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
1.2 Vorgehensweise
1.3 Was sind Policies?
1.4 Policies als Objekte
1.5 Anforderungen an das Policy-Processing-System

2. Die Dimensionen der Klassifikation

2.1 Anforderungen an eine Klassifikation für Policies
2.2 Hierarchien und Verfeinerung
2.3 Policy Conflict-Resolution
2.4 Trigger-Mode
2.5 Life-Time of Policy Object
2.6 Subject-Ebene
2.7 Type of Target-Ebene
2.8 Management Scenario
2.9 Management functionality of Policy-Activity
2.10 Activity-Ebene
2.11 Bewertungskriterien für Klassifikationen

3. Formale Beschreibungssprache für Policies

3.1 Anforderungen an eine formale Beschreibungssprache für Policies
3.2 Das Policy-Objekt und seine Parameter
3.3 SMF als Managementfunktionalität für Policy-Activities
3.4 Die 5 Viewpoints bei ODP
3.5 Activity-Implementierung in SMF

4. Zusammenfassung und Ausblick

4.1 Anwendung der Klassifikation an einem Beispiel
4.2 Policy-Beipiel in Objekt-Form
4.3 Zusammenfassung der erreichten Ziele

5. Literaturverzeichnis

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.

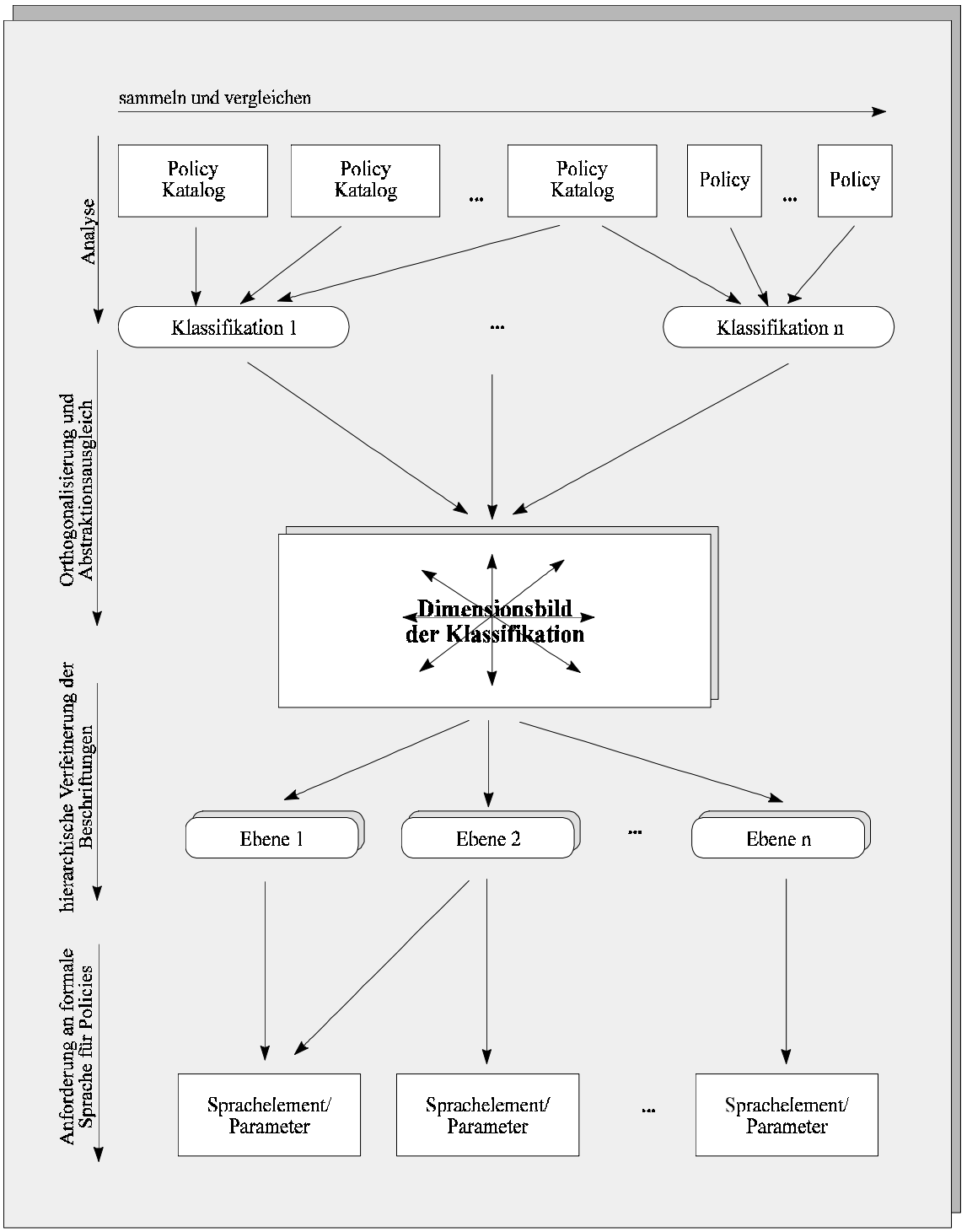
sammeln und vergleichen
Analyse
Policy Katalog
Policy Katalog
...
Policy Katalog
Policy
...
Policy
Klassifikation 1
...
Klassifikation n
Orthogonalisierung und Abstraktionsausgleich
Dimensionsbild der Klassifikation
hierarchische Verfeinerung der Beschriftungen
Ebene 1
Ebene 2
...
Ebene n
Anforderung 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.

