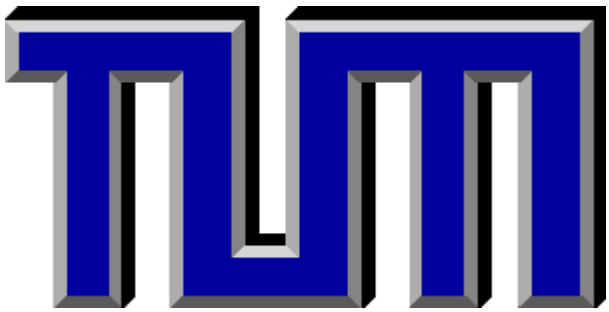


TECHNISCHE UNIVERSITÄT MÜNCHEN
Institut für Informatik

Konzeption eines Eventmanagements für ein PC-Cluster am Beispiel Microsoft Wolfpack

Bearbeiter: Alexander Razvan Balanescu

Aufgabensteller: Prof. Dr. Heinz.Gerd Hegering

Betreuer: Boris Gruschke

Rainer Hauck

Datum: 13. Februar 1998

Inhaltsverzeichnis

INHALTSVERZEICHNIS 1

1. EINLEITUNG 3

1.1 MOTIVATION 3
1.2 ZIELE DER DIPLOMARBEIT 4
1.3 VORGEHENSWEISE UND GLIEDERUNG 5

2 BESCHREIBUNG DES PC-CLUSTERS 7

2.1 MYLEX SCSI-RAID 9
2.1.1 Allgemeine Beschreibung 9
2.1.2 Softwarekomponenten 9
2.1.3 Zustände und Fehler 10
2.1.4 Operationen 10
2.2 MICROSOFT WINDOWS NT SERVER 4.0 11
2.2.1 Allgemeine Beschreibung 11
2.2.2 Softwarekomponenten 12
2.2.5 Operationen 15
2.3 MICROSOFT CLUSTER SERVER (MSCS) 16
2.3.1 Zustände 17
2.3.2 Operationen 17
2.3.3 NT-Integration 18

3. SYSTEMATISIERUNG DER EVENTS 20

3.1 BEGRIFFE 20
3.2 GRUPPIERUNGEN 21
3.2.1 Gruppierung nach der Eventberücksichtigung 21
3.2.2 Gruppierung nach der Eventquelle 22
3.2.3 Gruppierung nach der Eventinterpretation 23
3.2.4 Gruppierung nach der Eventverarbeitung 25
3.3 MYLEX-EVENTS 26
3.3.1 Gruppierung nach den Eventquellen 26
3.3.2 Systematisierung der Mylex-Events 27
3.4 NT-EVENTS 29
3.4.1 Einführende Betrachtungen 29
3.4.1.1 EventLog Service 30
3.4.1.2 Event-Logging 30
3.4.1.3 Event-Browsing 31
3.4.1.4 NT-Eventarten 32
3.4.2 NT-Netz-Events 35
3.4.2.1 Gruppierung nach den Eventquellen 36
3.4.2.2 Systematik der NT-Netz-Events 37
3.5 MSCS-EVENTS 39
3.5.1 Erschließung der MSCS-Events 40
3.5.2 Gruppierung nach den Eventquellen 42
3.5.3 Systematisierung derMSCS-Events 42
3.6 ZUSAMMENFASSUNG UND ERKENNTNISSE 43

4. KONZEPTION DES MSWEM 44

4.1 SPEZIELLE ZIELE 44
4.2 ANNAHMEN 45
4.3 KORRELATIONSGRUNDLAGEN 46
4.3.1 Kausale Sicht 46
4.3.2 Operationale Sicht 48
4.3.2.1 Model Based - Korrelationstechniken 49

4.3.2.2 Fault Propagation - Korrelationstechniken 49
4.3.2.3 Model Traversing - Korrelationstechniken 49
4.3.2.4 Case Based - Korrelationstechniken 50
4.3.2.5 Zusammenfassung 50
4.4 KORRELATIONSWAHL 51
4.4.1 Wahl der Korrelationsart *51*
4.4.2 Wahl der Korrelationstechnik *51*
4.5 KONSTRUKTION DES MSWEM 53
4.5.1 Kausalitätsgraph für Events (EG) *53*
4.5.1.1 Graphknoten im EG 53
4.5.1.2 Graphkanten im EG 54
4.5.1.3 Analyse des EG 55
4.5.2 Abhängigkeitsgraph für Eventquellen (EQG) *56*
4.5.2.1 Graphknoten im EQG 56
4.5.2.2 Graphkanten im EQG 58
4.5.2.3 Analyse des EQG 59
4.5.3 Eventklassen *62*
4.5.3.1 Erzeugung der Eventklassen 62
4.5.3.2 Aufbau des Graphen für Eventklassen (EKG) 62
4.6 KORRELATION IM MSWEM 69
4.6.1 Korrelations-Operationen des MSWEM *69*
4.6.2 Korrelations-Algorithmus des MSWEM *73*

5. REALISIERUNGSASPEKTE DES MSWEM 76

5.1 VERGLEICH MIT REGELBASIERTEN SYSTEMEN 76
5.2 INTEGRATION WEITERER TREIBER UND ANWENDUNGEN 77
5.2.1 Integration weiterer Treiber *77*
5.2.2 Integration weiterer Anwendungen *78*
5.3 INTEGRATION MIT DEM STATUSMANAGEMENT 78
5.3.1 Zustands-Aspekte der Integration *79*
5.3.2 Management-Aspekte der Integration *80*

6. ABSCHLIEßENDE BEMERKUNGEN 82

6.1 ZUSAMMENFASSUNG UND ERKENNTNISSE 82
6.2 BEMERKUNGEN ZUR INFORMATIONSBESCHAFFUNG 83
6.3 AUSBLICK 83

LITERATURVERZEICHNIS 84

ABBILDUNGSVERZEICHNIS 86

TABELLENVERZEICHNIS 86

ANHANG A: WICHTIGE DETAILS ZUR BESCHREIBUNG DES CLUSTERS 86

A.1 DETAILS ZUM MYLEX SCSI-RAID 86
Hintergrundwissen für die Event-Beschreibung *87*
Informations-Struktur „dga_event_info_t“ *89*

ANHANG B: DETAILS ZUR SYSTEMATISIERUNG DER EVENTS 89

B.1 EVENTKATALOG DER MYLEX - EVENTS 89
B.2 EVENTKATALOG DER NT - EVENTS 91
B.3 DETAILS ZU DEN MSCS EVENTS 99

1. Einleitung

1.1 Motivation

Das vorliegende Kapitel erklärt warum ein Eventmanagement für ein PC-Cluster notwendig ist. Unter **Event** oder Ereignis wird eine Meldung verstanden, die im laufenden PC-Cluster-Betrieb registriert wird und die einen Fehler aufzeigen kann.

1. Hardware-Entwicklung ⇒ steigende Notwendigkeit für Netz- und Systemmanagement für PCs:

Die Hardware-Entwicklung führt zu einer Leistungssteigerung in der Computerwelt, wobei die PCs als eine der „schwächsten Computerfamilien" am meisten davon profitieren, da sie eine kostengünstige (im Vergleich zu anderen Computerfamilien) Hardware voraussetzen. Außerdem bieten sich PCs für eine Stabilitätserhöhung durch Hardwareredundanz wegen ihre Kostengünstigkeit an.
Dies führt zu einer steigenden Notwendigkeit der Betreibung eines möglichst integrierten und einheitlichen Netz- und Systemmanagements (auch) in einem PC-Netz. Dafür gibt es nur Teillösungen, wie z.B. das „Server Management/Server View"-System der Firma Siemens Nixdorf AG (SNI).

2. PCs sind bezüglich Verfügbarkeit für kritische Anwendungen zu unzuverlässig:

Trotz hoher Hardwareredundanz und Bemühungen zur Stabilitätssteigerung wird die Oberschranke von 99 Prozent (entspricht 3,7 Ausfall-Tage im Jahr) Verfügbarkeit kaum von einem Server überschritten. Dabei erreichen „gute" PC-Arbeitsstationen höchstens 95 Prozent (entspricht 18 Ausfall-Tage im Jahr) Verfügbarkeit. Dies ist für kritische Anwendungen nicht tolerabel.

3. Zurückgreifen auf Cluster-Technologien:

Da eine alleinige Steigerung der Hardwareredundanz bezüglich Verfügbarkeit unzufriedenstellende Ergebnisse liefert, orientiert man sich in letzter Zeit zunehmend auf die Einführung von Cluster-Technologien (die für andere „Computerfamilien" schon erfolgreich eingesetzt wurden), die auf PC-Server basieren. Dabei wird in der vorliegenden Diplomarbeit unter einem Cluster ein Verbund von redundanten Maschinen, Nodes genannt, verstanden. Dieser Verbund von Maschinen verfügt über einen eigenen, direkten Kommunikationsmechanismus, der ein Verschieben von Ausfall-bedrohten Anwendungen ermöglicht, und vermittelt der Außenwelt die Sicht eines einzelnen Servers. In diesem Sinne ist ein PC-Cluster ein Cluster dessen Nodes PC-Server sind und dessen primäres Ziel eine „zumindest kontinuierliche Verfügbarkeit", d.h. 99,999 Prozent (entspricht 5 Ausfall-Minuten im Jahr) darstellt [KUR97].

4. Netz- und Systemmanagement im PC-Cluster:

Durch das PC-Cluster ist eine spezielle Netz-Architektur gegeben, die auf PCs basiert und die Hardwareredundanz beinhaltet. Wegen diesem speziellen Charakter und wegen den Überlegungen, die unter Punkt 1. angestellt wurden, wird die steigende Notwendigkeit eines möglichst integrierten und einheitlichen Netz- und Systemmanagements im PC-Cluster ersichtlich. Dabei wird eine Behandlung von spezifischen Management-Aspekten für die spezielle Netz-Architektur erforderlich sein.

5. Eventmanagement im PC-Cluster:

Allgemein ist ein Eventmanagement für das Mitteilen von wenigen, ungewöhnlichen Zustände an ein Management-System verantwortlich. Also wäre ein Eventmanagement als Teil einer Management-Plattform für ein PC-Cluster wegen Punkt 4. eine Motivation für das Thema der vorliegenden Diplomarbeit.
Als Basisanwendung einer Management-Plattform ist ein Eventmanagement - allgemein gesehen - für das Empfangen von Ressourcen-Events, für das Verarbeiten dieser und für das Weiterleiten an ein Management-System zuständig. Insbesondere behandelt ein Eventmanagement auch für die Ver-

fügbarkeit wichtige Events und kann sie im Sinne einer Ausfall-Verhinderung verarbeiten. Demzufolge ergibt sich - unter Berücksichtigung des primären Zieles eines PC-Clusters - eine zusätzliche, besondere Motivation zur Konzeption eines Eventmanagements für ein PC-Cluster.

Event- und Statusmanagement sind eng gekoppelte Managementbestandteile, da die Events, die das Eventmanagement behandelt, durch Zustandsänderungen von Managed Objects (MOs) ausgelöst werden, und Zustände (Stati) bekanntlich das Untersuchungsziel des Statusmanagements sind. Ein Statusmanagement für ein PC-Cluster wurde in einer vorangehenden Diplomarbeit [NAY97] basierend auf SNMP-Standards konzipiert.

Durch die vorherigen Überlegungen bleiben bezüglich des Eventmanagements einige Fragen offen:
- **Welche** Eingabe-Events gibt es?
- **Welche** Events sollen ausgegeben werden?
- Wie würde eine Realisierung für das Eventmanagement stattfinden?
- Wie werden neue PC-Cluster-Komponenten in das Eventmanagement eingebunden?
- Wie sollte das Eventmanagement mit dem konzipierten Statusmanagement kooperieren?

Die Aufgabenstellung für die vorliegende Arbeit resultiert aus der der Notwendigkeit eines Eventmanagements im PC-Cluster und aus den oben genannten Fragen.

1.2 Ziele der Diplomarbeit

Die Ziele der vorliegenden Diplomarbeit werden von den allgemeinen Zielen eines Eventmanagements, von den Gegebenheiten einer PC-Cluster-Architektur und von der Realisierung des zu erstellenden Konzepts geprägt.

Die **allgemeinen Ziele eines Eventmanagements** werden in ihrer Gesamtheit als Ziele der Diplomarbeit integriert und sind folgende:
- (*Fehlerlokalisierung*) Fehlerlokalisierung, d.h. eindeutige Eingrenzung der Event-Quellen,
- (*Minimierung*) Minimierung der Anzahl zu verarbeitender Eventmengen,
- (*Eventstorms*) Vermeidung von Eventstorms,
- (*Informationsgehalt*) Weiterleitung von Events mit kontextbezogenen, hohen Informationsgehalt,
- (*Weiterleitung*) Gezielte Weiterleitung von Events an zuständige Administratoren.

Unter Berücksichtigung der speziellen Gegebenheiten einer Netz- und System-Architektur geht ein Eventmanagement zur Realisierung dieser allgemeinen Ziele i.A. folgendermaßen vor:
1. Schaffung einer Informationsstruktur über die MOs,
2. Aufstellung von Methoden, die basierend auf der Informationsstruktur operieren und dadurch die allgemeinen Ziele eines Eventmanagements umsetzen.

Demzufolge setzt sich die vorliegende Diplomarbeit - die Gegebenheiten einer PC-Cluster-Architektur beachtend - folgende weitere Ziele:
- (*Auswertung*) Auswertung der Events im PC-Cluster,
- (*Information*) Erstellung einer Informationsstruktur der PC-Cluster-Komponenten,
- (*Verarbeitung*) Konzeption eines Verarbeitungsmechanismus, der auf das Ergebnis des (*Information*)-Zieles zurückgreift und die allgemeinen Ziele eines Eventmanagements realisiert.

Diese Ziele werden **Ziele der Konzeption eines Eventmanagements** genannt. Dabei wird in der vorliegenden Diplomarbeit unter **Korrelation** eine Zusammenfassung der Ziele (*Information*) und (*Verarbeitung*) verstanden.

Im Hinblick auf eine Realisierung der Konzeption und gewissermaßen als Prüfung des erstellten Konzepts werden noch die allgemeinen Ziele
- (*Vergleich*) Vergleich mit bewährte Konzepte,
- (*Erweiterbarkeit*) Untersuchung der Erweiterbarkeit,
- (*Integration*) Integration mit der vorangehenden Statusmanagement-Konzeption

verfolgt. Diese Ziele werden **Ziele bezüglich Realisierung** genannt.

Die Untersuchung des PC-Clusters wird wegen der speziellen Architektur weitere Ziele für das Eventmanagements aufdecken. Demzufolge werden diese weiteren Ziele **spezielle Ziele des Eventmanagements** genannt und werden nach der Untersuchung der PC-Cluster-Events, die einige Erkenntnisse aufbringen, angegeben.

Für die Realisierung dieser allgemeinen Ziele wurde von einem Kooperationspartner des MNM-Teams, und zwar die Firma Siemens Nixdorf AG in Augsburg, ein PC-Cluster zur Verfügung gestellt, das auf eine von Microsoft entwickelte Cluster-Technologie namens Wolfpack aufbaut. Deswegen wird sich die vorliegende Diplomarbeit - ohne Einschränkung der Allgemeinheit - auf dieses konkrete PC-Cluster, beziehen.

Der Eventmanager, der durch die auf Wolfpack ausgerichtete Konzeption des Eventmanagements entsteht, wird hiermit MNM-SNI-Wolfpack-Event-Manager, kurz **MSWEM**, benannt.

Bemerkung 1.1: Statt spezielle Ziele des Eventmanagements wird ab jetzt der Ausdruck spezielle Ziele des MSWEM verwendet.

1.3 Vorgehensweise und Gliederung

Im folgenden Kapitel wird die Vorgehensweise für das Erreichen der Diplomarbeitsziele übersichtlich dargestellt. Dies hat zur Folge, daß gleichzeitig auch eine Gliederung der Diplomarbeit erarbeitet wird.

Da sich diese Diplomarbeit bei der Konzeption des MSWEM an eine konkrete, exemplarische PC-Cluster-Architektur orientiert, wird zunächst eine auf das Eventmanagement ausgerichtete Darstellung dieser Architektur in Kapitel 2 vorgenommen.

Damit ist die Grundlage für eine eingehende Systematisierung der Events im PC-Cluster geschaffen. Diese Eventuntersuchungen stellen die Grundlage zur Konzeption des MSWEM dar und werden im Kapitel 3 durchgeführt. Dabei stellen die Ergebnisse der Systematisierung die Realisierung des (*Auswertung*)-Zieles dar und werden in Eventkatalogen zusammengefaßt.

Durch die gewonnenen Erkenntnisse wird das Kapitel 4 eingeleitet, das den zentralen Teil der Diplomarbeit darstellt. Dabei konzentriert man sich zunächst auf eine Aufstellung der speziellen Ziele des MSWEM, die aus den Gegebenheiten des PC-Clusters resultieren (siehe Kapitel 1.2). Es wird eine Top-Down Untersuchung der Korrelations-Grundlagen vorgenommen und dann werden die Ziele der Konzeption eines Eventmanagements und implizit die allgemeinen Ziele eines Eventmanagements und die speziellen Ziele des MSWEM realisiert.

Das Kapitel 5 beschäftigt sich mit den Realisierungsaspekten des MSWEM und erfüllt damit die allgemeinen Ziele (*Vergleich*), (*Erweiterbarkeit*) und (*Integration*).

Das abschließende Kapitel 6 stellt eine Zusammenfassung und gewonnene Erkenntnisse vor, gibt noch einige aus der Diplomarbeit resultierenden Bemerkungen zur Informationsbeschaffung an und schlägt weitergehende Überlegungen zum Thema im Ausblick vor.

Da die im Kapitel 3 analysierten Events sehr anzahlreich sind, werden die entsprechenden Eventkataloge in den Anhang verlagert. Außerdem wurden in den Anhang auch wichtige aber zu umfangreiche Erläuterungen der architektonischen Grundlagen eingefügt.

