, one with explicit and one with implicit members. The user can decide which membership is appropriate for his application. In ISO policies are defined in a more

narrow sense than in our approach. Policies are defined as a set of rules, which restrict the behaviour of managed objects. A system management rule is one of the following:

- a constraint on the allowed operations, including permissible parameters and their values,
- an assertion on the allowed attribute values,
- an assertion on the emissions of notifications, including permissible parameters and their values,
- an assertion on the replies of operations, including permissible parameters and their values.

Our notion of policy is sufficiently broad to include the ISO approach as a special policy class.

X/Open

Work on policy specification and support is also in progress in the X/Open consortium. A preliminary specification for support of policies in a CORBA environment (see [CORBA91]) is expected to appear in 1995. In the current working document policies seem to be restricted to the definition of default and allowed values of managed objects. This type of policies is a special case of our general policy definition. Managed objects can be grouped in policy regions, which are similar to domains. In contrary to our management domains, policy regions are not allowed to overlap. This restriction leads to a less flexible concept, because in many cases policy domains will have to overlap.

Research

In the research area several approaches deal with policy hierarchies. [Calo93] develops concepts and supporting tools for formalisation and enforcement of policies. A policy architecture is presented which is based on policy hierarchies. Several policy layers are defined: (1)societal policies, (2)directional policies, (3)organisational policies, (4)functional policies, (5)process policies, and (6)procedural policies. Layers (1)-(3) are abstract, whereas layers (4)-(6) are subject to formalisation and automated interpretation.

[Moffett93] investigates policy hierarchies and the formalisation of policy refinement. The concept of hierarchy is the prerequisite for the refinement of policies, where policies are transformed to lower level policies and actions. Policy hierarchy concepts are supported in our approach by policy classes and a policy refinement service (see 2.3 and 3.1).

The concepts in this paper are partially based on results of the DOMAINS ([Alpers93], [Becker94]) and Domino [Domino92] projects. DOMAINS defines domains as areas of authorisation including a manager whom can be given goals. These goals are roughly comparable to our state-based policies. From Domino we adopted the concept of domains as object groups and the representation of policies as relationships between subject and target sets and the class of manager action policies.

3. Realisation of Policy Concepts

3.1 Implementation Architecture

In this section we describe the services for supporting the domain-based policy concepts and an implementation architecture which is the basis for our implementation in the IDSM project [IDSM-D6]. The centrepiece of our implementation environment is a management platform which gives access to managed objects offered by OSI agents or by management applications acting as agents residing on the same or on a remote platform. Figure 3.1 shows the architecture with supporting services. These services are offered as Service Managed Objects to make them accessible for applications or other services. The Domain Service and Domain Service User Interface contained in the figure support the flexible definition of management domains. These components are described in [DSOM94].

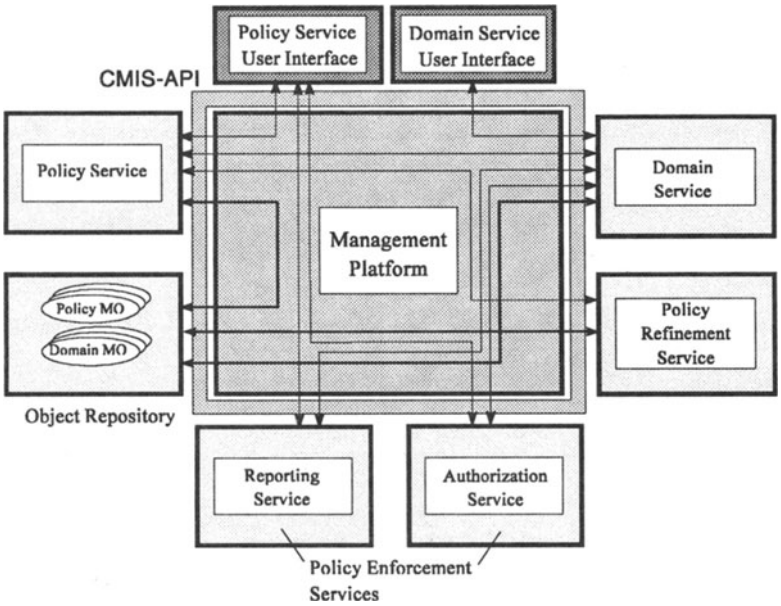

Figure 3.1: Platform-based Implementation Architecture

In this approach the policy concept is supported by a *Policy Service*, *Policy Service User-Interface*, *Policy Enforcement Services,* and *Policy Refinement Services*.