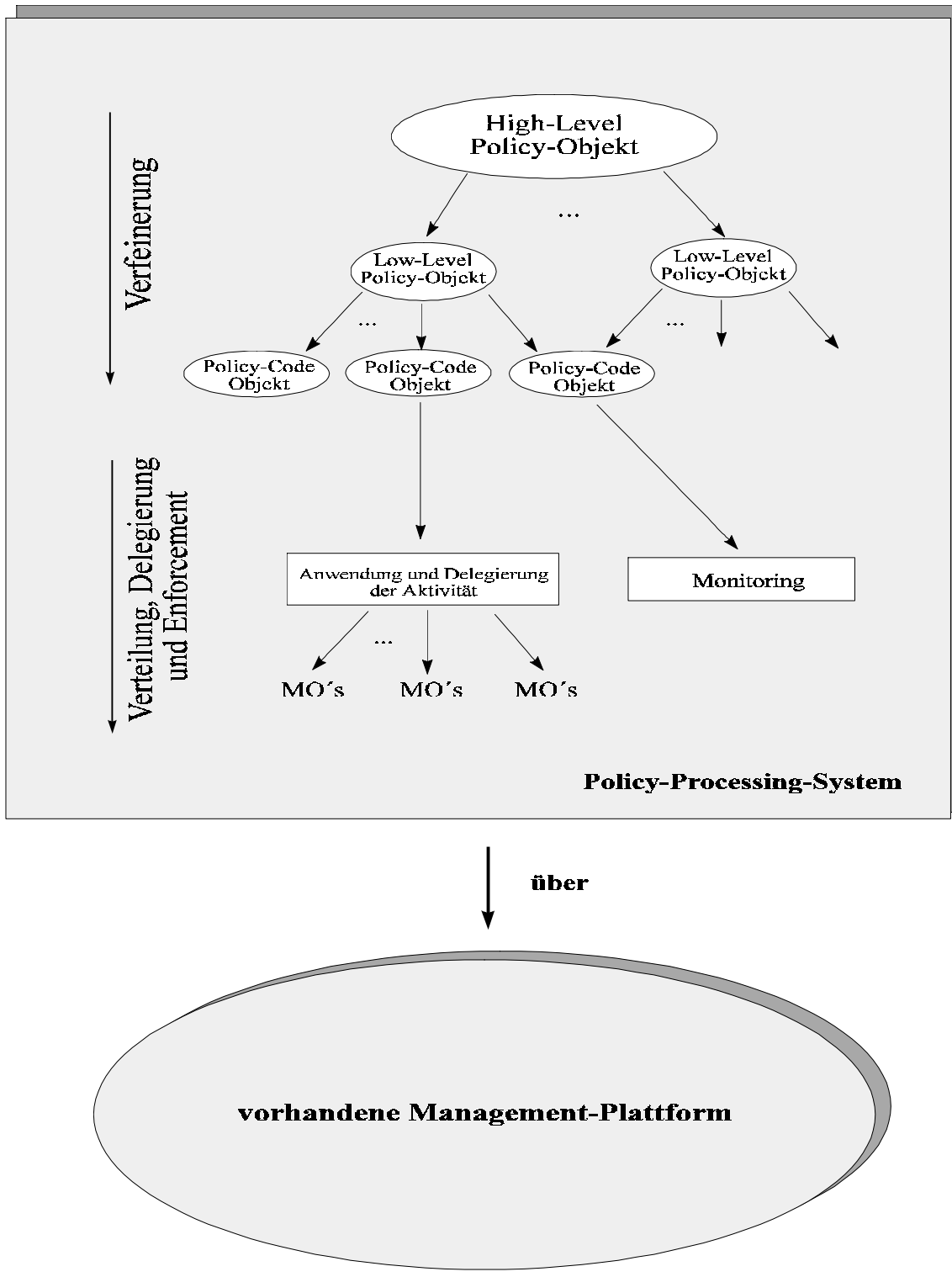
Verfeinerung
High-Level Policy-Objekt
...
Low-Level Policy-Objekt
Low-Level Policy-Objekt
...
...
Policy-Code Objekt
Policy-Code Objekt
Policy-Code Objekt
Verteilung, Delegierung und Enforcement
Anwendung und Delegierung der Aktivität
Monitoring
...
MO´s
MO´s
MO´s
Policy-Processing-System
über
vorhandene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

