INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Erstellung eines Klassifikationsschemas für
"Policies" aus dem Netz- und Systemmanagement

Bearbeiter: Thomas Berthold
Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering
Betreuer: René Wies
Abgabedatum: 15. 8. 1994

Ich versichere, daß ich diese Diplomarbeit selbständig verfaßt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

_____

Gliederung

1. Einleitung

1.1 Motivation

Mit der wachsenden Größe, Heterogenität und Komplexität verteilter Systeme und Netze wird es für Manager notwendig, immer mehr Detailwissen über verschiedene Netzkomponenten und Systeme zu haben. Deshalb empfiehlt es sich, den Grad der Abstraktion bei Managementanwendungen zu heben, um damit die Spezifikation einzelner Systeme zu verstecken.

Policies bieten die Möglichkeit, Managementziele auf einer hohen Abstraktionsebene festzulegen und wiederkehrende Arbeitsabläufe und Entscheidungen zu automatisieren. Policies sind nicht so sehr für einmalige Aktionen praktikabel, da der Arbeitsaufwand für das Spezifizieren, die Transformation in Policy-Code und die Wartung (Konfliktsuche und -auflösung) das Ergebnis um ein Vielfaches übersteigen würde.

Um die automatische Verarbeitung von Policies zu erreichen, ist es notwendig, eine formale Sprache für Policies zu definieren, die von Managern und Managementanwendungen verstanden werden kann.
Die Anforderungen an eine formale Sprache für Policies lassen sich aus der Klassifikation ableiten.
Diese Sprache muß für den Policy-'Spezifizierer', also den Personenkreis der die Policies erfaßt und betreut, einfach zu beherrschen sein, damit ein effizientes Management erreicht wird. Dennoch muß sie umfassend genug sein, um alle eventuell auftretenden Sonderfälle an Policies berücksichtigen und ein effektives Management garantieren zu können.

Ziele der Klassifikation und Motivation zum Thema der Diplomarbeit sind also:

- Einteilung und Gliederung von Policy-Katalogen nach verschiedenen Kriterien,

- ableiten einer Hierarchie über die Abstraktionsgrade verschiedener Policies für die (halb-) automatische Verfeinerung quer durch die Abstraktionsgrade von Policies,

- ableiten und überprüfen verschiedener Komponenten einer formalen Sprache aus der Klassifikation,

- den bestehenden Unterschied zwischen einer Klassifikation und einer daraus abgeleiteten formalen Sprache für Policies deutlich zu machen,

- durch die formale Sprache mögliche automatische Abarbeitung und Überwachung von Policies vorzubereiten,

- Policies sind bis jetzt, wenn überhaupt, immer nur in informeller Weise für ein Unternehmen niedergeschrieben [VIRN91], [FIDO91] etc.
  Deshalb ist auch eine formale Beschreibungssprache für Policies die nicht automatisch verarbeitet werden können wichtig,

- und damit, ein effizientes und effektives Management mit Hilfe von Policies zu ermöglichen.

1.2 Vorgehensweise

Will man ein Klassifikationsschema für Policies aus dem Netz- und Systemmanagement erstellen, ist es sinnvoll, verschiedene Policies aus ganz unterschiedlichen Zusammenhängen auf gemeinsame Kriterien und Besonderheiten zu untersuchen.
Die gemeinsamen Kriterien und Besonderheiten ergeben die verschiedenen Klassifikationskriterien, sowie die Anforderungen an eine formale Sprache und Transformation (Verfeinerung) von Policies.
Policy-Transformation ist ein Verfeinerungsprozess, der Policies auf eine Ebene bringt, auf der sie mittels vorhandener Management-Funktionalität automatisch anwendbar und ihre Aktionen ausführbar sind.

Für diesen Zweck war eine intensive Analyse verschiedenster Policy-Kataloge, zum Beispiel aus [FIDO91], [VIRN91] und [AMPER93] oder aus Gesprächen mit Netzverwaltern, Administratoren und Operateuren (debis-Systemhaus-Rechenzentren, Leibniz-Rechenzentrum, etc.), notwendig.

Die bei dieser Analyse herausgearbeiteten Kriterien wurden genutzt, um eine Vielzahl an (eindimensionalen-) Klassifikationen zu erstellen. Diese Klassifikationen wurden im nächsten Arbeitsschritt zu einem komplexen mehrdimensionalen Gefüge, also zu mehreren orthogonalen Dimensionen zusammengesetzt (siehe Abb.: 1).

Als Darstellungsform der Klassifikation wurde ein Dimensionsbild gewählt.
Ein Dimensionsbild ist günstig für diese Art der Klassifikation nach mehreren Ebenen oder Dimensionen, obwohl nicht alle Dimensionen voneinander unabhängig oder orthogonal zueinander sind, wie es das mathematische Vorbild eines Dimensionssystems suggeriert, sondern teilweise semantisch in Zusammenhang stehen.

Nachdem also einzelne Klassifikationen unter ihren Oberbegriffen zusammengesetzt worden sind, war es notwendig, die entstandene komplexe Klassifikation mittels verschiedener Kriterien (siehe 2.11 Bewertungskriterien für Klassifikationen) zu überprüfen, zu bewerten und zu verbessern.

Die Beschriftungen der einzelnen Dimensionen wurden semantisch getrennt und auf ein einheitliches Abstraktionsniveau gebracht.
In der Diplomarbeit werden bei der Erklärung der einzelnen Ebenen jeweils stellvertretend passende Policies aus der Praxis vorgestellt und erklärt. Diese Policies können auch in anderen Bereichen (Dimensionen) passend erscheinen, sind aber für die Darstellung des hervorzuhebenden Aspektes jeweils besonders geeignet.

Aus der entstandenen komplexen Klassifikation können die Anforderungen an eine formale Sprache für Policies hergeleitet werden.

Es gibt Aspekte, die in der Klassifikation nicht erfaßt werden und doch für die formale Sprache wichtig sind (z.B.: Policy Object-Namen und Beschreibung ...).

Ebenso werden durch die Klassifikation Einzelheiten erfaßt, die in der formalen Sprache, das heißt also in der Praxis, nicht relevant sind oder anders abgedeckt werden können (z.B.: verschiedene Klassifikationskriterien können durch Domänen [ISO 10164-19] ausgedrückt werden).

Besonderes Augenmerk verlangen die unterschiedlichen Aktivitäten, die erforderlich sind, um das Policy-Ziel oder Policy-Goal zu erreichen.
Zu untersuchen ist hierbei, in wieweit man sich beim Design der formalen Beschreibungssprache auf vorhandene Management-Funktionalitäten wie zum Beispiel SMF systems management functions [ISO 10164-X], OSF/DME distributed services [OSF DME92], CMIP common management information protocol [ISO 9596] oder SNMP simple network management protocol [CASE93] und Sprachmittel wie zum Beispiel ASN.1 [CCITT 208] abstützen kann.

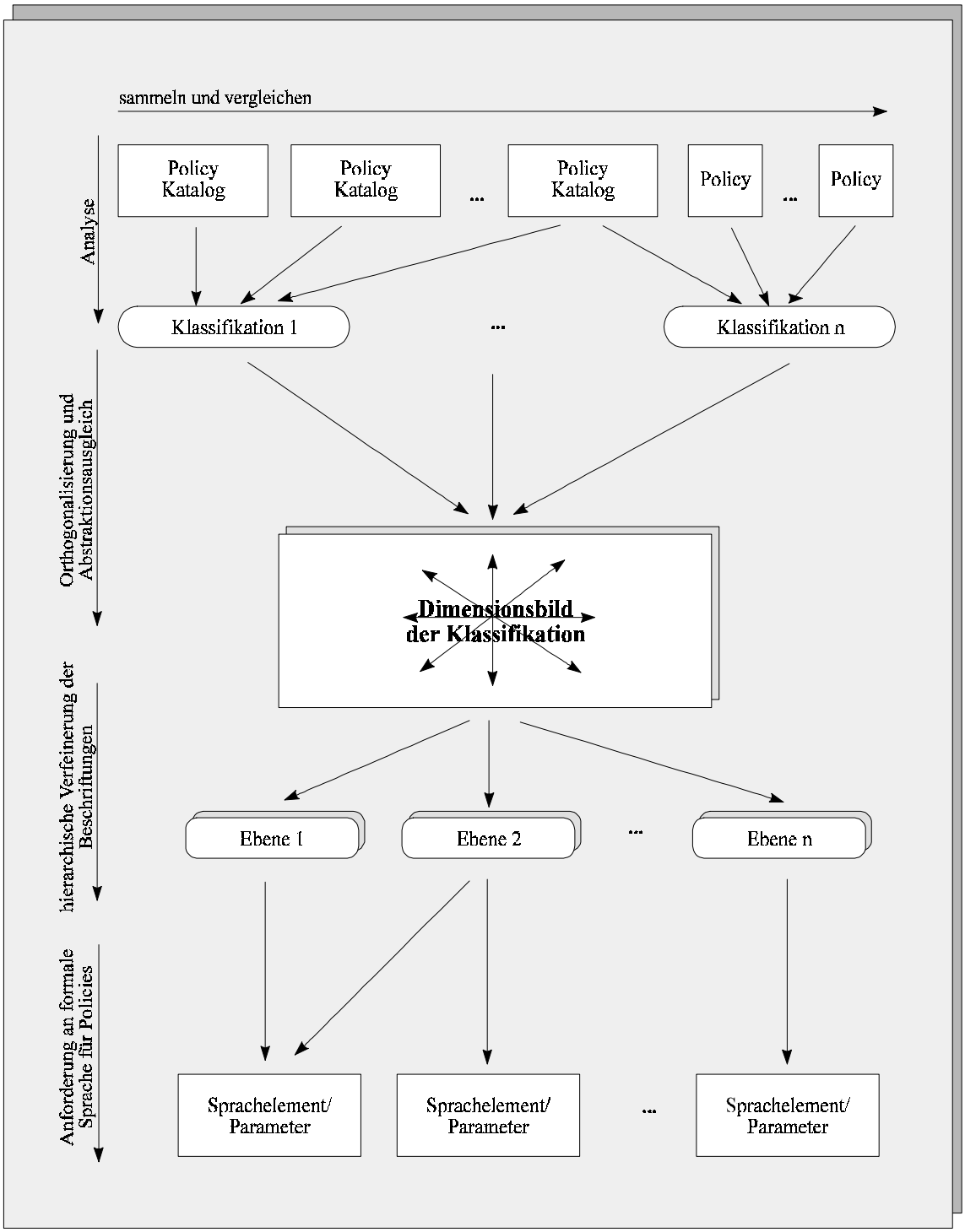
sammeln und vergleichen
Analyse
Policy Katalog
Policy Katalog
...
Policy Katalog
Policy
...
Policy
Klassifikation 1
...
Klassifikation n
Orthogonalisierung und Abstraktionsausgleich
Dimensionsbild der Klassifikation
hierarchische Verfeinerung der Beschriftungen
Ebene 1
Ebene 2
...
Ebene n
Anforderung an formale Sprache für Policies
Sprachelement/ Parameter
Sprachelement/ Parameter
...
Sprachelement/ Parameter

Abb.: 1

1.3 Was sind Policies?

In der Literatur besteht häufig einige Verwirrung zwischen Policies, betrieblichen Zielen, Strategien und Taktiken zum Management.

Ähnlich zu [WIES-PLEN94] definiere ich Policies wie folgt:

Def:

Policies sind aus den Zielen des (Netz- und System-) Managementes abgeleitet und definieren das geforderte Verhalten von verteilten, heterogenen Systemen und Netzen.

Policies sind also Vorgaben, die ein gewünschtes Verhalten und Eigenschaften der zu managenden Systeme, Netze und Managementsysteme widerspiegeln und beeinflussen. Policies stellen den informellen Aspekt des gewünschten Verhaltens das, arbeiten aber nicht selbst als Managementprozesse, um dieses Verhalten zu erreichen.

In dieser Arbeit werden Policies, wie in der Definition, nur aus dem speziellen Sichtfeld des Netz- und Systemmanagementes betrachtet, weil die Ziele dieser Arbeit in diesem Bereich liegen.

Netz- und Systemmanagement sind sich sehr ähnlich und haben viele Zusammenhänge. Ihre Unterscheidung und Abgrenzung voneinander ist in [KRUPCZAK93] beschrieben.

Man benötigt auch für Policies im integrierten Netz- und Systemmanagement eine geeignete Management-Architektur, die die verschiedenen Aspekte der vier Managementmodelle [HEG93], [HEG94] berücksichtigt:

- eine Beschreibungssprache zum Spezifizieren der MO´s,
- eine Möglichkeit, organisationelle Aspekte mittels Einschränkungen und Aktivitäten zu unterstützen (organizations subsystem),
- einen Kommunikationsmechanismus, zum Austausch von Informationen und Verteilen der Policy-Aktivitäten,
- eine strukturelle und gegliederte Ansicht auf Policy-Kataloge und ihre Funktionalitäten (functional model).

Bis jetzt ist das Management, also das Verwalten und Kontrollieren gerade von großen, heterogenen und unübersichtlichen Netzen ein Alptraum für jeden Systemadministrator.

Warum sollte man aber Policies den derzeitigen Methoden des zentralisierten Managementes (Telefonanrufe bei gewünschten Änderungen, Austausch von Papieren zur Festlegung von Verfahrensweisen, usw.) vorziehen?

Policies bieten die Möglichkeit zu einem dezentralen, (halb-) automatischen Management und zur automatischen Zielüberwachung (Monitoring) durch das System oder Netz selber.

Management Policies bieten zudem durch ihre mögliche hohe Abstraktionsebene die Möglichkeit, Strukturänderungen (neues Release, andere Personalstruktur) zu überdauern und bei oft wiederkehrenden Arbeitsvorgängen eine feste Vorgensweise zu erreichen, die gegenüber regelbasierten-Systemen eine höhere Flexibilität durch die Anpassung der Policies an äußere Gegebenheiten beinhaltet.

Mit diesen Eigenschaften sind Policies zukünftig für ein zielorientiertes Netz- und Systemmanagement notwendig und bieten eine einfache, aber mächtige Methode zum 'Customizing', also zum Anpassen des Managementsystems an Benutzeranforderungen, des durch Plattformen unterstützten Managements.

1.4 Policies als Objekte

Policies repräsentieren Information und Funktionalität, die verändert oder abgefragt werden kann. Es ist sinnvoll, Policies als Objekte mit verschiedenen Objektklassen in einem System zu sehen. Diese Objektklassen unterscheiden sich durch jene Hauptkriterien, die durch die Klassifikation deutlich werden.

Ein Policy-Service gestattet dann die Operationen create, delete, manipulate, ... auf den Policy-Objekten.

Die Operation create erzeugt automatisch eine Instantiierung einer Objektklasse mit teils ausgefüllten und teils freien oder unbelegten Attributen und Parametern.

Aus dieser Feststellung läßt sich folgern, daß Policies einen Lebenszyklus (Planung, Erstellung, Manipulation, Löschung...) wie auch andere managed objects (MO´s) [ISO 7498-4] in einem System aufweisen. Managed objects sind abstrakte Repräsentationen von jedem "Ding" im Netz und System, das administriert wird [PAVLOU93].

Die Frage, wer den Policy-Service oder Teile daraus benutzen darf, kann wiederum mit einer sogenannten *Meta-Policy* beantwortet werden.

Policy1: *Jeder Systemverwalter darf Policies, deren Targets (Ziele) und Aktivitäten innerhalb seines Aufgabenbereiches liegen, erstellen und ändern.*

Gerade der Policy-Service delete muß mit besonderer Vorsicht behandelt werden, da eventuelle Abhängigkeiten zwischen de Policies zerstört werden könnten.

Policy2: *Der Abteilungsleiter muß einmal in der Woche die CPU-Zeitkosten Abrechnung seiner Abteilung prüfen.*

Policy3: *Der Abteilungsleiter hat Zugriff auf die CPU-Zeitkosten Abrechnung seiner Abteilung.*

Wird hier Policy3 gelöscht, entsteht ein Konflikt.

Die Sichtweise, Policies als Objekte aufzufassen erleichtert die Darstellung von verschiedenen Beziehungsmöglichkeiten (siehe auch [ISO 10164-3]) zwischen ihnen.

Hierarchische Beziehung:	die Policies sind	durch den Prozeß der Verfeinerung miteinander verknüpft (siehe 2.2 Hierarchy-Ebene und Verfeinerung),
zeitliche Abhängigkeiten:	die Policies sind	sequentiell, gleichzeitig oder ohne Beziehung zueinander abzuarbeiten,
kausale Abhängigkeiten:	die Policies sind	durch kausale Zusammenhänge verknüpft (z.B.: Gültigkeitsdauer),
Prioritätsbeziehungen:	die Policies sind	durch Vorrangregelung bei gleicher Prioritätsstufe miteinander in Beziehung gesetzt (siehe 2.3).

Policies werden also während ihres Lebenszyklus auch geändert, manipuliert und gegebenenfalls transformiert oder verfeinert.

Gibt es aber auch Policies die während ihrer Gültigkeitsdauer nicht verändert werden und immer gelten, also 'fixed policies' [MOFF91:p8]?
Unserer Ansicht nach gibt es wahre 'fixed policies' nicht, denn das Konzept von Policies, wie wir es definiert haben, ist deshalb so mächtig, weil Policies sich flexibel ihrer äußeren Umgebung anpassen können.

Aus diesem Grund kann es keine wirklichen 'fixed policies' in unserem Konzept geben.
Es gibt aber wohl Policies, die eine lange Gültigkeitsdauer (siehe 2.5 Life-Time of Policy Object) aufweisen und sich in der Betriebszeit des Systems oder Netzes nicht ändern.

Policy4: *Ein User darf nur dann einloggen, wenn er sich durch eine gültige Kennung und Paßwort authentisiert.*

Doch 'fixed' kann man auch dieses Beispiel nicht bezeichnen, da sich die Parameter im Betrieb ändern (User, Kennung, Paßwort) und diese Policy auch transformiert oder verfeinert werden muß (z. B.: Verfeinerung durch Auswahl des Vorgangs der Authentisierung).

1.5 Anforderungen an das Policy-Processing-System

Das Policy-Processing-System stellt eine Managementeinheit dar, die sämtliche Aufgaben der Policy-Verwaltung (Eingabe, Manipulation, Verarbeitung, Löschung) im sogenannten Lebenszyklus einer Policy übernimmt.

Dazu muß das Policy-Processing-System unter anderem einen Editor zur Verfügung stellen, mittels dem Policies geeignet darzustellen und bearbeitbar sind.

Es kann mehrere Konzepte für eine formale Sprache für Policies geben.
So ist es denkbar, Policies die Autorisierungen vergeben mittels einer graphischen Darstellung (wie bei MIRÓ [MARR93:p14], TME Tivoli Management Environment [WELLS94]) zu beschreiben, während diese Methode für komplexere Policies aus Gründen der Übersichtlichkeit und aus Gründen der Vielfalt der verschiedenen Kriterien von Policies (Abhängigkeiten, Constraints, Hierarchien...) nicht passend erscheint (Vielfachheit der Kanten und deren Beschriftungen).

Ein Editor muß sich flexibel an die gewählte formale Sprache anpassen können.

Empfehlen würde sich, daß man ein sogenanntes "Ur-Objekt" für eine spezifische Policy-Art (Unterscheidung nach Klassifikationsebenen) wählen kann, um dieses geeignet zu modifizieren, bzw. etwaige Einschränkungen und Erweiterungen einbinden zu können.

Dieser objektorientierte Ansatz bietet viele Vorteile, zum Beispiel durch die Erweiterbarkeit, falls mit der Zeit deutlich wird, daß mehr oder andere Policy-Objekte bearbeitet werden müssen.
Auch das Policy-Processing-System sollte bei einer objektorientierten Sprache flexibel genug sein, neue Objekte zu akzeptieren bzw. verarbeiten zu können.

Der Editor sollte auch gleich nach Eingabe der neuen Policy einen etwaigen, vorraussehbaren Konflikt prüfen können.

Für große Policy-Konstrukte empfiehlt es sich ein Tool zu schaffen, mittels dem man sich durch die Policy-Abhängigkeiten bewegen kann.

Weiterhin sollte das Policy-Processing System auch für die korrekte Transformation/Verfeinerung (2.2 Hierarchy-Ebene und Verfeinerung), Aktivierung, Verteilung und Delegierung der Policy-Codes, gegebenenfalls mit menschlicher Unterstützung falls die Aktivität nicht durch die vorhandene Management-Funktionalität ausführbar ist, sorgen (siehe Abb.: 2).

Es sind bereits Managementplattformen (Netview, HP-Openview...) entwickelt worden, deren Ansatz ist, der Netz- und Systemadministration auch bei nichtlokaler Präsenz zu gestatten, ihre Aktivitäten zum Management durch remote-Funktionen, also Aktivität und

Ausführung von Funktionalität von entfernten Standpunkten auf managed objects (MO´s) (z.B.: Hub, Switch oder net-control-processor NCP), auszuüben.