Die folgende Abbildung 1.1 versteht sich als schematische Übersicht der Vorgehensweise in der vorliegenden Diplomarbeit:

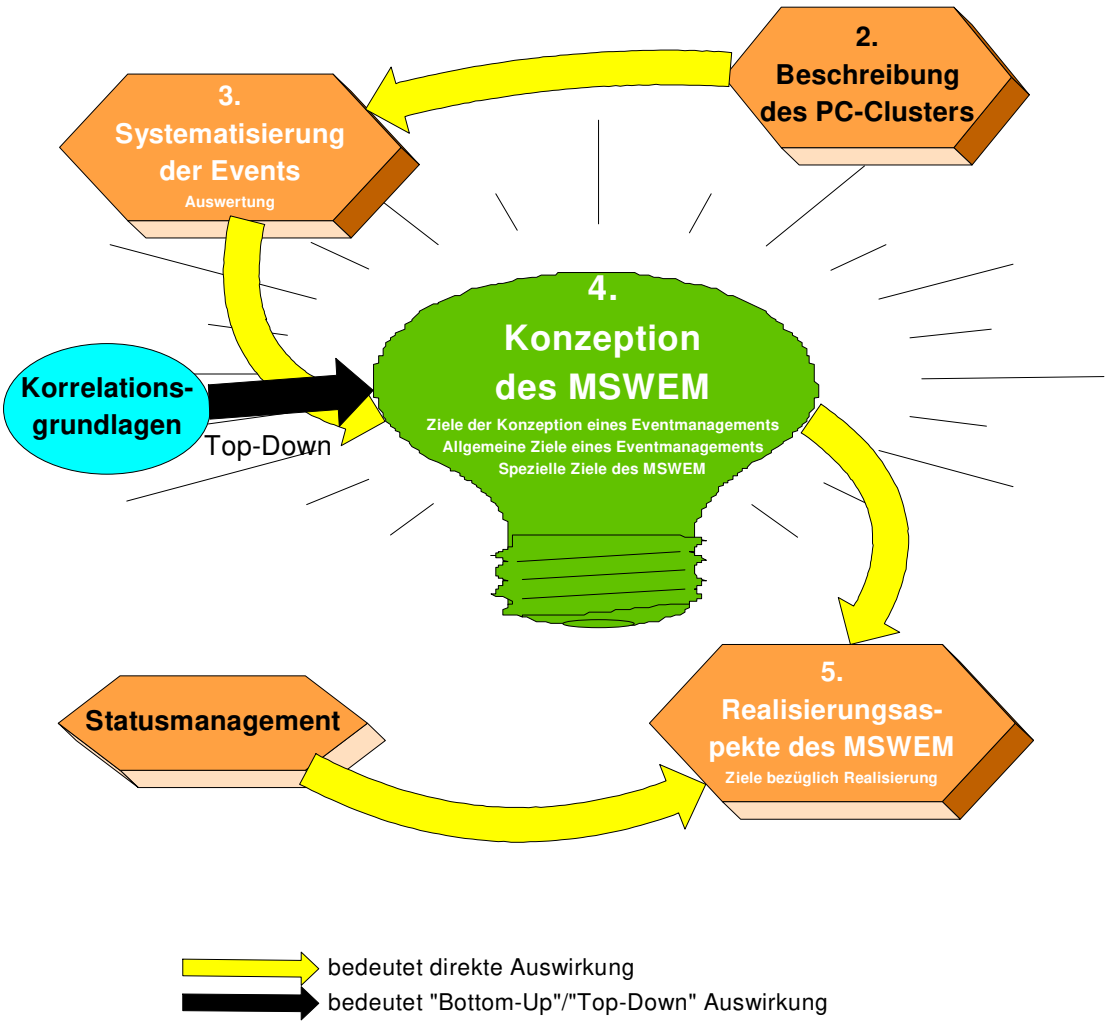


Abbildung 1.1: Vorgehensweise

An dieser Stelle möchte ich eine besondere Danksagung an meine Betreuer Boris Gruschke und Rainer Hauck vom Informatik-Institut der LMU München für ihre wertvolle Unterstützung und Diskussionen aussprechen. Außerdem möchte ich mich bei den Betreuern von der Firma Siemens Nixdorf AG in Augsburg, Herrn Stefan Freund und Herrn Manfred Randelzofer für ihre freundliche Unterstützung bedanken.

2 Beschreibung des PC-Clusters

Dieses Kapitel beschäftigt sich mit den grundlegenden Informationen, die für die Systematisierung der Events im PC-Cluster notwendig sind. Dabei wird verstärkt auf Hintergrundwissen über die gegebene PC-Cluster-Architektur eingegangen, wobei man sich auf folgende Annahme beschränkt:

- **(A1):** Es werden nur die architektonischen Grundlagen, die für das Analysieren der Events und für die Konzeption des MSWEM relevant sind, betrachtet,
- **(A2):** Eine Architektur-Beschreibung, die in der vorangehenden Diplomarbeit über das Statusmanagement angegeben wurde, wird hier höchstens ergänzend oder wegen der Übesichtlichkeit wiederholt.

Weiterhin wird an dieser Stelle die Menge aller Events, die im PC-Cluster aufreten können, mit **E** bezeichnet.

Einteilung bezüglich Hardware

Es wird davon ausgegangen, daß ein PC-Cluster ein Verbund von untereinander vernetzten PC-Servern darstellt, in dem jegliche Hardware möglichst redundant vorhanden ist. Dabei ist das PC-Cluster, rein hardwaretechnisch gesehen, aus folgenden Hauptbestandteilen aufgebaut:
1. Rechensysteme (Nodes, hier:PC-Server),
2. verschiedene Rechnernetze (Cluster-intern/Cluster-extern),
3. ein (hochperformantes) Speichersystem (i.d.R. RAID),
4. ein oder mehrere Busse für die Kommunikation zwischen Rechensystemen und Speichersystem.

Abbildung 2.1 gibt einen möglichen Aufbau eines PC-Clusters an, in dem vier PCs als „Nodes“, ein Intra-Cluster-Hochleistungsnetz als „Cluster-internes Rechnernetz“, ein Client-Netz, z.B. LAN, als „Cluster-externes Rechnernetz“, ein RAID als „hochperformantes Speichersystem“ und ein Shared SCSI Bus als „Bus für die Kommunikation zwischen Rechensystemen und Speichersystem“ gewählt wurden. Da die Intra-Cluster-Kommunikation über ein Cluster-internes Rechnernetz stattfindet, liegt ein sogenanntes **Enclosed Cluster** vor.

Jedes der Hauptbestandteile des PC-Clusters untergliedert sich mehr oder weniger hierarchisch in weiteren Hardware-Bestandteile, wie z.B. die Rechensysteme in CPU, Mainboard, Lüfter, etc.

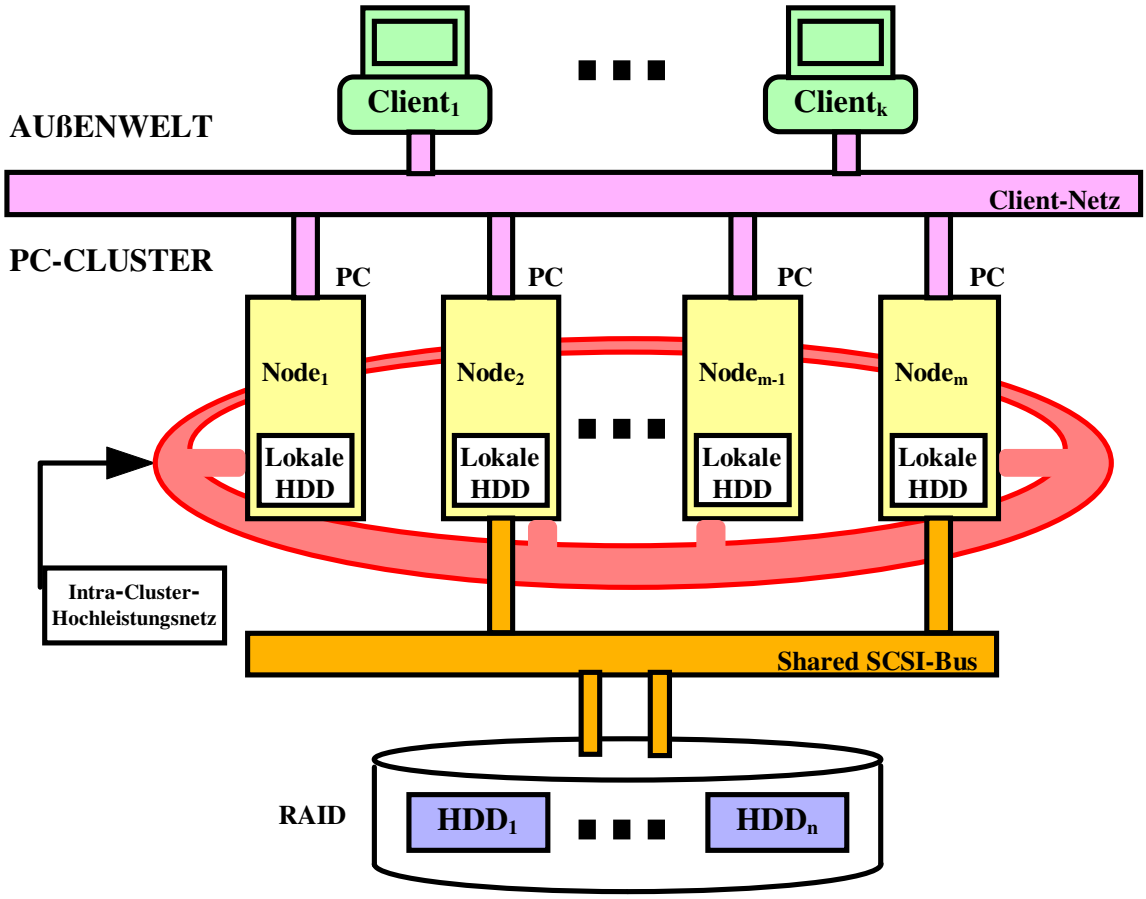


Abbildung 2.1: Mögliche Hardware-Architektur eines PC-Clusters

Das in Kapitel 1.2 erwähnte exemplarische PC-Cluster basiert auf eine von Microsoft im Oktober 1995 angekündigte und im August 1997 als Beta-3-Version herausgebrachte Cluster-Technologie, die folgende Hardware-Hauptbestandteile verwendet:

Speziell		**Allgemein**
Primergy 100	als	PC-Server
(zunächst „nur“) Ethernet	als	Rechnernetz-Transportmedium
Mylex SCSI RAID System	als	Speichersystem
Shared SCSI Bus	als	als Bus für die Kommunikation zwischen Rechensystemen und Speichersystem

Tabelle 2.1: Hardware-Wahl

Dabei stellt „Wolfpack“ nur einen Code-Namen für die o.g. Cluster-Technologie dar und ist mittlererweile veraltet. Demzufolge wird im folgenden durchgängig der Begriff **MSCS** (Microsoft Cluster Server) statt „Wolfpack“ verwendet, da dies der endgültig von Microsoft gewählte Name ist.

Die Bezeichnung MSWEM für den Eventmanager wird jedoch behalten, da sie vor der erwähnten endgültigen Namensgebung vorgenommen wurde.

Funktionale Einteilung

Für die Hauptbestandteile des PC-Clusters existieren zugeordnete **Softwarekomponenten**, die als repräsentierende, funktionale Einheiten verstanden werden, und die ihrerseits in einer eigenen, komplexen, von Abhängigkeiten geprägten Struktur aufgebaut sind. Dieses Verständnis der Softwarekomponenten deutet darauf hin, daß ihre Vereinigungsmenge, d.h. hier die Menge **K** aller Softwarekomponenten, u.a. alle Managed Objects (MOs) und alle möglichen Quellen von Events enthält.

Die zusammenfassende Softwarekomponente eines PC-Servers allgemein wird durch das Betriebssystem gegeben. Dafür wird im MSCS der Microsoft NT 4.0 Server (Enterprise Edition) gewählt.

Die Softwarekomponenten sind hardwareabhängig und eine Einteilung der Hardware in Hauptbestandteile, die dann von Tabelle 2.1 spezialisiert wurden, ist am Anfang dieses Kapitels vorgenommen worden. Allerdings werden die Softwarekomponenten für das/die Rechnernetz/e nicht nur durch die PC-Server realisiert, sondern auch in die Architektur der Softwarekomponenten für PC-Server fest integriert, wobei die Softwarekomponenten für das PC-Cluster an sich (obwohl sie auch durch die PC-Server realisiert werden) auf die der einzelnen PC-Servern nur aufbauen, aber mit einer übergeordneten Funktionalität belegt sind. Unter Berücksichtigung dieser Tatsachen wird nun eine Partitionierung von K vorgenommen. Da jedes Event bei seinem Auftreten eine eindeutig zugeordnete Softwarekomponente als Eventquelle hat, wird die Eventmenge E der Events im PC-Cluster analog zu K partitioniert:

K-Partitionierung	E-Partitionierung	Entsprechende Softwarekomponenten
K_{MLX}	E_{MLX}	Mylex SCSI-RAID Softwarekomponenten
K_{NT}	E_{NT}	Microsoft NT 4.0 Server Softwarekomponenten
K_{MSCS}	E_{MSCS}	Softwarekomponenten des PC-Clusters an sich

Tabelle 2.2: K-und E-Partitionierung

Bemerkung 2.1: Um Mehrdeutigkeiten zu vermeiden, wird im weiteren Verlauf der vorliegenden Diplomarbeit das gesamte PC-Cluster als „Cluster“ und das Cluster an sich als „MSCS“ bezeichnet.

Es wird nun - unter Berücksichtigung der Annahmen A1 und A2 - eine partitionsabhängige Beschreibung des PC-Clusters vorgenommen. Dabei wird i.a. eine allgemeine Beschreibung und eine Erklärung der für den MSWEM wichtigen Softwarekomponenten, deren Zustände, Fehler, Operationen und/oder Eigenschaften angegeben. D.h. es werden ausschließlich die Softwarekomponenten (und implizit deren Zustände, Fehler, Operationen und/oder Eigenschaften) betrachtet, die als Eventquellen der im MSWEM verarbeiteten Events dienen und die impliziten Betrachtungen sind von Events abgeleitet.

2.1 Mylex SCSI-RAID

Eine detaillierte Beschreibung des Mylex SCSI-RAID-Systems wurde im Anhang A.1 erstellt und dient als Referenz für die in diesem Kapitel kurz gehaltene Erklärung (siehe Annahmen A1 und A2) des RAIDs.

2.1.1 Allgemeine Beschreibung

Mit dem Einsatz eines Mylex SCSI-RAID-Systems hat man sich für einen **D**isk **A**rray **C**ontroller 960SX entschieden, der verschiedene SCSI-Formate und RAID-Speicherungsarten (RAID-Levels) unterstützen kann.

Die unterschiedlichen SCSI-Formate sind eventtechnisch nicht von Belang, und der RAID-Level 5 stellt eine optimale Kombination der Plattenausnutzung und der Ausfallsicherheit dar. Demzufolge wird im weiteren Verlauf dieses Kapitels eine kurze Erklärung des SCSI-RAID-Systems in einer RAID-Level 5 Konfiguration angegeben.

In einem Mylex-RAID können mehrere (höchstens 8) Platten zu einer Gruppe zusammengefaßt werden. Dabei können höchstens 8 Gruppen gebildet werden, in denen die Größe der Platten als konstant angenommen wird.

Eine RAID-Level 5 Konfiguration bedeutet, daß sowohl ein sequentieller Datenblock (Striping) als auch redundante Block-Parität (Parity) auf alle Platten einer Gruppe homogen verteilt wird.

2.1.2 Softwarekomponenten

- Controller
  Die wichtigste Softwarekomponente ist der **D**isk **A**rray **C**ontroller 960SX, da er das gesamte RAID steuert. Der Controller kann dual/parallel zu einem zweiten, identischen Controller betrieben werden (**dual-active-configuration**), wenn beide gleiche Anzahl an Host- und Platten-Kanäle und gleiche Anzahl an PHYSDEVs haben. In dieser Konfiguration kann jeder Controller ein Fehler seines dualen Controllers erkennen und automatisch dessen Funktionen übernehmen (fail-over).

- Physikalische Platten/Laufwerke
  Der RAID-Controller kann 5 SCSI-Kanäle a 6 physikalischen Platten (physical drives - **PHYSDEV**) unterstützen, d.h. insgesamt 30 Platten.

- Logische Platten/Laufwerke
  PHYSDEVs können zu einer logischen Platte (**L**ogical **UN**it, system drive - **SYSDEV**) so zusammenfaßt werden, daß man höchstens 8 SYSDEVs erzeugt

- Cache
  Dient zur Speicherung der Statusunterschiede in den PHYSDEVs und im Fall einer dual-active-configuration zur Inter-Controller redundanten Speicherung von Schreib-Daten

Folgende Softwarekomponenten sind sehr detailliert im Statusmanagement beschrieben:
- Lüfter
- Temperatursensoren
- Stromversorgungseinheiten

2.1.3 Zustände und Fehler

Zustände

- Physikalische Platten/Laufwerke
  ONLINE: Stromversorgt, gruppenzugehörend und richtig funktionierend,
  STANDBY/ HOTSPARE: Stromversorgt, keiner Gruppe gehörend und kann beim Ausfall einspringen,
  OFFLINE: Nicht betriebsbereit,
  DEAD: Ausgefallen.
- Logische Platten/Laufwerke
  ONLINE: Zugeordnete PHYSDEVs sind ausnahmslos ONLINE,
  CRITICAL: Nur 1 ausgefallenes zugeordnetes PHYSDEV
  OFFLINE: Ausgefallen, mindestens 2 ausgefallene zugeordnete PHYSDEVs

Fehler

- Physikalische Platten/Laufwerke
  PFA: Predictive Failure Analysis,
  MISC_ERROR: Automatische Erkennung von Fehlern durch Kommando-Timeouts,
  PARITY_ERROR: Automatische Erkennung von Fehlern durch Kommando-Timeouts,
  SOFT_ERROR: Nach acht Mal wird Platte Dead,
  HARD_ERROR: Platte wird Dead,

2.1.4 Operationen

- Physikalische Platten/Laufwerke
  COMMAND: Bewirken Timeouts (siehe Fehler),
  REBUILD: Falls RAID über STANDBY-PHYSDEV verfügt, findet ein automatischer, benutzertransparenter Wiederaufbau statt.

- Logische Platten/Laufwerke
  CHECK: Konsistenzüberprüfung der redundanten Parity-Information,
  REBUILD: Falls RAID über STANDBY-PHYSDEV verfügt, findet ein automatischer, benutzertransparenter Wiederaufbau statt.

2.2 Microsoft Windows NT Server 4.0

Die architektonischen Grundlagen von Microsoft Windows NT Server 4.0 (kurz: NT) sind komplex. Bei der Untersuchung der NT-Events wird eine Systematisierung und ein Eventkatalog der NT-Netz-Events (siehe Kapitel 3.4) angegeben. Das vorliegende Kapitel versteht sich - die Annahme A1 berücksichtigend- als Basis-Dokumentation der o.g. Untersuchung. In diesem Sinne werden hier zunächst einige Grundlagen beschrieben (Kapitel 2.2.1) und anschließend werden im Hinblick auf die NT-Netz-Events einige Softwarekomponenten und Operationen von NT erklärt.

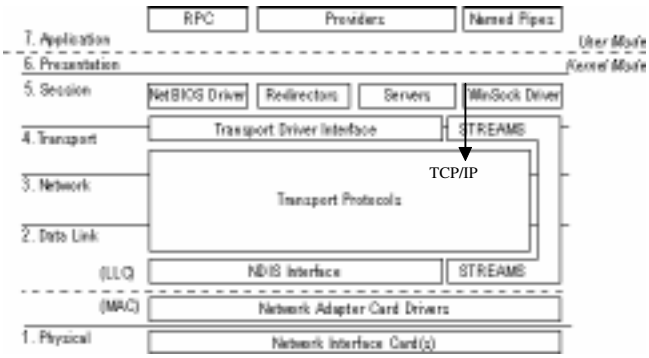
2.2.1 Allgemeine Beschreibung

Um das Ziel dieses Kapitels zu erreichen, wird zunächst ein kurzer Überblick der NT-Netz-Architektur unterbreitet. Danach werden allgemeine und wichtige Begriffe beschrieben.

NT-Netz-Architektur

Die NT-Netz-Architektur wird im Groben von der folgenden Abbildung beschrieben:

Abbildung 2.2: NT-Netz-Architektur



NT-Domänen

NT Arbeitsgruppen (workgroups) können im Gegensatz zu Domänen nicht mehrere zugeordnete TCP/IP Subnetze besitzen. Jede NT Arbeitsgruppe, der mehrere TCP/IP Subnetze zugeordnet sind, verhält sich de facto wie zwei verschiedene Arbeitsgruppen, die gleichen Namen haben. Deswegen werden im Verlauf der weiteren Diplomarbeit nur noch Domänen betrachtet. Diese stellen eine logische Gruppierung von Servern und Arbeitsstationen dar, die eine gemeinsame Verwaltung von Benutzer-Informationen haben. Dadurch wird eine einheitliche Sicht des Benutzers auf die Domäne und eine zentrale Administration eines Benutzers in der Domäme geschaffen.