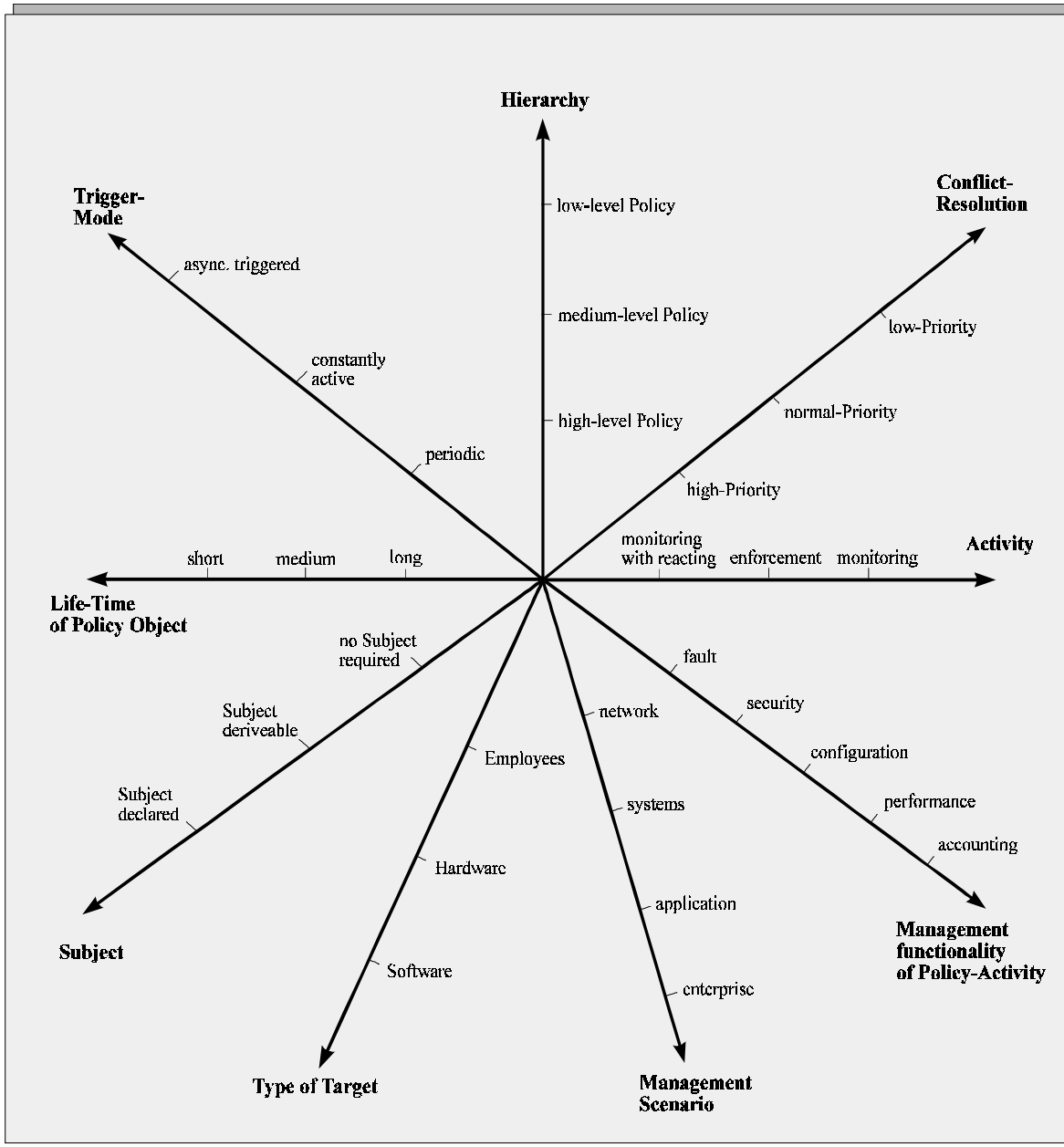

Abb.: 3

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,

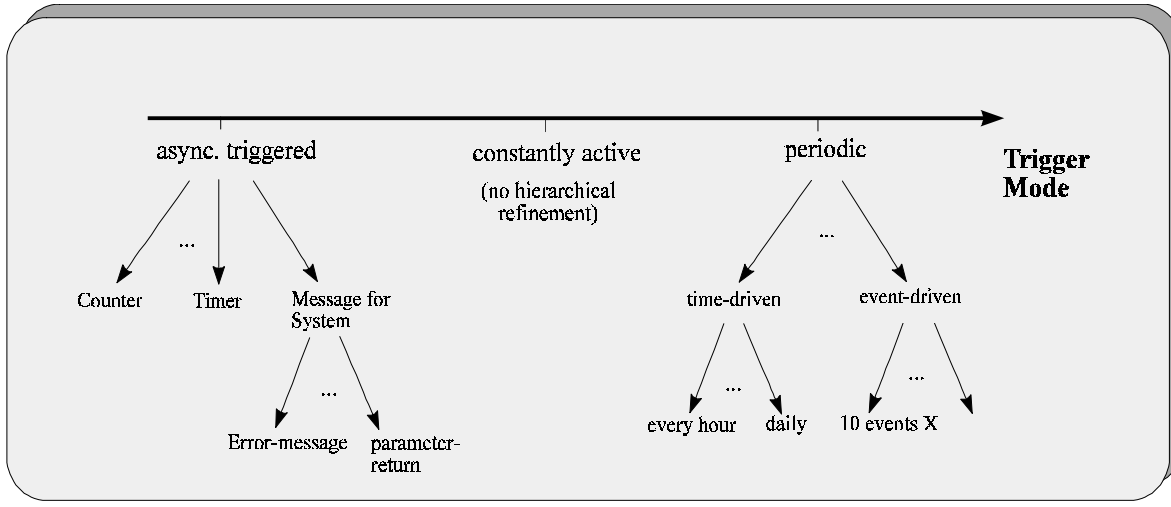