Sinnvoll ist sicher, das Policy-Processing-System genau dort zu integrieren, wo es Zugriff auf schon vorhandene Management-Funktionalität hat (SNMP, SMF, DME, TME). Voraussetzung für ein Management mit Policies ist, daß auf die Netzressourcen geeignet zugegriffen werden kann.

Die Einhaltung von Sicherheitskriterien ist eine weitere wichtige Anforderung an das in die Managementplattform integrierte Policy-Processing-System, denn es finden ja direkte Eingriffe auf das Netz und System und dessen Ressourcen statt.

Policies kontrollieren und aktivieren dann die von der Managementplattform ausgeführten Aktionen.

Als Erweiterung zu [SLOM93:p16] sollten zusammengefaßt verschiedene Dienste für Policies bereitgestellt werden:

- kreieren,
- lesen,
- manipulieren/verfeinern,
- speichern,
- löschen,
- delegieren der Activities,
- auflösen von Konflikten,
- und aktivieren/deaktivieren.

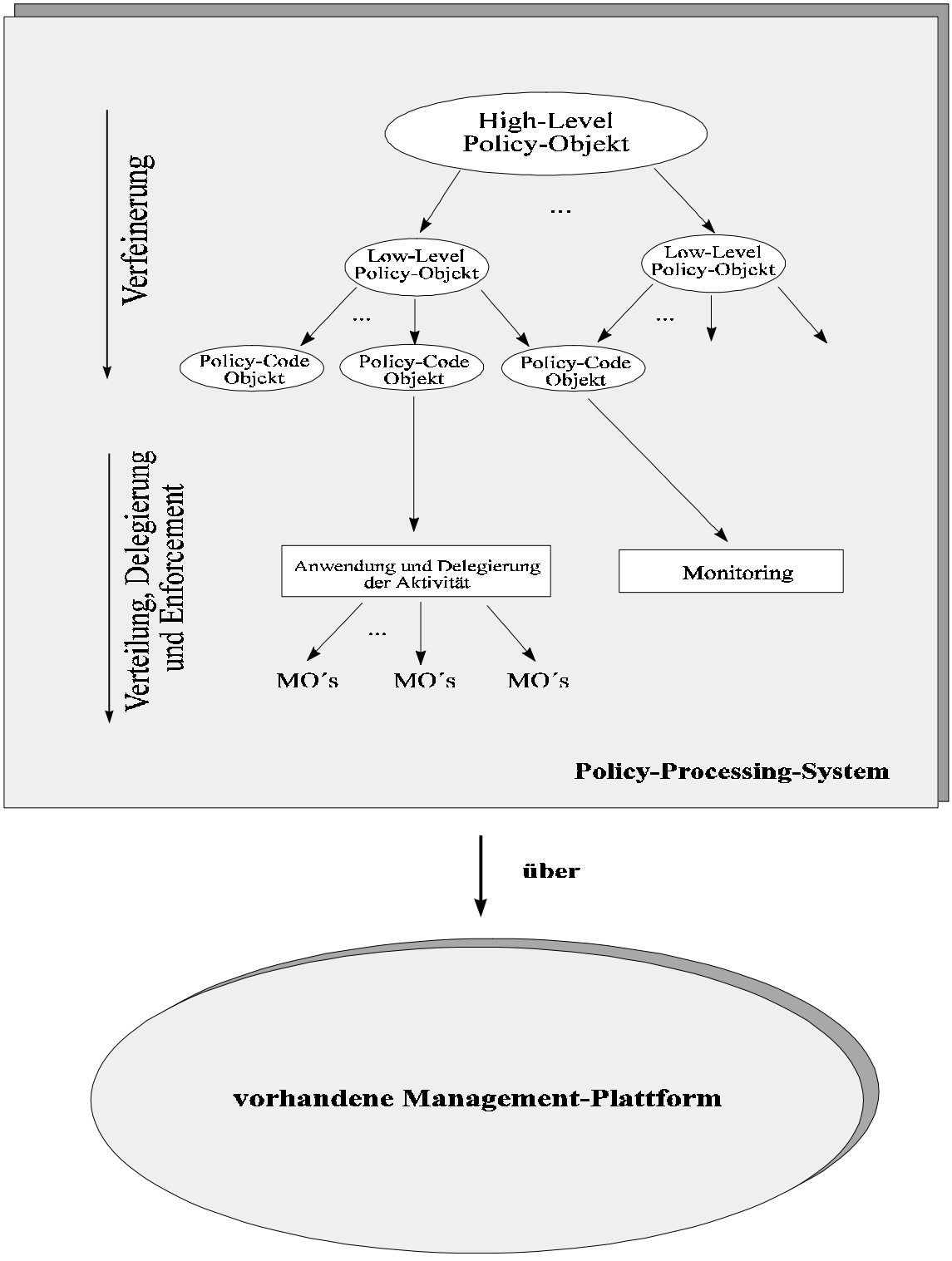
Verfeinerung
High-Level Policy-Objekt
...
Low-Level Policy-Objekt
Low-Level Policy-Objekt
...
...
Policy-Code Objekt
Policy-Code Objekt
Policy-Code Objekt
Verteilung, Delegierung und Enforcement
Anwendung und Delegierung der Aktivität
Monitoring
...
MO´s
MO´s
MO´s
Policy-Processing-System
über
vorhandene Management-Plattform

Abb.: 2

2. Die Dimensionen der Klassifikation

2.1 Anforderungen an eine Klassifikation für Policies

Def:

> *Eine Klasse ist eine Kollektion von Objekten mit mindestens einem gemeinsamen Merkmal [DUD89]*

Diese Definition ist auch für die Klassifikation von Policies sinnvoll. Wir haben Kataloge von Policies aus dem Netz- und Systemmanagement nicht nur auf ihre gemeinsamen Merkmale, sondern auch auf ihre Besonderheiten hin untersucht.

Daraus konnte ich die Anforderungen, sowie die Bewertungskriterien (2.11 Bewertungskriterien für Klassifikationen) an eine Klassifikation für Policies aus dem Netz- und Systemmanagement herrauskristallisieren.

Aus dieser Untersuchung sind Klassifikationen entstanden wie zum Beispiel:

- Policies gehören zur Verteilung von Ressourcen (auch Autorisierung),
- Policies gehören zur Überwachung der Ressourcen (MO-monitoring),
- Policies gehören zur Fehlerbehebung und Fehlerprävention.

- Klassifikation der Policies nach der Frage: Wen betrifft die Policy als Subject / Target?

- Klassifikation der Policies nach Gesichtspunkten der Implementierbarkeit, (kann die Policy nach "Stand der Technik" implementiert werden?

- Klassifikation der Policy-Activity nach OSI-Schichten.

Die häufigsten gemeinsamen Merkmale der Policies sind eingegangen in die eindimensionalen Ansätze zur Klassifikation (2.2 - 2.10).

Diese eindimensionalen Ansätze waren alle für sich sinnvoll, aber deckten nur einen 'Raum' ab, so daß bald klar wurde, daß man mehrere dieser Ansätze gleichzeitig ins Blickfeld rücken muß, will man eine Klassifikation erhalten, die alle oder zumindest die meisten wichtigen Gemeinsamkeiten, die zu berücksichtigen sind, herausstellt.

Das entstandene Dimensionsbild ist nur eine Form der Darstellung.

Ebensogut hätte man die einzelnen Kriterien nebeneinander stehen lassen können, aber das Prinzip, Policies wie in einem mehrdimensionalen Raum zu betrachten, also jeweils Punkte auf den Achsen zuweisen zu können, ist sehr anschaulich. Inspiriert dazu wurden wir durch die Darstellung der Dimensionen des Managements aus [HEG93].

Der erste Eindruck der Klassifikationsgraphik mag vielleicht abschreckend erscheinen durch die multi-Dimensionalität.

Dazu ist zu bemerken, daß die Klassifikation als Erstes dem Bewertungskriterium der Vollständigkeit genügen sollte.
Für gezielte Anwendungsfälle für Klassifizierungen von Policies ist es natürlich möglich, die Dimensionen auf ein Mindestmaß zu verringern. Das Mindestmaß ergibt sich dabei aus dem jeweiligen Anwendungsfall.

Gefordert ist, daß die Dimensionen relativ sauber orthogonal aufeinander stehen, das heißt, sie müssen semantisch unabhängig voneinander sein. Auch die Beschriftungen der Kanten sollten sich nicht semantisch überlappen, soweit dies möglich und sinnvoll ist.

Die Dimensionen stehen in der vorgestellten Klassifikation relativ orthogonal aufeinander, das heißt, eine Abhängigkeit der Ebenen voneinander besteht nur durch inhaltliche Bezugnahmen. Zum Beispiel ist die Ebene *Activity* inhaltlich abhängig von der Ebene *Type of Target* aber die Funktionalitäten, nach denen unterschieden wird, können semantisch unabhängig voneinander betrachtet werden.

Die Beschriftungen der einzelnen Ebenen sind dagegen nicht völlig voneinander unabhängig.
Zum Beispiel liegt in der Ebene *Management functionality of Policy-Activity* eine Überlappung der einzelnen Beschriftungen vor, da beim Management meist eine Funktionalität nicht ohne die andere vonstatten gehen kann. So ist ein Sicherheits- oder Accountingmanagement undenkbar ohne einen Zugriff auf die Konfiguration des Systems, und ein Fehlermanagement nicht ohne Performancemanagement möglich.

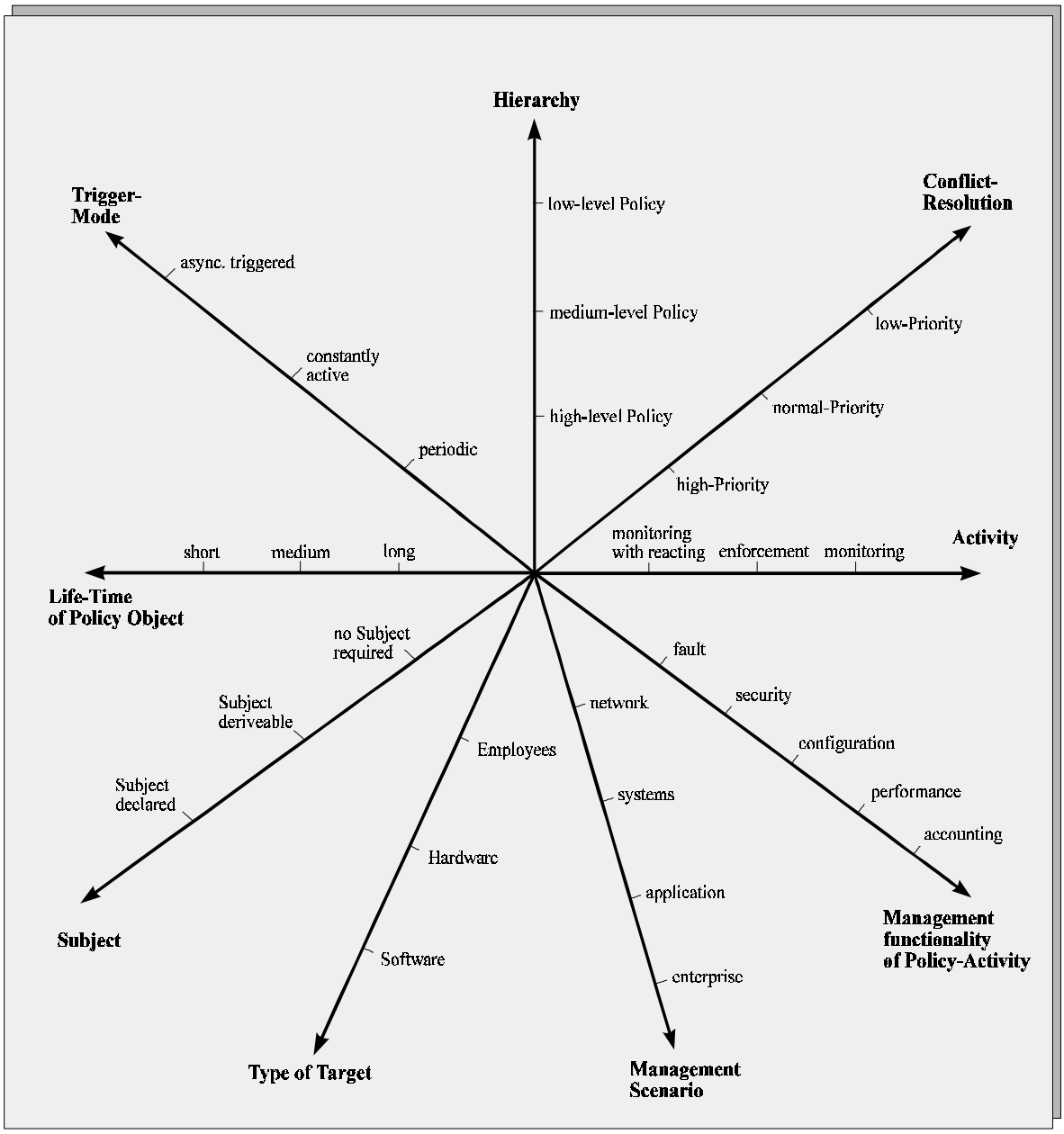

Abb.: 3

Will man eine Policy mit diesem Schema klassifizieren, muß man sich vorstellen, daß man ihr jeweils pro Achse einen (oder mehrere) Punkte, also pro Dimension eine (oder mehrere) Koordinaten, zuweist.

2.2 Hierarchien und Verfeinerung

Nach intensivem Studium verschiedenster Policies haben wir festgestellt, daß im normalen Anwendungsfall Policies in verschiedenen Abstraktionsebenen auftreten.
Dies ist ein Kriterium, nach dem man Policies klassifizieren kann.

Will man dieser Feststellung in der Sprache Rechnung tragen, muß eine für Policies in Frage kommende formale Sprache so flexibel sein, daß man mit ihr verschiedene Grade der Abstraktion von Policies ausdrücken kann und diese innerhalb des Sprachgefüges verfeinern kann.

Daraus folgt, daß eine Policy in der Klassifikation gleichzeitig mehrere Punkte auf der *Hierarchy-Ebene* belegen kann.

Ein anderer Ansatz währe, die Policies nur in einer Abstraktionsebene, nämlich der Niedrigsten, zuzulassen, was eine Einschränkung in der Flexibilität bei der Definition von Policies bedeutet, aber dafür eine bessere und eindeutigere Klassifikation zur Folge hätte, denn die Verfeinerung der Policies und die Zusicherung ob die Ziele der Aktionen der höheren Policy nach der Verfeinerung erfüllt oder abgedeckt werden, fallen weg.

Unterschieden werden muß zwischen der (halb-) automatischen Verfeinerung von Policies durch die Hierarchiestufen in der Klassifikation, und des in der Anwendung der Verfeinerung der formalen Sprache entstandenen Policy-Codes auf die Target-Objekte. Dies sind zwei getrennte Arbeitsschritte (siehe Abb.: 2).

Die Informationen auf ein einheitliches Abstraktionsniveau zu bringen und Konflikte zu erkennen und eventuell auch zu lösen sind die schwierigsten Aufgaben bei der Verfeinerung in der Klassifikation. Ob dieser Arbeitschritt jemals ganz automatisch und ohne menschliche Entscheidungen vonstatten gehen kann, sei dahingestellt.

Die Unterscheidung, welcher Hierarchie- oder Abstraktionsebene die Policy angehört, kann entweder vom System generiert werden, wenn die Policy schon formal eingegeben wurde, oder muß von der eingebenden Person, gegebenenfalls in Intervention mit dem Policy-Processing-System, stattfinden.

Diese Entscheidung ist manchmal auch deshalb schwer zu treffen, weil die Klassengrenzen oft nicht sauber definierbar sind. Aus diesem Grund ist es vorteilhaft, in der Ebene der Hierarchien nicht zu viele exakte Unterscheidungen vorzusehen, damit eine eventuelle Fehleinstufung der Policy nicht zu Klassifizierungs- oder Verfeinerungsfehlern führt.

In der angegebenen Klassifikation werden drei verschiedene Abstraktionsebenen (Hierarchieebenen) für Policies vorgesehen:

- **high-level Policies:** definieren Aktivitäten oder Ziele in Form von grob definierten Operationen, die mehr den betrieblichen Zielen abgeleitet sind, als technisch orientiert, und die erst transformiert oder verfeinert werden müssen, bevor sie vom Management-Personal oder den Management-Applikationen abgearbeitet werden können,

- **medium-level Policies:** definieren Aktivitäten in Form von Funktionen und Operationen, welche durch Zielbestimmung und Verfeinerung zu Policy-Codes oder Policy-Rules abgearbeitet werden können, oder auf bestehende Management-Applikationen anwendbar sind (siehe systems management functions (SMF) von OSI [ISO 10164-X] oder OSF/DME distributed services [OSF DME92]),

- **low-level Policies:** die Aktivitäten der low-level Policies können durch die Management-Funktionalität ohne weitere Verfeinerung auf die managed objects (MO´s) der Target-Ressourcen angewandt werden.

Policies sollen also in mehreren Hierarchieebenen eingegeben werden können. Das Policy-Processing-System stellt gegebenenfalls fest, zu welcher Abstraktionsebene die Policy gehört, oder splittet zusammengesetzte Policies in mehrere Teile.

Der Grad für die Auswahl der Abstraktionsebenen bezieht sich auch auf den Abstraktionsgrad der Policy-Activity, das heißt die Detailtreue in der Festlegung der Aktionen, die noch auszuführende Hilfsmittelauswahl, Verteilung der Aufgaben der Policies und Zuordnung der Targets (Policy-Ziele) und Subjects (Policy-Ausführende) sowie die Auflösung der Domänen, falls notwendig.

Die Verfeinerung drückt sich im Klassifikationsschema durch jeweils andere, spezifischere Beschriftungen an den Ebenen je niederem Abstraktionsgrad aus (siehe Verfeinerung der einzelnen Ebenen).

Wann ist aber eine Transformation oder Verfeinerung der Policy in der Klassifikation abgeschlossen? Wenn jede einzelne Domäne aufgelöst ist, oder in der Verfeinerung der Klassifikation keine weitere Untergliederung vorgesehen ist? Ist es überhaupt sinnvoll, Policies so genau zu klassifizieren?

Wenn man eine Klassifikation zu stark betreibt, hat man keine gemeinsamen Merkmale der einzelnen Objekte mehr, und muß für jedes Objekt eine eigene Klasse schaffen. Das ist sicher nicht das Ziel einer Klassifikation.

Wie oben erwähnt beinhalten viele Policies auch verschiedene Abstraktionsebenen, und sind somit mehreren Hierarchiestufen zugeordnet.

Beispiel einer Policy mit gemischten Hierarchien:

Policy5: *An einem Ticket-Buchungssystem teilnehmende Reisebüros sind über eine Standleitung angeschlossen. Im Falle eines Netzfehlers haben die Reisebüros die Möglichkeit, sich über eine Wählverbindung einzuloggen und müssen sich dann über einen Identifikationsvorgang authentisieren.*

Eine zusammengesetzte Policy läßt sich schwer in die Klassifikation einordnen und selten richtig automatisch (oder zumindest halbautomatisch) verfeinern.

Beispiel-Policy5 könnte man in drei verschiedene Policies mit jeweils nur einer Hierarchiestufe trennen:

Policy5a: *Der Betreiber des Ticket-Buchungssystems ist dafür verantwortlich, neben den Standleitungen für den Fall eines Netzfehlers auch Modems für Wählverbindungen zu unterhalten.*

Diese Policy ist high-level, da die Datenraten, Anzahl der Modems oder Art der Fehlererkennung usw., also die auszuführenden Funktionen und angesprochenen managed objects (Instantiierungen des Target-Parameters), nicht definiert werden.

Policy5b: *Im Falle eines Netzfehlers müssen die Modems für die Wählverbindung aktiviert werden.*

Diese Policy ist medium-level, da bereits eine Funktion (die Aktivierung der Modems), aber noch nicht der Ausführende oder die Art der Fehlererkennung angesprochen werden.

Policy5c*:* *Melden sich Teilnehmer über eine Wählverbindung an, muß der Identifikationsvorgang dafür sorgen, daß sich nur Teilnehmer mit korrekter Kennung und Paßwort einloggen können.*

Diese Policy besteht aus sogenannten Rules und ist low-level einzuordnen, denn sie spricht direkt den Ausführungsteil (Identifikationsvorgang) und das Target an.

Ein Beispiel für eine Policy mit nur einem Abstraktionsgrad ist Policy6:

Policy6: *Jeden Tag sind Daten zwischen den Firmen X und Y in der Zeit zwischen 18.00 und 20.00 Uhr im 'encrypted mode' auszutauschen.*

Diese Policy ist unter low-level einzuordnen, denn mit einer kleinen Verfeinerung ist sie direkt abzuarbeiten:

Verfeinerung Policy6: *Operateure der Firmen X und Y haben jeden Tag in der Zeit zwischen 18.00 und 20.00 Uhr den 'Encrypter' vor bzw. nach dem Modem zwischenzuschalten und dann die Datenübertragung zu starten.*

Weitere Beispiele von Policies mit verschiedenen Hierarchieebenen und deren Verfeinerung findet man auch in [MOFF92:p6 ff].