- **Policy Service (PS):** The PS supports storage, retrieval and analysis of policy objects belonging to certain policy object classes. It allows to create, delete, query and modify instances of these classes. It is important to note that the Policy Service *does not enforce policy*. Dedicated enforcement services for certain classes of policies interpret instances and enforce them using the mechanisms provided by the underlying infrastructure.
- **Policy Service User Interface (PS-UI):** The PS-UI offers the operations of the PS in a user friendly manner. It allows to look up policy objects and change attributes, to create new policy objects from scratch or to copy and modify existing policies. Moreover, the user can activate and deactivate policies; this does not lead to an operation on the PS but on the respective policy enforcement service.
- **Policy Enforcement Services (PES):** A policy enforcement service is an application, which is able to interpret policy objects and map the object information onto the mechanisms of the infrastructure. Therefore, an enforcement service hides the details and peculiarities of the available mechanisms allowing the user to concern himself with the higher-level abstraction provided by the respective policy object class (POC). Since an enforcement service has to deal with available mechanisms, there might be several different enforcement services for one POC. Moreover, there will be different enforcement services for different POCs. We envisage an increasing set of formalised policy classes and correspondingly an increasing set of enforcement services such that policy-based management will more and more determine the structure and semantics of the management system.
  In the IDSM project [IDSM-D7] we build policy enforcement services for the instantiable *authorisation* and *reporting policy* classes we formalised, i.e. the authorisation resp. reporting service for enforcing authorisation resp. reporting policy objects. These services offer operations to activate and deactivate policies.
  - Authorisation Service: Authorisation policy objects contain information on the subject domain, the target domain, and the rights members of the subject domain should have on

members of the target domain. The underlying mechanisms supported by the infrastructure in our environment are access control lists which are used by the platform or by services to perform access control when invocations occur. Thus, the authorisation service transforms domain-based authorisation into information used by the environment for doing the actual access control but is not involved in the latter itself.

- Reporting Service: Reporting policy objects contain information on the subject domain, the target domain, event discrimination and destination of reports. The underlying mechanisms supported by the infrastructure in our environment are OSI event forwarding discriminator objects which can be created, manipulated and deleted in OSI agents via a management platform. The reporting service transforms domain-based reporting policies into information used by the underlying platform and the agents to perform the actual event reporting but is not involved in the latter itself.

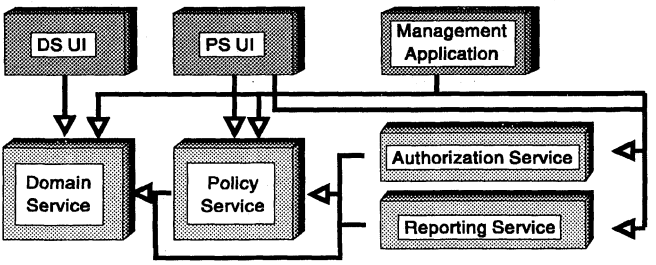

Figure 3.2: Functional Architecture

Figure 3.2 shows the usage relationship between the supporting components and management applications. The Domain Service (DS), the Policy Service (PS), and the policy enforcement services can be accessed by human managers via the respective user interfaces. The PS uses the DS to realise its analysis functionality (detection of potential authority conflicts). When the enforcement services (authorisation and reporting) are requested to activate or deactivate a policy object, they invoke the PS to look up the content of the policy (subject, target etc.). Then they use the DS to determine the scope of the policy to be activated or deactivated. Having this information they invoke the appropriate available infrastructure mechanisms. Other management applications use the DS to determine their scope, they use the PS to store and retrieve policy objects and the available enforcement services to activate and deactivate policies.