Abb.: 5

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.*

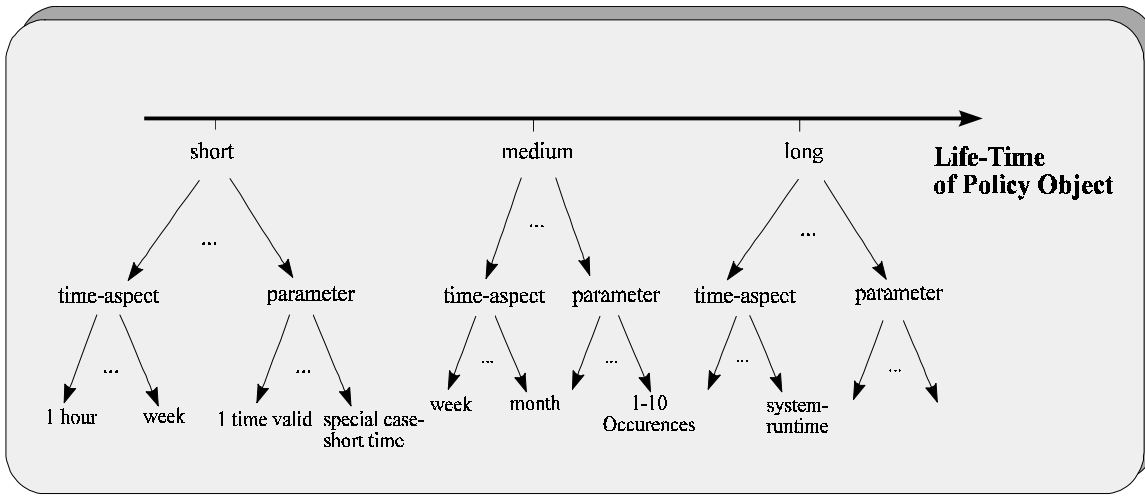

Abb.: 6