Nicht alle Policies sind zur Verfeinerung geeignet. (Zum Beispiel gewisse Policies für menschliche Manager oder Policy-Rules bzw. low-level Policies).

Beispiel zweier nicht zur Verfeinerung geeigneten Policies:

Policy7: *Der Systementwickler hat dafür zu sorgen, daß das Paßwort beim Einloggen in das System nicht auf dem Bildschirm sichtbar wird.*

Policy8: *Der Operateur muß jeden Abend die Endezeiten der Batchverarbeitung in Datei X eintragen.*

Um spätere Konflikte lösen zu können, ist es notwendig, den Verfeinerungsprozeß aufzeichnen, und umkehrbar machen zu können (z.B.: mittels Querverweisen auf die übergeordnete Policy). Auch beim Verfeinerungsprozess selbst können Konflikte ausgelöst werden (zum Beispiel durch Wahl eines durch eine andere Policy gesperrten Tools).

Es bestehen also auch Relationen zwischen den Hierarchiestufen der Policies (siehe 1.4 Policies als Objekte). Diese Relationen müssen aber nicht eindeutig sein, da eine low-level Policy zum Beispiel von mehreren high-level Policies beeinflußt werden kann, und umgekehrt.

Bei der Anwendung der Klassifikation durch das Policy-Processing-System können diese vermehrten Zusammenhänge vor allem auch dann entstehen, wenn man low-level Policies nicht mehrfach für alle angesprochenen Targets oder Subjects, welche sich zum Beispiel aus einer Domäne ergeben, speichert. Damit würde durch die Anwendung der Klassifikation Information verloren gehen.

Zu Untersuchen bleibt, ob die neuen Policy-Objekte, die bei der Verfeinerung in der Klassifikation entstehen, das gewünschte Ergebnis der Ursprungspolicy auch abdecken. Dies gilt natürlich auch bei der Transformation in der formalen Sprache, wo *Activities* nicht nur klassifiziert werden, sondern auch ausführbar gemacht werden müssen.

2.3 Policy Conflict-Resolution

Es gibt mehrere denkbare Formen der Konfliktauflösung zwischen zwei oder mehreren konkurrierenden Policies.
Policies lassen sich auch anhand ihrer Prioritäten klassifizieren. Prioritäten können durch Rangordnungen in der Personalhierarchie oder durch Forcierung eines Ziels entstehen.

So wird eine Policy, die von einem Abteilungsleiter definiert wird sicherlich beim Mitarbeiter mehr Gewicht haben, als eine Policy von seinen Kollegen. In anderen Fällen ist es in der Anwendung der Policies sicher nützlich, mittels Prioritäten eine Forcierung gewisser Ziele zu erreichen.

Es sind folgende drei Prioritätsstufen in der Klassifikation ausreichend:

- *low-Priority* haben Policies, die bei einem Konflikt der anderen Policy den Vorzug geben,

- *normal-Priority* Policies die "stärker" als *low-Priority* und "schwächer" als *high-Priority* bei der Konfliktauflösung zwischen zwei Policies sind,

- *high-Priority* haben Policies, die bei einem Konflikt in jedem Fall gültig sind.

Beispiele:

für Priorität *low*:	***Policy9:***	*Kein User darf die Dateien seiner Kollegen lesen,*
für Priorität *normal*:	***Policy10:***	*Jeder User kann anderen Mitarbeitern durch Eintrag in File x gestatten, seine dafür gekennzeichneten Dateien zu lesen,*
für Priorität *high*:	***Policy11:***	*Der Superuser hat alle Zugriffsberechtigungen im lokalen System.*

Man kann deutlich den Trend erkennen, daß mit niedrigen Prioritäten Policies ausgestattet werden, die sich mehr auf allgemeine Ziele und Unternehmensstrategien beziehen.

Die Wertetabelle in Abb.: 4 zeigt an, welche Policy bei einem Konflikt zweier Policies mit unterschiedlichen Prioritäten gültig ist:

Policy B \ Policy A	gering	normal	hoch
gering	?	A	A
normal	B	?	A
hoch	B	B	?

Abb.: 4

Nicht sinnvoll erscheint eine genauere Unterteilung in mehr als 3 Prioritätsstufen, da die Zuteilung zu einer Priorität Ermessenssache ist, und leicht falsch gewählt werden kann.

Zum Beispiel kann bei 10 Prioritäten durch falsche Entscheidung in den mittleren Prioritätsstufen 3-7 leicht eine falsche Klassifikation entstehen. Wenn dann auf diese Klassifikation durch das Policy-Processing-System zurückgegriffen wird, entstehen Fehler bei der Konfliktauflösung.

Noch weiter zum Thema *Conflict-Resolution* untersucht werden muß, ob man die Vorgenhensweise bei der Konfliktauflösung durch Beziehungen zwischen den Policies ausdrücken kann.

2.4 Trigger-Mode

Die meisten Policies benötigen zur Ausführung oder zum Anstoß einen sogenannten Trigger-Mechanismus. Ein solcher Mechanismus kann natürlich auch dazu benutzt werden, die Aktivitäten der Policy zu deaktivieren.

Es werden in der Klassifikation drei verschiedene Trigger-Modi unterschieden:

- async. triggered: die Policy wird durch ein asynchrones Ereignis aktiviert.
- constantly active: ist eigentlich kein wirklicher Trigger; die Policy ist permanent aktiv,
- periodic: die Policy wird in einem festgeschriebenen periodischem Intervall aktiviert,

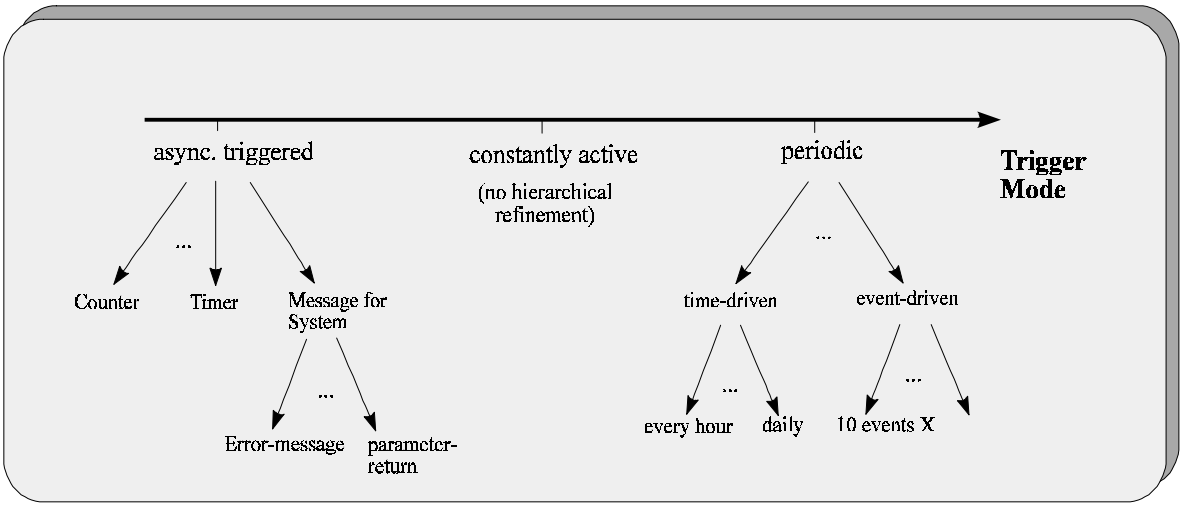

Abb.: 5

Ist eine Policy *constantly active* so benötigt sie eigentlich keine Aktivierung, da ihre activity ständig ausgeführt wird.
Wie in Abb.: 5 dargestellt, gibt es für diesen Punkt keine hierarchische Verfeinerung in der Klassifikation.

Policy12: *Die Netz-Discovery-Funktion (Netzmanagement-Funktion zur Überprüfung der am Netz beteiligten Geräte) läuft immer im Hintergrund der Netzaktivität ab.*

Policy13: *Auf dem Drucker X dürfen keine personenbezogene Listen gedruckt werden.*

Verfeinerungen in der Klassifikation zum Punkt *periodic* sind folgende Aktivierungen oder Deaktivierungen:

- time-driven: die Policy wird nach einem festen zeitlichen Plan periodisch aktiviert oder deaktiviert,

- event-driven: die Policy wird durch Auftreten bestimmter Ereignisse periodisch aktiviert oder deaktiviert.

Policy14: *Alle 20 Werktage ist der Benutzer aufzufordern, sein Paßwort zu ändern, da es in 5 Tagen ungültig wird.*

Policy15: *Der Fehlerschwellwert bei der Datenübertragung auf Line X ist alle 5 Minuten zu überprüfen und neu einzutragen.*

Policy16: *Die Datenbanken der Münchner Firmen werden von 17.00 bis 19.00 Uhr aus dem Online entfügt.*

Policy14, Policy15 und Policy16 sind *periodic time-driven* aktiviert. Man kann diese Policies nicht unter *async. triggered-Timer* einordnen, auch wenn dieses Kriterium ähnlich erscheint, denn es handelt sich hier eindeutig um eine *periodische* und keine *asynchrone* Aktivierung.

Folgende Policy17 (siehe Policy14) dagegen ist *periodic event-driven*:

Policy17: *Nach jedem zwanzigsten Einloggen in das System ist der Benutzer aufzufordern, sein Paßwort zu ändern, da es in 5 Tagen ungültig wird.*

Bei der Verfeinerung *event-driven* ist eine Parallele zum Punkt *Counter* festzustellen. Vorstellbar ist, daß ein Counter bestimmte Ereignisse mitzählt und bei einem Schwellwert die Policy auslöst.

In der Klassifikation wurden diese beiden Punkte dennoch unterschieden, da sich ihre übergeordneten Merkmale *async. triggered* und *periodic* grundsätzlich schon voneinander unterscheiden.

Der Auslöser *event-driven* ist hier so beschrieben, daß nach einer ganz festen Anzahl bestimmter Ereignisse der Trigger erfolgt. Dieses Merkmal ist eindeutig unter *periodic* einzuordnen, während *Counter* die asynchronen Ereignisse zählt.

Verfeinerungen in der Klassifikation zum Punkt *async. triggered* sind Aktivierungen oder Deaktivierungen mittels:

- Timer: die Policy wird durch einen Timer aktiviert,
- Counter: die Policy wird aktiviert, wenn der Counter X den Schwellwert erreicht hat, oder darüber liegt,
- Message for System: die Policy wird durch eine bestimmte (oder alle) Fehlermeldungen, oder durch einen bestimmten Systemparameter (-zustand) aktiviert.

Policies mit asynchronen Aktivierungen durch Counter, Timer oder Systemmeldungen (-zuständen) sind zum Beispiel:

Policy18: *Sind beim CRC-Verfahren mehr als X Bytes falsch, wird die Übertragung ab dem letzten START-Signal wiederholt.*

Bei Policy18 sind die asynchronen Ereignisse:

- Zustand: CRC-ON,
- Counter: X falsche Bytes.

Policy19: *Batchverarbeitung beginnt, nachdem die Onlinedatenbanken aus dem Onlinesystem entfügt sind.*

Auch Policy19 ist *async. triggered* (non-periodic event), denn sie wird aktiviert durch ein asynchrones Ereignis (Onlinedatenbanken werden entfügt).

Auch Abhängigkeiten zwischen Policies zählen zu asynchronen Ereignissen, die eine Trigger-Funktion ausführen können.

Policy20: *Erst muß PolicyA aktiviert sein, dann aktiviere Aktionen von PolicyB.*

Async. triggered-Counter:

Policy21: *Falls mehr als 10 FTP-Anwendungen gestartet sind, soll das Netz auf Durchsatz überprüft werden,*

Diese Policy ist nur aktiv, wenn 10 FTP-Anwendungen gleichzeitig gestartet sind.

Auch diese Policy ist nicht *periodic triggered*, da zwar 10 Events abgewartet werden (10 FTP-Anwendungen gestartet), diese jedoch nicht periodisch sondern unzusammenhängend auftreten, denn das Starten einer FTP-Anwendung ist nicht (zeitlich) abhängig von anderen FTP-Anwendungen.

Async. triggered-Timer:

Policy22: *An Kurzarbeitstagen wird nach 20.05 Uhr kein Eintrag mehr in die Zeiterfassungsdatei über das Kartenlesegerät genehmigt, außer der Mitarbeiter ist Schichtarbeiter.*

Die Abb.: 5 zeigt natürlich, ebenso wie alle folgenden Abbildungen einer Ebene, nur exemplarische Ausschnitte einer Verfeinerung in der Klassifikation. Die Kürzung in der Abbildung ist aber so gehalten, daß auf die vollständigen Verfeinerungen geschlossen werden kann.

Constraints:

Nicht in der Klassifikation vermerkt, existiert noch ein den Triggern verwandtes Klassifikations-Kriterium: die Constraints. Warum die Ebene der Constraints in die Klassifikation keinen Eingang gefunden hat, wird im Folgenden erklärt.

Constraints oder Einschränkungen können optional als ein Teil einer Policy definiert sein. Ein großer Teil der Policies aus dem Netz- und Systemmanagement beinhaltet Constraints. Constraints sind Einschränkungen, die erfüllt sein müssen, bevor die Policy einen Effekt hat, oder Aktionen auslöst.

Im Gegensatz zu Triggern werden Constraints erst dann überprüft, wenn die Policy aktiviert wird, also zum Beispiel nach einer Aktivierung durch einen Trigger.
Mögliche Constraints sind bestimmte Vorbedingungen, die erfüllt sein müssen, bis die Policy gültig ist:

Policy23: *Aktion A ist nur auszuführen, wenn Netz-Controll-Prozessor (NCP) neu geladen wird.*

Policy24: *Job X darf nur anlaufen, wenn genügend Hauptspeicher zur Verfügung steht.*

Policy25: *Wenn ein Node im Netz übermäßig verärgerndes Verhalten aufzeigt wird er exkommuniziert (wird er aus dem Netz entfernt) [FIDO91].*

Und Parameter-bezogene Einschränkungen:

Policy26: *Paßwörter dürfen nur 8 Ziffern mit einem Sonderzeichen haben.*

Policy27: *Ist der Modus einer Policy 'enforcement', muß die Autorisierung für das Subject auf das Target automatisch mitgeneriert werden.*

Policies können auch abhängig von ihren Parametern verschiedene Aktionen ausführen:

Policy28: *Die CPU-Auslastung (CPU=central processing unit) soll möglichst gleichmäßig auf einem hohen Niveau gehalten werden. Steigt die CPU-Auslastung über den Schwellwert x, sind alle Batchverarbeitungen auf 'Hold'-Status zu setzen.*

Bei Policy28 werden abhängig vom Schwellwert x verschiedene Aktionen ausgeführt.

Policy29: *Der Abteilungsleiter darf Reisen der Mitarbeiter bis zu 10.000.-DM genehmigen. Über diesen Betrag muß er sich mit dem Bereichsleiter absprechen.*

Policy30: *Sinkt die Empfangsstärke der D-Netz Mobilstation unter Schwellwert x ab, muß die Mobilstation automatisch auf die nächste Sendestation mit größter Empfangsstärke umschalten und sich dort anmelden.*

Es gibt auch noch zeitliche Constraints und geographische Einschränkungen.

Wie man sieht, könnte man alle diese Constraints auch geschickt über Trigger ausdrücken. Das ist auch der Grund, warum die Ebene der Constraints keinen Eingang in das Klassifikationsschema gefunden hat, hier aber doch erwähnt werden sollte. Zum Beispiel kann man Policy28 in zwei Policies trennen:

constantly active:	Die CPU-Auslastung soll möglichst auf einem hohen Niveau gehalten werden;
async. triggered -> parameter-return:	Steigt die CPU-Auslastung über Schwellwert x...

Der formale Unterschied besteht in der Zeit der Überprüfung. Trigger aktivieren oder deaktivieren eine Policy, während Constraints sie nach der Aktivierung einschränken, also eine Veränderung oder Einschränkung der *Activity* einer Policy darstellen.

2.5 Life-Time of Policy Object

Die einzelnen Policies werden wie Anfangs beschrieben in unserer Sichtweise eines Policy-Systems als einzelne Objekte betrachtet. Diese Objekte haben einen Lebenszyklus (Planung, Erstellung,...) und eine Lebensdauer.

Anhand des Aspekts *Life-Time of Policy Object* sollen die Policies in der Klassifikation und in der Praxis (Policy-Processing-System) nach Gültigkeit sortiert werden können.

In der Klassifikation wird nach dem Kriterium Lebensdauer unterschieden, da dies ein wichtiges Merkmal einer Policy ist, und manchmal sogar direkt im Text der Policy mitgegeben wird.

***Policy31**:* *Die PolicyX gilt solange, bis das neue Release der Software Y eingesetzt wird.*

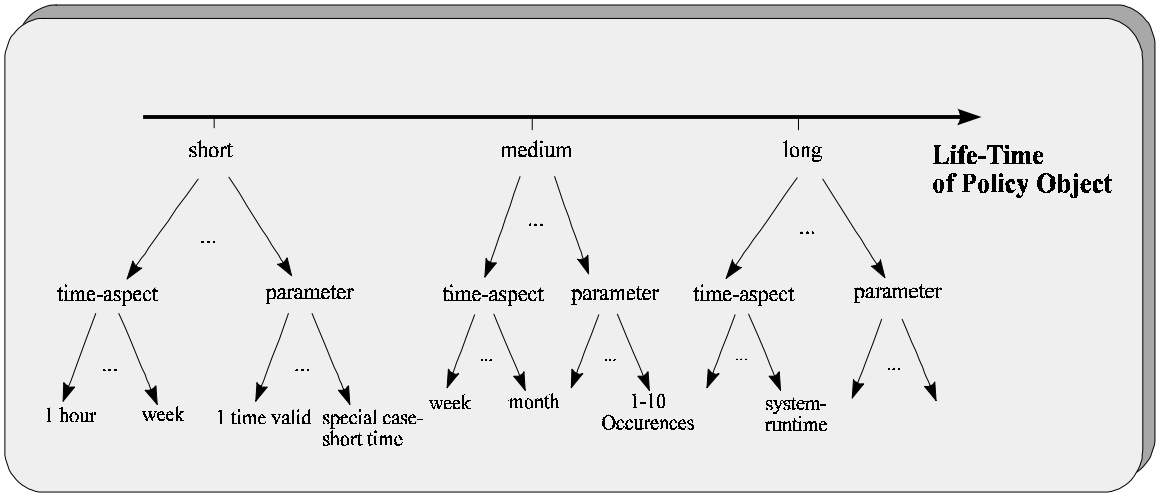

Abb.: 6

Wie in Abb.: 6 ersichtlich unterscheiden wir in drei ineinander fließende Gruppierungen von Lebenszeiten einer Policy.

Während *short* Life-Time Policies wegen des zu großen Aufwandes bei der Policy-Handhabung für ein oder wenige Auftreten, weniger vorkommen werden, ist die Klassifizierung bei *medium* und *long* öfter besetzt.

Beide Punkte beinhalten eine Verfeinerung *parameter*. Hier wird die Lebenszeit einer Policy zum Beispiel nach möglichen Auftreten von Ereignissen unterschieden. Bei der anderen Verfeinerung time-aspect wird nach zeitlichen Kriterien unterschieden, wie zum Beispiel 1 Woche oder die ganze Systemlaufzeit.

2.6 Subject-Ebene

Eine andere Form von Constraints in einer Policy sind sogenannte räumliche Constraints, die sich aus Einschränkungen der Subject- und Targetobjekte ergeben. Die Bezeichnung 'räumlich' bezieht sich nicht nur auf die geographischen Gegebenheiten (z.B.: im Raum Süddeutschland, etc.), sondern auch auf logische Räume (z.B.: Domänen oder Rollen).

In großen Systemen ist die Anzahl der zu adressierenden Objekte so groß, daß es unpraktisch wäre, Policies nur für einfache Objekte spezifizieren zu können, deshalb werden Policies meist für eine ganze Anzahl von Objekten spezifiziert und können natürlich auch nach diesen Gesichtspunkten klassifiziert werden.

Als Target oder als Subject einer Policy sollte es also möglich sein, Rollen bzw. Domänen ausdrücken zu können. Diese Rollen werden erst bei der Verfeinerung aufgelöst.
Die räumlichen Constraints stellen in der Klassifikation zwei gesonderte Ebenen dar, wobei in diesen Ebenen die Sparten oder Unterteilungen überlappend sein können und dürfen:

- Subject-Ebene und
- Type of Target-Ebene.