Registry

Die Registry (auch „Configuration File“) ist eine System-Datenbank, die auch von Anwendungen zur Speicherung von Konfigurationsdaten verwendet werden kann. Sie hat eine spezielle baumartige Struktur, wobei die einzelnen Knoten „Keys“ bezeichnet werden. Die Keys können „Subkeys“ (Nachfahren im Baum) und/oder parametrisierte Daten, die sogenannten „Values“, enthalten. Dabei können Konfigurations-Informationen durch die Values eines Keys gespeichert werden oder manchmal durch bloße Anwesenheit eines Keys. Folgende Abbildung zeigt auf der linken Seite die baumartige Struktur, in der ein Subkey

„LanmanWorkstation“ (siehe auch Dienste) markiert ist, und auf der rechten Seite die Values des entsprechenden Subkeys:

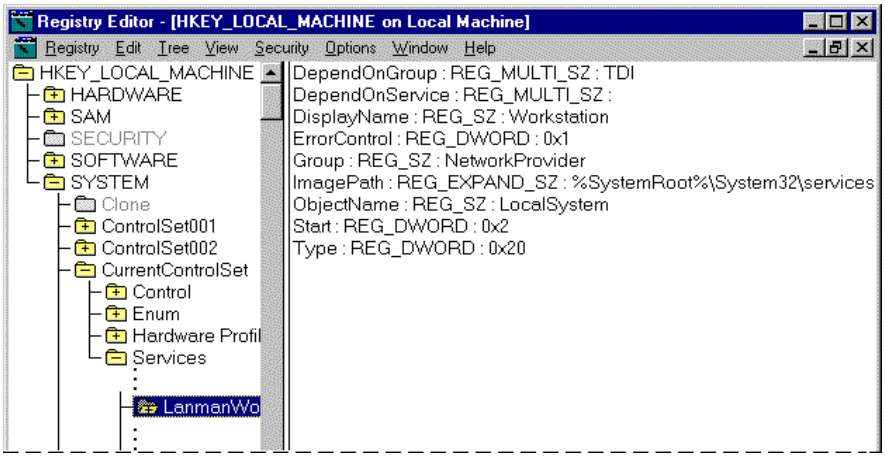


Abbildung 2.3: Registry-Struktur

2.2.2 Softwarekomponenten

Die Softwarekomponenten, die Eventquellen der vom MSWEM verarbeiteten Events darstellen, sind sehr anzahlreich. Dabei stellen Dienste den größten Anteil dieser Softwarekomponenten dar und sind in einer eindeutigen Abhängigkeitsstruktur angeordnet.

Allgemein bekannte Softwarekomponenten, die in NT ihre Bedeutung beibehalten haben, sind:
- Lokaler Plattenplatz,
- Netz,
- Netz-Ressource,
- Stromversorgung,
- System,
- System-Ressource,
- System-Uhr,
- UPS (Uninterruptible Power Supply),
- Verzeichnis.

Diese werden hier nicht weiter beschrieben.

Im folgenden wird eine Erklärung der Softwarekomponenten angegeben, wobei die Untersuchung der Dienste wegen den o.g. Überlegungen besonders gehandelt wird.

- DLL (Dynamic-Link Libraries)
  Windows-typische Datei, die einen Mechanismus zur Modularisierung von Anwendungen realisiert. Eine DLL enthält Daten und Funktionen, die getrennt von den benutzenden Prozessen (i.d.R. in Executables laufend) compiliert, gelinkt und gespeichert werden. Dabei kann die DLL vom Betriebssystem in den Adreßraum des aufrufenden Prozesses auch während er rechnet eingebunden werden. Die Win32 API ist z.B. durch DLLs realisiert.
  DLLs werden von einigen NT-Dienste eingebunden und stellen die Basis der Resource-Typen von MSCS dar (siehe Kapitel 2.3) .

- PDC (Primary Domain Controller)
  Allgemein stellen Domänen Controller Server dar, die ein Shared Directory für die Speicherung der Interaktionen zwischen Benutzer und Domäne zur Verfügung stellen. Dabei werden diese Informationen bei NT in dem SAM gehalten (siehe nächste Softwarekomponente). In diesem Zusammenhang gibt es:

- Einen eindeutigen PDC, der die o.g. Informationen in seinem SAM für die Domäne zentral verwaltet und
- mehrere BDCs (Backup Domain Controller), die eine Kopie des PDC-SAMs beinhalten. Diese muß periodisch per Synchronisation (siehe Operationen) aktualisiert werden.

- SAM (Security Account Manager)
  Diese Softwarekomponente, die auch „directory database“ oder „security database“ genannt wird, speichert und verwaltet alle Benutzer-Accounts, -Paßwörter und -Rechte.

Dienste

Dienste können in der Registry unter dem Key
„HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\“
konfiguriert werden. Sie haben eine komplexe Abhängigkeitsstruktur, die im Hinblick auf den MSWEM besonders wichtig ist. Demzufolge wird diese gesondert betrachtet. Folgende Beschreibungen werden nur für zentrale Dienste vertieft angegeben, wobei sie sich für die restlichen Dienste auf das Wesentliche beschränken.

- Workstation Dienst („LanmanWorkstation“)
  Jeder Datenzugriffswunsch, der im user-mode abgesendet wird, wird von einer user-mode Schnittstelle (in Services.exe enthalten) des Workstation Dienstes empfangen. Dabei wird er dann folgendermaßen bearbeitet:
  - Falls die Daten lokal gespeichert sind, wird die Anfrage an einen entsprechenden Dateisystem-Treiber weitergeleitet,
  - Falls sich die Daten auf einen Remote-Computer befinden, wird die Anfrage an den sogenannten Redirector (durch Rdr.sys realisiert) weitergeschickt. Dieser ist als kernel-mode Dateisystem-Treiber implementiert und befindet sich direkt über der TDI. Dadurch kann er mittels TDI den Datenzugriffswunsch an die darunter liegenden Netztreiber weiterleiten, die dann die Kommunikation mit dem Remote-Computer realisieren.

  Die Arbeitsweise des Workstation Dienstes wird in der folgenden Abbildung graphisch dargestellt:

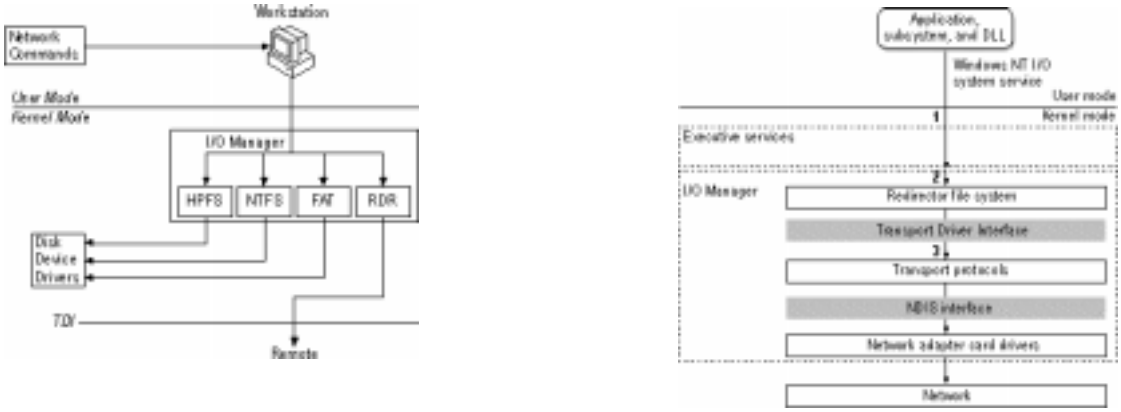


Abbildung 2.4: Worstation-Dienst

- Server Dienst („LanmanServer“)
  Der Server Dienst ist in der Services.exe realisiert besteht aber im Wesentlichen aus einem Server Treiber, namens SRV (Srv.sys). Dieser stellt das Server-seitige Gegenstück zum Redirector dar und ist ebenfalls als Dateisystem-Treiber, der direkt über der TDI zu lokalisieren ist, implementiert. Demzufolge kann er mit den eigentlichen Dateisystem-Treiber kommunizieren, um Client-seitigen Datenzugriffswünsche entsprechend genügen zu können: Er empfängt die Daten von den Dateisystem-Treiber und sendet sie über TDI und Netztreiber an die anfragenden Systeme.
  Die folgende Abbildung beschreibt den Server-Dienst:

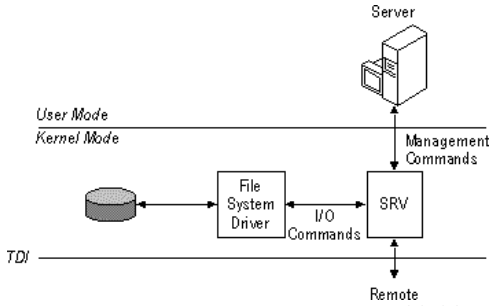


Abbildung 2.5: Server-Dienst

- Alerter Dienst
  Dieser Dienst wird verwendet um Computer oder Benutzer von System-Alarmen in Kenntnis zu setzen.

- Messenger Dienst
  Der Messenger Dienst sendet bzw. empfängt Nachrichten, die vom Alerter Dienst oder von einem Administrator geschickt wurden bzw. erwartet werden.

- NetLogon Dienst
  Dieser Dienst realisert für Domänen-Benutzer eine eindeutige Zugangsstelle zum PDC bzw. zu den BDCs. In diesem Sinne ist er in die Authentifikation für Domänen-Benutzer involviert (siehe Operationen).
  Außerdem ist er für die Durchführung der Synchronisation (siehe Operationen) zuständig.

- Replikator Dienst
  Der Replikator Dienst ist für das Updaten ausgewählter Verzeichnisse von einem sogenannten Export Server auf seinen Clients.

- Dienst-Controller oder Service Control Manager (SCM)
  Der Dienst-Controller (Services.exe) wird an dieser Stelle nicht als Dienst sondern als Softwarekomponente mit Dienst-Bezug angegeben. Er ist ein RPC Server und stellt durch folgende Aufgaben einen Kontrollmechanismus für Dienste dar:
  - Verwaltung eine Datenbank von installierten Diensten
  - Starten und Stoppen von Diensten,
  - Aufzählung installierter Dienste,
  - Verwaltung von Status-Informationen bezüglich Dienst-Ausführungen,
  - Absenden von Kontrollabfragen an laufende Dienste.

Mehrere Dienste können, je nach Funktionalität und Startart, zu einer Dienst-Gruppe zusammengefaßt werden. Dies ist in NT für einige Dienste vorgenommen worden, so daß eine Reihenfolge der zu startenden Dienst-Gruppen beim Hochfahren angegeben werden kann. Einige NT-Dienst-Gruppen werden in der Tabelle 2.3 angezeigt:

Dienst-Gruppe	Enthaltene Dienste
NDIS	EE16, NDIS
Network	Mup, Rdr, Srv
NetworkProvider	LanmanWorkstation
PNP_TDI	NetBT, Tcpip
RemoteValidation	NetLogon
TDI	Afd, DHCP

Tabelle 2.3: Dienst-Gruppen

Dienst-Abhängigkeiten

Dienst-Informationen werden durch Values in den entsprechenden Dienst-Subkeys des o.g. Keys gespeichert. Die für Dienst-Abhängigkeiten wichtige Values sind „DependOnGroup“ bzw. „DependOnService“, die die Abhängigkeit des jeweiligen Dienstes von der Dienst-Gruppe bzw. von anderen Diensten anzeigen. Folgende beiden Tabellen geben Bespiele für Dienst- und Dienst-Gruppen-Abhängigkeiten an:

Dienst	Dienst-Gruppen-Abhängigkeit
LanmanServer	TDI
LanmanWorkstation	TDI
NetBIOS	TDI
LmHosts	Network Provider

Tabelle 2.4: Dienst-Gruppen-Abhängigkeiten

Dienst	Dienst-Abhängigkeit
Alerter	LanmanWorkstation
Browser	LanmanWorkstation, LanmanServer, LmHosts
DHCP	Afd, NetBT, TCP/IP
Messenger	LanmanWorkstation, NetBIOS
NetBT	TCP/IP
NetLogon	LanmanWorkstation, LmHosts
Replicator	LanmanServer, LanmanWorkstation

Tabelle 2.5: Dienst-Abhängigkeiten

In diesem Kontext hat eine Untersuchung der entsprechenden Registry-Subkeys folgende Dienst-Abhängigkeiten für die vorher erklärten Dienste ergeben:

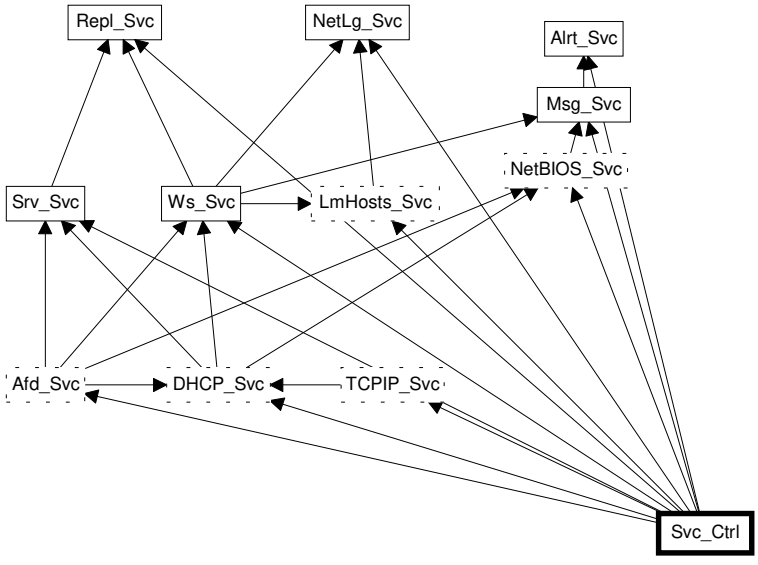


Abbildung 2.6: Dienst-Abhängigkeiten

Dabei stellen die mit gestrichelten Ränder angezeigten Dienste bezogene Dienste dar, die nicht direkt für die weiteren Ausführungen in dieser Diplomarbeit relevant sind. Durch die Abhängigkeiten des Dienst-Controllers (fett umrandet) wird seine hohe Bedeutung ersichtlich.

2.2.5 Operationen

An dieser Stelle werden bis auf UPS-Shutdown nur Operationen vorgestellt, die nicht direkt vom MSWEM behandelt werden. Ihre Erklärung ist jedoch sehr wichtig im Hinblick auf das Verständnis der Eventanalyse und der Eventkataloge.

- Authentifikation
  Der Prozeß der domänenweiten Überprüfung von Accounts, Paßwörter und Rechte bei einer Benutzer-Anmeldung. Auf lokaler Ebene sind nur die SAMs einbezogen während für Remote-Überprüfungen auch der NetLogon Dienst verwendet wird. Die folgende Abbildung soll dies verdeutlichen

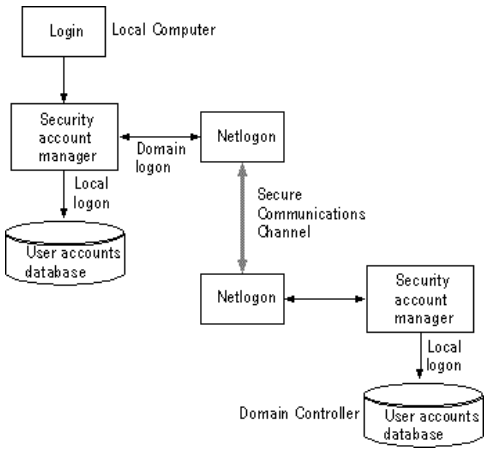


Abbildung 2.7: Authentifikation bei Benutzer-Anmeldungen

- Replikation
  Der Vorgang des Updatens von Verzeichnissen, der vom Replikator Dienst durchgeführt wird. Die Verzeichnisse werden durch Konfigurations-Einträge des Replikator Dienstes gegeben.

- Synchronisation
  Synchronisation bedeutet der Abgleich von BDC-SAMs mit PDC-SAMs. In NT werden zwei Arten von Synchronisation durchgeführt:
  „Partial Synchronization“: Stellt einen inkrementellen, periodischen Abgleich dar,
  „Full Synchronization“: Führt einen vollständigen Abgleich aus, wenn ein neuer BDC oder wenn keine Information für die Inkremente mehr vohanden ist.

- UPS_Shutdown
  Diese Operation fährt einen Rechner herunter. Sie wird von der UPS, die eigentlich als Dienst realisiert ist, bei Problemen mit der Stromversorgung angestoßen. Dabei wird Benutzern die Möglichkeit gegeben, in einer Konfigurationsdatei Befehle zu hinterlegen, die vor einem Shutdown noch ausgeführt werden sollen.

2.3 Microsoft Cluster Server (MSCS)

Microsoft Cluster Server (MSCS) ist ein Enclosed Cluster (siehe Abbildung 2.1), der die Verfügbarkeit der Client-Anwendungen als primäres Ziel hat.

Der MSCS wird in diesem Kapitel ausschießlich auf Event-relevante Hintergründe untersucht. In einer vorangehenden Diplomarbeit über das Statusmanagement wurde der MSCS als „Eine PC-Cluster-Architektur“ (Kapitel 3) beschrieben. Um Wiederholungen zu vermeiden (Annahme A2) und im Hinblick auf dem MSWEM (Annahme A1) wird hier auf eine allgemeine Beschreibung und auf eine Erklärung der Softwarekomponenten verzichtet und nur eine Auflistung der einzelnen Zustände und Operationen angegeben. Dies wird zusätzlich durch die spätere Feststellung, daß die MSCS-Events nur indirekt über Zustände und Operationen erfaßt werden können, motiviert.

Um eine möglichst aktuelle Version zu untersuchen, wurden die Zustände und Operationen aus Header-Dateien herausgearbeitet.

2.3.1 Zustände

Bei der Untersuchung der Zustände wurde mit „nicht aktiv“ eine Softwarekomponente bezeichnet, die am Clusterbetrieb nicht teilnimmt. Zustände für die keine, keine genaue oder keine ausreichende Dokumentation vorhanden war, werden mit „doku“ bezeichnet.

- Node-Zustände:
  - *unknown:* unbekannt,
  - *up:* richtig funktionierend,
  - *down:* ausgefallen,
  - *paused:* nicht aktiv,
  - *joining:* tretet dem Cluster bei.

- Gruppen-Zustände:
  - *unknown:* unbekannt,
  - *online:* richtig funktionierend,
  - *offline:* nicht aktiv,
  - *failed*: ausgefallen,
  - *partial online:* mindestens eine zugeordnete Ressource nimmt am Clusterbetrieb nicht teil.

- Ressourcen-Zustände:
  - *unknown:* unbekannt,
  - *inherited:* doku,
  - *initializing:* doku,
  - *online:* richtig funktionierend,
  - *offline:* nicht aktiv,
  - *failed:* ausgefallen,
  - *pending:* doku,
  - *online pending* versucht in den Online-Zustand zu kommen,
  - *offline pending:* versucht in den Offline-Zustand zu kommen.

- Netz-Zustände:
  - *unknown:* unbekannt,
  - *up* richtig funktionierend,
  - *down* ausgefallen,
  - *unavailable* doku,
  - *partitioned* doku.

- NIC-Zustände:
  - *Unknown:* unbekannt,
  - *Unavailable:* doku,
  - *Failed:* ausgefallen,
  - *Unreachable:* doku,
  - *Up:* richtig funktionierend.

2.3.2 Operationen

- Nodes:
  - *pause*: Node wird kurzzeitig aus dem Clusterbetrieb ausgeschlossen,
  - *resume*: Node wird in den Clusterbetrieb eingebracht,
  - *evict*: Node wird aus dem Clusterbetrieb gelöscht,
  - *join*: Node wird in den Clusterbetrieb eingefügt.