- **Policy Refinement Service (PRS):** For the interpretation of policy objects we propose in addition to the policy enforcement services a policy refinement service. A policy enforcement service maps a policy object to the infrastructure mechanisms of the system, whereas a refinement service should be independent from the infrastructure. The main purpose of a PRS is to enable policy hierarchies. A PRS should be able to map a policy object to other policy objects. Since a policy object consists of policy attributes, the refinement process depends on the policy attributes. In contrast to the enforcement services, where a specific service is responsible for a certain policy class, there will be only one type of refinement service. This service will refine all policy objects which cannot be directly enforced by PESs.

3.2 Policy Engineering Model

Policy influences the behaviour of manager and managed objects. It is the centrepiece of each management task. If such a task is to be automated based on policy concepts, one has to perform several activities which we describe in the sequel:

1. Structured Informal Policy Description
In this step, policies must be informally described as objects and their attributes. The description comprises the policy subjects and targets, the desired/permitted behaviour and applicable constraints.
2. Formalising the Policy using GDMO
In section 2.2 we defined a generic policy object class which contains attributes for subject, target, and constraints, from which specific policy classes can inherit. So, the main task consists of formalising the desired behaviour. Here, we have to determine a formal description of the actions and procedures of a manager or of the desired managed object behaviour. For the reporting policies which we formalised in IDSM, this results in attributes for specifying the events to be reported and the destination of event reports.
3. Extending the Policy Service by the new Policy Object Class
In order to make instances of the new class available for documentation and analysis by the Policy Service, the new class must be known in the Policy Service. It depends on the implementation of the latter whether or not adaptations need to be made.
4. Implementation of a Policy Enforcement Service (PES) for the new Class
Once a policy object class has been formalised, one has to investigate how the abstraction provided by the policy can be mapped onto management mechanisms of the underlying platform and systems or onto functionality of already existing management applications. If this is possible, then the enforcement of the policy can be fully automated. In this case, a PES specific to the policy class under consideration is to be designed and implemented. This enforcement service is a special management application which is able to interpret and enforce instances of the newly created policy object class. Enforcement is usually not restricted to a one-time or periodically performed sequence of actions. Does, for example, the policy specify a desired system state, then it is not sufficient to set up this state once. One has also to monitor the state and to take measures in case of deviations. For monitoring purposes the enforcement service to be implemented can ideally use a Reporting PES by creating and activating a reporting policy object. So, Policy Enforcement Services can have usage relationships.
5. Creation of policy object instances and activation
In the operational phase a policy, is introduced into the management system by creating an instance of the policy class. The policy is enforced by invoking the "activate" operation of the PES for the class. The latter enforces and monitors the policy until the policy is deactivated. Finally, deletion terminates the life-cycle of a policy object.

4. Application

We apply the policy concepts to two scenarios:

- In the IDSM project domains and policies are specified for managing the X.400 service over interconnected LANs, i.e. in the area of distributed system and service management.
- In the telecommunications area we identify domains and policies for specifying the interactions between managers in customer network management.

We give an outline of these applications in the sequel of this chapter (for a broader treatment of the first resp. second scenario see [Veldkamp94] resp. [Alpers95]).

4.1 Managing X.400 over interconnected LANs

In the IDSM project we use a pilot site consisting of several Local Area Networks (LANs) which are interconnected by a Wide Area Network (WAN). On top of this network an X.400 message handling application is provided. The local networks contain PCs and workstations. Those stations which are attached to the mail system have an X.400 User Agent and dedicated workstations serve as X.400 Message Transfer Agent (MTA). For this pilot a management system is being built based on industrial platforms providing access to OSI or SNMP managed objects. This system consists of the Domain Service, Policy Service, Authorisation Service, Reporting Service and specialised management applications which use the services. Moreover,

the applications can be used for realising higher-level policies not yet formalised (see Figure 3.1).

We specify authorisation policies to separate the areas of authority for the managers involved in managing the whole system. For each local site we define a domain of local managers and a domain of local managed objects and create an *authorisation policy* object which gives the manager objects rights (GET, SET, ACTION, CREATE, DELETE) on the members of the managed domain. Local managers can delegate rights to sub-managers which are in charge of managing specific services like the local X.400 service. For this, the managed objects relevant for X.400 are grouped in a sub-domain, a sub-domain of the local managers is created, and the domains are related to each other by another authorisation policy object. The target domain should also include objects like the PCs and workstations running X.400 software but the X.400 managers should only have read rights thus allowing them to see for example whether the configuration is adequate. Furthermore, we construct technology-specific domains for which we create reporting policy objects. It is for example useful to create a reporting policy which states that managers should be informed if the file systems in the workstations running MTA software are filled by more than 95%.