Als Subject bezeichnet man das die Policy ausführende Objekt oder die ausführende Einheit.
Der Ausführungs- oder Aktionsteil einer Policy stellt also eine Relation zwischen Ausführendem (Subject) und Ziel (Target) dar. Diese Beziehung muß nicht einfach sein. Schon bei den einfacheren Policies treten Mehrfach-Beziehungen auf:

Policy32: *Alle Abteilungsleiter dürfen die Directory-Dateien der in ihrer Abteilung Angestellten lesen.*

In den untersuchten Policies fanden sich viele Beispiele mit angegebenen Subject. Bei einigen der Policies wurde das Subject nicht direkt angegeben, bei manchen war gar keines erforderlich um das Ziel der Policy über die auszuführenden Aktionen zu erreichen.

Nach diesen drei Eigenschaften kann man Policies klassifizieren. Diese Klassifizierung ist sogar für den praktischen Ansatz einer formalen Beschreibungssprache sehr günstig, da dem Policy-Processing-System mitgeteilt wird, ob es das Subject der Policy entnehmen kann, oder es aus dem Aufgabenbereich und den Aktionen, die in der Policy beschrieben werden, herleiten muß.

In der Verfeinerung der Subject-Klassifiaktion findet man wie bei der Target-Ebene die Unterteilungen in *Employees*, *Software* und *Hardware.*

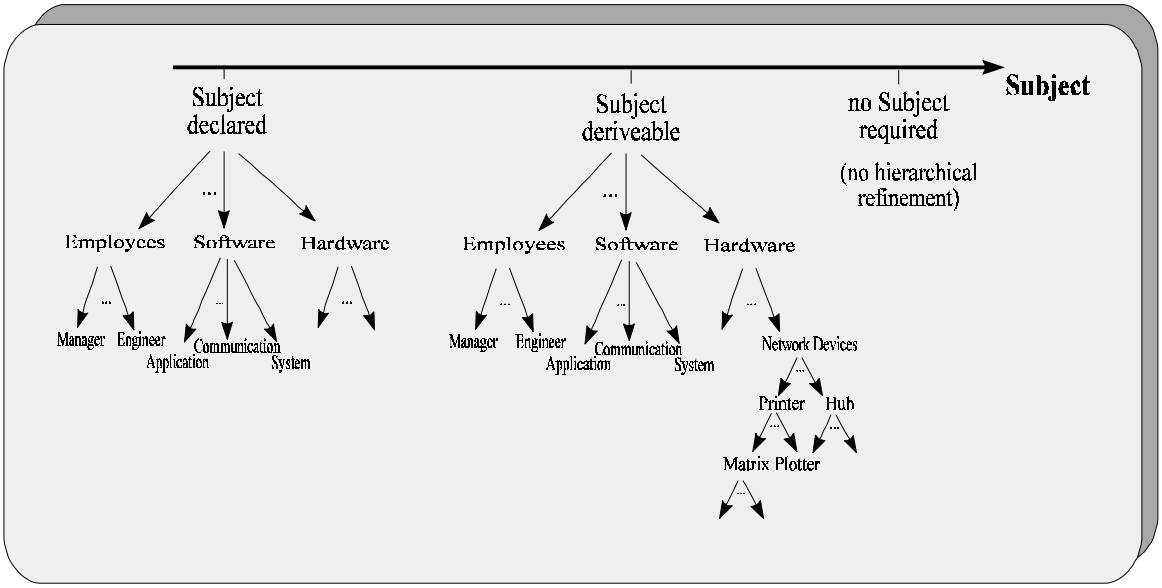

Abb.: 7

Bei dem Punkt *no Subject required* ergibt sich natürlich keine weitere Verfeinerung. Die anderen Unterteilungen lassen sich wie in Abb.: 7 ersichtlich beliebig weiter verfeinern.

- Employees: Angestellte als Subject der Policy können zu verschiedenen Gruppen oder Rollen gehören, wie zum Beispiel Manager, Ingenieure oder Operateure.

- Software: Software als Subject einer Policy kann sich gliedern in Applications-Software, Communications-Software und das eigentliche System selber.

- Hardware: Betrifft die Policy als Subject Hardware, kann man diese je nach Spezifität in der Policy verfeinern bis zu den einzelnen Grundkomponenten und Herstellerfirmen (-eigenarten).

Eine andere Form der Geltungseinschränkung tritt in manchen Policies als geographische Einschränkung für Subject und/oder Target auf:

Policy33: *Netzkomponenten im Werk München-Neuaubing sind vom Operating-Personal des Netzkontrollzentrums München-Allach zu warten.*

Dieser geographische Gesichtspunkt muß zwar in der Sprache Berücksichtigung finden, hat aber keine Bedeutung in der Klassifikation selber, da er sich aus den Ebenen *Subject*, *Target* und *Management Scenario* ableiten läßt.

2.7 Type of Target-Ebene

Das Target oder Ziel einer Policy ist in der Klassifikation in ähnlicher Weise wie das Subject unterteilt. Nur wird hier nach den verschiedenen Typklassen unterschieden.

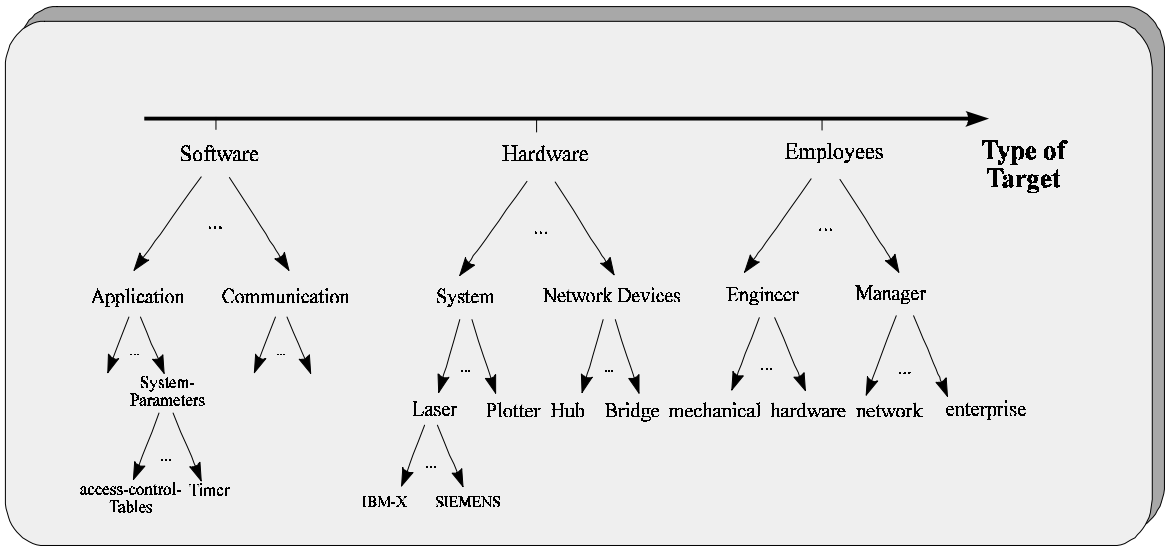

Abb.: 8

Auch hier findet man wie in der Subject-Ebene die Aufteilung in Software, Hardware und Employees und deren Verfeinerungen, jedoch ohne übergeordnete Punkte *deriveable* und *not required*.

Das Ziel oder Target einer Policy muß immer angegeben werden, denn sonst ist nicht sichergestellt, daß die Aktivität, die die Policy ausführen soll, Erfolg hat.

2.8 Management Scenario

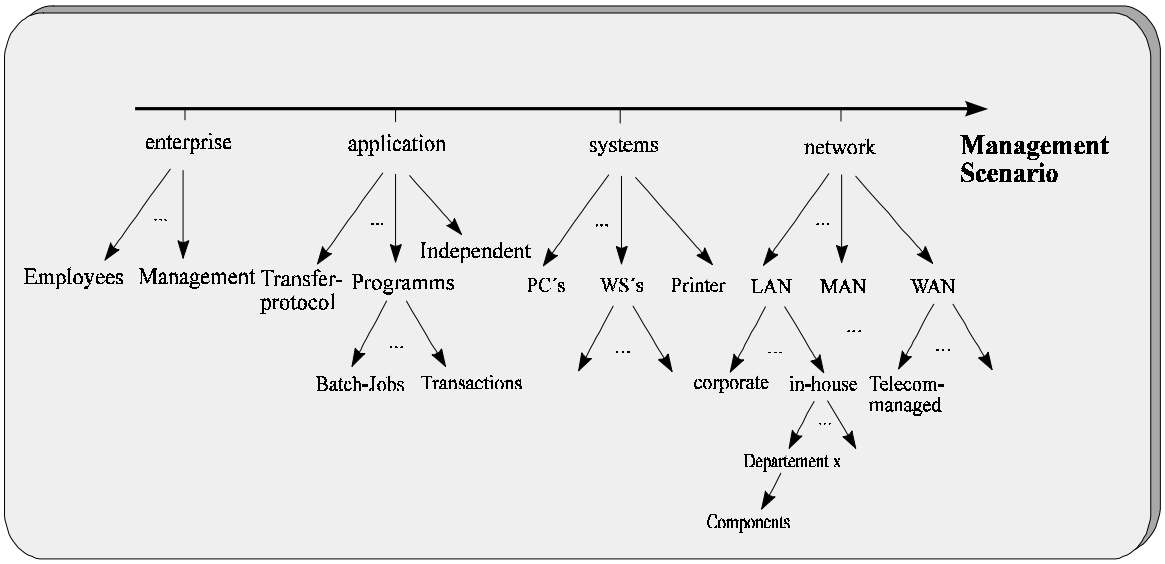

Abb.: 9

Policies lassen sich nach ihren Anwendungsgebieten und Zielobjekten, unterteilt in die Management-Szenarien, klassifizieren.
Vorgesehen in der Klassifikation ist die Unterteilung in:

- network-Management Policies, deren Aktivitäten zum Netzmanagement gehören,
- systems-Managment Policies, deren Aktivitäten zum Systemmanagement gehören,
- application-Management Policies, deren Aktivitäten zum Management für Anwendungen gehören,
- enterprise-Management Policies, deren Aktivitäten zum Enterprise-Management gehören.

Dies sind die in der Netz- und Systemmanagement-Welt gebräuchlichen Unterteilungen [HEG93].

Die Verfeinerung der Sparte network-Management sieht folgende Unterteilungen vor:

- Management des LAN (Local Area Network),
- MAN (Metropolitan Area Network) und
- WAN (Wide Area Network),

wobei zum Beispiel Letzteres sich wiederum unterteilen läßt in Bereiche des WAN´s, die (zum Beispiel in Deutschland) von der Telekom betreut sind und Teile, die vollständig von der eigenen Firma betreut werden. (Wobei das erst bei Wegfall des Monopols der Telekom an Bedeutung gewinnen wird). Die Zwischenlösung existiert natürlich auch. Das Übertragungsmedium wird (in Deutschland) von der Telekom bereitgestellt, die Technik der Übertragung ist dabei ja frei wählbar.

So kann ein Network-Management über und auf dem Übertragungsmedium von der jeweiligen Firma stattfinden.

Eine andere Möglichkeit das Network-Management zu unterteilen ist durch die Klassifikation nach den Funktionalitäten möglich. Dieser Aspekt wird aber schon in der Ebene *Management functionality of Policy-Activity* berücksichtigt.

Das LAN kann sich untergliedern in Haus-, Stockwerk-, Abteilungs- und Geländeweit.
Danach kann man das LAN-Management noch unterscheiden durch die verschiedenen Technologien und Topologien der einzelnen Netze (Token-Verfahren, Ethernet-CSMA/CD, Sterntopologie, logische Busstruktur, physische Busstruktur,...).

Die jeweiligen weiteren möglichen Verfeinerungen der anderen Sparten ergeben sich aus Abb.: 9.

Die meisten Policies sind durch diese Unterteilung nicht sauber voneinander abgrenzbar, da viele der Policies mehrere Szenarien durch ihre Subjects oder Targets gleichzeitig ansprechen können.

***Policy 34**:* *Wird die Benutzung eines Programms über das Netz angeboten, muß durch den Systemverwalter des Serversystems sichergestellt sein, daß aus Lizensierungsgründen nicht mehr als X Anwendungen gleichzeitig aktiv sein dürfen.*

Diese Policy spricht sowohl die Sparte des *network* (über Netz) an, sowie auch *systems* (Server), *application* (Anwendungen) und durch die Sicherstellung des Policy-Zieles über den Systemverwalter auch die Sparte *enterprise*.

2.9 Management functionality of Policy-Activity

Diese Dimension oder Ebene in der Klassifikation befaßt sich mit der Zuordnung der Policy-Aktivitäten zu bestimmten typischen Management-Bereichen [ISO 7498-4].

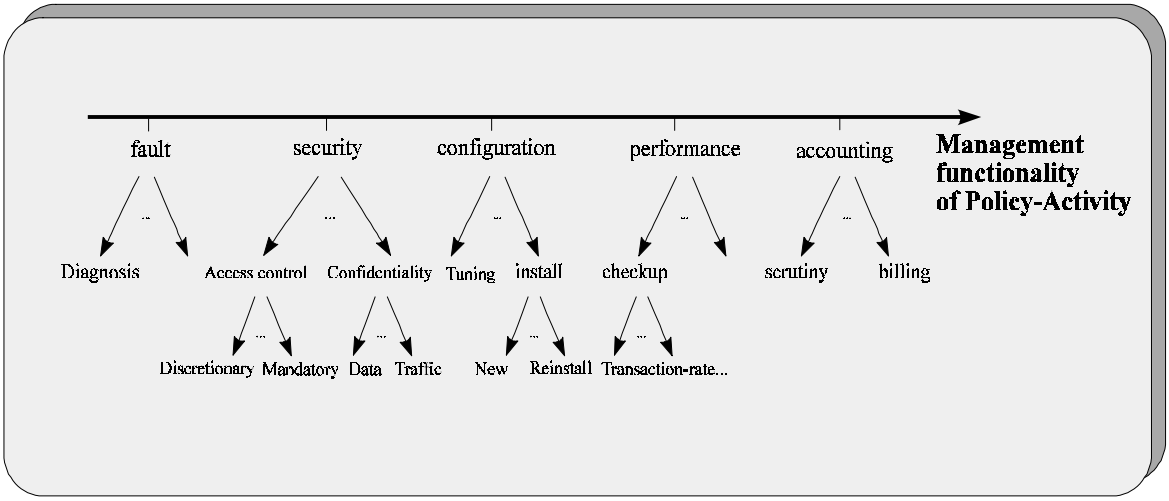

Abb.: 10

Die Policies werden in dieser Ebene unterteilt in:

- fault — also Policies, die sich mit Fehlererkennung und Fehlerdiagnose beschäftigen,
- security — Policies, die sich mit Sicherheit (auch Datensicherheit) im System und Vertraulichkeit (Zugangskontrolle) beschäftigen,
- configuration — Policies, die Konfigurationsaspekte beinhalten,
- performance — Policies, die eine Überwachung und Verbesserung der Performance und Leistung der Systeme gewährleisten, und schließlich
- accounting — Policies, die sich mit Abrechnungsmodalitäten und Kontierungsproblemen beschäftigen (auch das Mitzählen von Datenverkehr).

Die Untersuchung vieler Policies aus der Praxis haben ergeben, daß Policies mehrere Funktionalitäten in sich vereinen können. Daher ist Zuordnung nicht sauber trennbar, denn viele Kriterien überlappen sich. Beispiele für sich überlappende Management-Funktionalität in einer Policy wären:

Policy35: *Nur die Systemverwalter dürfen eine neue Kennung einrichten.*

Policy36: *Der Anwender der Gruppe X hat zu den Programmen Y Zugang.*

Policy37: *Falls unter einer Kennung X-mal mit falschem Paßwort Zugang der zum System versucht wird, ist diese Kennung zu revocen (zu sperren).*

Policy38: *Paßwörter müssen beim Daimler-Benz Konzernverbund mindestens 6, höchstens 8 Buchstabe und mindestens eine Ziffer beinhalten.*

Diese Policies betreffen zum Beispiel die Managementbereiche Konfiguration und Sicherheit.

Policy39: *Fällt in einem Ticket-Buchungssystem X die Standleitung zum Anwender aus, so sind die Modems für den Anwender innerhalb von 10 Minuten zugänglich zu machen.*

Besonderes Augenmerk verlangt auch die Sicherheit der Aktionen des Policy-Processing-Systems selber.
Es findet ein Zugriff auf die Netzaktivitäten statt (z.B.: Deaktivieren von Lines, abhören von Leitungsaktivität), der ja auch geschützt werden muß.
Diese Policy ist ein Beispiel für die Managementbereiche Fehler und Konfiguration.

Es gibt aber auch eine Reihe von Policies mit nur einer Management-Funktionalität.

Policy40: *Der Direktor der Abteilung x bekommt y Mega Byte disc quota (maximal verfügbarer Speicherplatz auf Plattenmedium) zugeteilt. Jede studentische Hilfskraft bekommt z Mega Byte disc quota.*

Eine Einteilung der Policies in funktionale Bereiche ist sinnvoll.
So wird diese Angabe etwa bei der Verfeinerung von Policies im *Policy-Processing-System* herangezogen werden, denn es besteht zum Beispiel ein Unterschied bei der Subject-Auflösung (z.B.: Policies mit Sicherheitskriterien), wenn der Managementbereich bekannt ist.

Andererseits kann eine Angabe auf dieser Ebene wichtig für automatische Abarbeitung des Policy-Codes sein, denn Teilbereiche von Managementplattformen können durch Benutzung schon vorhandener Funktionen und Tools die Aktionen der Policy ausführen. Aber diese Funktionen sind in den Managementplattformen nach den Managementbereichen getrennt und somit leichter zur Abarbeitung der *Policy-Activity* zu gebrauchen.

Die Entscheidungsfindung [BECKER93:p284], was ein Managementobjekt zu tun hat, um die Aktivitäten der Policy auszuführen, ist je höher der Abstraktionsgrad der Policy ist, um so schwieriger. Es kann durchaus passieren, daß die Lösung eines Management-Zieles in Konflikt mit anderen Lösungen anderer Ziele kommt (siehe auch 2.3 Policy Conflict-Resolution).

Eine Lösung zur Verminderung dieses Problems besteht unter anderem darin, eine passende Möglichkeit zu finden, die Managementzugehörigkeit in der formalen Sprache für Policies genau ausdrücken zu können.

2.10 Activity-Ebene

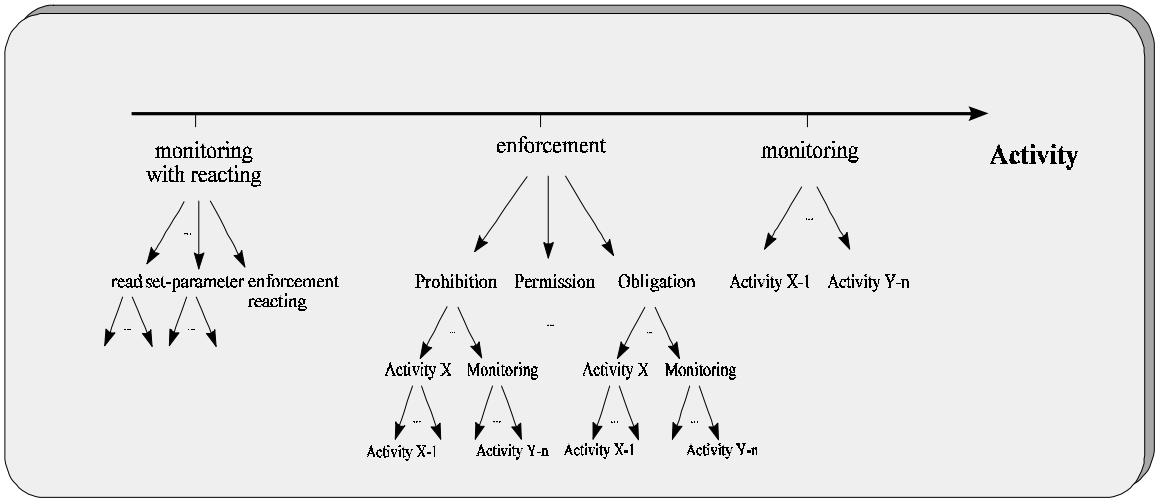

Abb.: 11

In der Activity-Ebene werden Policies nach den unterschiedlichen Aktivitäten, die auszuführen (oder zu unterlassen) sind, klassifiziert. Es werden hierbei drei Sparten unterschieden:

- monitoring,
- enforcement und
- monitoring with reacting.

In der Sparte monitoring werden die Policies eingeordnet, die eine reine Überwachungsfunktion haben.

Policy41: *Die Durchführung der Aktivitäten von Policy X-Y ist zu überwachen.*

Policy42: *Die Discovery-Function (Netzüberwachung) ist permanent im Hintergrund aktiv.*

Doch diese Art von Policies ist eher selten. Die meisten Policies überwachen etwas, werden durch einen Trigger (2.4 Trigger-Mode) aktiv und stoßen dann eine oder mehrere Managementaktionen an.