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.*

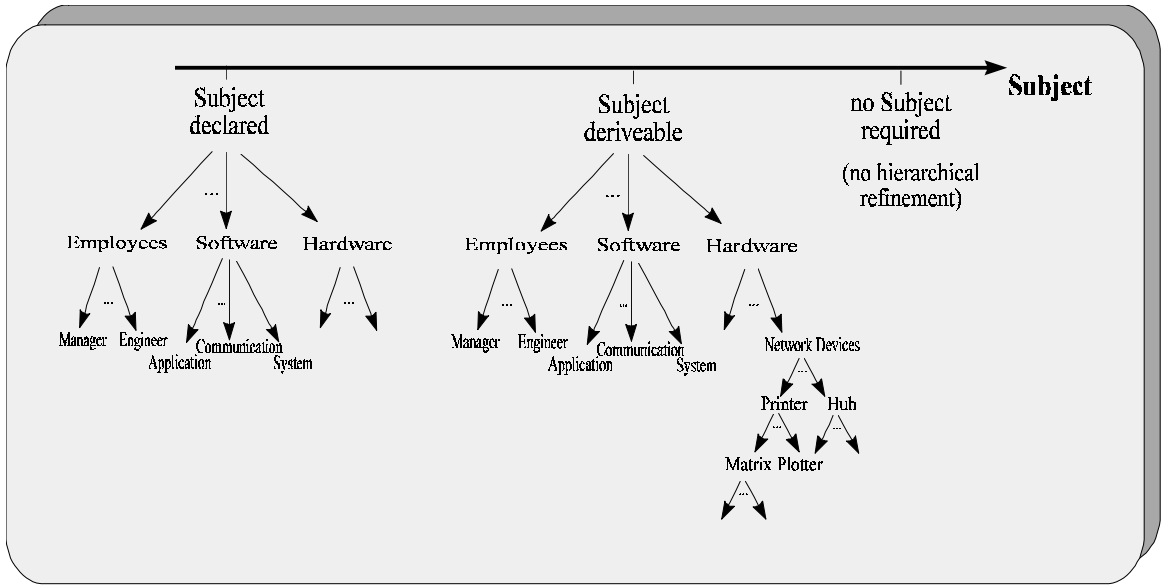

Abb.: 7

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.