- Gruppen:
  - *create*: Einfügen,
  - *delete*: Löschen,

bringOn: Bringe Gruppe im Online-Zustand,
takeOff: Bringe Gruppe im Offline-Zustand,
moveTOnode: Bewege Gruppe auf anderen Node.

- Ressourcen:
  create: Einfügen,
  delete: Löschen,
  bringOn: Bringe Ressource im Online -Zustand,
  takeOff: Bringe Ressource im Offline-Zustand,
  initiate failure: Bringe Ressource im Offline-Zustand - zum Testen von Failover
  moveTOgroup: Bewege Ressource zu anderer Gruppe.

- Netz:
  enable: Aktiviere Netz für Clusterbetrieb,
  disable: Desaktiviere Netz für Clusterbetrieb.

2.3.3 NT-Integration

In Zusammenhang mit dem MSWEM ist an dieser Stelle die Einbettung des MSCS in NT ein wichtiges Untersuchungsziel.

Die Integration von MSCS in NT erfolgt durch vier Dienste, die an den Dienst-Controller angebunden werden:
- ClusterDisk-Dienst,
- ClusterNet-Dienst,
- Cluster-Dienst,
- Time-Dienst.

Die Abhängigkeiten dieser vier Dienste - in NT eingebettet - werden von Abbildung 2.8 aufgezeigt. Dabei stellen die mit gestrichelten Ränder angezeigten Dienste bezogene Dienste dar, die nicht direkt für weiteren Untersuchungen in dieser Diplomarbeit relevant sind.

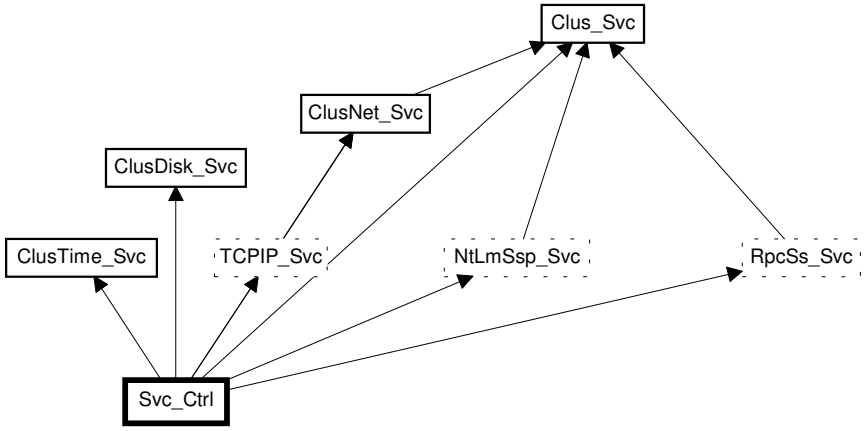


Abbildung 2.8: Cluster-Dienst-Abhängigkeiten

Ein weiterer Aspekt der NT-Integration ist was sich für die NT-Rechner ändert oder was vorausgesetzt wird, wenn sie einer MSCS-Installation unterliegen.

Durch den Bezug des MSCS auf NT können folgende Konfigurationsannahmen getroffen werden:
- Die NT-Rechner sind in einer Domäne mit einem PDC und mindestens einem BDC,
- Auf allen Nodes ist die gleiche NT-Version installiert,

- Alle Netz-Software-Versionen sind gleich,
- Es gibt keine Drucker im Cluster.

3. Systematisierung der Events

Das vorliegende Kapitel eine umfassende Systematisierung aller Events im Cluster dar und leitet damit die Konstruktion des MSWEM (siehe Kapitel 4) ein. Dies wurde aufgrund einer eingehenden Untersuchung der Event-Meldungen - unter Berücksichtigung der Informationen aus dem Kapitel 2 - durchgeführt. Die Systematisierung stellt eine Art Filterung der Event-Meldungen dar und hat als Ziel sinnvolle Kriterien für das Eventmanagement zu synthetisieren. Das Ergebnis ist ein Eventkatalog, der die einzelnen Events und ihre Kriterienwerte enthält und der wegen seines Umfangs in den Anhang plaziert wurde.
Um dies zu erreichen werden am Anfang einige Begriffe eingeführt, auf die sich die darauffolgenden Untersuchungen beziehen. Das Kapitel 3.2 stellt die o.g. Kriterien auf. Im weiteren Verlauf des Kapitels wird unter Hinzunahme der eingeführten Begriffe und der Ergebnisse aus 3.2 eine Systematisierung der Events vorgenommen. Dabei wird entsprechend der E-Partitionierung aus Tabelle 2.2 jeweils für Mylex-Events, NT-Events und MSCS-Events je ein Unterkapitel (3.3 - 3.5) aufgewendet. Demzufolge gliedert sich auch der o.a. Eventkatalog in entsprechenden drei Eventkataloge auf. Im letzten Unterkapitel 3.6 wird eine Zusammenfassung und Erkenntnisse der Vorkapiteln präsentiert.

3.1 Begriffe

Dieses Kapitel führt einige Begriffe ein, die für das allgemeine Verständnis eines Events notwendig sind. Dies ist notwendig um eine einheitliche Sprache bei der anstehenden Systematisierung der Events verwenden zu können.

Für eine Eventmenge E gibt es einen zugehörigen **Kombinationsraum**, der alle möglichen Kombinationen von Events enthält die ein Eventmanager verarbeiten muß. Theoretisch stellt also der Kombinationsraum die Potenzmenge von E dar und hat dadurch eine Maximalzahl gleich $2^{|E|}$ von Kombinationen. Der Kombinationsraum der Cluster-Events wird noch im weiteren Verlauf dieses Kapitels beschrieben.

Jede Softwarekomponente befindet sich zu einem gegebenen Zeitpunkt in einem gewissen Zustand. Von einem Zeitfortschritt abhängig, kann sich der Zustand einer Softwarekomponente ändern.
Die Art und Weise, in der aus gegebenen Zuständen/Zustandsänderungen von Softwarekomponenten ein/e Clusterzustand/Clusterzustandsänderung gefolgert werden kann, wurde in einer eigenen, vorangehenden Diplomarbeit über Statusmanagement untersucht. Deswegen geht man in der vorliegenden Arbeit von einem generischen Mechanismus zur Bildung eines Clusterzustandes und zur Beschreibung der Clusterzustandsänderungen als gegeben aus.

Eine Softwarekomponente k2 kann von einer anderen Softwarekomponente k1 abhängig sein, d.h. daß eine Zustandsänderung von k1 eine Zustandsänderung von k2 bewirken könnte. Eine derartige Beziehung zwischen Softwarekomponenten wird **Abhängigkeit** genannt.

Im laufenden Clusterbetrieb finden verschiedene Aktionen (wie z.B. Programm- / Unterprogrammausführungen, Überprüfungen, An- / Abmeldungen, Initialisierungen, Terminierungen, Synchronisationen, Anforderungen) statt. Hiermit werden all diese Aktionen zu **Operationen** verallgemeinert, deren Ausführungen i.d.R. von einer Softwarekomponente ausgelöst werden und die sich primär durch Zustandsänderungen von einer oder mehreren Softwarekomponenten auswirken. Die auslösenden Softwarekomponenten werden **initiierende** Softwarekomponenten und die Softwarekomponenten, auf denen sich die Zustandsänderungen beziehen, werden **implizierte** Softwarekomponenten genannt.

Im Kapitel 1.1 wurde behauptet, daß die Eventquelle eines Events die Softwarekomponente sei, die für sein Auftreten zuständig ist. Hiermit wird diese Behauptung erweitert, indem man als Eventquelle q(e) = **q** für ein Event auch Operationen als zusätzliche Verantwortungsträger für das Auftreten zuläßt. Die Motivation dieser Erweiterung wird im Kapitel 3.2.2 beschrieben. Die Menge aller Eventquellen einer Eventmenge E wird mit Q(E) = **Q** bezeichnet.

Als Gegenteil einer Eventquelle, die Events auslöst, wird an dieser Stelle der Begriff eines **Eventbrowsers** eingeführt, der für das Anzeigen von tatsächlich aufgetretenen Events zuständig ist. Wenn eine Eventquelle

ein Event sendet, spricht man von **Event-Logging** und analog wird das Anzeigen eines Events durch einen Eventbrowser als **Event-Browsing** bezeichnet.

Ein Event hat eine eindeutige **Identifikationsnummer,** und ein **Meldungstext**, der beim Auftreten des Events angezeigt wird, zugeordnet.

Jedem Event und jeder Eventmenge wird ein Informationsgehalt zugeordnet. Der Informationsgehalt eines Events stellt die Aussage - die tatsächliche Information dar, die man beim Auftreten eines Events über seine Eventquelle in Erfahrung bringen kann. In diesem Sinne wird der Informationsgehalt eines Events als Maß für die Information über

- den Zustand von k, falls die Eventquelle eine Softwarekomponente $k \in K$ ist,
- die einzelnen Zustandsübergänge, die eine Operation als Eventquelle auslösen könnte,

in dieser Diplomarbeit verstanden.
Man spricht von einem „guten“ Event bzw. von einer „guten“ Eventmenge, je höher sein bzw. ihr Informationsgehalt ist. Es kommt häufig vor, daß der Informationsgehalt eines Events nicht aus dem Meldungstext ersichtlich wird (siehe Ziel „Weiterleitung von Events mit kontextbezogenen, hohen Informationsgehalt“).

Das Auftreten eines Events geschieht zu einem gewissen Zeitpunkt, der in einer aufsteigend sortierten Zeitachse eingeordnet werden kann.

Als Zusammenfassung dieses Kapitels wird nun folgendes angegeben:

> Ein Event besitzt eine eindeutige Identifikationsnummer, einen frei wählbaren Meldungstext, eine auslösende Quelle und einen Zeitstempel. Der Informationsgehalt eines Events ist selten aus dem Meldungstext ersichtlich.

3.2 Gruppierungen

Um eine Systematik in E hineinzubringen wird E nach verschiedenen Kriterien, die durch die Beschreibung des Clusters (siehe Kapitel 2) und durch die in der Dokumentation verfügbaren Eventinformationen motiviert sind, gruppiert. Ziel des vorliegenden Kapitels ist es diese Kriterien und ihre entsprechenden Gruppierungen vorzustellen. Dabei werden alle einzelnen Werte der Gruppierungen angegeben und die in den Eventkatalogen (siehe Anhang B) verwendeten Abkürzungen dieser Werte werden in Klammern notiert.

Beispiele von Events werden in der Notation „*Eventnummer: Meldungstext*“ angegeben.

3.2.1 Gruppierung nach der Eventberücksichtigung

Diese Art von Gruppierung zeigt an, ob die Events von dem MSWEM bei der Korrelation berücksichtigt werden oder nicht und stellen damit eine a priori Filterung dar. Das bedeutet, daß die Events ganz gewöhnlich empfangen und damit auch registriert werden, aber sie werden - falls sie nicht berücksichtigt werden sollen - einfach nicht verarbeitet und auch nicht weitergeleitet. Grund dieser Gruppierung ist folgende Feststellung, die bei der Erstellung der Eventkataloge aufgefallen ist:

> Wegen der **vorbestimmten Konfiguration** (Kapitel 2) oder einem **nicht aufschlußreichen Informationsgehalt** von Events ist ein Verarbeiten oder Weiterleiten sinnlos.

Demzufolge werden hier folgende Gruppierungswerte sind:

(1)	Ja	Events werden berücksichtigt	(J),
(2)	Nein	Events werden nicht berücksichtigt	(N).

Die o.a. Gründe, die zu einer nicht-Berücksichtigung führen können, lassen sich - zusammenfassend für die drei E-Partitionen - in weitere Kriterien detaillieren:

Vorbestimmte Konfiguration

- Konfiguration
  Beispielsweise setzt das Event „*2115: The queue is empty*.“ einen Drucker voraus, der bei MSCS nicht konfiguriert wird.

- Fehlende Implementierung
  Der in der Dokumentation vorkommende Event „*26: MLXEV_PHYSDEV_ACTIVESPARE*“ ist z.B. nicht implementiert. Da zusätzlich die Mylex-Dokumentation i.W. aus einer Header-Datei besteht, kann hier auch keine Erklärung des Events angegeben werden.

Nicht aufschlußreicher Informationsgehalt der Events

Hierfür läßt sich eine Erklärung der Beispiele aufgrund des Kriteriums an sich nicht angeben.

- Fehlende Dokumentation
  Das NT-Event „*3732: Default permissions cannot be set for that resource*.“ hat z.B. keine Erklärung in der Dokumentation.

- Nicht aufschlußreiche Dokumentation
  „*3534: The service did not report an error*.“ wird beispielsweise von folgendem Text „Try the task later. If the problem persists, contact your network administrator.“ erklärt. Jedoch kann man diesem Text keinen brauchbaren Informationsgehalt entnehmen.

Die Eventberücksichtigung ist also eine Gruppierung, die die Zulassung zur Verarbeitung der Events im MSWEM steuert.

3.2.2 Gruppierung nach der Eventquelle

Diese Gruppierung teilt die Events nach ihren Eventquellen $q \in Q$ ein (siehe Kapitel 3.1). An dieser Stelle sollte geklärt werden, was genau unter einer Eventquelle verstanden wird. In vorherigen Kapiteln wurde einer Eventquelle immer das Auslösen eines Events oder die Verantwortung für den Auftreten eines Events zugeschrieben. Allerdings hat aber diese Formulierung zwei mögliche Bedeutungen:

1) Die Eventquelle ist die Instanz, die für das technische Senden einer Event-Meldung zuständig ist,
   oder
2) Die Eventquelle ist eine Softwarekomponente oder Operation, auf die sich der Informationsgehalt eines Events bezieht.

Da eines der allgemeinen Ziele eines Eventmanagements die „Weiterleitung von Events mit kontextbezogenen, hohen Informationsgehalt“ ist und da ein derartigen Informationsgehalt die Angabe „WORÜBER eine Aussage getroffen wird“ impliziert, wird hiermit für eine Eventquelle die zweite Bedeutung, also 2) gewählt. Nun stellt sich die Frage, warum auch Operationen als Eventquellen modelliert werden. Zur Beantwortung werden folgende Überlegungen angestellt:

a) Durch ihre Definition in Kapitel 3.1 sind Operationen von ihren initiierenden Softwarekomponenten abhängig und bewirken Abhängigkeiten zu ihren implizierten Softwarekomponenten,
b) In der vorgegebenen Cluster-Architektur kommen Operationen mit mehreren initiierenden Softwarekomponenten und mit mehreren implizierten Softwarekomponenten vor (z.B. moveTOnode im MSCS: von allen Nodes zu allen Gruppen). Sei n die Anzahl der initiierenden Softwarekomponenten und m die Anzahl der implizierten Softwarekomponenten,
c) Demzufolge (aus a und b) wird mit einer Modellierung der Operationen als eigenständige Eventquellen eine Reduktion der Abhängigkeiten-Anzahl zwischen zwei Eventquellen von (n * m) auf (n + m) erreicht,
d) Das allgemeine Ziel „Fehlerlokalisierung“ wird im MSWEM über eine Abhängigkeitssuche der Eventquellen realisiert werden. Demnach ist eine geringe Anzahl an Abhängigkeiten wünschenswert,
e) Es gibt Events, die Zustände von Operationen bzw. ihrer Ausführung angeben (z.B. für Rebuild eines Physikalischen Laufwerks bei Mylex).

Wegen c), d) und e) werden die Vorteile einer Modellierung der Operationen als eigenständige Eventquellen ersichtlich.

Die Frage nach einer Identifikation der Eventquelle für ein gegebenes Event läßt sich meistens durch Untersuchung der Dokumentation lösen. Falls dies nicht der Fall ist, liegt eine nicht aufschlußreiche Dokumentation vor und die Events werden nicht berücksichtigt (siehe Kapitel 3.2.1).

Warum eine Eventquelle als Gruppierungskriterium gewählt wird, ist durch das spätere Korrelations-Vorgehen des MSWEM, der die Events für verschiedene Eventquellen unterschiedlich behandelt, motiviert. Außerdem wird die Begründung dieser Gruppierung im Hinblick auf das allgemeine Eventmanagement-Ziel „Gezielte Weiterleitung von Events an zuständige Administratoren“ durch folgende Überlegungen ersichtlich:

- Die Gruppierungswerte ordnen den Events implizit Softwarekomponenten oder Operationen zu,
- In einer vernetzten Rechnerstruktur gibt es vordefinierte Zuständigkeiten von Administratoren zu Softwarekomponenten oder Operationen,
- Also werden durch Gruppierungswerte implizit Event-Administrator-Zuordnungen geschaffen.

Da die Eventquellen neben Operationen hauptsächlich Softwarekomponenten darstellen und diese für die einzelnen Partitionen K_{MLX}, K_{NT} und K_{MSCS} von K sehr unterschiedlich sind, werden die einzelnen Werte dieser Gruppierung dementsprechend erst in den Kapiteln 3.3, 3.4 und 3.5 beschrieben.

3.2.3 Gruppierung nach der Eventinterpretation

Diese Gruppierung ist aus der Notwendigkeit eines „Verstehens der Aussageart“ von Events durch den MSWEM entstanden. Demzufolge wurde eine weitere Analyse der Event-Dokumentationen vorgenommen, um den Informationsgehalt bezüglich der Frage „WAS wird ausgesagt“ zu untersuchen. Um eine unnötige Erschwerung der Bearbeitung von Events im MSWEM durch Semantik-Verfeinerungen zu vermeiden, ist man bestrebt eine kleine Anzahl an Gruppierungswerten zu erzeugen.

1. Untersuchung

Die durch die Eventberücksichtigung und den Annahmen (siehe Kapitel 4.2) gefilterten Events von dieser Untersuchung wurden ausgeschlossen.

Das erste, nicht auf ein allgemeines Eventmanagement-Ziel bezogene Ergebnis der Untersuchungen war eine hohe Anzahl an möglichen Gruppierungswerten. Dies beruht hauptsächlich auf der Betrachtung der Erklärungen und vorgeschlagenen Handlungen bei den NT-Netz-Events.

Demzufolge wurde eine allgemeine Eventmanagement-Ziel bezogene Untersuchung erforderlich. Hierfür wurden folgende Überlegungen angestellt:

- Die allgemeine Eventmanagement-Ziele „Minimierung der Anzahl zu verarbeitender Eventmengen“, „Vermeidung von Eventstorms“ und „Gezielte Weiterleitung von Events an zuständige Administratoren“ können hier nur indirekt über das Verständnis der Events verfolgt werden,
- Eine ausschließliche Verfolgung des allgemeinen Eventmanagement Zieles „Weiterleitung von Events mit kontextbezogenen, hohen Informationsgehalt“ würde wiederum wegen der Erzeugung von vielen Gruppierungswerten wenig Sinn machen,
- Also wird die „Fehlerlokalisierung“ herangezogen.