Beipiel-Policy zur Sparte monitoring with reacting:

Policy43: *Wenn der Anwender sein print-quota (zulässige Ausdrücke pro festgelegtem Zeitrahmen) überschreitet, wird ihm eine Warnmeldung gesendet und seine Druckberechtigung deaktiviert.*

Diese Policy überwacht einen Systemparameter (print-quota des Anwenders). Wenn dieser über den Schwellwert steigt, wird eine Aktion ausgelöst (Warnmeldung und Parameter "Druckberechtigung" wird auf NO gesetzt). Die Aktion des Entzugs der Druckberechtigung kann wiederum als ein enforcement der Policy angesehen werden.

In der Sparte enforcement unterscheide ich drei Modi:

- Prohibition — Policies, die Zugriff auf etwas verbieten (z.B.: durch Löschen eines Eintrages in Autorisierungstabellen),

- Permission — Policies, die Autorisierungen vergeben und Zugriff erlauben (z.B.: durch Eintrag in eine Autorisierungstabelle),

- Obligation — Policies, die das Subject veranlassen, etwas zu tun (siehe auch *motivation-Policies* [MOFF91:p11ff] und *imperatival-Policies* [MARR93].

Obligation-Policies können als Subject einen automatisierten oder menschlichen Manager haben.
Es kann, um das System etwas zu vereinfachen, angenommen werden, daß sich automatische Manager, sobald sie eine sich betreffende Policy bemerken und das System korrekt läuft, korrekt verhalten und versuchen werden, die Ziele der Policy mittels den vorgeschriebenen Aktionen zu erreichen [MOFF92:p10].
Wenn man davon ausgeht, daß menschliche Manager sich immer korrekt verhalten, würde man das System etwas zu sehr vereinfachen. Es muß also damit gerechnet werden, daß Obligation-Policies für Angestellte, aber bei ungünstigen Verhältnissen auch bei automatisierten Managern, ihr Ziel aus irgendeinem Grund nicht erreichen.

Es sollte also eine Überwachung stattfinden, ob die Ziele der durchgeführten Aktionen erreicht wurden. Diese sollte automatisch vom Policy-Processing-System mitgeneriert werden.
Man muß also auch und ganz besonders bei Policies, die Angestellte (Employees) ansprechen, dieses monitoring nach Aktionsabschluß durchführen, denn es nützt im Sinne

des Managementes nichts, wenn eine Policy definiert wird, die den Abteilungsleiter einer Firma dazu bringen soll, die Telefonrechnung seiner Angestellten zu überprüfen, man sich aber dann darauf nicht verlassen kann.
Das gleiche Problem ist, wenn ein Staat Gesetze erläßt, aber keine Executive aufweisen kann, die die Durchführung und Einhaltung garantiert.

Wie man nun die Durchführung und Einhaltung der Policies überwachen kann (zum Beispiel mittels SNMP-Abfragen bei Targetobjekten, oder durch SMF gesteuertes Management) ist wiederum eine sehr komplexe Frage und wird im Kapitel 3 der Diplomarbeit behandelt.

Für eine Prohibition oder Permission ist das monitoring nicht so wichtig. Hier kann man davon ausgehen, daß eine Autorisierung sinnvoll im Sinne des Managements derart definiert werden kann, daß der Anwender sie auch hat oder nicht hat. Man muß allerdings trotzdem die Aktionen die ausgeführt werden, um zum Beispiel die Autorisierung zu vergeben (z.B.: Eintrag in eine Tabelle) auf Korrektheit überprüfen.

Policies mit einer enforcement-Obligation sprechen eher das Subject an, während die anderen Modi für das Target einer Policy bestimmt sind. Hier darf man Subject und Target einer Policy nicht miteinander verwechseln. Mit dieser Betrachtung kann auch die *Policy-Distribution-Function* [SLOM93:p10] im *Policy-Processing-System* eine Verteilung der Policies zu den einzelnen Objekten oder Serversystemen vornehmen.

Wie in Abb.: 11 ersichtlich kann man eine rekursive Struktur zwischen den Sparten monitoring with reacting und enforcement feststellen. Die Reaktion auf ein monitoring kann sich durch eine Prohibition, Permission oder Obligation ausdrücken, deren Aktivitäten wiederum durch ein monitoring überprüft werden müssen.

Löscht man eine Domäne, so fällt auch für sie das monitoring weg und die entsprechenden Policies müssen dann entweder geändert werden oder neu generiert werden.

2.11 Bewertungskriterien für Klassifikationen

Untersucht man mögliche Klassifikationsschemata für Policies muß man sich auch eine "Schablone" zurechtlegen mittels derer man entscheiden kann, ob die Klassifikation für die verfolgte Intension (bei uns: formale Sprache für Policies) sinnvoll und korrekt ist.
Wichtige Kriterien zur Bewertung der untersuchten Klassifikationen waren:

- **Orthogonale Dimensionen:**

  stehen im mehrdimensionalen Klassifikationsschema die einzelnen Dimensionen wirklich semantisch unabhängig voneinander, oder haben sie Wechselwirkungen aufeinander oder Abhängigkeiten unter sich? Orthogonalität ist das Prinzip, unabhängige Funktionalität voneinander zu separieren.

- **Anzahl der Dimensionen:**

  wieviel Dimensionen werden durch die Klassifikation erfaßt und sind alle Ebenen wirklich für die verfolgte Intension sinnvoll? Diese Frage stellt sich besonders für das angegebene Klassifikationsschema, das aus vielen einzelnen Ebenen besteht. Es besteht aber keine Verpflichtung, eine Policy nach allen Kriterien einzuordnen. Man kann sie auch nur in den für den Anwendungsfall relevanten Ebenen klassifizieren. Deshalb ist es besser, viele verschiedene und sinnvolle Möglichkeiten der Klassifikation zu bieten, aus denen dann im Spezialfall ausgewählt werden kann.

- **Abgeschlossenheit der Klassifikation:**

  lassen sich alle Policies in das Klassifikationsschema einordnen (wichtiges Kriterium für formale Sprache entwickelt aus Klassifikation) oder gibt es nicht berücksichtigte Sonderfälle, die für ein effektives und effizientes Management mittels Policies trotzdem Eingang in die formale Sprache finden müssen?

- **Disjunktheit in den einzelnen Dimensionen:**

  findet in den einzelnen Dimensionen des Klassifikationsschemas eine semantische Überlappung der Sparten (Achsenbeschriftungen) statt? Kann eine Policy in einer Dimension mehrfach eingeordnet werden, und wie wirkt sich diese Überlappung in diesem Fall auf die Verarbeitbarkeit der formalen Sprache im *Policy-Processing-System* aus?

- **Abhängigkeiten zwischen den Policies geht nicht verloren:**

  durch Trennung und Einordnung der Policies in das Klassifikationsschema dürfen Abhängigkeiten und Ablaufordnungen der Policy-Objekte nicht verlorengehen. Das heißt: die formale Sprache, die aus der Klassifikation erarbeitet wird, muß eine Möglichkeit bieten, Abhängigkeiten zwischen den Policies ausdrücken zu können.

- **Ausgewogenheit:**

  findet eine ausgewogene Einteilung der Policies durch die Dimensionen und deren Sparten statt? Wird eine Dimension nie bei Policies berücksichtigt? Sind in einer Sparte einer Dimension extrem viele Policies einzuordnen, in den anderen keine?

- **Erweiterbarkeit:**

  Ist das Klassifikationsschema für neue, erst später in Betracht kommende Ansprüche erweiterbar?

Nachdem nun die Kriterien zur Bewertung einer Klassifikation aufgestellt sind, stellt sich die Frage, wie das oben vorgestellte Klassifikationsschema bewertet werden kann.

Das Klassifikationsschema besitzt wie in der Vorstellung erwähnt orthogonale Dimensionen, auch wenn inhaltliche Bezüge zwischen den Ebenen vorhanden sind.

Die Anzahl der Dimensionen überschreitet wegen der Allgemeinheit der Fragestellung die zu erwartende, sinnvolle und minimale Anzahl. Dafür wurde eine hohe Anwendbarkeit für viele Fragestellungen der Klassifikation erreicht. Falls einzelne Dimensionen für eine bestimmte Anwendung nicht in Fragte kommen, können diese auch völlig unberücksichtigt bleiben.
Trotz der hohen Anzahl der Dimensionen ist meiner Ansicht nach keine überflüssige Ebene im normalen Anwendungsfall vorhanden.

Die von uns untersuchten Policies ließen sich alle in das Klassifikationsschema einordnen, auch wenn manche Policies mehrere Punkte auf einer Ebene belegten. Dieses Problem läßt sich durch die Verfeinerung oder Trennung der Policy-Objekte lösen.

Eine Abhängigkeit zwischen den Policy-Objekten wurde in der Klassifikation nicht explizit berücksichtigt.
Dennoch kann man hierarchische Abhängigkeiten durch die Hierarchieebene, zeitliche Abhängigkeiten durch die Triggerebene und Prioritätsbeziehungen durch die Conflict-Resolution Ebene ausdrücken.

Ist die Einordnung der Policies in die einzelnen Dimensionen oder Sparten der Dimensionen nicht ausgewogen hat man die Möglichkeit, nicht belegte Ebenen ganz wegzulassen, oder für eine hohe Anwendbarkeit der Klassifikation, diese weiter zu berücksichtigen.

Das Klassifikationsschema ist jederzeit auf neue Anforderungen durch Erweiterung, also durch Hinzunahme neuer Ebenen oder Beschriftungen erweiterbar, wenn diese die Orthogonalität der Klassifikation nicht zerstören.

Die vorgestellte Klassifikation für Policies erfüllt also die meisten Kriterien und ist demnach als effektiv und gut anwendbar zu betrachten.

3. Formale Beschreibungssprache für Policies

3.1 Anforderungen an eine formale Beschreibungssprache für Policies

Klassifikationen und formale Beschreibungssprache für Policies haben wie oben (1.2) erwähnt einen festen Zusammenhang, unterscheiden sich jedoch in einigen wichtigen Merkmalen. Diese Unterschiede ergeben sich aus den verschiedenen Zielen beim Entwurf von Klassifikation und Sprache.

Eine Klassifikation für Policies sollte eine Gliederung und übersichtliche Einteilung von großen Policy-Objektkatalogen erlauben. Man kann anhand der Klassifikation eine Menge von Policies heraussuchen, die zum Beispiel einem bestimmten Typ von Target zugeordnet sind, eine bestimmte Tätigkeit ausführen, einem Management-Scenario zugeordnet sind oder eine hohe und gemischte Hierarchiestufe aufweisen.
Diese Eigenschaft kann zum Beispiel für das Lösen von Konflikten im Policy-Processing-System hilfreich sein.

Bis jetzt sind Policy-Kataloge von Unternehmen und Anbietern von Netzservices, wenn überhaupt, nur in informeller Weise beschrieben [VIRN91], [FIDO91]. Sie sind sortiert nach dem Verantwortlichkeitsbereich der einzelnen Systemoperatoren. Die Policies sind dabei in normalem Text aufgeschrieben und lassen mehrdeutige Begriffe wie "übermäßig verärgerndes Verhalten", "bleibt dem Ermessen überlassen" zu.

Das Ziel beim Design oder Entwurf einer formalen Beschreibungssprache ist die Policy in einer festgelegten Weise so ausdrücken zu können, daß ihr Goal durch Delegierung der *Activity* entweder in der Angestellten-Rangfolge ohne "Hilfe von oben" oder in der automatischen Verarbeitung erreichbar ist.

Das heißt insbesondere, daß sich die Activity und deren Ausführungsparameter wie Trigger etc. dafür in einer dem Subject verständlichen, delegierbaren oder ausführbaren Weise darstellen läßt.

Ich will im Folgenden besonders auf das Modell der automatischen Abarbeitung von Policies durch das Netz oder System selber eingehen. Hier stellen sich die Fragen:

- Wie kann man die verlangten Activities für das Ziel einer Policy beschreiben, damit sie auf vorhandene Management-Funktionalität (SMF, SNMP ...) anwendbar sind?

- Kann man eine high-level Policy durch einen hochentwickelten "Compiler" so umformen, daß in den hervorgebrachten low-level Objekten die Activity schon in geeigneter Weise ausgedrückt ist, oder muß die Person, die die Policies für das Netz oder System formal beschreibt und in das Policy-Processing-System speichert, bereits eine Umformung der Policy vornehmen und die Activity und deren Parameter in vorhandener Management-Funktionalität festlegen?

- Kann eine high-level Activity einer Policy dadurch abgearbeitet werden, daß sich die Hilfsmittelauswahl auf Tools und Werkzeuge höherer Abstraktionsgrade beschränkt?

Diese Fragen werden in Zukunft zunehmende Bedeutung gewinnen, da die automatische Transformation und Verfeinerung der wohl am schwierigsten zu lösende Teil des Policy-Processing-Systems ist.

Wichtig beim Design der formalen Beschreibungssprache ist nicht zu vergessen, daß auch Policies mit höherer Hierarchie durch die Sprache beschreibbar und ausdrückbar sein müssen. Das heißt, daß auch eine Verfeinerung in der Sprache, ebenso wie in der Klassifikation stattfinden muß, um zum Beispiel Domänen aufzulösen.

In den Ansatz zur formalen Beschreibungssprache für Policies sollen sprachliche Konzepte, die schon für verschiedene Arten von Policies vorhanden sind (z.B.: [MARR93], [WIES-ISNM94]) zu einem einheitlichen Konzept zusammengeführt und so gestaltet werden, daß sie für alle Arten und Hierarchien von Policies anwendbar sind.

Hierfür wurde ein sogenanntes Policy-Objekt kreiert, dessen Parameter verschieden belegbar sind.

Informationen, die durch die Klassifikation erhalten wurden, sollten natürlich durch die Sprache nicht verloren gehen, deshalb müssen beim Design der formalen Beschreibungssprache sämtliche Aspekte der Klassifikation, wie sie für die jeweilige Anwendung ausgesucht wurden, berücksichtigt werden und in Form von Parametern in verschiedener Weise in das Policy-Objekt einfließen.

3.2 Das Policy-Objekt und seine Parameter

Wie oben schon angesprochen, wird im Policy-Objekt, also im formalen Ansatz der Beschreibung von Policies der Unterschied zu der Klassifikation deutlich.

Zur Darstellung wurde ein Objekttyp mit zu besetzenden Parametern gewählt. Dieser ähnlich dem objektorientierten Ansatz hat den Vorteil, daß er sich auch bei neu auftretenden Anforderungen flexibel erweitern bzw. modifizieren läßt.

Die folgende Abbildung zeigt das Policy-Objekt. Danach werden die einzelnen Parameter informell erklärt:

POLICY OBJECT

Policy Name: (identifier)
Description of Goal: (free text)
Access-rights on Policy-Object: (Key of persons that are permissed to manipulate)
Creation Date/Valid Time: (dd/mm/yy/ LifeTime)
State of Refinement: (splitted, not yet refined, ...)
Relationship to Policies: (hierarchical with PolicyX, time-Relationship with PoliciyX)
Conflict-Resolution: (high-priority, normal-priority...)
Management Scenario of activity: (network, systems, application,...)
Activity: (detailed Description of Management-functionality, Tools / Actions) (s. 3.5)
Triggered by: (Trigger-Events)
Monitored by: (Domain of Monitor-Objects)
Target Domain: (specified Domain of Target-Objects)
Subject Domain: (specified Domain of Subject-Objects)

REGISTERED AS { ... }

Abb.: 12

Policy-Namen

Policies sollten aufgrund ihres Namens (identifier) eindeutig identifizierbar sein.
Daraus folgt, daß ein Name für eine Policy in einem zu managenden Verbundsystem nicht zweimal vergeben werden darf. Der Name einer Policy sollte mit der Klasse, von der die Policy instantiiert wurde, in Zusammenhang stehen. Die formale Sprache muß also die Möglichkeit geben, eindeutige Namen für Klassen von Policies und für Policies sowie deren Verfeinerung vergeben zu können, denn im Fall von Konflikten muß auch ein Rückbezug auf die ursprüngliche, noch nicht transformierte und übergeordnete Policy möglich sein, um bei der Auflösung des Konfliktes die Prioritäten oder Ziele der übergeordneten Policies überprüfen zu können.

Policy-Goal

Ebenso wichtig ist eine Möglichkeit zur Angabe des Zusammenhangs und des Ziels der Policy in normaler Textform, um bei einem späteren Bearbeiten der Policies deren Inhalt leichter zu verstehen.

Access-rights on Policy-Object

Da mit der Policy auch sicherheitsrelevante Aktionen angestoßen werden können, ist es unerläßlich zu definieren, wer Schreib- und Manipulationsrechte an der Policy selbst besitzt.
In diesem Attribut kann eine oder mehrere Domänen angegeben werden, mit ihren dazugehörigen Rechten auf dem Policy-Objekt.

Creation Date/Valid Time

In diesem Feld wird das Initiierungsdatum der Policy im Format dd/mm/yy angegeben. Auch die Gültigkeitsdauer muß angegeben werden. Dies kann entweder im Format hh/dd/mm/yy geschehen, oder Event abhängig gesteuert sein.
Ein Lebenszyklus (Gültigkeitszeitraum) der Policy kann auch extern von anderen Policies gesteuert werden.

State of Refinement

Da die Policy im Policy-Processing-System verarbeitet wird, ist es wichtig ihren Status am Objekt selbst erkennen zu können.
Dabei ist ersichtlich, ob die Policy anwendbar, noch zu splitten oder schon für ungültig erklärt wurde.
Die Eintragungen in diesem Attribut müssen teilweise durch das Policy-Processing-System selbst (zum Beispiel beim Transformationsvorgang) vorgenommen werden.
Nicht alle Policies sind zur Verfeinerung geeignet. (Zum Beispiel gewisse Policies für menschliche Manager oder Policy-Rules bzw. low-level Policies). Es muß also für diesen Fall eine Kennzeichnung möglich sein.

Im normalen Anwendungsfall können Policies, wie schon bei der Klassifikation gezeigt, in verschiedenen Abstraktionsebenen auftreten.
Also muß man in der Sprache die Möglichkeit haben, verschiedene Grade der Abstraktion von Policies ausdrücken zu können. Daraus folgt, daß eine Policy in der Sprache mehrere Kombinationen von Belegungen des Attributes *State of Refinement* haben kann.

Ein anderer Ansatz währe, die Policies nur in einer Abstraktionsebene zuzulassen, was eine Einschränkung in der Flexibilität bei der Definition von Policies bedeutet, aber dafür eine bessere Verarbeitung im Policy-Processing-System (Kombination von Verfeinerung/Transformation und durchführende Managementeinheit, die Policies verarbeitet, siehe 1.5), also in der Praxis, zur Folge hat.

Die Informationen auf ein einheitliches Abstraktionsniveau zu bringen und Konflikte zu erkennen und eventuell auch zu lösen sind die schwierigsten Aufgaben des Policy-Processing-Systems.

Die Unterscheidung, welcher Hierarchie- oder Abstraktionsebene die Policy angehört, kann entweder vom System generiert werden, zum Beispiel mit Unterstützung der vorher durchgeführten Klassifikation, oder muß von der eingebenden Person, gegebenenfalls in Intervention mit dem Policy-Processing-System, stattfinden.

Der Grad für die Auswahl der Abstraktionsebenen bezieht sich auch auf den Abstraktionsgrad der Policy-Activity, das heißt die Detailtreue in der Festlegung der Aktionen, die noch auszuführende Hilfsmittelauswahl, Verteilung der Aufgaben der Policies und Zuordnung der Targets (Policy-Ziele) und Subjects (Policy-Ausführende) sowie die Auflösung der Domänen, falls notwendig.

Zur Abarbeitung werden die Policies von einem Compiler/Verfeinerer in eine anwendbare Abstraktionsebene gewandelt (low-level Policies oder Policy-Code/Rules).
Das Endprodukt der Transformation sind nicht anwendbare Funktionen, sondern Spezifizierungen, wie vorhandene Management-Funktionalität anzuwenden ist [WIES-ISINM94].

Das heißt insbesondere, daß auch die Hierarchie der Policy eine Auswirkung auf die Hilfsmittelauswahl bei der Durchführung der Policy-Activity hat, und damit bei nicht gleichen Hilfsmitteln, daß die Policies nicht unbedingt auf eine gleiche Abstraktionsebene gebracht werden müssen.

Diese Policy-Rules oder Policy-Codes werden mit Hilfe von Managementaktionen auf die managed objects (MO´s) der Target-Objekte angewandt. Dies kann zum Beispiel mit Hilfe von SNMP (simple network management protocol), ODP (open distributed processing) oder SMF (systems management functions) geschehen (siehe [HEG93], [CASE93], [ISO 10164-X], [ISO 10746-1-3]).