The fault management application which is being developed deals with fault diagnosis in interconnected LANs. It will help to analyse problems within the pilot site by providing status information of network and logical components for the human manager. The application observes errors and establishes correlations between them to detect the faulty component. Errors are observed by retrieving management information from MIBs, by polling attribute values, or by receiving event notifications. Additionally, diagnostic tests must be performed. The application uses domains and obligation policies to collect the management information. The obligation policies are:

- *reporting policy* in order to receive event notifications;
- *polling policies* for reading attributes of managed objects;
- *testing policies* for obtaining information on state and connectivity of managed objects.

The reporting policy will be enforced by the IDSM Reporting Service (see 3.2). The polling and testing policies are specific to the fault management application. The targets of the policies are specified by domains.

4.2 Customer Network Management

As a consequence of deregulation in the telecommunication area several new players have entered the scene. Beside the network operator providing bearer services there are providers of value-added services like virtual private network (VPN). Moreover, Customer Network Management (CNM) allows customers to manage subscribed services which are offered by service providers who in turn are customers for network services offered by public network providers. Figure 4.1 shows an example of the hierarchy of services and the corresponding hierarchy of CNMs.

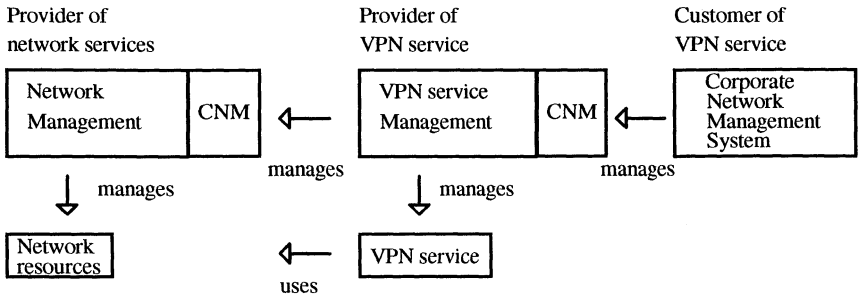

Figure 4.1: Roles in CNM-VPN environment

In this scenario, the service provider has to control the interaction with the customer managers and has to decide which customer requests can and should be fulfilled. In the sequel we show how this problem can be solved by using domains and policies for authorisation and reporting.
In order to organise access of different customers we introduce *authorisation policies*. For each customer we create a customer manager domain where customer manager objects, i.e. object representations for customer managers, are to be included. Moreover, we create a domain of managed objects for which the customer managers get access rights. In this domain we group all managed objects which are of concern for a specific customer, e.g. the VPN object representing the customer's VPN, link objects representing the links within the VPN, objects representing usage and accounting information and so on. Note that these objects are partly specific to CNM and partly overlap with managed objects the service manager uses also for service management purposes; i.e. the respective domain might contain references to objects which are also members of domains used for service management. Given these domains, we specify an authorisation policy which gives the members of the customer manager domain (possibly restricted) access to the objects in the domain of managed objects. The access rights are specified in terms of allowed operations. This way areas of authority can be easily specified along the borderlines of customer concern thus providing a clear authorisation structure in the management system.
Giving customers access to managed objects is not sufficient. Besides this, the customer must be informed in case the service cannot be provided as fixed in a service level agreement or in case of other situations (e.g. excessive usage) the customer might be interested in. For specifying information to be reported, we create or reuse domains and specify *reporting policies*. If, for example, we have already created a managed object domain for a customer, then we can reuse this domain as target for a reporting policy unless we want to reduce the scope and specify different policies for subsets which we then have to include in sub-domains. E.g., we can group all link objects for a VPN in a sub-domain and specify a reporting policy for these objects. The subject domain consists of managers who are responsible for the reporting policy and who will use the Reporting Service for setting up the infrastructure mechanisms accordingly.