2. Untersuchung

Die Einbeziehung des allgemeine Eventmanagement-Zieles „Fehlerlokalisierung“ impliziert eine Verfügbarkeits-bezogene Untersuchung des Informationsgehaltes eines Events. Mit diesem Ansatz wurde eine weitere Untersuchung der Events und ihrer Dokumentation vorgenommen. Das Ergebnis war eine akzeptable Anzahl von Gruppierungswerte, in denen sich die Verfügbarkeit der Eventquellen wiederspiegelt. Dabei wurde festgestellt, daß mehr Information, als nur für eine einfache Einteilung in „verfügbar“ und „nicht verfügbar“ bereitsteht:

a) Events wie z.B. „*01: MLXEV_PHYSDEV_ONLINE*“ zeigen eine Verfügbarkeit an,
b) Es gibt Events, die über ihre Eventquelle eine Aussage der Form „ist zwar noch verfügbar, aber funktioniert fehlerhaft“ treffen. Z.B. sagt „*2247: The security database is corrupted*“ über den SAM aus, daß er zwar noch funktioniert - also verfügbar ist - aber seine funktionsweise keinesfalls fehlerfrei ist (falls z.B.

ein noch bestehender Account nicht mehr in dem SAM vorhanden ist, wegen der Inkonsistenz, dann wird eine darauf bezogene Abfrage falsch beantwortet),

c) Eine ähnliche Interpretation bezüglich Verfügbarkeit läßt sich den Events zuordnen, die eine momentane Unverfügbarkeit aufgrund einer Beschäftigung aufweisen. Beispielsweise zeigt das Event „*02: MLXEV_PHYSDEV_HOTSPARE*“ an, das aufgrund eines Rebuilds das Physikalische Laufwerk z.Z. nicht verfügbar ist,

d) Ein eindeutiger Ausfall ist durch einige Events zu beobachten. „*2453: Could not find the domain controller for this domain*.“ deutet in diesem Sinne auf eine fehlende Verfügbarkeit des NetLogon-Dienstes am PDC hin,

e) Manche Events erlauben eine frühzeitige Erkennung eines Ausfalls, wie z.B. das Event „*04: MLXEV_PHYSDEV_PFA*“, das auf einen möglichen Fehlereintritt hindeutet,

f) Allerdings drohen einige Events mit ihrem Auftreten einen unmittelbaren Ausfall an, wie es beispielsweise bei „*03: MLXEV_PHYSDEV_HARD_ERROR“* wegen des nachfolgenden DEAD-Zustandes des Physikalischen Laufwerks der Fall ist,

g) Es gibt Events, denen man gar keine Verfügbarkeits-Bedeutung zuteilen kann. Ein Beispiel wäre „*28: MLXEV_PHYSDEV_REQSENSE*“, das die Notwendigkeit weiterer Informationen über sogenannte „sense-bytes“ anzeigt.

Für Operationen als Eventquellen wird zunächst die o.a. Einteilung mit aufgenommen, allerdings wird festgestellt, daß zusätzliche Events bezüglich ihrer Ausführungsergebnisse vorkommen. Dabei konnten in diesem Sinne nur zwei mögliche Aussagen, nämlich „Erfolg“ und „Mißerfolg“, aus der Dokumentation synthetisiert werden:

h) Das Event „*3231: The UPS service performed server shutdown*.“ zeigt beispielsweise einen Erfolg an. Obwohl eigentlich ein Server nicht mehr verfügbar ist, handelt es sich hier um eine Operation, die ihre Aufgabe erfüllt hat (siehe auch Kapitel 3.2.4),

i) Ein Mißerfolg wird z.B. durch „*42: MLXEV_SYSDEV_REBUILD_ERROR*“ angezeigt.

Erkenntnisse:

- Eine Untersuchung der angegeben Einteilung ergibt, daß b) und c) gleichermaßen bezüglich der Verfügbarkeit eine Unzulänglichkeit darstellen. Deswegen werden diese Events hiermit als „unzureichend verfügbar“ betrachtet. Um das allgemeine Eventmanagement-Ziel „Weiterleitung von Events mit kontextbezogenen, hohen Informationsgehalt“ trotzdem mitzuverfolgen, wird in Kapitel 3.2.4 ein Konzept entwickelt. Damit findet eine Interpretations-Auslagerung in die Verarbeitung statt,
- Verfügbarkeits-Einteilungen für Operationen ergaben zwei zusätzliche Werte, die ihre Ausführungsergebnisse charakterisieren - siehe h) und i). Also ist es sinnvoll die Gruppierungswerte für Operationen gesondert zu betrachten. Es wird festgestellt, das dies eine Erfolgs-Untersuchung ist, die sich allerdings auf die Verfügbarkeits-Untersuchung auswirkt,
- e) und f) können als „Ausfall-drohende“ Gruppierungswerte aufgenommen werden. Allerdings bezieht sich e) auf mögliche, spätere Ausfälle, während f) einen konkreten, baldigen Ausfall anzeigt. Um diesen Unterschied der Bedeutungen beizubehalten wird e) eine Drohung der „unzureichenden“ und f) eine Drohung der nicht vorhandenen Verfügbarkeit zugesprochen. Hierbei findet keine Auslagerung der Bedeutung nach Kapitel 3.2.4 statt (s.o.), weil die Art der Drohung bezüglich der Verfügbarkeit relevant ist.

Demnach werden nun folgende Gruppierungswerte aufgestellt, die die Erkenntnisse der o.a. Untersuchungen wiederspiegeln:

Für eine beliebige Softwarekomponente k als Eventquelle:

(1)	k ist verfügbar	(SK_V),
(2)	k ist unzureichend verfügbar	(SK_UV),
(3)	k ist nicht verfügbar	(SK_NV),
(4)	k droht (2)	(SK_DUV),
(5)	k droht (3)	(SK_DNV),
(6)	k befindet sich in einem sonstigen Zustand	(SK_SZ),

Für eine beliebige Operation o als Eventquelle:

(7)	o ist verfügbar	(O_V),
(8)	o ist unzureichend verfügbar	(O_UV),
(9)	o ist nicht verfügbar	(O_NV),

(10)	o droht (8)	(O_DUV),
(11)	o droht (9)	(O_DNV),
(12)	o erfolgreich beendet (d.h. Erfolg)	(O_E),
(13)	o gescheitert (d.h. Mißerfolg)	(O_M),
(14)	o befindet sich in einem sonstigen Zustand	(O_SZ),

Warum diese Einteilung vorgenommen wurde, wird durch die Argumentationskette dieses Kapitels motiviert.

Die Eventinterpretation enthält implizit eine Gruppierung nach der Art der Eventquelle (Softwarekomponenten und Operationen). Um die Verfügbarkeit bzw. den Erfolg bestmöglich zu beschreiben wird bei dieser Gruppierung versucht eine Minimalanzahl von Events einem „sonstigen Zustand“ zuzuordnen.

3.2.4 Gruppierung nach der Eventverarbeitung

Bisher wurden Events nach der Berücksichtigung, nach ihrer Eventquelle oder nach einer Interpretation bezüglich Verfügbarkeit oder Erfolg analysiert. Die Gruppierung nach der Eventverarbeitung hat zwei Motivationen:
A) Eine erste Orientierung für die Konzeption der Korrelations-Operationen im MSWEM zu geben,
B) Den Informationsgehalt eines Events ergänzend zur Eventinterpretation zu erschließen.

A)
Die vorliegende Gruppierung setzt sich als Ziel die Events hinsichtlich ihrer Verarbeitungsweise, d.h. nach ihrer Behandlung durch den MSWEM und nicht etwa durch einen Administrator, zuzuordnen. Dafür wird aus der Untersuchung der Event-Dokumentation ein erster, nicht kontextbezogener Handlungsbedarf synthetisiert. Dieser wird als Kombination möglicher Verarbeitungsweisen des MSWEM, von denen man sich zu diesem Zeitpunkt eine einfache Vorstellung macht, verstanden. Die vorgestellten Verarbeitungsweisen sind:
- S „Speichern“,
- V „Verarbeiten“ als eine Folge von Handlungen des MSWEM,
- M „Melden“ als Weiterleitung von Events,
- KA „Katastrophen-Alarm“ als besonders dringliche Weiterleitung von Events.

Diese Vorstellung resultiert aus der allgemeinen Sicht eines Eventmanagers, als Verantwortungsträger für den Empfang, die Verarbeitung und die Weiterleitung von Events.

Die Untersuchung der Event-Dokumentation ergab folgende Einteilung des Handlungsbedarfes:
- Für die meisten Events ist eine Verarbeitung, Speicherung und Meldung nötig. So z.B. wird für „*23: MLXEV_PHYSDEV_SOFT_ERROR*“, das beim achtmaligen Auftritt einen DEAD-Zustand des Physikalischen Laufwerks bewirkt, festgestellt: V - um das vorherigen Auftreten zu überprüfen, S - im Hinblick auf weitere Überprüfungen und M - falls das Auftreten mehr als z.B. 3 Mal vorkommt muß der Administrator gewarnt werden.

B)
Bei einer gestarteten Operation ist in der Regel die Eventinterpretation O_V und die Eventverarbeitung M oder SM. Allerdings, wenn die Operation das UPS-ShutDown darstellt, ist die Eventinterpretation weiterhin O_V aber die Eventverarbeitung wird zunächst (ohne Cluster-Betrachtungen) auf VSKA gesetzt. Also ergänzt die Eventverarbeitung manchmal bezüglich der Erfassung des Informationsgehaltes.

Nach der Eventverarbeitung lassen sich die Events einer der folgenden hybriden Hauptverarbeitungsweisen zuordnen:

(1)	Speichern und Melden	(SM),
(2)	Verarbeiten und Melden	(VM),
(3)	Verarbeiten, Speichern und Melden	(VSM),
(4)	Verarbeiten, Speichern und Katastrophen-Alarm	(VSKA),
(5)	Melden	(M),

Die Reihenfolge von „Speichern“ und „Melden“ während eines gemeinsamen Auftretens ist im Grunde irrelevant. Dennoch wurde in 1 und 3 „Speichern“ vor „Melden“ gesetzt, um eventuelle Speicherungsfehler der Events an dem „Melden“ anknüpfen zu können.

3.3 Mylex-Events

Beim Empfangen eines Events aus E_{MLX} kann man jederzeit unter Angabe der Eventnummer (die zyklisch im Intervall [0,1023] iteriert) durch das IOCTL-Kommando „DACIOC_GETEVENT“ Eventinformationen abfragen. Diese werden dann in der Struktur „dga_event_info_t“ (siehe Anhang A.1) zurückgeliefert. Dabei werden wichtige Informationen über die Events, wie z.B. der Meldungstext, die Zeit, der Controller, die LUN, die Lüfter, die Stromversorgungseinheit oder der Temperatursensor, mitgeschickt.

Bei der Analyse des gegebenen Speichersystems ergaben sich einige Probleme. Trotz hohem Zeitaufwand (mindestens 20 Stunden) und Einsatz von nicht zu übertreffenden Mitteln, wie z.B. Originaldokumentation, Diskussionen mit dem SNI-Mitarbeiter, der die Mylex-MIB geschreiben hat, und mit den SNI-Mylex-Experten (eigenes Team), Untersuchung der SNI-Mylex-MIB, kommerzielle Supportanfrage an Mylex, konnten diese Probleme aber nicht geklärt werden. Diese Tatsache beruht z.T. auf die nicht ausreichend korrelationsbezogene Dokumentation und auf nicht implementierte aber aufgezählte Features. Im folgenden zählen wir diese Probleme auf:

Probleme bei der Informationsbeschaffung:

- (**MLXP0**) Mit „*“ wurden Controller-Charakteristika bezeichnet, die aufgrund der Event-Semantik mit relativ hoher Wahrscheinlichkeit angenommen werden können. Eine genaue Erklärung der Ursachen, die zum Event geführt haben, ist nicht zu erhalten,
- (**MLXP1**) Keine vorhandene Spezifikation um die Mylex-Events 33 und 34 zu verstehen oder um sie voneinander unterscheiden zu können;
- (**MLXP2**) Mylex-Events 26 (ACTIVESPARE) und 27 (WARMSPARE) sind nicht implementiert;
- (**MLXP3**) Keine vorhandene Spezifikation um die Mylex-Events 43 und 44 zu verstehen oder um sie voneinander unterscheiden zu können;
- (**MLXP4**) Den SIZECHANGED-Event (ID = 55) kann man als Folgerung einer EXPANDCAPACITY - Operation oder als Folgerung eines Plattenwechsels oder einer Konfigurationsänderung interpretieren;
- (**MLXP5**) Mylex-Events, die sich auf die **B**attery **B**ackup **U**nit beziehen (64, 65 und 66), sind nicht implementiert.

Aus den oben dargestellten Problemen ergibt sich eine a priori Einschränkung der Eventmenge E_{MLX}:

Erzwungene Annahmen durch die mangelhafte Information:

- (**MLXPA1**) Mylex-Events 10 und 11 werden ignoriert;
- (**MLXPA2**) Mylex-Events 33 und 34 werden ignoriert;
- (**MLXPA3**) Mylex-Events 43 und 44 werden ignoriert;
- (**MLXPA4**) Mylex-Events 26 und 27 werden ignoriert;
- (**MLXPA5**) Mylex-Events 64, 65 und 66 werden ignoriert.

3.3.1 Gruppierung nach den Eventquellen

Dieses Kapitel stellt die Mylex spezifische Gruppierung der Events nach ihrer Eventquelle vor.

Da alle Mylex-Events eigentlich vom Controller berichtet werden, müßten sie diesen als Eventquelle haben. Alledings würde diese Gruppierung in diesem Fall sehr flach ausfallen und würde damit keine Hilfe bei der Systematisierung der Events leisten. Deswegen wird hier jedem Begriffeinheit und jeder Operation des RAIDs eine abstrakte Softwarekomponente oder Operation zugeordnet. Diesen wird dann die Fähigkeit einer eigenständigen Meldung von Events verliehen, ohne daß der tatsächliche Ablauf der Eventmeldung für weitere Betrachtungen ungültig wird.

In diesem Sinne unterscheidet man folgende (abstrakte) Eventquellen für Mylex-Events:

(1)	Gesamtes RAID	(R)
(2)	Physikalisches Laufwerk	(PL)
(3)	Operation: Rebuild eines PL	(PL_Rbld)
(4)	Operation: Expand Capacity eines PL	(PL_ExC)
(5)	Logisches Laufwerk	(LL)
(6)	Operation: Check eines LL	(LL_Chk)
(7)	Operation: Rebuild eines LL	(LL_Rbld)
(8)	Operation: Initialisierung eines LL	(LL_Init)
(9)	Operation: Expand Capacity eines LL	(LL_ExC)
(10)	Cache	(Buf)
(11)	Controller (Adapter)	(Ctrl)
(12)	Lüfter	(Fan)
(13)	Stromversorgungseinheit	(Pwr)
(14)	Temperatursensor	(Tmp)

Als nebenläufiges Ergebnis dieser Gruppierung wird hier festgestellt, daß die Werte PL_Rbld, PL_ExC, LL_Chk, LL_Rbld, LL_Init und LL_ExC 6 Operationen darstellen, während die restlichen 8 Werte Softwarekomponenten simulieren.

Durch die Eventquelle wird implizit eine Einteilung der Operationen und Abhängigkeiten nach den Softwarekomponenten, auf die sich diese auswirken vorgenommen. Dies wird sowohl aus der Erklärung als auch aus den Abkürzungen ersichtlich: „Ctrl“ aus „Ctrl_Add“ bezieht sich beispielweise auf eine Abhängigkeit, die mit einem Controller zusammenhängt.

3.3.2 Systematisierung der Mylex-Events

Dieses Kapitel beschreibt eine Systematisierung der Mylex-Events, die auf dem von Mylex gelieferten Eventkatalog (siehe Anhang B.2) basiert.

Eventberücksichtigung:

Von insgesamt 79 Mylex-Events wurden durch die Eventberücksichtigung 10 Events aus dem Eintritt in die MSWEM-Verarbeitung ausgeschlossen. Demnach bleiben noch 69 zu analysierende Mylex-Events übrig.

Eventquelle:

Nach der Eventquelle ergeben sich insgesamt 21 verschiedene Werte für die Ursache eines Mylex-Events.

Dabei wurden Events, die eine Aussage der Form „FOUND“ bzw. „GONE“ beinhalten, nicht als Operationen der betreffenden Softwarekomponente (hier nur LLs oder PLs oder Ctrl) sondern als Abhängigkeiten deren modelliert. Dies beruht auf Bemerkung 3.2. In diesem Sinne werden

„13:MLXEV_PHYSDEV_FOUND“ als PL_Add für PL
und
„63:MLXEV_ADAPTER_GONE“ als Ctrl_Del für Ctrl

eingestuft.

Events, die einen „ERROR“ signalisieren, werden der entsprechenden Softwarekomponente oder Operation zugeordnet. So z.B. bezieht sich

„23:MLXEV_PHYSDEV_SOFT_ERROR“ auf ein PL
und
„32:MLXEV_SYSDEV_CHECK_ERROR“ offensichtlich auf eine LL_Chk - Operation.

Die „HEAT“ - Events beziehen sich auf die Temperatur, die vom Temperatursensor gemessen wird.

Die weiteren Werte der Eventquellen-Gruppierung sind durch ihren Meldungstext relativ leicht einzuordnen.

Eventinterpretation:

Von 17 möglichen Interpretationen wurden bei Mylex-Events 11 verwendet.

„DEAD“ - oder „OFFLINE“ - Events bedeuten allgemein SK_NV, während „ONLINE“ - Events zu einer SK_V - Interpretation führen. In diesem Sinne werden

„12:MLXEV_PHYSDEV_DEAD“ als SK_NV für PL,
„35:MLXEV_SYSDEV_OFFLINE“ als SK_NV für LL
und
„37:MLXEV_SYSDEV_ONLINE“ als SK_V für LL

gedeutet.

Wegen der Annahme einer atomaren Zustandsänderung wird ein „START“ - Event als O_V für die entsprechende Operation interpretiert. So z.B. bezieht sich

„38:MLXEV_SYSDEV_AUTO_REBUILD_START“

auf ein O_V für die LL_Rbld - Operation.

Eine Operation, deren Status „CANCELED“, „ERROR“ bzw. „DONE“ ist, wird als O_DUV, O_M bzw. O_E angenommen:

„31:MLXEV_SYSDEV_CHECK_CANCELED“ bedeutet O_DUV für LL_Chk,
„46:MLXEV_SYSDEV_INIT_DONE“ bedeutet O_E für LL_Init
und
„53:MLXEV_SYSDEV_EXPANDCAPACITY_ERROR“ bedeutet O_M für LL_ExC.

Die „ERROR“ - , „CRITICAL“ - bzw. „RESET“ - Events, die sich auf Softwarekomponenten beziehen, haben eine SK_DNV bzw. SK_DUV Semantik bezüglich der Softwarekomponenten. Für „ERROR“ - und „CRITICAL“ - Events ist dies offensichtlich. Für „RESET“ - Events folgt die vorherige Behauptung aus der Tatsache, daß ein Reset nach dem Auftreten eines (noch) nicht gravierenden Fehlers durchgeführt wird. In diesem Sinne werden z.B.

„23:MLXEV_PHYSDEV_SOFT_ERROR“ auf SK_DNV für PL
und
„61:MLXEV_ADAPTER_RESET“ auf SK_DUV für Ctrl

gesetzt.

Für die Lüfter und für die Stromversorgungseinheiten des RAID Systems gibt es allgemein die „OK“ - , „FAILED“ - oder „NOTPRESENT“ - Events. Diese werden sinngemäß den Werten SK_V, SK_NV bzw. SK_NV für die entsprechende Softwarekomponente zugeordnet:

„68:MLXEV_FMTFAN_OK“ wird als SK_V,
„69:MLXEV_AEMI_FAN_FAILED“ wird als SK_NV,
und
„74:MLXEV_FMTPOWER_NOTPRESENT“ wird als SK_NV

interpretiert.

Obwohl die Temperatur als Zustand eines Temperatursensors - also einer Softwarekomponente - modelliert ist, ist es sinnlos eine SK_NV oder SK_DNV Semantik auf sie zu beziehen, da sie über Schwellwert - Festlegungen geregelt wird. Deswegen werden dafür folgende Zuordnungen vorgenommen:

„75:MLXEV_FMTHEAT_BAD“ bedeutet SK_UV,
„76:MLXEV_FMTHEAT_CRITICAL“ bedeutet SK_DUV,
„77:MLXEV_FMTHEAT_OK“ bedeutet SK_V,
„78:MLXEV_AEMI_OVER_TEMPERATURE“ bedeutet SK_UV.