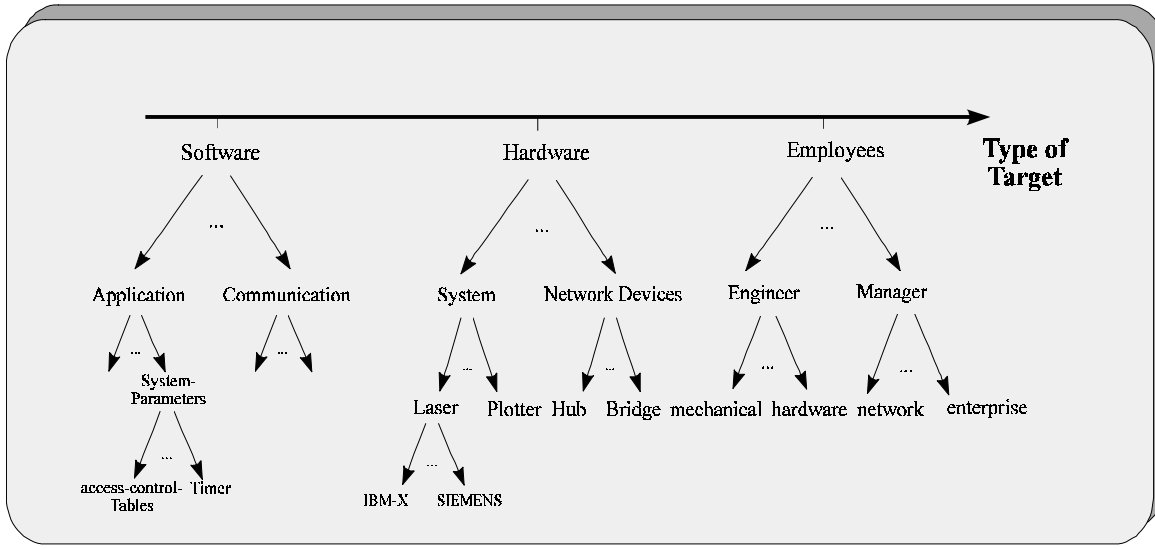

Abb.: 8

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

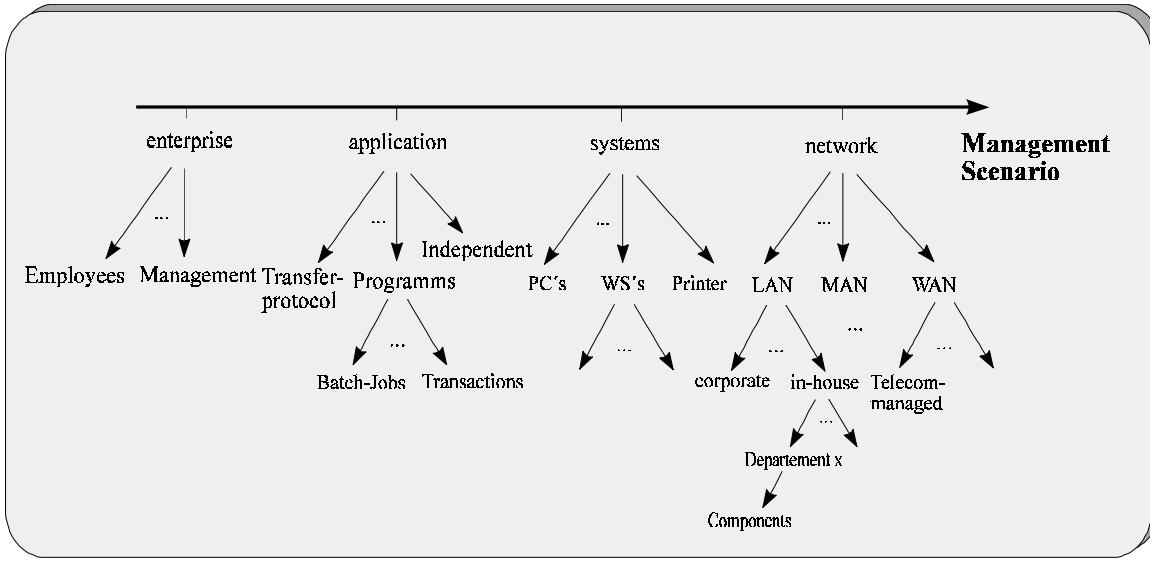

Abb.: 9

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].