Bei dem Policy-Target Management-Personal entspricht eine Verfeinerung auch einer Delegierung von Arbeiten [MOFF92:p6].
Man kann sagen, daß bei Management-Personal eine high-level Policy einen ("Top-")Manager bei der Arbeit anleitet. Dieser wiederum erreicht seine Ziele durch Verfeinerung der Policy zu low-level Policies, welche er den anderen Mitarbeitern tiefer in der Hierarchie zuteilt und so weiter [BECKER93].

Wann ist aber eine Transformation oder Verfeinerung der Policy im Anwendungsfall abgeschlossen?

Bei Policies, deren Aktivität automatisch generiert werden kann ist diese Frage einfach zu beantworten, nämlich dann, wenn sich das Ergebnis der Transformation auf vorhandene Managementfunktionalität anwenden läßt. Falls dieser Zustand nach X Schritten nicht erreicht wird, muß die Policy entweder neu definiert werden, oder mit menschlicher Hilfe verfeinert werden.

Ebenso wie in der Klassifikation bleibt zu Untersuchen, ob die Aktionen, die bei der Verfeinerung entstehen, das gewünschte Ergebnis der Ursprungspolicy auch abdecken und was die Verhaltensweise des Policy-Processing-Systems ist, wenn ein Objekt aus der Domäne gelöscht wird, für die eine Policy gültig ist.

Auch die Frage, wo der von dem "Verfeinerungsinterpreter/-compiler" erzeugte Abarbeitungscode der Policy zu halten ist (verteilt beim Agenten, in Autorisierungstabellen zentral...), besonders im Hinblick auf Konflikte, sollte berücksichtigt werden, gehört aber nicht mehr zum Thema der Diplomarbeit.

Relationship to Policies

In diesem Attribut ist eine Abhängigkeitsbeziehung zu anderen Policy-Objekten darzustellen. Abhängigkeiten können zeitlich, prioritätsbezogen bei gleichen Prioritäten, kausal oder hierarchisch sein.

Eine mögliche und gute Form, Abhängigkeiten zu formalisieren finden sich in einem regelbasierten Ansatz. Einschränkungen jeglicher Art kann man mit dem Wort "wenn" formalisieren:

WHEN Condition x DO action y,
oder *IF Condition x DO action y.*

Regelbasierte Systeme sind heute durchaus sinnvoll einzusetzen, wie etwa das Beispiel von XCON[1] der Firma Digital Equipment zeigt.

Conflict-Resolution

Wie in der Klassifikation erwähnt, ist es möglich der Policy eine Priorität mitzugeben, anhand derer bei Konflikten schnell und automatisch entschieden werden kann, ob die Policy Vorrang vor anderen hat.

Um Konflikte zu lösen, die einerseits durch konfliktäre Managementziele (z.B.: Performance vs. Security) sowie durch (halb-)automatische Verfeinerung und andererseits durch Versehen bei der Eingabe oder Manipulation von Policies entstehen können, ist es zweckmäßig, Prioritäten für Policies vergeben zu können, für den Fall, wenn sich der Konflikt nicht automatisch durch Rückbezug auf andere oder die noch nicht verfeinerten Policies lösen läßt.

Die bei den Klassifikationen vorgestellte Wertetabelle (Abb.: 4) zeigt an, welche Policy gültig ist bei einem Konflikt zweier Policies mit unterschiedlichen Prioritäten.
Je definierter und ausgefallener die Aktionen in der Policy sind, desto mehr Priorität sollte der Policy zugeteilt werden.

Entstehen Konflikte, die nicht automatisch gelöst werden können (z. B.: bei gleicher Prioritätsstufe, Rückbezug auf andere Policies löst nicht Konflikt), ist Intervenierung eines Managers von außen notwendig.

Will man Prioritäten nicht direkt in eine Sprache aufnehmen, kann man auch einen Ansatz wie in [MOFF91] wählen. Hier werden andere Policies definiert, die Prioritäten auf bestimmte Aktionen verteilen, um damit im Konfliktfall eine Lösung zu erreichen. Dieser Ansatz ist aber umständlicher, und bei Sonderfällen unhandlicher, als die Methode, den Policies direkt Prioritäten mitzugeben.

[1] XCON ist ein Regelbasiertes-Deduktionssystem zur Konfigurierung von Computersystemen. Einsatz und Entwicklung durch die Firma Digital Equipment. XCON beinhaltet ca. 20.000 Regeln und Fakten über mehrere Komponenten von Vax-Computern.

Noch weiter zum Thema *Conflict-Resolution* untersucht werden muß:

- Welche Prioritäten bei der Verfeinerung von Policies vererbt werden und wie Konflikte zwischen Policies mit gleichen Prioritäten (vielleicht automatisch) aufgelöst werden können.

- Was passiert, wenn die Aktionen einer Policy aktiviert wurden und im Zeitraum der Abarbeitung eine andere Policy mit gleicher oder höherer Priorität aktiviert wird, die gegenteilige Aktionen auszuführen hat? Wird die erste Policy abgebrochen, werden ihre ausgeführten Aktionen wie beim Transaktionsprinzip zurückgesetzt oder wird sie fertig abgearbeitet wie im Warteschlangenprinzip?

Management Scenario of activity

Anhand dieses Attributes kann im Policy-Processing-System entschieden werden, welchem Anwendungsgebiet die Policy und ihre Verfeinerungen zugeordnet sind.

Activity

Die auszuführende Activity der Policy ist in diesem Attribut detailliert zu beschreiben. Dabei soll Rücksicht auf die vorhandene Managementfunktionalität, auf die zurückgegriffen werden kann, genommen werden. (siehe auch 3.3-3.5)

Triggered by

Die Auslöser oder Trigger der Activity, wenn sie nicht schon in der Activity mitgegeben sind, sollen in diesem Attribut eingetragen werden.

Zum Beispiel kann ein Auslöser der Art: async. triggered-Timer, wie in Policy22:

Policy22: *An Kurzarbeitstagen wird nach 20.05 Uhr kein Eintrag mehr in die Zeiterfassungsdatei über das Kartenlesegerät genehmigt, außer der Mitarbeiter ist Schichtarbeiter.*

in der formalen Beschreibungssprache durch den Event-Service (EVS) von OSF/DME [OSF DME92] ausgedrückt werden.

Monitored by

Wenn die Objekte oder Flags bekannt sind, durch die die Aktivität der Policy überwacht werden kann, so sind sie in diesem Attribut mitanzugeben.

Es muß darauf hingewiesen werden, daß *Employees* als Target oder Subject einer Policy besondere Monitormechanismen verlangen, da ihre ausgeführten Aktivitäten der Policies nicht wie in den anderen Fällen automatisch überprüft werden können (siehe 2.10 Activity-Ebene).

Target Domain und Subject Domain

Wichtig ist um eine (halb-)Automatisierung erreichen zu können, die Eingabe von dem Target der Policy, die durch den Formalismus verbindlich gesichert sein muß (auch die Eingabe von 'any' kann erlaubt sein). Die erforderlichen Eintragungen für das Subject können aus der Klassifikation abgeleitet werden.

In großen Systemen ist die Anzahl der zu adressierenden Objekte so groß, daß es unpraktisch wäre, Policies nur für einfache Objekte spezifizieren zu können, deshalb werden Policies meist für eine ganze Anzahl von Objekten spezifiziert.

Um Mehrfach-Beziehungen und logische Zusammenhänge mehrerer Targets und Subjects in der Policy erfassen zu können, braucht man ein Domänen-Konzept. Eine Domäne steht stellvertretend für eine Anzahl oder Gruppierung von Objekten, die durch gemeinsame Eigenschaften selektierbar sind. Diese Objekte können nun wiederum eine logische Rollenstruktur darstellen (Alle Abteilungsleiter, Direktory-Dateien).

Rollen beschreiben funktionelle oder organisatorische Gruppierungen innerhalb eines Systems oder einer Organisation. (Siehe zu Rollen und Domänen auch [SLOM93], [ISO 10164-19]).

Als Target oder als Subject einer Policy sollte es also möglich sein, Rollen bzw. Domänen ausdrücken zu können. Diese Domänen werden erst bei der Verfeinerung aufgelöst.

Einem User wird z.B.: während der Login-Prozedur eine Rolle oder eine Anzahl von Rollen zugewiesen, die dann aufgrund aller Policies für diese Rollen gewisse Rechte und Pflichten folgern lassen.

3.3 SMF als Managementfunktionalität für Policy-Activities

Die Arbeit an der Normung von SMFs (Systems Management Function) [ISO 10164-X] ist von ISO (International Standardisation Organisation) noch nicht abgeschlossen. In der Standardisierung werden Funktionen vorgestellt, die für eine Management-Applikation in einer zentralen oder dezentralen Managementumgebung für den Zweck des Systemmanagements definiert wurden.

Die in der Klassifikation angesprochenen Managementfunktionalitäten (security, accounting, fault...) wurden auch für die Aufstellung und Einteilung von Netzmanagement-Hilfsfunktionen zu Grunde gelegt, denn die verschiedenen SMFs sind wiederum verschiedenen SMFAs (System Management Functional Areas) zugeordnet, die sich aus den ISO-Netzmanagement-Funktionsbereichen ableiten lassen.

SMFs sind relativ komplexe Funktionen. Dieser Nachteil wird dadurch ausgeglichen, daß sie eine maximal erreichbare Anwendbarkeit erzielen, und so viele Bereiche des Netz- und Systemmanagementes abdecken.

Die eigentliche Managementinformation bei SMFs wird in sogenannten Support Managed Objects gehalten, die sich aus der Oberklasse Support Managed Object Class ableiten. SMFs stützen sich bei ihren Aktivitäten auf MOs auf CMIS (Common Management Information Service [ISO 9596]) ab.

Die bei [ISO 10164-X] definierten SMFs und ihre Beschreibungen werden im folgenden als Erweiterung zu [HEGAB 93] erklärt.

Object Management Function

Mittels der Object Management Function sind bestimmte Aktivitäten auf den MOs und deren Attributen möglich. Die Aktivitäten umfassen das Erzeugen und Löschen von MOs in einem System, sowie das Ändern und Lesen von MO-Attributen.
Diese Funktion ist wichtig um eine einheitliche, vom Managementstandpunkt aus überschaubare und bearbeitbare Umgebung zu schaffen.

State Management Function

Zustände und deren Übergänge beobachten und manipulieren zu können ist ein wichtiger Teil des Netz- und Systemmanagements. Diese Funktion stellt Operationen zur Verfügung, um Zustände von MOs in einem System zu managen.

Attributes for representing Relationships

Diese Funktion definiert Attribute, um Beziehungen (z.B.: Backup relationships, Service relationships, peer relationships,...) von MOs, und Rollen in diesen Beziehungen (secundary, primary, owner, member) zu erstellen, darzustellen und zu manipulieren. Zur Manipulation werden eine Menge von Operationen bereitgestellt.

Alarm Reporting Function

Diese (generisch) definierte Funktion befaßt sich mit Alarmen, die durch genau klassifizierte Ereignisse und Ursachen angestoßen werden.
Die Ereignisse (Events) werden in Typklassen unterschieden (Communication alarm type, Quality of service alarm type, Environmental alarm type) welche sich wiederum in genauere Ursachenkataloge untergliedern lassen. Zum Beispiel untergliedert sich Environmental alarm type in:

- Enclosure Door open,
- Excessive Vibration,
- Fire detected,
- Heating/Ventilation/Cooling Problem detected, etc.

In dieser Funktion wird also nur eine Meldung oder Notification, aber keine Fehlerbehebung oder Test der MOs berücksichtigt. Sie arbeitet auf Grund von lokalen Ereignissen, die herstellerabhängig in diese Klassifikation zum Report eingepaßt werden müssen.

Event Report Management Function

Auch diese Funktion bearbeitet Events, die erst, nachdem sie lokal durch einen Event Forwarding Discriminator (EFD) gefiltert werden, selektiert und abgeschickt werden. Der EFD wiederum ist als ein MO zu betrachten, das über Managementaktivität manipuliert werden kann.

Log Control Function

Für viele Managementfunktionen (Alarm Reporting Function, Security Audit Trail Function, Accounting Metering Function) ist es wichtig, ihre Informationen über Events durch sogenannte Log-Objects zu erlangen. In dieser Definition wird ein generisches Modell für Log-Objects und deren Manipulation bereitgestellt.

Ein Log-Object besteht aus einzelnen Log-Records, die anfänglich durch ein MO als Notification erzeugt wurden, und durch ein sogenanntes Log-Preprocessing umgeformt und in das Log-Object eingetragen wurden.
Ein Log-Package besteht aus:

- einem Log-Identifier, der eindeutig eine Instanz eines Logs relativ zu dem übergeordneten MO definiert,
- einem administrative state und operational state, welche den Status des Logs darstellen sollen,
- eine Beschreibung des Typs der Information, die geloggt werden soll,
- das Verhalten, wenn das Log-Object seine maximale Kapazität erreicht,
- Anmerkungen, wann das Log-Object erzeugt, gelöscht oder modifiziert wurde.

Das Management von Log-Objecten teilt sich in zwei Units:

a) log control functional unit,

b) monitor log functional unit.

Security Alarm Reporting Function

Diese Funktion arbeitet ähnlich wie die Alarm Reporting Function, nur werden hier speziell Belange des Sicherheitsmanagements gefiltert. Diese sind nach ISO insbesondere:

- authentication,
- confidentiality,
- integrity,
- non-repudation,
- security policy,
- security service.

Als Alarmauslöser werden verschiedene Möglichkeiten in Attributen definiert:

- Duplicate Information,
- Authentication failure,
- Delayed Information, etc.

Der Typ des Alarms gliedert sich in

- Integrity Violation: Verletzung der Datenintegrität,
- Operational Violation: ein Indikator dafür, daß der verlangte Service wegen fehlender Autorisierung oder Fehlfunktion nicht zugänglich war,
- Physical Violation: eine physische Ressource wurde zerstört oder beschädigt. Die Art der Beschädigung läßt auf eine Sicherheitsverletzung schließen,
- Security service or mechanism Violation: ein Indikator dafür, daß eine Attacke auf einen Sicherheitsmechanismus oder eine Sicherheitsfunktion entdeckt wurde (z.B.: Anbringen einer Abhörvorrichtung),
- Time domain Violation: ein Indikator dafür, daß ein Ereignis in einer unerwarteten oder dafür verbotenen Zeit stattgefunden hat (z.B.: Zugriff auf Paßwortdatei außerhalb der Bürozeiten).

Security Audit Trail Function

Diese Funktion ist aus der Log Control Function abgeleitet, aber davon unabhängig. Sie verfeinert die oben besprochene Filterung auf das Ablegen und Sammeln nur sicherheitsrelevanter Ereignisse in Form von sogenannten Audits.

Objects and Attributes for Access Control

Als letzte Funktion speziell für das Sicherheitsmanagement, ist diese Funktion zusammen mit Security Alarm Reporting Function und Security Audit Trail Function auf die besonderen Anforderungen an Sicherheit beim Netz- und Systemmanagement hin entworfen worden.

Hier werden Operationen zum managen von Zugriffs- und Autorisierungsregeln definiert. Dadurch können die MOs in einem System gegen unzulässige Eingriffe und Zugriffe geschützt werden.

Zu diesem Zweck werden Access control policies definiert, die eine Anzahl von Regeln beinhalten. Nach ISO - und entgegen der obigen Auffassung von Policies - ist genau eine solche Policy einer Domäne von MOs zugeordnet.

Die Access control shemes der Policies beinhalten:

- Access control lists,
- Capabilities,
- context based shemes,
- security labeling.

Innerhalb einer solchen Sicherheitsdomäne ist nach ISO eine einzige Access control Policy gültig. Zur Entscheidung, ob ein access gewährt wird, benötigt die sogenannte Access decision function Informationen.

Eine Access control information beinhaltet:

- access control Regeln,
- die Identität des Initiators der Zutritt anfordert (access request),

- die Ziele (targets) wofür der Zutritt (access) beantragt wurde (das können MOs sein und unter anderem auch management informations),

- die zugelassenen Operationen auf den Zielen ,

- Information, die durch die Kontrolle des Zugriffs entsteht und damit im Zusammenhang stehende Information.

Accounting Metering Function

In einem Netz oder System ist es sowie dem Benutzer als auch dem Betreiber ein Bedürfnis, Daten sammeln und mitzählen zu können, um eine Abrechnungshandhabe zu besitzen. Die Informationserlangung ist in dieser Funktion spezifiziert.

Die Accounting Metering Function definiert ein einheitliches Beschreibungsschema für Abrechnungsdaten und spezifiziert die Funktionalität zur Erfassung und zum Austausch dieser Daten.
Dazu werden Accounting Meter Objects als Unterklasse von den schon oben erwähnten Support Managed Objects definiert.

Über das Accounting Meter werden die Aspekte der Kontrolle des Aufzeichnungsvorgangs auf einem MO und die Identifizierung bzw. Filterung der aufgezeichneten Daten abgedeckt. Bei einer Operation dieser SMF wird entweder das MO selbst angesprochen, wenn darin "accountable objects" definiert sind, oder das Kontrollobjekt des MOs.

Die Attribute des accounting meter control capabilities package sind:

- units of usage, unterstützt einerseits zeitgesteuerte, andererseits volumenabhängige Einheiten,

- recording triggers, definiert Ereignisse, die veranlassen, daß accounting Daten aufgezeichnet werden,

- reporting triggers, definiert Ereignisse, durch die accounting Daten weitergegeben werden,

- accountable objects reference list, ist ein read-only Attribut, das die Domäne der Objekte beschreibt, für die das accounting meter object Daten sammelt,

- data objects reference list, ist ebenso ein read-only Attribut, welches die Objekte angibt, für die "accounting control" unterstützt wird.

Zusammen mit den accounting meter actions ist diese SMF ein mächtiges Werkzeug, um den wichtigen Managementaspekt der Kontrolle der Abrechnung in einem System oder Netz abzudecken.

Workload Monitoring Function

Diese Funktion arbeitet eng mit den schon vorher vorgestellten Alarmfunktionen zusammen. Durch die Workload Monitoring Funktion können wählbare Schwellwerte überwacht und Alarme ausgelöst werden.

Test Management Function

Mit dieser SMF werden verschiedene Formate für Testaktivierung und -beendung, sowie Ergebnismeldung unterstützt. Ein Test kann durch verschiedene Trigger (Time, Event) ausgelöst werden. Es werden verschiedene Testarten unterstützt (Test request controlled parameters, test request uncontrolled parameters). Im Test-result Type werden die Ergebnisse zum Test Action Performer übermittelt.

Summarization Function

Dies ist eine Funktion für das Performance Management und beschäftigt sich mit der Messung von Durchsatz, Antwortzeiten, Verfügbarkeit oder Ressourcen-Auslastung.
Dabei werden Informationen zum Beispiel aus Log-Objekten eingeholt, verarbeitet und verbreitet.

Ein sogenanntes Summarization-Object enthält Attribute für die Einstellung des sogenannten Scanners, für den reporting process und für verschiedene Mechanismen um zu beobachtende Objekte zu selektieren.

Confidence and Diagnostic Test Categories

Die in der Test Management Function eingeführten Tests werden hier durch verschiedene Verfeinerungen erweitert.

Scheduling Function

Die Scheduling Function unterstützt externes und intern gesteuertes Scheduling von Aktivitäten.
Dabei können die Perioden oder Intervalle frei gewählt und kombiniert werden, in denen Aktionen gestartet oder beendet werden sollen (multiple weekly scheduling, periodic scheduling).
Durch diese SMF wird ein komplexes, automatisches Management möglich gemacht.

Management Knowledge Management Function

Große Systeme und heterogene Netze benötigen spezifisches Wissen über die Strukturen des Systems, um das Management erst möglich zu machen. Verschiedene Typen des Management-Wissens sind:

- Instance information: aus der MIB erhaltene Informationen, welche Objekte existieren.
- Repertoire: Information was das Managed System verarbeiten kann.
- Definition Information: zum Beispiel Template Definitionen für Klassen und Namengebung.

Software Management Function

Die Software Management Function unterstützt das Managementsystem durch das Verteilen von Software und das Management von Software im System.
Einerseits kann Software als reine Daten interpretiert werden, im anderen Fall werden die Ressourcen betrachtet, die die Software benutzt.

Die Definition beschreibt und unterstützt folgende Aktionen:

- Initiierung und Transfer von Software,
- Transferkontrolle von Software,
- Aktivierung (incl. Versionsupdate),
- Deaktivierung der Software,
- Software Gültigkeits- und Korrektheitsüberprüfung.

Die Software wird dabei als MO mit einem eigenem Lebenszyklus angesehen.

Management Domain and Management Policy Management Function

Diese Funktion erlaubt die Modularisierung der Management-activities durch Gruppierung und Policy Definition für Domänen. Die Policy Definition ist unterschiedlich zu der weiter vorne in dieser Arbeit vorgestellten, da hier einer Domäne eine Policy zugeordnet wird.