Dabei bezieht sich der „NOTPRESENT“ - Event in diesem Zusammenhang auf den Temperatursensor und wird demnach als S_NV eingestuft.

Warum START = O_V ?

Bei der Eventinterpretation im RAID wurde eine Minimalanzahl = 2 an sonstigen Zuständen erreicht, und zwar bei der Softwarekomponente PL und bei der Buf Operation.

Eventverarbeitung:

Allgemein wird im RAID-Kontext die Verarbeitung eines als SK_V, als O_V oder als O_E interpretierten Events auf Melden (M) gesetzt:

„46:MLXEV_SYSDEV_INIT_DONE“	M
„51:MLXEV_SYSDEV_EXPANDCAPACITY_START“	M
“57:MLXEV_SYSTEM_STARTED“	M

Für Einfügen bzw. Löschen von Softwarekomponenten, was eine A_V bzw. A_NV Semantik hat, wird eine VSM bzw. VM Verarbeitung gewählt. Ersteres resultiert aus der erforderlichen Bekanntmachung im System und beim zweiten wird auf das Speichern verzichtet, weil die Softwarekomponente eben gelöscht wird.

Wenn Operationen, die keine Aussage über die Verfügbarkeit einer Softwarekomponente beinhalten, fehlschlagen - also O_M - dann wird dies gespeichert und gemeldet:

„18:MLXEV_PHYSDEV_EXPANDCAPACITY_ERROR“ wird mit SM verarbeitet.

Sonstige Zustände werden hier immer als SM behandelt.

Der Cache ist bei den Mylex-Events durch den „WRITEBACK“ - und den „CHG_TABLE“ - Event vertreten und wird als Operation modelliert. Da sich ein „WRITEBACK“ - Fehler nur auf das Bekanntgeben des Schreibens an das Rechner bezieht, wird es hier auf O_SZ und damit auf SM gesetzt.

Da es höchstens zwei Controller redundant im RAID-System geben kann, bedeutet ein Controller - Ausfall ein schwerwiegender Fehler, da keine Absicherung mehr vorhanden ist und bei einem erneuten Controller - Ausfall eine RAID - Katastrophe entstehen würde. Deswegen wird

„60:MLXEV_ADAPTER_DEAD“ als VSKA

gedeutet.

Alle restlichen fehleranzeigende Eventinterpretationen, die bei Mylex-Events vorkommen, also SK_DUV, SK_UV, SK_DNV, SK_NV, O_DUV und O_M, werden durch Verarbeiten, Speichern und Melden - also VSM - behandelt.

3.4 NT-Events

Dieses Kapitel setzt sich als Ziel die Beschreibung und Systematisierung der NT-Events exemplarisch durchzuführen. Dafür werden am Anfang einige einführende Betrachtungen zu Event-Logging, zu Event-Browsing und zu den verschiedenen Eventarten von Windows NT vorgenommen. Anschließend werden beispielhaft die NT-Netz-Events nach ihren Eventquellen gruppiert und systematisiert.

3.4.1 Einführende Betrachtungen

Dieses Kapitel stellt ausführliche Behandlungen des Event-Logging, des Event-Browsing und der Eventarten, die NT-spezifisch sind, vor. Die Motivation dieser Behandlungen an dieser Stelle wird

- für die Behandlung der Eventarten durch die deutlichen Mehrzahl der NT-Events im Vergleich zu den anderen Events und
- für die Behandlung des Event-Logging und des Event-Browsing durch die Tatsache, daß die MSCS-Software diese Mechanismen verwendet

gerechtfertigt.

3.4.1.1 EventLog Service

Windows NT verfügt über drei binäre, systemglobale Log-Dateien, die von allen Softwarekomponenten im System zugreifbar sind:

- APPEVENT.EVT für Events von Anwendungs-Softwarekomponenten,
- SECEVENT.EVT für Events von Security-Softwarekomponenten,
- SYSEVENT.EVT für Events von System-Softwarekomponenten.

Diese Dateien werden hiermit **EventLogFile** bezeichnet.

Der Zugriff auf die EventLogFiles wird von einem speziellen Systemdienst, dem EventLog Service gesteuert. Der EventLog Service unterstützt für alle EventLogFiles die Eventtypen:

Eventtyp	Erklärung
EVENTLOG_ERROR_TYPE	zeigt ein Fehler-Event an
EVENTLOG_WARNING_TYPE	zeigt ein Warnungs-Event an
EVENTLOG_INFORMATION_TYPE	zeigt ein Informations-Event an
EVENTLOG_AUDIT_SUCCESS	zeigt eine erfolgreiche Authentifizierung an
EVENTLOG_AUDIT_FAILURE	zeigt eine mißlungene Authentifizierung an

Tabelle 3.1: Erklärung der Eventtypen

3.4.1.2 Event-Logging

Der EventLog Service ermöglicht jeder Eventquelle unter folgenden Voraussetzungen ein einheitliches Schreiben auf ein EventLogFile:

1. Es wird mindestens ein sogenanntes **event message file** benötigt, das die Zuordnungen der Event - Identifier zu den Meldungstexten realisiert. Zusätzlich kann man noch zum Unterteilen (Kategorisieren) bzw. Parametrisieren der geloggten Events weitere **category message files** bzw. **parameter message files** angeben. Allerdings ist es möglich die Unterteilungs- und Parametrisierungsinformationen auch in der event message file zu halten.

2. Es müssen Zugriffsbeschreibungen in der folgenden Registry-Struktur abgelegt werden:

   HKEY_LOCAL_MACHINE
   SYSTEM
   CurrentControlSet
   Services
   EventLog
   Application
   Security
   System

   Die Registry-Subkeys des EventLog Service heißen **logfiles**. Standardmäßig werden die Application, Security und System logfiles beim Installieren erzeugt, die den bestehenden EventLogFiles entsprechen. Die Subkeys der logfiles werden **sourcen** bezeichnet, da sie die Zugriffsbeschreibungen von Eventquellen beinhalten. Sourcen sind für jedes logfile erweiterbar und enthalten als Registry-Blätter folgende beschreibende Parameter:

Parameter-Bezeichnung	Parameter-Beschreibung
EventMessageFile	Pfad des/der event message file/s als REG_EXPAND_SZ.
CategoryMessageFile	Pfad des/der category message file/s als REG_EXPAND_SZ.
ParameterMessageFile	Pfad des/der parameter message file/s als

	REG_EXPAND_SZ.
CategoryCount	Anzahl der Kategorien als REG_DWORD.
TypesSupported	Bitmask der unterstützten Eventtypen als REG_DWORD.

Tabelle 3.2: Registry-Parameter der Eventquellen

Wenn die beiden o.g. Voraussetzungen erfüllt sind, kann jede Eventquelle in folgender Weise ihre eigenen Events loggen:

- Durch Aufruf der Funktion „RegisterEventSource“ oder „OpenEventLog“ öffnet sich die Eventquelle einen Kommunikationskanal zur EventLogFile,
- Durch Aufruf der Funktion „ReportEvent“ schreibt die Eventquelle tatsächlich ihr Event in die EventLogFile.

Nach Beendigung des Eintrags wird die Funktion „CloseEventLog“ zum Schließen des geöffneten Kommunikationskanals aufgerufen.

3.4.1.3 Event-Browsing

Der von Microsoft beim Vertrieb von Windows NT mitgelieferte Eventbrowser wird **Event Viewer** genannt.

Der Event Viewer zeigt standardmäßig die geloggte Information der EventLogFiles an. Dies wird folgendermaßen realisiert:

- Durch Aufruf der Funktion „OpenEventLog“ öffnet sich der Event Viewer einen Kommunikationskanal zur EventLogFile,
- Durch Aufruf der Funktion „ReadEventLog“ liest der Event Viewer die geloggten Events aus der EventLogFile.

Dabei werden die anzuzeigenden Events in der einheitlichen Struktur EVENTLOGRECORD vom entsprechenden EventLogFile zurückgeliefert:

```
typedef struct _EVENTLOGRECORD { // evlr
      DWORD       Length;
      DWORD       Reserved;
      DWORD       RecordNumber;      = //Position im EventLogFile
      DWORD       TimeGenerated;     = //Zeit des Entstehens in der source
      DWORD       TimeWritten;       =  // Zeit des Eintrags ins EventLogFile
      DWORD       EventID;
      WORD        EventType;         //siehe Tabelle
      WORD        NumStrings;
      WORD        EventCategory;     = //source-spezifische Kategorie
      WORD        ReservedFlags;
      DWORD       ClosingRecordNumber;
      DWORD       StringOffset;
      DWORD       UserSidLength;
      DWORD       UserSidOffset;
      DWORD       DataLength;
      DWORD       DataOffset;
      //
      // Then follow:
      //
      // TCHAR    SourceName[]
      // TCHAR    Computername[]
      // SID      UserSid
      // TCHAR    Strings[]
      // BYTE     Data[]             = event-spezifische Information
      // CHAR     Pad[]
      // DWORD    Length;
      //
} EVENTLOGRECORD;
```

Beim Empfang eines EVENTLOGRECORDs kann der Event Viewer anhand des „EventCategory“- und des „SourceName[]“-Parameters die entsprechenden event message files und event category files über die Registry-Einträge lokalisieren. Durch Angabe der EventID bzw. der Kategorie kann er dann den genauen Meldungstext aus den event message files bzw. die Kategoriebeschreibung aus den event category files abfragen.

Wenn im Meldungstext Marken für Ersetzungstexte vorkommen, werden diese Ersetzungstexte
- entweder direkt über die „StringOffset“- und „Strings[]“-Parameter der Struktur, falls eine Sortierung der Marken im Meldungstext vorliegt
- oder indirekt über Abfrage der parameter message files bei gegebener Marke, falls die Marken Platzhalter darstellen

ermittelt.

Nach Beendigung des Lesens wird die Funktion „CloseEventLog“ zum Schließen des geöffneten Kommunikationskanals aufgerufen.

Als Zusammenfassung von 3.4.1.2 und 3.4.1.3 wird folgende Behauptung aufgestellt:

Behauptung 3.1: Jede Softwarekomponente im Cluster kann durch eine richtige Konfiguration oder durch zusätzliche Implementierungen ihre eigenen Events einem EventLogFile berichten, so daß alle Events aus E einem Eventbrowser zur Verfügung stehen können.

3.4.1.4 NT-Eventarten

Die Eventarten beziehen sich in diesem Kontext nicht auf eine in dieser Diplomarbeit vorgenommenen Gruppierung sondern auf eine Einteilung, die in der Dokumentation vorliegt.

Die Dokumentation der NT-Events befindet sich zu Einen
- auf der MSDN- und redundant auf der TechNet-CD in den Dokumenten „Message Reference“ (1) und „Windows NT Executive Messages“ (2)

und zum Anderen
- auf der ResourceKit-CD in dem Dokument „Message Database“ (3).

Dabei ist (2) eine Untermenge von (1), allerdings etwas genauer. (3) ist von Microsoft in Form einer Access-Tabelle mit 4384 Datensätzen realisiert und (1) wurde nach einigen Anstrengungen in selber Form gebracht. Dabei enthält (2) ca. 5500 Datensätze - was wegen wenigen Duplikaten in der Tabelle, als ziemlich genaue Schätzung für die NT-Eventanzahl gelten kann.

Eine Untersuchung dieser Dokumente ergab folgende Einteilung nach Eventarten, die von Microsoft festgelegt wurden:

I. Meldungen der Executive,
   - A. Zeichenorientierte Meldungen,
   - B. Fensterorientierte Meldungen,
     1. System-Informations-Meldungen,
     2. Warnungs-Meldungen,
     3. Anwendungs-Unterbrechungs-Meldungen,

II. Netz-Meldungen,
   - A. NT-Netz-Meldungen, die von Microsoft als Netz-Meldungen bezeichnet werden,
   - B. Down-Level-Meldungen,

III. System-Meldungen.

Dabei werden bis auf III. alle Eventarten explizit benannt, während die „System-Meldungen“ eine in dieser Diplomarbeit gewählte Beschreibung für alle restlichen Events darstellt.

I:

Meldungen der Executive werden sowohl in der Message Database als auch in der Message Reference und in (2) beschrieben.

I.A:
Zeichenorientierte Meldungen - es gibt 130 davon - sind Meldungen, bei denen der Kernel nicht mehr reagiert: „character-mode messages occur when the Windows NT Kernel detects an inconsistent condition from which it cannot recover“. Dabei erscheint das typische NT-Absturz-Bildschirm mit blauem Hintergrund und weiß angezeigtem Hex-Code. Das Auftreten dieser Meldungen bedeutet, daß jede Management-Aktion sich erübrigt, da sie „zu spät“ erfolgen würde. Demnach werden die zeichenorientierten Meldungen vom MSWEM nicht berücksichtigt.

I.B:
Fensterorientierte Meldungen sind Meldungen, die in einer „Message-Box“ am Bildschirm angezeigt werden und die - laut Microsoft - nur einen Status eines Prozesses anzeigen, über die der Benutzer Bescheid wissen sollte. Dabei wird mit Prozeß auch eine Anwendung oder ein Thread gemeint.

I.B.1:
System-Informations-Meldungen haben - wie der Name schon andeutet - informativen Charakter. Ein Lesen und Schließen der Message-Box ist eine ausreichende Behandlung, da der Kernel den Prozess normal fortsetzt. Es gibt insgesamt 18 System-Informations-Meldungen.

I.B.2:
Warnungs-Meldungen - 28 an der Zahl - zeigen an, daß entweder

- a) eine Aktion erforderlich ist, damit der Kernel den aktuellen Prozeß / Thread rechnend hält

oder daß

- b) trotz rechnendem Prozeß / Thread der Kernel fehlerhafte Ergebnisse liefern könnte.

Ein typisches Beispiel für a) wäre das Ereignis „Device Power Is Off“, das als „Schalten Sie das Gerät ein, sonst kann ich den Zugriff nicht fortsetzen“ zu verstehen ist.

Für b) wird „Filemark Found“ beispielsweise betrachtet, was soviel bedeutet wie: „Da ich beim Bearbeiten des Prozesses / Threads einen nicht passenden Recordelement auf dem Band gefunden habe, wird das Ergebnis u.U. fehlerhaft sein“.

Problem:
Allgemein wurde festgestellt, daß diese Meldungen sehr schlecht dokumentiert sind; die Standarderklärung dafür war meistens:

> „This Windows NT Executive STATUS message is a warning. Choose one of the options from the message box.“

Dabei werden die Optionen in der Dokumentation nicht einmal aufgelistet - d.h. die Meldungen müßten zuerst auftreten um sich ein näheres Verständnis darüber schaffen zu können.

I.B.3:
Anwendungs-Unterbrechungs-Meldungen besagen, daß der Kernel kurz vor der Beendigung eines Prozesses / Threads steht. Es gibt insgesamt 78 Anwendungs-Unterbrechungs-Meldungen, davon 13 EXCEPTIONS und 65 Fehler.

Bei den EXCEPTIONS taucht das obige Problem wieder auf: alle EXCEPTIONS werden mit folgendem Kommentar versehen:

> „This is a Windows NT Executive STATUS error message. Choose an option from the message box. Then contact your technical support group.“

Dabei wird allerdings die Fehlerursache aus dem Meldungstext meistens verständlich:

> „EXCEPTION:Integer division by zero.“ bedeutet offensichtlich, daß beim Bearbeiten der entsprechenden Anwendung ein Teilen durch Null aufgetreten ist.

Die Fehler können in harmlose und schwerwiegende Fehler nach ihrer Semantik aufgeteilt werden. So wäre z.B.:

„File Not Found:The file filename does not exist.“ ein harmloser Fehler, der den „Dateiöffnungs-Thread“ eines Anwendungsprozesses zu beenden droht,

während

„Hard Disk Error:While accessing the hard disk, a disk operation failed even after retries.“ ein Beispiel für ein schwerwiegender Fehler wäre, der den HD-Zugriffs-Prozess zu beenden droht.

II:
Netz-Meldungen werden nur in der Message Reference erklärt. Dabei geschieht dies leider (auch) nicht vollständig. Microsoft behauptet, die fehlenden Erklärungen basieren auf trivial-Meldungen, wie z.B. „Password expires“. Es wurde aber festgestellt (siehe Kapitel 3.4.3), daß diese Behauptung falsch ist.

II.A:
Die NT-Netz-Meldungen sind Meldungen, die ein NT-Server oder eine NT-Workstation senden, wenn ein Problem bei einer Netz-Operation festgestellt wurde. Dabei werden als Problemquellen nicht nur das Netz (intensiv) an sich, sondern auch Systeme (oberflächlich) oder Anwendungen (oberflächlich) betrachtet.

II.B:
Down-Level-Meldungen sind das Pendant der NT-Netz-Meldungen - mit dem Unterschied, daß sie von früheren Microsoft - Betriebsystemen (Windows for Workgroups oder LAN Manager) gesendet werden. Da im Cluster nur NT Server vorhanden sind, erübrigt sich die Betrachtung dieser Meldungen.

III:
Als System-Meldungen wurden alle restlichen (also nicht zu I. oder II. gehörend) Events bezeichnet, die in der Message Reference oder in der Message Database erscheinen.