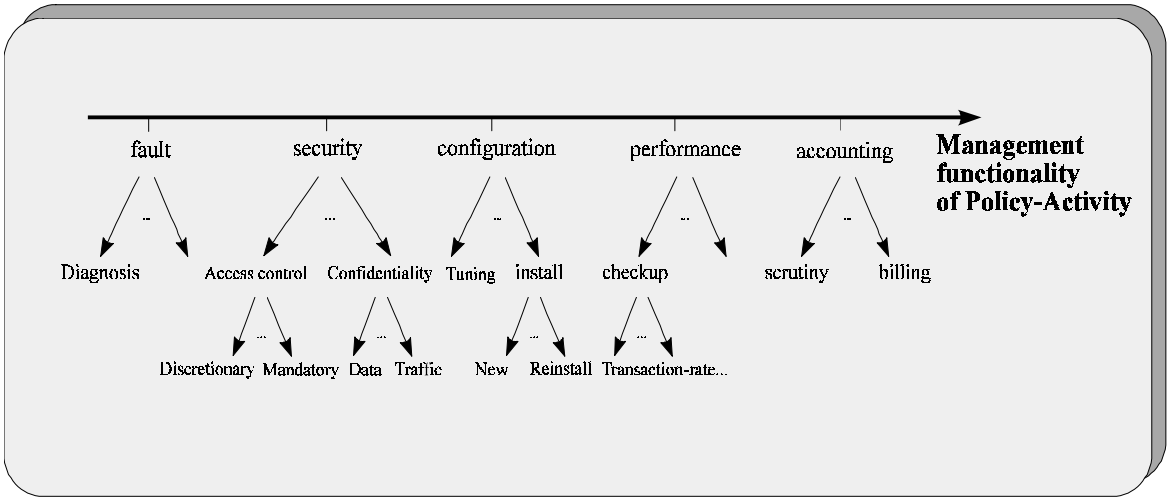

Abb.: 10

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

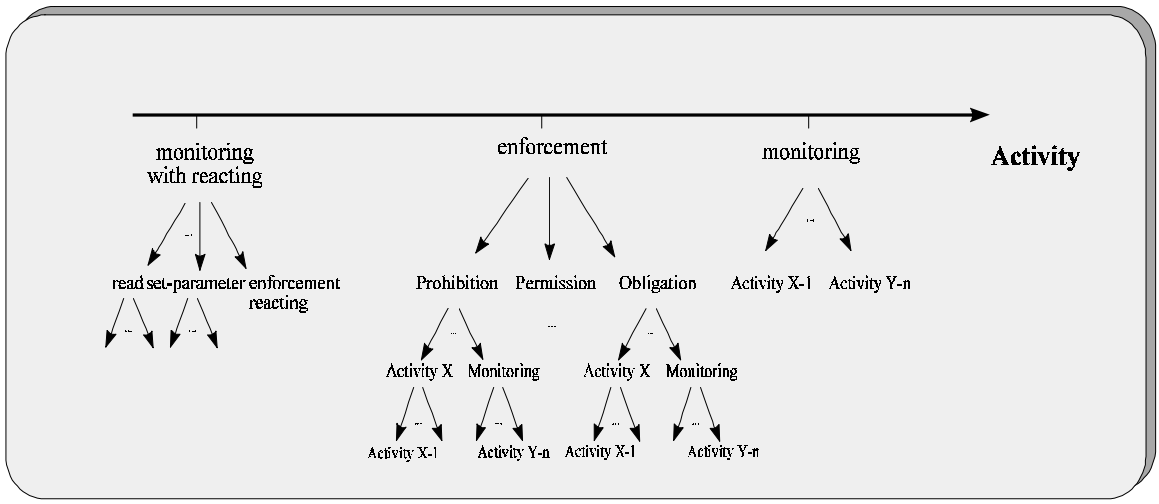

Abb.: 11

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.