Die Management-Domäne ist wiederum als MO modelliert. In dieses MO können neue Ressourcen aufgenommen und herausgenommen werden.
Also unterstützt diese Defnition die Informationen im Zusammenhang mit Management-Domains, deren Repräsentation und die Operationen, die auf den MOs der Domänen ausgeführt werden können. Zudem wird das Verhalten der Management Policy Objects nach ISO festgelegt.

Alles in Allem kann man durch die Komplexität und dennoch gute Struktur der Systems Management Functions erkennen, daß damit ein gutes und brauchbares Werkzeug für das System- und Netzmanagement vorliegt, auf das auch zum Beispiel ein Policy-Processing-System mit den verschiedenen auszuführenden Aktionen aufsetzen könnte.

Die schon oben in diesem Kapitel angesprochenen SMFAs (System Management Functional Areas) nehmen eine funktionelle Unterteilung, also eine Klassifikation der SMFs vor, die sich verglichen mit der oben vorgestellten Klassifikation für Policies nur auf die Sparten der Ebene *Management functionality of Policy-Activity* bezieht.

So werden zum Beispiel die *Accounting Metering Function* und die *Summarization Function* der SMFA zugeteilt, die für Accounting-Management steht und die *Alarm Reporting Function*, sowie die *Test Management Function* der SMFA für Fault-Management.

Eine SMF kann auch mehreren SMFAs zugeteilt werden. Zum Beispiel wird die *Test Management Function* sicher auch für das Configuration Management notwendig sein.

Die Policies, die in der *Management Domain and Management Policy Management Function* definiert werden, gelten nur für festgelegte Domänen.
Diese Definition ist sehr einschränkend und starr. Policies sind in dieser Anschauungsweise nur eine Ablauffolge von Management-activities jeweiliger Domänen.

Nach unserer Definition sind Policies viel umfassender.
Aber ihre Activities können auf die jeweiligen SMFs abgebildet werden (siehe 3.5 Activity-Implementierung in SMF).
Die Auswahl der SMFs kann über die Klassifikation der Policies im Klassifikationsschema auf der Ebene *Management functionality of Policy-Activity* und der entsprechenden SMFAs stattfinden.

3.4 Die 5 Viewpoints bei ODP

Mit den Standards zu ODP (Open Distributed Processing) [ISO 10746-1-3] will ISO eine Festlegung erreichen, wie in einem verteiltem System, bei heterogenen Ressourcen von verschiedenen Organisationen, Informationsverarbeitung betrieben werden kann.
Dafür wird gerade der Versuch gemacht, ein Reference Model zu beschreiben, in dem eine Architektur gezeigt wird, in der die Verteilungs- und Zusammenarbeitstechniken von verschiedenen Systemen integriert werden können.

Zu diesem Zweck wird in der Festlegung zuerst der Begriff 'Distributed Processing' definiert.
Distributed Processing Systeme sind die Klasse von informationsverarbeitenden Systemen, bei denen diskrete Einheiten auf mehr als nur eine Lokation verteilt sind, oder wo es einen Grund gibt, eine Kommunikation zwischen den Komponenten zu unterstützen. Open Distributed Processing entspricht dem Distributed Processing welches nach den ODP Standards abläuft.

Ein System entwickelt sich aus folgenden Gründen verteilt:

- das System wird restrukturiert und im Netz angeboten, um es billiger und mehr flexibel bezogen auf neue Dienste zu machen,

- das System wird auf der Basis schon vorhandener Systeme umgeändert, um neue Anforderungen zu erfüllen,

- das System entwickelt sich heterogen aus Gründen des schnellen Wechsels der Technologien und Märkte.

Aber die Informationsverarbeitung muß die Möglichkeit bieten, mit Systemen anderer Organisationen oder Organisationseinheiten zusammenzuarbeiten. Zum Beispiel, wenn Joint-Ventures geschlossen werden. Das bedeutet, daß die Firmen Informationen aus dem anderen System bearbeiten können müssen.

Die Vorteile, die sich Firmen von einem verteilten System erwarten sind also:

- hohe Verfügbarkeit und Fehlertoleranz,

- Remote-Management der Ressourcen und verteilte und damit bessere Wartung der Informationen,

- die Möglichkeit, Ressourcen (Software...) zu teilen um damit Kosten zu senken,

- dezentrales und damit kostengünstigeres Management,

- Management by Delegation,
- durch geschlossene Informationsbereiche höhere Sicherheit,
- Zusammenfassung heterogener Systemwelten, etc.

Um ein Distributed System und dessen Vorteile nutzen zu können, ist es notwendig, einige Standards in Bezug auf Datenaustausch und Systemstruktur festzusetzen.

Die ODP-Standards definieren zum Beschreiben des verteilten Systems eine gemeinsame Sprache, die in verschiedene Gesichtspunkte (s.u.) unterteilt ist, und benutzen diese, um Funktionen festzulegen die die erforderlichen Aktivitäten übernehmen. Das Ergebnis der Standards kann dazu benutzt werden, um ein ODP-System zu implementieren.

Ein solches System benutzt Komponenten und Ressourcen, welche:

- heterogen sind,
- nebenläufig arbeiten können,
- physisch oder logisch verteilt sind,
- dynamisch und statisch auswechselbar sind, und
- die Fähigkeit haben, zusammenzuarbeiten.

Dabei wird Heterogenität in dem Standard wie folgt definiert:

- Komponenten-Heterogenität: ergibt sich aus Zusammenfügung verschiedener Firmen oder durch die Entwicklung der Technologien,
- Betriebsystem-Heterogenität: ergibt sich aus den verschiedenen Einsatzgebieten in der Organisation (Büro, Lagerverwaltung...),
- Computational-Heterogenität: ergibt sich aus der Benutzung von verschiedenen Programmiersprachen, oder Datenbank-Anwendungen,
- Autoritäts-Heterogenität: ergibt sich dort, wo Zusammenarbeit verschiedener Einheiten und Hierarchieebenen gefordert ist,
- Applikations-Heterogenität: ergibt sich dort, wo verschiedene Applikationen benutzt werden, um eine Arbeit zu machen. Zum Beispiel: Integrieren von Design und Produktion.

Um die Arbeit in der Standardisierung und das Ergebnis davon überschaubarer zu machen, wird die Diskussion über ODP auf Grund der hohen zu verarbeitenden Informationsmenge in fünf Gruppierungen organisiert und klassifiziert.

Diese fünf Anschauungsweisen werden bei ODP als ***Viewpoints*** bezeichnet.
Jeder dieser Viewpoints hat seine eigene zugehörige Sprache, welche die Regeln dieses Anschauungsstandpunktes ausdrückt.

Ebenso, wie in der oben beschriebenen Klassifikation für Policies, sind diese Viewpoints nicht ganz voneinander unabhängig; aber jeder gibt einen partiellen, auschnittweisen Blick auf das Gesamtsystem.

Einige Ausschnitte des Systems treten in mehr als nur einem Viewpoint auf. Die Zusammenhänge der Viewpoints sind in sogenannten Reference-Points beschrieben, welche auch graphisch darstellbar sind.

Die fünf in dem Reference Model für ODP definierten Viewpoints sind:

- **computational Viewpoint:**

  Aus der Sicht des computational Viewpoint kann ein ODP-System als eine Menge interagierender Objekte gesehen werden, welche gewisse spezifische Funktionen unterstützen. Dabei wird nicht betrachtet, wie das vonstatten geht.

  Dazu gibt es eine sogenannte Infrastruktur, welche eine Umgebung zur Verfügung stellt, die die transparente Verteilung der Interaktionen unterstützt.
  In der computational Sprache werden die Objekte eines ODP Systems und die darauf stattfindenden Aktivitäten und Interaktionen definiert.
  Dafür werden sogenannte transactional properties festgelegt, die für die Aktivitäten gelten (visibility, consistency, recoverability).

  Die computational Sprache bedient sich eines remote procedure call oder eines kontinuierlichen Datenflußes zwischen den Objekten und verschiedener Programmiertechniken.
  Dabei wird in dieser Sichtweise der aktuelle Grad der Verteiltheit einer Applikation der Sicht des Programmierers entzogen. Aus diesem Grund kann die Konfiguration und der Grad der Verteiltheit der Hardware, auf denen ODP läuft ohne Probleme, also auch ohne Wechsel der Software verändert werden.

- **enterprise Viewpoint:**

  Der enterprise Viewpoint beschäftigt sich nicht nur mit einer einzelnen Organisation, wie der Ausdruck vielleicht suggeriert. Hier werden besonders Management-Policies, Geschäftsstrategien und Angestellten-Rollen in Hinsicht auf ihren Zusammenhang mit dem verteilten Systemen betrachtet. Eine Policy kann in Form einer Prohibition oder Permission ausgesprochen werden. Mit der enterprise Sprache können Regeln wie :

  resource usage rules,
  domain rules,
  conflict resolution rules,
  organization rules,
  business rules,
  transfer rules,
  security rules, etc. festgelegt werden.

  Der enterprise Viewpoint eines ODP-Systems zeigt an, was für Funktionen unterstützt werden, aber nicht, wie diese implementiert werden können. Deshalb ist dieser Viewpoint auf einer abstrakten Ebene, um die Belange der Firma nach außen darzustellen. Es werden aber nur die Funktionen und Transitionen betrachtet, die sich mit der enterprise Sicht beschäftigen.

- **information Viewpoint:**

  Ebenso wie der enterprise Viewpoint setzt der information Viewpoint Festlegungen, die sich nicht mit dem Verteilungsprozeß der Information direkt beschäftigen.
  Manche Elemente des enterprise Viewpoints treten auch im information Viewpoint auf und umgekehrt. Zum Beispiel kann eine Aktivität aus dem enterprise Viewpoint im information Viewpoint als Beschreibung einiger Prozesse welche eine Zustandsänderung einer Informationseinheit (z.B.: MO) hervorrufen, gesehen werden.
  Ein ODP System ist vom information Viewpoint aus ein formales System, dessen Verhalten genau definiert ist durch die Aktivitäten und Transitionen von der Inbetriebnahme bis zur Gegenwart.
  Das System wird modelliert durch Informations-Objekte und deren Beziehungen wie in einem Entity-Relationship-Modell, wobei die Informationsobjekte Abstraktionen der realen Ressourcen darstellen. Eine Informations-Spezifizierung ist ein Schema, welches alle möglichen Aktionen, Konzeptklassen und Beziehungsmöglichkeiten zwischen den Objekten beschreibt.

- **engineering Viewpoint:**

  Die letzten beiden Viewpoints beschreiben die Umgebung, in der die Beschreibung, die im computational Viewpoint festgelegt wurde, interpretiert werden kann. Dabei ist der engineering Viewpoint mit der Definition der Verteilungsmechanismen und deren benötigte Infrastruktur beschäftigt.
  Dafür muß eine Abbildung der computational objects auf die engineering functions stattfinden.

  Die engineering Sprache wird benützt um:

  die Organisierung einer abstrakten Infrastruktur zu beschreiben, die die Ausführung der Funktionen eines ODP Systems möglich macht,

  die Abstraktionen zu identifizieren, die erforderlich sind, um physisch verteilte und lokale Systemressourcen zu managen,

  die Rolle der verschiedenen Objekte zu definieren, die ODP Funktionen unterstützen, und

  um die sogenannten reference points zwischen den verschiedenen Objekten zu beschreiben.

  Der Informationsfluß zwischen mit ODP modellierten Systemkomponenten kann durch mehr als nur einen reference point laufen. So kann zum Beispiel eine computational Viewpoint Beschreibung eine Interaktion zwischen zwei Komponenten A und B aufrufen, aber die Kommunikation zwischen den beiden kann durch mehrere Interface gehen.

  Im engineering Viewpoint des gleichen Schemas können die Komponenten mehrere Interfaces auf dem Kommunikationspfad aufweisen.

- **technology Viewpoint:**

  Der technology Viewpoint beschäftigt sich mit den Details bei den Komponenten und Verbindungen und mit der Wahl der technischen Hilfsmittel um das System zu unterstützen.

  In einem ODP System sind die einzelnen Objekte selbst verantwortlich für ihr eigenes Management. Natürlich können sie von außen veranlaßt werden, ihre Verhaltensweise zu ändern. Dazu ist sicherzustellen, ob die Objekte die geforderten Managementaktivitäten auch ausführen.

Die zum technology Viewpoint zugehörige Sprache spezifiziert die genaue Art und Weise, wie die Spezifikationen eines ODP Systems implementiert sind.
Sie ist beschränkt durch die Kosten die entstehen, wenn man sie in Hardware oder Software Komponenten implementiert.

Diese Einteilung in Viewpoints nimmt intuitiv eine Klassifikation der einzelnen Aktivitäten in einem System vor.

Man kann den Ansatz zur Klassifikation mit dem oben für Policies vorgestellten Klassifikationsschema vergleichen. Dies wird im Folgenden versucht:

- Der computational Viewpoint von ODP könnte gesehen werden als eine abstrakte Verbindung (*high-level*) zwischen den Ebenen *Type of Target* und *Subject*. Unabhängig vom *Management-Scenario* sind die Aktivitäten und deren Funktionalitäten nicht einzuordnen.

- Der enterprise Viewpoint beschäftigt sich vor allem mit dem *Management-Scenario* enterprise. Die Aktivitäten oder Vorgaben der Policies können je nach Fall in die *Activity*-Ebene (Prohibition, Permission) übernommen werden. Die jeweiligen Rules (conflict resolution, security...) bestimmen, welche *Management functionality* zu wählen ist. Auch in dieser Sichtweise wird der Grad der Abstraktion (*high-level*) nicht gesenkt.

- Im informational Viewpoint werden Zustandsübergange und Beziehungen der Informationseinheiten betrachtet. Diese Informationseinheiten können wie bei uns als Domänen der Quelle (*subject*) und der Senke (*target*) betrachtet werden. Die Aktivitäten entsprechen nur Statusänderungen, was beim Klassifikationsschema für Policies nicht eingeordnet werden kann, da hier die Aktivitäten auf den Informationseinheiten als Handlungen betrachtet werden. Wenn man weiter abstrahiert, kann man natürlich auch dieses als Statusänderung nämlich der entsprechenden repräsentierenden Flags der MOs betrachten. Nur bei Angestellten als Subject und Target ist diese Betrachtungsweise nicht möglich.

- Der engineering Viewpoint beschreibt die Funktionen im ODP auf einem geringeren Abstraktionsgrad (*medium*). Vor allem die Organisierung des Systems oder Netz wird in dieser Sichtweise hervorgehoben. Je nach Fall also in die Ebene Management-Scenario einzuordnen.

- Im technology Viewpoint werden die Details der Komponenten (*Subject, Type of Target*) unabhängig von der *Management-functionality* beschrieben. Diese Beschreibung muß extrem *low-level* stattfinden. So wird eine Gruppe oder eine Einheit genau technisch beschrieben. Dabei ist unberücksichtigt, zu was für einer Organisationseinheit oder Funktionalität diese Komponenten gehören.

Umgekehrt betrachtet, ist die gesamte Klassifikation für Policies nicht mit den Viewpoints von ODP austauschbar, da viel mehr Einzelheiten klassifiziert werden müssen, und sich die Policies, für die das Klassifikationsschema gemacht wurde, nicht nur auf Informationsverteilung in verteilten Systemen beziehen.
Einzelne Ebenen der Klassifikation aber kann man mit verschiedenen Viewpoints vergleichen.

So werden die Ebenen Type of Target und die Verfeinerung von der Subject-Ebene dem informational Viewpoint zuzuordnen sein, wenn man sie zusammen, also ihre Beziehung betrachtet. Die Verfeinerungen jedoch können mit dem technology Viewpoint, der die Betrachtungsweise ja auf Details lenkt, verglichen werden.

Die Ebene Activity und Management functionality of Policy Activity und Management Scenario könnten dem enterprise Viewpoint zugeordnet sein. Hier werden auch die Rechte und Pflichten der einzelnen Komponenten und Personal einer oder mehrerer Firmen im Verbund betrachtet.

Dem engineering Viewpoint sind die Ebenen Life-Time, die geographische Verfeinerung der Ebenen Subject und Target, sowie die Verfeinerungen der enterprise-Sparte in der Management-Scenario Ebene zuzuordnen. Der engineering Viewpoint selektiert das Interesse auf einem niedrigen Abstraktionsniveau, deshalb sind auch nur die Verfeinerungen der genannten Ebenen vergleichbar.

Wie man sieht, muß man auch bei der umgekehrten Zuordnung die Abstraktionsebenen vermischen um gegenseitige Referenzen zwischen den beiden Klassifikationen, Viewpoints von ODP und meinem Klassifikationsschema aufstellen zu können.

3.5 Activity-Implementierung in SMF

Wie vorher beschrieben, kann sich eine Implementierung einer Policy auf vorhandene Managementfunktionalität abstützen.

Policy44: *meldet eine Netzkomponente der Domäne X Überhitzung ist sie sofort zu deaktivieren. Nach 10 Minuten ist sie wieder zu aktivieren und auf ihre Funktionstüchtigkeit zu überprüfen. Bei Korrektheit wird die Accounting-Function der Netzkomponente gestartet.*

Diese vereinfachte Policy soll als Beispiel für eine in SMF [ISO 10164 - X] gehaltene Implementierung der *Activity* einer Policy stehen.

Vereinfacht ist die obige Policy aus folgenden Gründen:

- die Netzkomponente wird deaktiviert, ohne Rücksicht auf Verkehrsfluß und Verkehrs-Umleitungsmöglichkeiten,
- es kann sein, daß die Komponente beim Policy-Aktivierungszeitpunkt bereits inaktiv ist, und nicht mehr reagiert,
- der Grund oder Auslöser der Überhitzung muß schnell vergehen (10 Minuten),
- nach der erneuten Aktivierung wird nur beispielhaft die Accounting-Function der Netzkomponente gestartet.

Trotzdem ist diese Policy ein gutes Beispiel um die Wirkungsweise einer in zu SMF ähnlich gehaltenen Notation der Policy-Activity zu zeigen.

Der Ablauf der Protokolle bei diesem Beispiel aussehen.

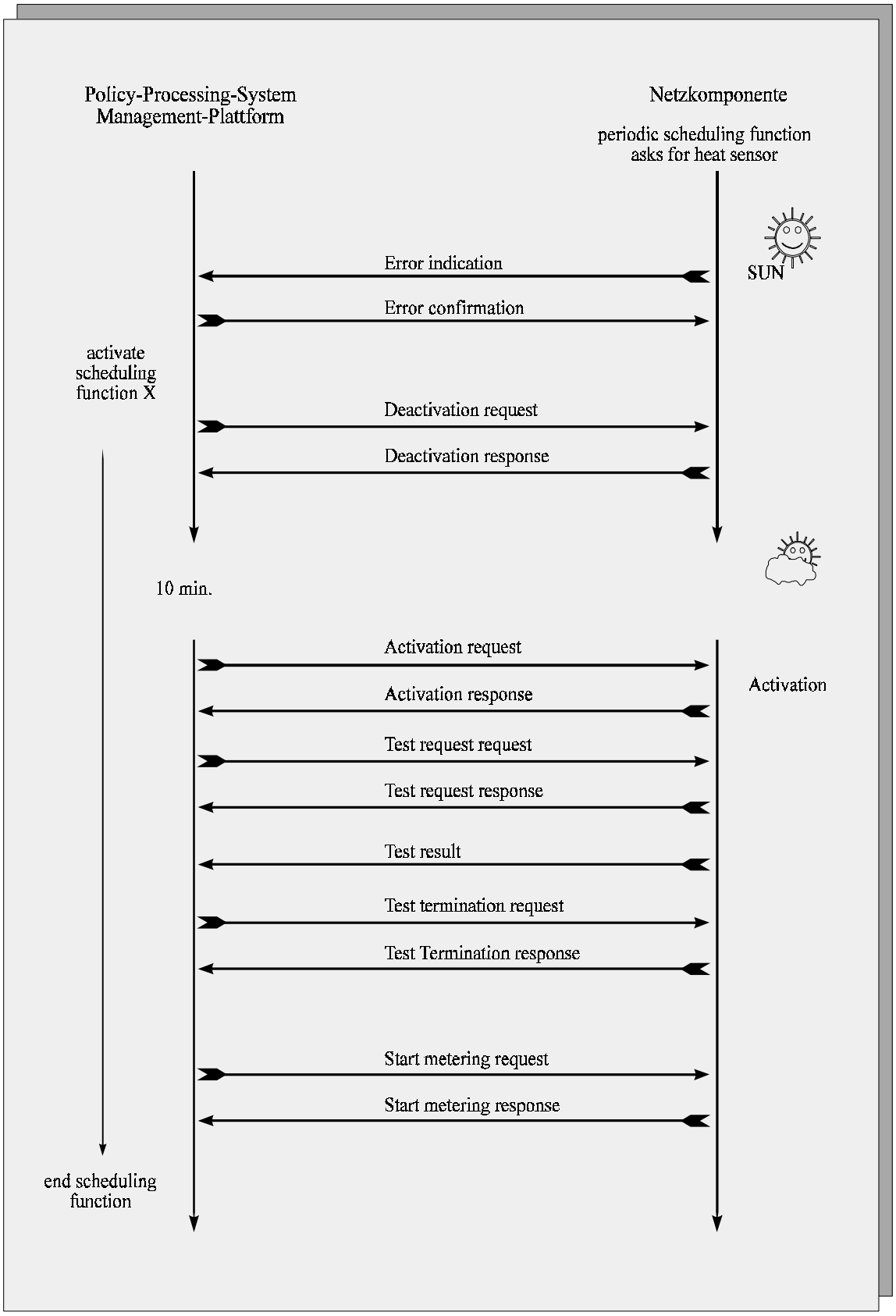
Policy-Processing-System
Management-Plattform
Netzkomponente
periodic scheduling function
asks for heat sensor
SUN
Error indication
Error confirmation
activate
scheduling
function X
Deactivation request
Deactivation response
10 min.
Activation request
Activation response
Activation
Test request request
Test request response
Test result
Test termination request
Test Termination response
Start metering request
Start metering response
end scheduling
function

Abb.: 13

Andere Lösungen, die Policy-Activity mitels SMF´s auszuführen bestehen auch. So kann man für manche SMF´s (z.B.: test management function [ISO 10164-12]) eigene Attribute zur Zeitsteuerung einsetzen, oder die alarm reporting function [ISO 10164-4] übernimmt die Warnung zur Überhitzung, und damit den obigen Ablauf verändern, bzw. vereinfachen. Zur Demonstration wird in diesem Beispiel der Ablauf von oben ausgeführt.