Die Abbildung 3.1 zeigt eine Zusammenfassung der dargestellten Eventarten, ihrer Dokumentation und ihrer Anzeige:

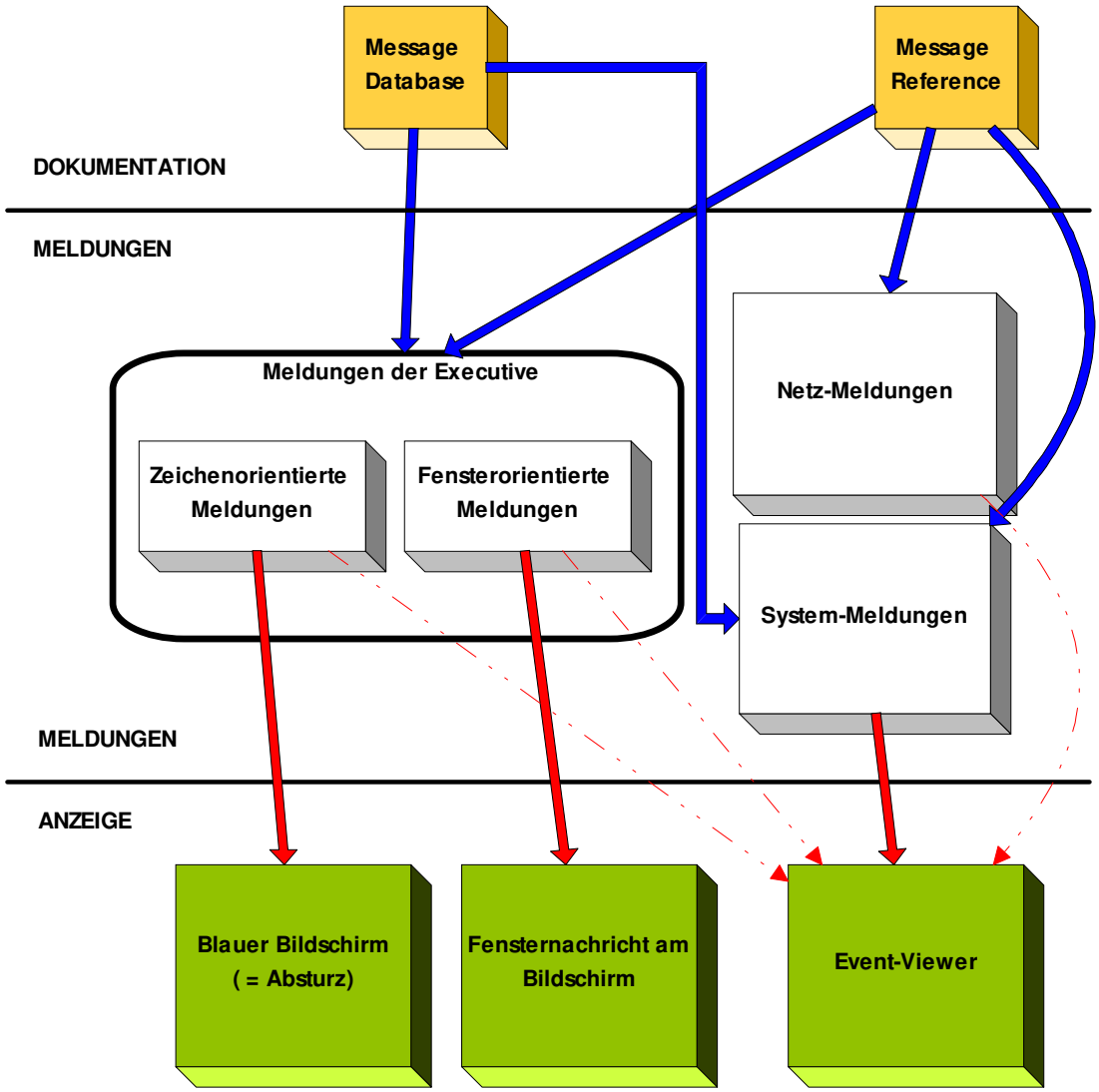


Abbildung 3.1: NT-Eventarten

Die Betrachtung der Gesamtanzahl von ca. 5500 NT-Events für den MSWEM würde offensichtlich den Rahmen dieser Diplomarbeit sprengen. Deswegen wird im folgenden eine Einschränkung der zu betrachtenden NT-Event vorgenommen. Da die NT-Netz-Events den Zustand des Netzes und teilweise auch der Systeme beschreiben, da sie akzeptabel dokumentiert sind und da sie eine noch zu bewältigende Anzahl, nämlich 283, aufweisen, wird nun die Analyse und Systematik der NT-Events auf die NT-Netz-Events beschränkt. Außerdem haben Netze als Cluster-Bestandteile eine höhere Gewichtung als Systeme, da sie z.T. die Verbindung des Clusters mit der Außenwelt und damit seine Verfügbarkeit in der angeschlossenen Umgebung realisieren.

3.4.2 NT-Netz-Events

Das Ziel dieses Kapitels ist es die NT-Netz-Events, ähnlich wie die Mylex-Events zu systematisieren. Als Grundlage dafür diente eine Informationstabelle *NET*, die durch folgendes Vorgehen erstellt wurde:

1. Cut & Paste der unerklärten Eventliste in numerischer Reihenfolge von CD in ein Text-Dokument,
2. Importieren des Word-Dokuments in eine MS-Access-Tabelle *NetMsg*,
3. Cut & Paste der ca. 5500 Einträge der Message Reference von CD in ein Text-Dokument,
4. Importieren des Word-Dokuments in eine MS-Access-Tabelle *AllMsg*,
5. Korrigieren von Formatfehler, die beim Importieren erzeugt wurden,

6. Filtern aller Netz-Meldungen aus *AllMsg* durch geeignete Abfrage,
7. Filtern der NT-Netz-Meldungen durch geeignete Abfrage in eine Tabelle *Net_Expl*,
8. Aufbau einer mit allen notwendigen Informationen ausgestatteten Tabelle *NET* aus den Tabellen *NetMsg* und *Net_Expl* durch einen komplexen, mehrstufigen Abfragekonstrukt (unter Verwendung von Inner Joins usw.) .

Das Ergebnis der vorgenommenen Systematisierung spiegelt sich in den Eventkatalog der NT-Netz-Events, der im Anhang B2 angegeben wird, wieder.

3.4.2.1 Gruppierung nach den Eventquellen

Die Wahl der Eventquellen bei den NT-Netz-Events ist durch viel Background-Recherche erst möglich, da nur in seltenen Fällen der Meldungstext eine hinreichende Aussage über die Bedeutung der Events enthält. Außerdem reicht die Untersuchung der Erklärung und der empfohlenen Aktionen oft nicht aus um eine eindeutige Eventquelle herausfiltern zu können. Demnach muß in solchen Fällen eine Wahl getroffen werden - was folgendermaßen geschieht:

1. Falls aus der Erklärung oder aus den empfohlenen Aktionen eine erste konkrete Fehlerlokalisierung erkenntlich wird, so wird diese Ursache als Eventquelle gewählt:
   Für das Event
   „2453: Could not find the domain controller for this domain.“
   mit der empfohlenen Aktion
   „Start the Netlogon service on the domain controller.“
   wird nicht das System, das eine Operation am PDC ausführen will, als Eventquelle gewählt, sondern der NetLogon-Dienst des PDC,

2. Falls aus der Erklärung oder aus den empfohlenen Aktionen nur eine vage Fehlerlokalisierung aber eine konkrete Nebenwirkung ersichtlich ist, so wird diese Nebenwirkung als Eventquelle interpretiert. Die vage Fehlerlokalisierung wird hinsichtlich einer ergänzenden Angabe beim Melden gespeichert:
   Beispielsweise wird für das Event
   „3042: Network error code occurred.“
   mit der Erklärung
   „The Replicator service stopped because the listed Windows NT error occurred.“
   der Replicator Dienst als Eventquelle gesetzt und der „error code“ gespeichert,

3. Falls „quasi-variable“ Eventquellen vorkommen, d.h. Eventquellen, die vor dem Event-Auftreten nicht genau bestimmt sind, sondern erst aus einem im Meldungstext spezifizierten Namen konkret ersichtlich werden, so werden diese als solche gekennzeichnet. Das geschieht durch Einfügen des Präfixes „Spec“ in ihre Abkürzung:
   So wird z.B. für
   „5724: Could not register control handler with service controller name.“
   die Eventquelle als „Spec_Svc_Ctrl“ und nicht „Svc_Ctrl“ gedeutet,

4. Falls sowohl eine Softwarekomponente als auch eine Operation als Eventquelle Sinn ergeben würden, wird erstere gewählt:
   Beim Event
   „3031: The replication server could not update directory name from the source on name due to error code.“
   fällt z.B. die Wahl auf die Softwarekomponente „Spezifiziertes Verzeichnis“, das sich wegen dem Mißerfolg der Operation „Verzeichnis Updaten“ nun in einem inkonsistenten Zustand befindet. Dabei wird im Sinne des 2. Falles der „error code“ mitgespeichert.

Außerdem ist bei der Systematisierung der Eventquellen für NT-Netz-Events eine Eventquelle „Config“ erforderlich, die Konfigurationsmeldungen beschreibt. Dies folgt aus der großen Anzahl und aus der Unterschiedlichkeit dieser Konfigurationsmeldungen, die erst im Kapitel 3.4.2.2 deutlich werden.

Zur Vereinfachung weiterer Betrachtungen wird bei dieser Gruppierung versucht die Eventquelle zusätzlich „örtlich“ zu charakterisieren. Das bedeutet, daß Eventquellen, bei denen aus der Beschreibung eine Information der Form „Lokal“, „Remote“ bzw. „am PDC“ über ihren Aufenthalt verfügbar ist, auch als solche zu kennzeichnen sind. Demnach wurden solche Eventquellen mit einem Präfix der Form „L“, „R“ bzw. „PDC“ versehen.

In diesem Sinne wurden folgende Eventquellen für NT-Netz- Events aufgestellt:

(1)	Operation: Ändere Computer-Passwort	(Ch_CompPW)
(2)	Konfiguration	(Config)
(3)	Lokaler Alerter Dienst	(L_Alrt_Svc)
(4)	Lokale Uhr	(L_Clock)
(5)	Lokaler Plattenplatz	(L_DiskSp)
(6)	Lokaler Message Dienst	(L_Msg_Svc)
(7)	Lokaler NetLogon Dienst	(L_NetLg_Svc)
(8)	Lokale Stromversorgung	(L_Pwr)
(9)	Lokaler Replicator Dienst	(L_Repl_Svc)
(10)	Lokaler Server Dienst	(L_S_Svc)
(11)	Lokaler SAM	(L_SAM)
(12)	Lokaler Dienst Controller	(L_Svc_Ctrl)
(13)	Lokales System	(L_Sys)
(14)	Lokale System-Ressource	(L_Sys_Res)
(15)	Lokaler Workstation Dienst	(L_WS_Svc)
(16)	Netz	(Net)
(17)	Netz-Ressource	(Net_Res)
(18)	NetLogon Dienst des PDC	(PDC_NetLg_Svc)
(19)	Primärer Domänen Controller	(PDC_Sys)
(20)	Remote Message Dienst	(R_Msg_Svc)
(21)	Remote Workstation Dienst	(R_WS_Svc)
(22)	Remote Server Dienst	(R_S_Svc)
(23)	Remote Replicator Dienst	(R_Repl_Svc)
(24)	Spezifiziertes Verzeichnis	(Spec_Dir)
(25)	Spezifizierte DLL	(Spec_Dll)
(26)	Spezifizierte Operation	(Spec_O)
(27)	Spezifizierter SAM	(Spec_SAM)
(28)	Spezifizierter Dienst	(Spec_Svc)
(29)	Spezifizierter Dienst Controller	(Spec_Svc_Ctrl)
(30)	Spezifiziertes System	(Spec_Sys)
(31)	Operation: UPS initiiert ShutDown	(UPS_SDown)
(32)	Operation: Benutzer einer Gruppe zuordnen	(Usr_2_Grp)

Feststellungen zur Gruppierung:

- Von den 32 Werten dieser Gruppierung konnte die Anzahl der Operationen auf 3, nämlich *Ch_CompPW*, *UPS_SDown* und *Usr_2_Grp*, begrenzt werden.
- Bis auf die „quasi-variablen“ Eventquellen und die Operationen konnten alle Eventquellen als *Lokal* oder *Remote* oder *PDC* eingestuft werden. Für *Config* bzw. *Net* hätte diese Einteilung ohnehin keine Bedeutung (siehe Kapitel „Systematik der NT-Netz-Events“) bzw. keinen Sinn.

3.4.2.2 Systematik der NT-Netz-Events

Die Systematik der NT-Netz-Events erfolgt aufgrund des im Anhang B2 angegebenen Eventkatalog und wendet sich der Reihe nach jeder einzelnen Gruppierung zu.

Eventberücksichtigung:

Bei der Gruppierung nach der Eventberücksichtigung wurden folgende 4 Kriterien für ein Ausschluß von der Verarbeitung eingesetzt:

- Events, die Drucker - Informationen vermitteln, werden wegen der Abwesenheit von Druckern im Cluster ignoriert. Es gibt 5 solche Events, wie z.B.:
  „*2115: The queue is empty*."

- Auf Down-Level Betriebssysteme bezogene Events, wie z.B.:
  „*2470:Try down-level (remote admin protocol) version of API instead*."
  oder
  „*3817:You have specified a value that is incompatible with servers with down-level software. Please specify a lower value*."
  werden ausgelassen, da alle Server im Cluster unter NT 4.0 laufen.

- Events, für die überhaupt keine Erklärung in der Dokumentation gefunden wurde, werden nicht betrachtet. Diese Annahme basiert auf der Unabdingbarkeit einer genauen Erklärung bei NT-Netz-Events, die aus der häufigen Unterschiedlichkeit des Meldungstextes von der Erklärung folgt. Es gibt 40 dereartige Events.

Auch wegen der Forderung nach einer genauen Erklärung wurden Events, die durch „A Software Error Occured. Contact technical support." beschrieben werden, bis auf 4 Ausnahmen nicht berücksichtigt. Die restlichen Events dieser Art, d.h. ohne die 4 Ausnahmen, sind 12 an der Zahl.
Außerdem wurden 2 Events bearbeitet, deren Beschreibung zwar nicht die obige Form hat, aber auch keine eigentliche Information liefert. Diese sind:
„3534:The service did not report an error.", mit der Beschreibung:
„Try the task later. If the problem persists, contact your network administrator."
und
„3547:A service specific error occurred: code.", mit der Beschreibung:
„Refer to the Help or documentation for that service to determine the problem."
Event 3534 wurde ausgelassen, da keine vernünftige Semantik zu erkennen ist, und das Event 3547 wurde weiterbearbeitet, da eine Einordnung als Spec_Svc, SK_DNV in die Eventinterpretation denkbar wäre.

Berücksichtigung	Anzahl von Events
J	211
N	72

Eventquelle:

Quelle	Anzahl von Quelle
Ch_CompPW	1
Config	133
L_Alrt_Svc	2
L_Clock	1
L_DiskSp	1
L_DiskSp / L_NetLg_Svc	1
L_DiskSp / L_SAM	1
L_Msg_Svc	1
L_NetLg_Svc	3
L_Pwr	3
L_Repl_Svc	2
L_S_Svc	2
L_SAM	8
L_SAM / Spec_SAM	1
L_Svc_Ctrl	1

L_Sys	5
L_Sys_Res	3
L_WS_Svc	2
Net	2
Net_Res	1
PDC_NetLg_Svc	1
PDC_Sys	4
R_Msg_Svc	1
R_Msg_Svc / R_WS_Svc	1
R_S_Svc / R_Repl_Svc	1
Spec_Dir	2
Spec_Dll	1
Spec_O	1
Spec_SAM	3
Spec_Svc	13
Spec_Svc_Ctrl	2
Spec_Sys	3
UPS_SDown	3
Usr_2_Grp	1

Eventinterpretation:

Interpretation	Anzahl von Events
O_E	1
O_M	4
O_V	1
SK_DNV	5
SK_DUV	6
SK_DUV / SK_DUV	1
SK_NV	39
SK_NV / SK_NV	1
SK_NV / SK_UV	2
SK_UV	17
SK_V	1

Eventverarbeitung:

Verarbeitung	Anzahl von Events
SM	3
SM / SM	1
VM	1
VSKA	10
VSM	60
VSM / VSM	3

3.5 MSCS-Events

Bei der Systematisierung der MSCS-Events sind massive Probleme aufgetreten. Diese beruhen auf die Abwesenheit einer Auflistung oder Beschreibung der MSCS-Events. Allerdings wurden nach intensiven Recherchen, die viel Zeit in Anspruch genommen haben, einige Informationen gefunden, die die Erschließung / Bestimmung der Menge aller MSCS-Events E_{MSCS} ermöglichen.

3.5.1 Erschließung der MSCS-Events

Die Erschließung der MSCS-Events basiert auf real verfügbare Daten aus Header-Dateien und Schnittstellenbeschreibungen und auf Annahmen, die zur Bildung einer sinnvollen E_{MSCS} getroffen werden. Da die Bestimmung von E_{MSCS} nur indirekt die realen MSCS-Events berücksichtigen kann, wird die Erhaltung der Verfügbarkeit als ein Ziel des Eventmanagements zunächst in den Vordergrund gesetzt. In diesem Sinne wird eine erste Annahme zur Bestimmung von E_{MSCS} getroffen:

(CLUA1) Mögliche MSCS-Events, die keine direkte oder indirekte Aussage über Verfügbarkeit beinhalten, werden bei der Erschließung von E_{MSCS} ignoriert.

Einen Teil der o.g. Informationen sind für die Gruppierung nach der Eventquelle erforderlich und wird deswegen an dieser Stelle präsentiert:

Laut Schnittstellenbeschreibung „clusapi.doc“ (siehe []) wird einer Anwendung durch Aufruf der Prozedur `CreateClusterNotifyPort` die Möglichkeit geboten auf einen Kanal für den Zugriff auf eine Queue von MSCS-Events aufzubauen und die Bedingungen zu setzen, die einen Eintrag in diese Queue (Eventlogging) verursachen. Diese Bedingungen werden auch Event-Filter-Flags genannt und wurden in der Header-Datei „clusapi.h“ ausfindig gemacht. Folgender Auszug der o.g. Header-Datei zählt die Event-Filter-Flags auf:

```
1.    CLUSTER_CHANGE_NODE_STATE
2.    CLUSTER_CHANGE_NODE_DELETED
3.    CLUSTER_CHANGE_NODE_ADDED
4.    CLUSTER_CHANGE_NODE_PROPERTY
5.    CLUSTER_CHANGE_REGISTRY_NAME*
6.    CLUSTER_CHANGE_REGISTRY_ATTRIBUTES*
7.    CLUSTER_CHANGE_REGISTRY_VALUE*
8.    CLUSTER_CHANGE_REGISTRY_SUBTREE*
9.    CLUSTER_CHANGE_RESOURCE_STATE
10.   CLUSTER_CHANGE_RESOURCE_DELETED
11.   CLUSTER_CHANGE_RESOURCE_ADDED
12.   CLUSTER_CHANGE_RESOURCE_PROPERTY
13.   CLUSTER_CHANGE_GROUP_STATE
14.   CLUSTER_CHANGE_GROUP_DELETED
15.   CLUSTER_CHANGE_GROUP_ADDED
16.   CLUSTER_CHANGE_GROUP_PROPERTY
17.   CLUSTER_CHANGE_RESOURCE_TYPE_DELETED
18.   CLUSTER_CHANGE_RESOURCE_TYPE_ADDED
19.   CLUSTER_CHANGE_NETWORK_STATE
20.   CLUSTER_CHANGE_NETWORK_DELETED
21.   CLUSTER_CHANGE_NETWORK_ADDED
22.   CLUSTER_CHANGE_NETWORK_PROPERTY
23.   CLUSTER_CHANGE_NETINTERFACE_STATE
24.   CLUSTER_CHANGE_NETINTERFACE_DELETED
25.   CLUSTER_CHANGE_NETINTERFACE_ADDED
26.   CLUSTER_CHANGE_NETINTERFACE_PROPERTY
27.   CLUSTER_CHANGE_QUORUM_STATE
28.   CLUSTER_CHANGE_CLUSTER_STATE
29.   CLUSTER_CHANGE_CLUSTER_PROPERTY*
30.   CLUSTER_CHANGE_HANDLE_CLOSE*
```

Eine Analyse der o.g. Event-Filter-Flags ergibt folgendes:

- Die mit * markierten Event-Filter-Flags (siehe 5. bis 8. und 29. bis 30) werden in der MSCS-Beta3-Version nicht von der Prozedur `CreateClusterNotifyPort` unterstützt und werden demzufolge hier nicht weiter berücksichtigt.

- Die „`CLUSTER_CHANGE`_ClusObj_`PROPERTY`“ - Event-Filter-Flags (siehe 4., 12., 16., 22. und 26.) beziehen sich auf Änderungen der Konfiguration (siehe auch Kapitel „Microsoft Cluster Server“) und finden meistens bei der Installation statt.

- Den „`CLUSTER_CHANGE`_ClusObj_`DELETED`“ - bzw. „`CLUSTER_CHANGE`_ClusObj_`ADDED`“ - Event-Filter-Flags (siehe 2., 3., 10., 11., 14., 15., 17., 18., 20., 21., 24. und 25.) entspricht jeweils ein Event mit der Eventinterpretation *SK_DEL* bzw. *SK_ADD*.