5. Conclusions and Future Work

Management of complex networked systems with many parties involved needs concepts and services for organisation and semantic specification of management activity and intention. In this paper we presented the domain-based policy concept for this purpose. We formalised a generic policy object class using GDMO from which specific classes can be derived. Two such classes for authorisation and reporting policies have also been formalised. Formalisation is the prerequisite for implementing enforcement services which realise policies using underlying mechanism. For reporting and authorisation policies such enforcement services are being built on top of OSI-based platforms. The application of our concepts to management of X.400 over interconnected LANs as well as to customer network management shows their value for structuring and abstraction.
In our future work we plan to extend the area of policy-based management by formalising more policy classes and constructing the respective enforcement services. Candidates for formalisation are in particular generic policies which are relevant in many application areas. Here, we think of reporting policies for more complex situations and model-based policies to specify the desired system state. Moreover, technologies to support the enforcement of such policies have to be investigated.
Other conceptual areas where more work is required are policy hierarchies and formal refinement as well as policy analysis where conflict definition, detection and resolution must be investigated.

Acknowledgements

We gratefully acknowledge contributions of our partners in the IDSM and SysMan projects from Bull, Imperial College London, IITB Fraunhofer Institute, MARI, NTUA, AEG, Alcatel Austria, and ICL.

6. References

[Alpers93] Alpers, B., Becker, K., Raabe, U.: DOMAINS: Concepts for Networked Systems Management and their Realisation, Proc. GLOBECOM 93, Houston 1993

[Alpers95] Alpers, B., Plansky, H.: Applying Domains and Policy Concepts to Customer Network Management, to appear in proceedings of ISS '95

[Becker94] Becker, K., Holden, D.: Specifying the Dynamic Behaviour of Management Systems, J. Network Systems Management, vol. 1, no. 3,1994

[Calo93] Calo, S. B., Masullo, M. J., "Policy Management: An Architecture and Approach", Proc. IEEE Workshop on Systems Management, UCLA, Calif., 14.-16.April 1993

[CORBA91] Object Management Group: The Common Object Request Broker: Architecture and Specification, Doc. No. 91.12.1, 1991

[Domino92] Sloman, M., Moffett, J., Twidle, K.: Domino Domains and Policies: An Introduction to the Project Results", Domino Report Arch/IC/4, February 1992

[DSOM94] Alpers, B., Plansky, H., "Domain and Policy-Based Management: Concepts and Implementation Architecture", 5th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Toulouse, 10.-12. October 1994

[IDSM-D6] IDSM Deliverable D6/SysMan Deliverable MA2V2: "Domain and Policy Service Specification", October 1993

[ISO-10040] ISO/IEC IS 10040/PDAM 2: Systems Management Overview - Amendment 2: Management Domains Architecture, 2.11.1993

[ISO-10164-19] ISO/IEC JTC1/SC21/WG4: Management Domain and Management Policy Management Function, Committee Draft, 21.1.1994

[Mazumdar91] Mazumdar, S., Lazar, A.A., "Objective-Driven Monitoring", Integrated Network Management II, Ed. I. Krishnan, 1991, p. 653-678

[Moffett93] Moffett, J. D., Sloman, M. S., "Policy Hierarchies for Distributed Systems Management", IEEE Journal on Selected Areas in Communications, Vol. 11, No. 9, December 1993, S.1404-1414

[Veldkamp94] Veldkamp, W., Mitropoulos, S.: Integrated Distributed Management in LANs, Proc. NOMS 94, Orlando 1994

[Wies94] Wies, R.: Policies in Network and Systems Management - Formal Definition and Architecture, J. Network Systems Management, vol. 2, no. 1, 1994)

7. Biography

Burkhard Alpers received his Ph.D in mathematics from the University of Hamburg in 1988, where he specialised in geometry and algebra. Since 1989 he has been working with the research and development laboratories at Siemens, Munich. His research interests are in the field of network and system management.

Herbert Plansky received his Ph.D in electrical engineering from the Technical University of Munich in 1993, where his area of study was picture coding algorithms, digital signal processing, and VLSI architectures. Currently, he is a member of the research and development laboratories at Siemens, Munich. His research interests are in the field of network and system management of data and telecommunication networks. He is member of VDE/ITG (German association of electrical engineers).