Sensortest: Abstützung auf M-EVENT-REPORT

TestActionPerformer...
DERIVED FROM...
CHARACTERIZED BY....
PACKAGES
ControlledTestRequest:

PARAMETER:
TestRequestControlledType:
TestRequestControlledInfo:
ControlledTestRequestType:
TestObjectList:
TOClass:
Filter:

TestObject...
DERIVED FROM...
CHARACTERIZED BY....
PACKAGES
ControlledTestResponse:

PARAMETER:
CourrentTime:
TestInvokationId:
TestObjectResponseList:

TestObject:
TestResultIndication:

PARAMETER:
InvokeId:
TestResultType:
EventTime:
MonitoredAttributes:
TestOutcome:

TestActionPerformer:
TestResultResponse:

PARAMETER:
InvokeId:
TestResultType:
CourrentTime:

4. Zusammenfassung und Ausblick

4.1 Anwendung der Klassifikation an einem Beispiel

Zur Verdeutlichung, wie die vorgestellte Klassifikation für Policies zu verwenden ist, wird auf eine schon in Kap. 2.2 erklärte Policy zurückgegriffen.

Policy5: *An einem Ticket-Buchungssystem teilnehmende Reisebüros sind über eine Standleitung angeschlossen. Im Falle eines Netzfehlers haben die Reisebüros die Möglichkeit, sich über eine Wählverbindung einzuloggen und müssen sich dann über einen Identifikationsvorgang authentisieren.*

Diese Policy ist als high-level einzuordnen. In der Klassifikation werden ihr gewisse Punkte oder Punktmengen auf den Dimensionen zugewiesen. Als Graphik ist das in Abb.: 14 dargestellt:

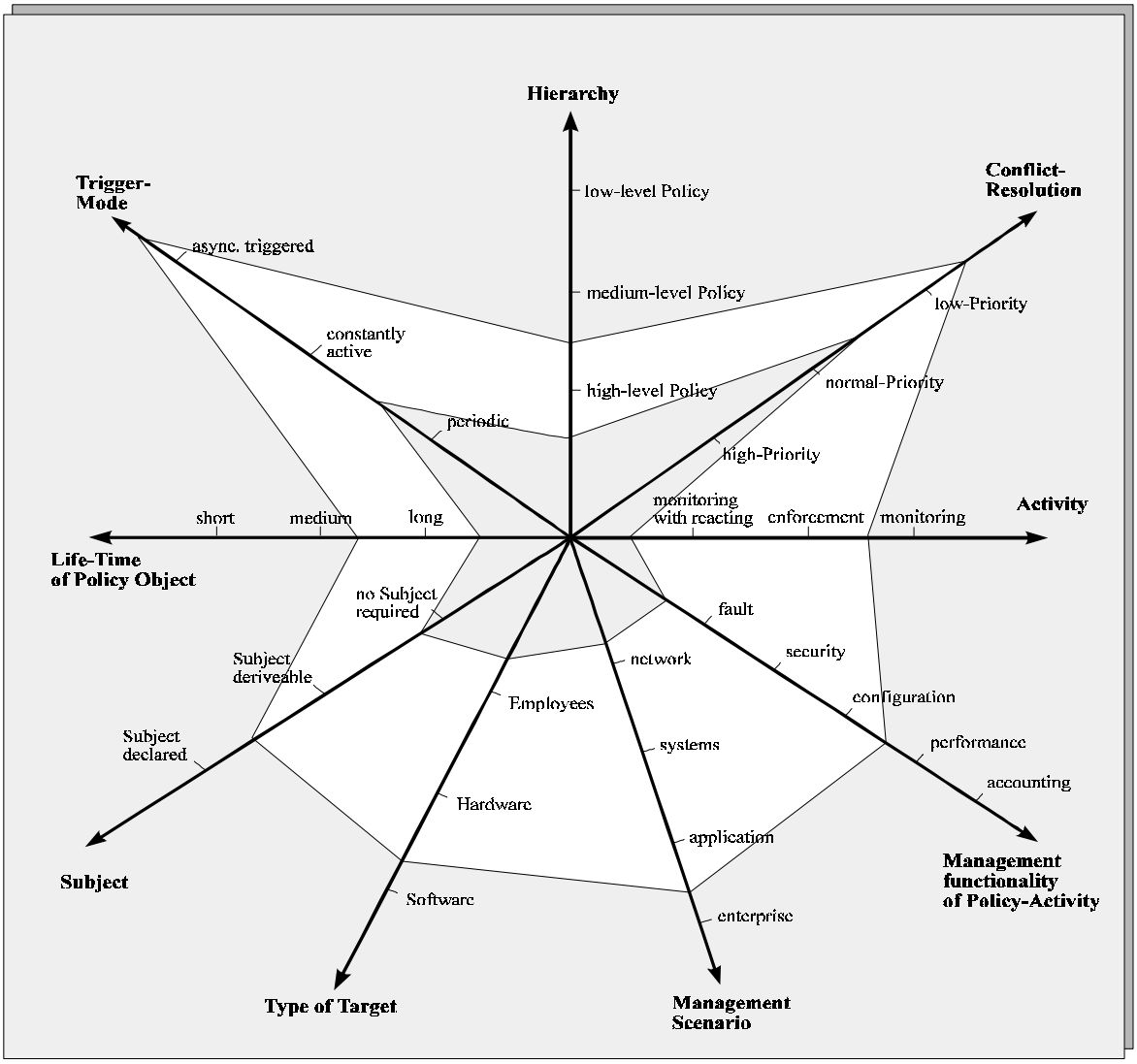

Abb.: 14

In der Verfeinerung bei der Klassifizierung muß jetzt noch genauer unterschieden werden. Dazu kann die Policy wie in 2.2 gezeigt zuerst gespalten werden. Zum Beispiel einer verfeinerten Klassifikation wird die Policy5b in der Activity-Ebene klassifiziert.

Policy5b: *Im Falle eines Netzfehlers müssen die Modems für die Wählverbindung aktiviert werden.*

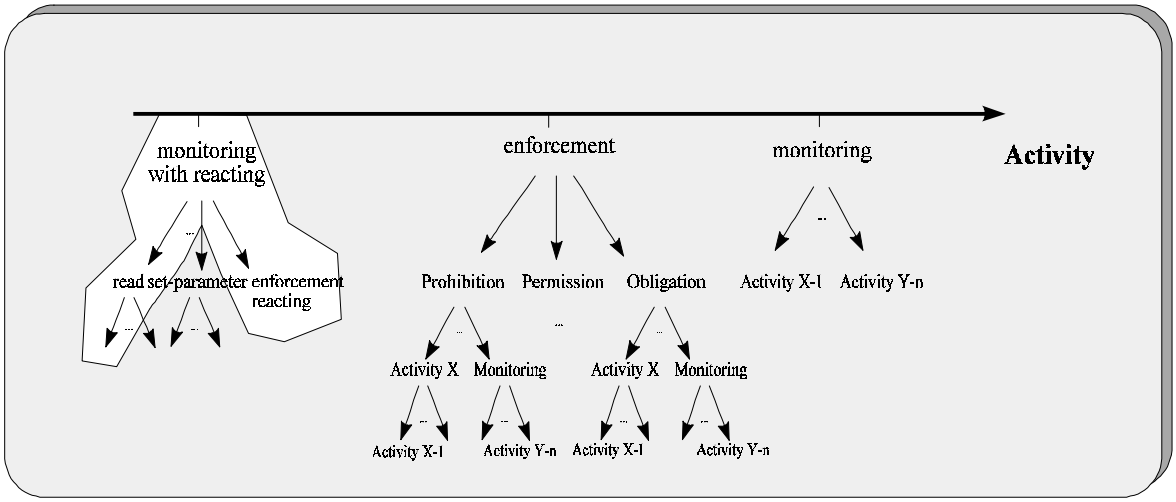

Abb.: 15

Und im zweiten Schritt:

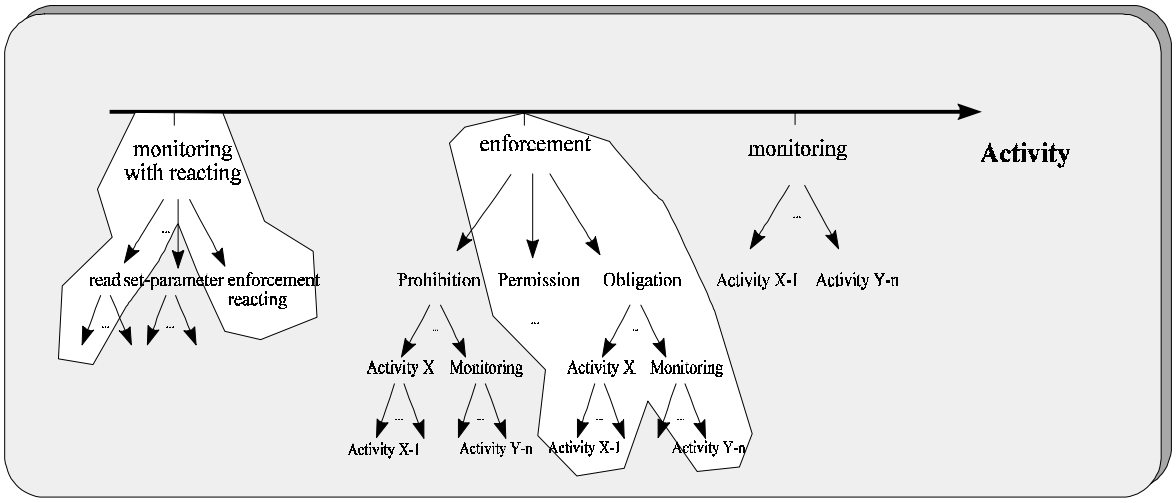

Abb.: 16

4.2 Policy-Beispiel in Objekt-Form

Policy5: *An einem Ticket-Buchungssystem teilnehmende Reisebüros sind über eine Standleitung angeschlossen. Im Falle eines Netzfehlers haben die Reisebüros die Möglichkeit, sich über eine Wählverbindung einzuloggen und müssen sich dann über einen Identifikationsvorgang authentisieren.*

POLICY OBJECT

Policy Name: **Fault-Sec-Conf.Mod**
Description of Goal: **Im Falle eines Netzfehlers der StandleitungenX sind die Modems für externe TeilnehmerY zu aktivieren, und der Identifikationsvorgang für Wählverbindungen zu aktivieren.**
Access-rights on Policy-Object: **AuthorX & Senior_Operator_Keys**
Creation Date/Valid Time: **10.06.1994/99.99.9999**
State of Refinement: **Not yet refined**
Relationship to Policies: **Relation to Policy YXZ**
Conflict-Resolution: **normal-priority**
Management Scenario of activity: **network, systems, application, enterprise**
Activity: **to be defined by tranformation**
Triggered by: **to be defined....**
Monitored by: **Senior_Operator ...**
Target Domain: **ModemsX-Y of IBM, Identification_Service**
Subject Domain: **automated Management-System.Event-Generation**

REGISTERED AS { ... }

Abb.: 17

Policy5b: *Im Falle eines Netzfehlers müssen die Modems für die Wählverbindung aktiviert werden.*

POLICY OBJECT

Policy Name: **Fault_Sec.Mod**
Description of Goal: **Im Falle eines Netzfehlers sind die Modems für externe Teilnehmer zu aktivieren**
Access-rights on Policy-Object: **AuthorX & OperatorKeys**
Creation Date/Valid Time: **20.06.1994/99.99.9999**
State of Refinement: **splitted, ready for action**
Relationship to Policies: **no relation**
Conflict-Resolution: **normal-priority**
Management Scenario of activity: **network, systems, application**
Activity: **send Activation-Flag to Targets**
Triggered by: **in Activity-Attribut**
Monitored by: **Operator, Fault-Log, ...**
Target Domain: **ModemsX-Y of IBM**
Subject Domain: **automated Management-System.Event-Generation**

REGISTERED AS { ... }

Abb.: 18

4.3 Zusammenfassung der erreichten Ziele

In dieser Arbeit wurden Kriterien zur Klassifikation von Policies aufgestellt und Ansätze einer formalen Beschreibungssprache dargebracht. Es wurden mehrere Ansätze untersucht, erforscht und miteinander verglichen. Viele noch offenstehende Fragen wurden aufgedeckt. Trotzdem erscheint die Klassifikation und Sprache "powerful" im theoretischen wie im praktischen Sinne zu sein.
Mit der erarbeiteten Klassifikation können große Mengen von Policies aus dem Netz- und Systemmanagement eingeteilt und mit der formalen Beschreibungssprache beschrieben werden. Der Ansatz zur formalen Beschreibungssprache gibt die Möglichkeit, Policies auf einer formalen Ebene zu definieren und für das Policy-Processing-System anwendbar machen zu können.
Ebenso sind in dieser Arbeit zum Punkt formale Beschreibungssprachen Untersuchungen darüber angestellt worden, wie die Policy-Activity auf bereits vorhandene Managementfunktionalität aufsetzbar ist.

Danksagungen:

Ich möchte mich besonders bei René Wies für seinen zeitintensiven Aufwand in der Betreuung meiner Diplomarbeit bedanken. In vielen Diskussionen hat er mich immer wieder auf den Kern der Fragestellung hingeleitet.
Ebenso möchte ich mich bei Tobias Kick bedanken, der mir als Node[2] in vielen privaten Netzen (FIDO, VIRNET, AMPER, MISTIE, etc.) deren Policies verschafft hat.

[2]Private Netze, wie zum Beipiel das internationale Fido-Netz, sind auf mehreren Ebenen gruppiert und die Verwaltung ist dezentralisiert. Ein Node ist dabei die vorletzte Organisationseinheit, die sich wie alle darüberliegenden (Hub, Koordinator) verpflichtet, Mails und zugelassene Files kostenfrei für alle Points, die an diesem Node angeschlossen sind, zu transprotieren und zu verteilen.

5. Literaturverzeichnis

[AMPER93]
Policy des Amper-Netzes, 28. 05. 93, Amper-Netz-Deutschland.

[BECKER93]
Karsten Becker, David Holden, Specifying the Dynamic Behavior of Management Systems, from Journal of Network and Systems Management, Sept. 1993, Volume 1, Number 3, ISSN 1064-7570.

[BROY90]
Prof. Dr. Manfred. Broy (1990), Vorlesung zur Einführung in die Informatik, 3. Semester, Eigenmittschrift.

[CASE93]
J. D. Case & Marshall. T. Rose, Advanced Topics in SNMP (Tutorial), in International Sysmposium on Integrated Network Management, April 18-23, 1993 San Francisco, USA.

[CCITT 208]
CCITT - Rec.X.208, Abstract Syntax Notation One, Blue Book, 1988.

[DUD89]
Duden Informatik, (1989) F. A. Brockhaus AG. Mannheim, Dudenverlag, ISBN 3-411-02421-6.

[FIDO91]
FIDONET POLICY 4.07, Offizielle deutsche Übersetzung von RC 24 (11. November 1991).

[HEGAB93]
Heinz-Gerd Hegering, Sebastian Abeck, Integriertes Netz- und Systemmanagement, Addison-Wesley, 1993, ISBN 508-4.

[HEG93]
Heinz-Gerd Hegering, Sebastian Abeck, R. Valta, Integriertes Netzmanagement - Konzepte und Realisierungen, erschienen in: Kommunikation in Verteilten Systemen, Tutorium anläßlich der ITG/GI Fachtagung der TUM 1993, Karl M.Lipp Verlag München, Herausgeber : N. Gerner, H.-G. Hegering, J. Swoboda, ISBN 3-87490-510-1, Seiten 243-323.

[HEG94]
Heinz-Gerd Hegering, B. Neumair, M. Gutschmidt, Cooperative Computing and Integrated System Management - A Critical Comparison of Architectural Approaches - ,1994 , to be published.

[ISO 7498-4]
Information Processing Systems - Open Systems Interconnection - Basic Reference Model, IS 7498 - 4, Management Framework.

[ISO 9596]
Information Technology, Common Management Information Protocol Specification (CMIP), IS 9596.

[ISO 10040/2]
Information Technology - Open Systems Interconnection - Systems Management Overview - Amendment 2: Management Domains Architecture, PDAM 10040/2, ISO/IEC, Nov. 1992.

[ISO 10164-X]
Information Technology - Open Systems Interconnection - Systems Management - Management Functions, ISO 10164-X, ISO/IEC.

[ISO 10164-19]
Information Technology- Open Systems Interconnection- Systems Management- Part 19: Management Domain and Management Policy Management Function, January 1994, CD 10164-19, ISO/IEC.

[ISO 10165-4]
Information Technology - Open Systems Interconnection - Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects, IS 10165-4, ISO/IEC August 1991.

[ISO 10746-1-3]
Information Technology - Basic Reference Model of Open Distributed Processing (ODP), Part 1 to Part 3. IS 10746-1, ISO/IEC Commitee Draft, 1993.

[KRUPCZAK93]
B. Krupczak, Unix Systems Management via SNMP, p289-299, in Heinz-Gerd Hegering, Yechiam Yemini (Editors), Proc. of the 3rd International Symposium on Integrated Network Management, San Francisco, IFIP Transactions C-12, April 1993, ISBN 0 444 89982 0.

[MARR93]
Damian A. Marriott, Management Policy Specification, Department of Computing, Imperial College of Science, Technology and Medicine, London, England, November 1993.

[MOFF91]
Jonathan D. Moffett & Morris S. Sloman , The Representation of Policies as System Objects, In Conference on Organizational Computing Systems, volume 12 of COCS´91, Atlanta, SIGOIS Bulletin, p.171-184, November 1991.

[MOFF92]
Jonathan D. Moffett & Morris S. Sloman, Policy Hierarchies for Distributed Systems Management, Domino Report: Arch/IC/6, Juni 1992.

[MOFF93]
Jonathan D. Moffett, Dirk Jonscher, John A. McDermid, The Policy Obstacle Course: A Framework for Policies Embedded within Distributed Computer Systems, July 1993.

[OSF DME92]
Open Software Foundation, OSF Distributed Management Environment (DME) Architekture, 1992.

[PAVLOU93]
George Pavlou, Implementing OSI Management, A Tutorial for the 3rd International Symposium on Integrated Network Management, April 1993, San Francisco, USA.

[SIEG91]
H.-J. Siegert, (1991), Betriebssysteme: Eine Einführung, Oldenburgverlag München Wien, 3. Auflage, ISBN 3-486-21930-8.

[SLOM93]
Morris S. Sloman, Specifying Policy for Management of Distributed System, In Proc. IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, October 1993.

[VIRN91]
VIRNET POLICY, Policy of 'Virus Information Resource Net' (1991) by Mikael Larsson.

[WELLS94]
Caroline Wells, Tivoli Systems, Inc., Tivoli Management Environment (TME), in Data Pro, Datapro International Network Management, Januar 1994.

[WIES-DSOM94]
René Wies, Policy Definition and Classification: Aspects, Criteria and Examples, In Proc. of teh IFIP/IEEE International Workshop on Distributed Systems: Operations & Management, IFIP, October 1994, to be published.

[WIES-PLEN94]
René Wies, Policies in Network and Systems Management-Formal Definition and Architecture-In Journal of Network and Systems Management, March 1994, M. Malek (editor), Plenum Publishing Corp., New York.

[WIES-ISINM94]
René Wies, Using Policy Classification for Policy Specification and Transformation, Draft: ISINM `95, Version: May 30, 1994.