- Die restlichen Event-Filter-Flags stellen Statusmeldungen der Form „`CLUSTER_CHANGE`_ClusObj _`STATE`“ dar, die spezifisch für die jeweilige Softwarekomponente sind. Demnach werden daraus folgende Events, unter Einbeziehung der Informationen aus dem Kapitel „Microsoft Cluster Server“, erschlossen:
  - Für `CLUSTER_CHANGE_NODE_STATE` die Events
    „Cluster_Node_State_Unknown“
    „Cluster_Node_State_Up“
    „Cluster_Node_State_Down“
    „Cluster_Node_State_Paused“
    „Cluster_Node_State_Joining“
    mit der entsprechenden Bedeutung
  - Für `CLUSTER_CHANGE_RESOURCE_STATE`
    „Cluster_Node_State_Unknown“
    „Cluster_Node_State_Up“
    „Cluster_Node_State_Down“
    „Cluster_Node_State_Paused“
    „Cluster_Node_State_Joining“
    mit der entsprechenden Bedeutung

unknown	
inherited	nicht dokumentiert
initializing	nicht dokumentiert
online	
offline	
failed	
pending	
online pending	versucht in den Online-Zustand zu kommen. Falls
offline pending	

  - Für `CLUSTER_CHANGE_GROUP_STATE`
  - Für `CLUSTER_CHANGE_NETWORK_STATE`
  - Für `CLUSTER_CHANGE_NETINTERFACE_STATE`
  - Für `CLUSTER_CHANGE_QUORUM_STATE`
  - Für `CLUSTER_CHANGE_CLUSTER_STATE`

Eine Anwendung, die einen NotifyPort aufgebaut hat, kann über die Prozedur `RegisterClusterNotify` bekanntgeben, daß sie sich nur für ein bestimmtes Cluster-Objekt interessiert und dementsprechend nur die dazugehörenden Events aus der Queue erhalten möchte. Durch Aufruf der Prozedur `GetClusterNotify` kann nun die Anwendung den nächsten Eventeintrag aus der Queue abfragen.

Die restlichen Event-Filter-Flags, also die ohne Markierung und die mit ** markiert sind, kann man nun in 3 Kategorien einteilen:

- Status-Bedingungen,
- Hinzunahme-Bedingungen,
- Ausschluß-Bedingungen.

Wegen der Annahme CLUA1, der obigen Feststellung und den Verfügbarkeits- bzw. Erfolg-Überlegungen, die bei der Gruppierung nach der Eventinterpretation (siehe Kapitel 3.2.3) angestellt wurden, wird eine erste Erschließung der MSCS-Events folgendermaßen vorgenommen:

1. Erschließung:
- Für jede Status-Bedingung wird eine Eventmenge gebildet, die für jede Eventinterpretation einer Softwarekomponente ein entsprechendes Event enthält,
- Für jede Hinzunahme-Bedingung wird eine Eventmenge gebildet, die für jede Eventinterpretation einer Abhängigkeiten ein entsprechendes Event enthält,
- Für jede Ausschluß-Bedingung wird eine Eventmenge gebildet, die für jede Eventinterpretation einer Abhängigkeiten ein entsprechendes Event enthält.

Durch Vereinigung aller von der 1.Erschließung erzeugten Mengen erhält man eine erste brauchbare Version der Menge aller MSCS-Events E_{MSCS} , die hiermit E_{MSCS1} bezeichnet wird.

Weiterhin wurden zwei Header-Dateien analysiert, und zwar Clusmsg.h und Cluserr.h, die Error-Codes für die in „clusapi.doc“ beschriebenen Prozeduren beinhalten. Diese können durch Aufruf der Prozedur „GetLastError“ in Erfahrung gebracht werden. Wegen der Behauptung 3.1 könnten diese Error-Codes auch für die Erschließung der Menge aller MSCS-Events E_{MSCS} wichtig sein und werden deswegen analysiert.

Einige dieser Error-Codes haben gleichen Aussagewert wie Events, die in E_{MSCS1} enthalten sind, und werden deswegen ignoriert

3.5.2 Gruppierung nach den Eventquellen

A) Änderung der Eigenschaften eines Nodes
B) Node
C) Änderung der Eigenschaften einer Ressource-Gruppe
D) Ressourcegruppe
E) Änderung der Eigenschaften einer Ressource
F) Ressource
G) Ressource-Typ
H) Quorum
I) Cluster

3.5.3 Systematisierung derMSCS-Events

ID	Meldungstext	Berück-sichti-gung	Quelle	Interpre-tation	Verar-beitung
	Events aus der Datei Cluserr.h:				
5001L	ERROR_DEPENDENT_RESOURCE_EXISTS	J	Res_2_Gr	O_M	
5002L	ERROR_DEPENDENCY_NOT_FOUND	J	CL_Dpd	A_NV	

5003L	ERROR_DEPENDENCY_ALREADY_EXISTS	J	CL_MkDpd	O_M	
5004L	ERROR_RESOURCE_NOT_ONLINE	J	CL_Res	SK_NV	
5005L	ERROR_HOST_NODE_NOT_AVAILABLE	J	CL_Spec_O	O_M	
5006L	ERROR_RESOURCE_NOT_AVAILABLE	J	CL_Res	SK_NV	
5007L	ERROR_RESOURCE_NOT_FOUND	J	CL_Res	SK_NV	
5008L	ERROR_SHUTDOWN_CLUSTER	J	CL	SK_NV	VSKA
5009L	ERROR_CANT_EVICT_ACTIVE_NODE	J	CL_Evct	O_M	
5010L	ERROR_OBJECT_ALREADY_EXISTS	J	Config		
5011L	ERROR_OBJECT_IN_LIST	J	Config		
5012L	ERROR_GROUP_NOT_AVAILABLE	J	CL_Gr	SK_NV	
5013L	ERROR_GROUP_NOT_FOUND	J	CL_Gr	SK_NV	
5014L	ERROR_GROUP_NOT_ONLINE	J	CL_Gr	SK_NV	
5015L	ERROR_HOST_NODE_NOT_RESOURCE_OWNER	J	CL_MvRes	O_M	
5016L	ERROR_HOST_NODE_NOT_GROUP_OWNER	J	CL_MvGr	O_M	
5017L	ERROR_RESMON_CREATE_FAILED	J			
5018L	ERROR_RESMON_ONLINE_FAILED	J			
5019L	ERROR_RESOURCE_ONLINE	J	CL_Spec_O	O_M	
5020L	ERROR_QUORUM_RESOURCE	J	CL_ResDel oder CL_ResOff	O_M	
5021L	ERROR_NOT_QUORUM_CAPABLE	J	CL_MkQ	O_M	
5022L	ERROR_CLUSTER_SHUTTING_DOWN	J	CL	SK_UV	VSKA
5023L	ERROR_INVALID_STATE	J	CL_Res oder CL_Gr	SK_NV	
5024L	ERROR_RESOURCE_PROPERTIES_STORED	J	Config		
5025L	ERROR_NOT_QUORUM_CLASS	J	CL_MkQ	O_M	
5026L	ERROR_CORE_RESOURCE	J	CL_ResDel	O_M	
5027L	ERROR_QUORUM_RESOURCE_ONLINE_FAILED	J	CL_QOn	O_M	VSKA
5028L	ERROR_QUORUMLOG_OPEN_FAILED	J			
5029L	ERROR_CLUSTERLOG_CORRUPT	J			
5030L	ERROR_CLUSTERLOG_RECORD_EXCEEDS_MAXSIZE	J			
5031L	ERROR_CLUSTERLOG_EXCEEDS_MAXSIZE	J			
5032L	ERROR_CLUSTERLOG_CHKPOINT_NOT_FOUND	J			
5033L	ERROR_CLUSTERLOG_NOT_ENOUGH_SPACE	J			
	Events aus der Datei Clusmsg.h:				
	ERROR_NETWORK_NOT_FOUND				
	ERROR_NETWORK_NOT_AVAILABLE				
	ERROR_NODE_NOT_AVAILABLE	N	analog	zu 5005L	

Tabelle 3.2: Übersicht der Cluster-Errors

3.6 Zusammenfassung und Erkenntnisse

Gruppierung	Allgemeines Eventmanagement-Ziel
Eventberücksichtigung	Minimierung des Korrelationsraumes ca. 25% Filterung
Eventquelle	Informationsgehalt, Weiterleitung
Eventinterpretation	Informationsgehalt vs. Fehlerlokalisierung
Eventverarbeitung	Informationsgehalt

Tabelle 3.3: Zusammenfassung der Systematisierung

- Sehr hoher Kombinationsraum, da ca. 1500 Events übrig.
- Schlechte Dokumenation.

4. Konzeption des MSWEM

Das Kapitel „Systematisierung der Events“ (Kapitel 3) stellt eine umfassende Analyse der Events im Cluster dar. Darauf aufbauend wird jetzt der konzeptionelle Schwerpunkt dieser Diplomarbeit, nämlich der MSWEM, vorgestellt.

Am Anfang (Kapitel 4.1) werden spezielle Ziele des MSWEM dargestellt, die u.a. die Notwendigkeit einer Eventkorrelation aufdecken. Anschließend werden in Kapitel 4.2 die Annahmen zur Minimierung der Eventanzahl gegeben. Danach werden in Kapitel 4.3 einige wichtige Korrelationsgrundlagen vorgestellt um dann in Kapitel 4.4 die Konzepte des MSWEM besser verstehen zu können. Schließlich wird in Kapitel .4 eine Übersicht der im Vorkapitel eingeführten Abhängigkeitsgraphen offengelegt.

4.1 Spezielle Ziele

Das Ziel dieses Kapitels ist es die Gegebenheiten des Clusters und seiner Events (siehe Kapitel 2 bis 4) auf spezielle Forderungen an das MSWEM zu untersuchen. Diese werden dann als spezielle Ziele des MSWEM, d.h. als zusätzliche Ziele des MSWEM im Vergleich zu einem allgemeinen Eventmanagement oder als Bestätigung eines allgemeinen Eventmanagement-Ziels für das MSWEM, formuliert.

SZ1: Minimierung des Kombinationsraumes im Cluster (*Korrelation*)

Dieses spezielle Ziel wird an dieser Stelle durch die im Kapitel 3.6 auf 2^{1500} (enorm hoch) geschätzte Anzahl der tatsächlich zu berücksichtigenden Eventmengen im Cluster unbedingt erforderlich. Die Minimierung des Kombinationsraumes kann durch zwei Maßnahmen realisiert werden:

1. Formulierung einiger sinvollen Annahmen, die die Eventanzahl und implizit die Anzahl der möglichen Eventmengen reduziert,
2. Korrelation der Events, d.h. Verarbeitung von auftretenden Eventmengen um einen gebündelten, hohen Informationsgehalt zu erlangen.

Die erste Maßnahme wird im nächsten Kapitel 5.1.2 realisiert und die Korrelation setzt die Entwicklung eines Eventkorrelators voraus. Dies ist Gegenstand der Kapiteln 5.2 bis 5.4.

SZ2: Berücksichtigung von Komponenten-Dynamik (*Dynamik*)

Einerseits kann die im Cluster auftretende Hardware und Software zum Teil ausgetauscht werden und andererseits kann Hardware und Software in bzw. aus dem Cluster eingefügt bzw. gelöscht werden. Dies wirkt sich dementsprechend auf alle zugeordneten Softwarekomponenten, Abhängigkeiten und Operationen aus, die hiermit allgemein als **Komponenten** bezeichnet werden. Die Auswirkung auf Komponenten kann entweder als Einfügen oder Löschen der jeweiligen Komponente stattfinden und wird als Komponenten-Dynamik verstanden. Der MSWEM sollte auf eine beabsichtigte Komponenten-Dynamik anders reagieren als auf einen unbeabsichtigten Ausfall einer Komponente.

SZ3: Berücksichtigung von Fail-Over (*Failover*)

Im Cluster wird das Fail-Over Konzept zur Ausfallsicherheit sowohl für Nodes als auch für RAID-Platten eingesetzt. Dabei werden gewisse Abhängigkeiten zwischen Komponenten umgesetzt von
Failover-Operationen / Normal-Operationen
Redundanz.

SZ4: Hauptsächlich auf Verfügbarkeit ausgerichtet (*Verfügbarkeit*)

Wie in Kapitel 2.3 dargestellt, ist der MSCS ein Verfügbarkeits-Cluster. Hiermit wird zwischen einer internen Verfügbarkeit, die sich auf Komponenten ohne direkte Vebindung zur Außenwelt bezieht, und einer externen Verfügbarkeit, die sich auf Komponenten mit einer direkten Vebindung zur Außenwelt bezieht, unterschieden. Die interne Verfügbarkeit kann sich auf die externe Verfügbarkeit auswirken, wenn der Ausfall einer Komponente mit direkten Vebindung zur Außenwelt den Ausfall einer Komponenten mit einer direkten Ve-

bindung zur Außenwelt verursacht. Der MSWEM muß die beiden Arten der Verfügbarkeit trennen und unterstützen können. Weil der MSCS ein Verfügbarkeits-Cluster ist, soll der MSWEM hauptsächlich auf Verfügbarkeit ausgerichtet sein.

4.2 Annahmen

Da diese Annahmen die Minimierung des Kombinationsraumes als Ziel haben, aber nur eine Reduktion dessen vornehmen, werden sie im weiteren Verlauf der Diplomarbeit als **Reduktions-Annahmen** bezeichnet.

Allgemeine Reduktions-Annahmen

- (ARA1) Jede einzelne Komponente des Clusters und implizit das Cluster selbst ist zu jedem Zeitpunkt korrekt konfiguriert,

Reduktions-Annahmen für E_{MLX}:

- (**MLXRA1**) Die Rebuild-Events für SYSDEVs (die trotz Abwesenheit eines REBUILD-Zustandes für SYSDEVs existieren) sind redundant zu den Rebuild-Events für PHYSDEVs. Daraus folgt, daß eine ausschließliche Betrachtung der Rebuild-Events für SYSDEVs ausreicht;
- keine - expcap
- (**MLXRA2**) Der „BADBLOCK“-Event kann noch früh genug über die PHYSDEV-Errors abgefangen werden oder er tritt als Folge dieser auf und bedarf implizit keiner weiteren Behandlung ;
- (**MLXRA3**) Der „TYPECHANGED“-Event kann wegen der Annahme MLXA1 ignoriert werden;
- (**MLXRA4**) Der „WRITEBACK_ERROR“-Event signalisiert einen Fehler, der bei der „write-back“-Strategie entstanden ist. Da diese alleine der Leistungssteigerung dient, er ignoriert werden;
- (**MLXRA5**) Mylex-Events, die sich auf die **A**rray **E**nclosure **Ma**nagement **I**nterface beziehen (69, 73 und 78), stellen nur ein zusätzliches, physikalisches (LED-Aufleuchten) Warnmechanismus für die Mylex-Hardware dar und werden deswegen nicht betrachtet;
- reqsense = config
-

Reduktions-Annahmen für E_{NT}:
Alle „Remote“ NT-Netz-Events sind als Local-Events in den jeweiligen Systemen zu betrachten
Alle NT-Netz-Events mit „Spec“ als Eventquelle sind für das jeweilige System lokal zu betrachten, aber Berücksichtigung von Treiber und CluSvc, NetSvc, Time Svc,
Usr_2_Grp , Ch_CompPW (3224: An error occurred while changing this computer's password.) und Spec_O (3513:More data is available than can be returned by Windows NT.) sind Eventquellen, die für die Verfügbarkeit unwichtige Events beschreiben und werden deswegen nicht weiter behandelt.
PDC_NetLg als NetLg, da direkt Netz-Auswirkg.
Keine Config

Quelle	Anzahl von Quelle
L_Alrt_Srv	2
Clock	1
DiskSp	1
Msg_Srv	1
NetLg_Srv	3
Pwr	3
Repl_Srv	2
S_Srv	2
SAM	8
Srv_Ctrl	1
Sys	5
Sys_Res	3

WS_Srv	2
Net	2
Net_Res	1
PDC_Sys	4
Dir	2
Dll	1
Spec_Srv	13
UPS_SDown	3

Reduktions-Annahmen für E_{MSCS}:
Alle Cluster-Anwendungen als Gruppen mit abh. Ressourcen, da sonst sinnlos und da empfohlen.
Alle Ressourcen innerhalb einer Cluster-Anwendungs-Gruppe werden nicht anderen Gruppen zugeordnet, da sinnlos. Speziell hat jede Cluster-Anwendungs-Gruppe eine Ressource vom Typ „Fault-Tolerant Disk Set“, so, daß jeder dieser Ressource eine RAID-LUN zugeordnet ist.
Keine Res-Ops
Es kann nur MoveGroup, BringOnlineGr oder TakeOfflineGr ausgeführt werden
Es gibt mehrere interconnects

Fazit

Als Ergebnis dieses Kapitels wurde eine Reduktion des Kombinationsraumes durch Weglassen unwichtiger oder redundanter Events erzielt. Im Sinne der Minimierung des Kombinationsraumes wird nun die im vorherigen Kapitel erwähnte 2. Maßnahme, nämlich Korrelation, ergriffen. Die Korrelation greift auf den entstandenen, reduzierten Kombinationsraum zurück, der im folgenden nunmehr als **Korrelationsraum** bezeichnet wird.

4.3 Korrelationsgrundlagen

Das vorliegende Kapitel stellt eine allgemeine Analyse der Entwicklung eines Eventkorrelators dar. Dies wird durch das Bestehen verschiedener Vorgehensweisen bei der Entwicklung eines Eventkorrelators und durch die Notwendigkeit dieser Entwicklung (siehe spezielles Ziel *Korrelation*) motiviert.

Während der Entwicklung eines Eventkorrelators (kurz: Korrelator) werden zwei Sichten auf die zu behandelnden Events unterschieden:
- Die **kausale Sicht**, die Ursachen für das Auftreten der Events zu beschreiben versucht,
- Die **operationale Sicht**, die allein die Behandlung der Events betrachtet.

Die kausale Sicht wird vor Beginn der Entwicklung eines Korrelators festgelegt und stellt Annahmen für das Verhalten von Events auf. Darauf greift dann die operationale Sicht zurück und erstellt die Bearbeitungsweisen der Events, was die Erzeugung eines Korrelators bedeutet.

4.3.1 Kausale Sicht

Die Vorgehensweise bei der Untersuchung der Ursachen von Eventsendungen ist komplex. Deswegen abstrahiert man diese reale, informatische Aufgabe und faßt sie in ein mathematisches Modell um einen strukturierten Überblick des Problems zu erreichen.

Während dieses Vorgehens werden alle technischen Schwierigkeiten, die bei der Beschaffung einer vollständigen Informationsmenge (IB) und bei der zeitlichen Koordination dieser Beschaffung zu bewältigen sind, ignoriert.

Man geht zunächst von einer Eventmenge E aller auftretbaren Events und einer Kausalitätsbeziehung „→“ aus, die zwei Events e1,e2 ∈ E verbinden kann. Dabei bedeutet **e1→e2**, daß **e1 e2 auslösen könnte**. Die Kausalitätsbeziehung stellt eine partielle Ordnung auf E dar und kann über einen sogenannten Kausalitätsgraph, der Elemente aus E als Knoten enthält, beschrieben werden.
Desweiteren teilt man E in zwei Teilmengen U bzw. W auf, die Ursachen- bzw. Wirkungsevents darstellen. U enthält Events, auf die man reagieren muß, wobei W solche Events enthält, die man wahrnehmen / beobachten kann. Damit wird klar, daß U und W nicht unbedingt eine Partition von E darstellen.

Durch den bisher eingeführten Formalismus haben wir nun die Voraussetzungen für folgende:

Definition 4.1: Das **Ziel der kausalen Sicht einer Eventkorrelation auf E** ist jedem Event aus W \ U, das man **reine Wirkung / Beobachtung** nennt, ein/mehrere Event/s aus U \ W, das/die man **ursprüngliche Ursache/n** nennt, streng zuzuordnen, d.h. daß man für jede reine Beobachtung alle ihr zugrundeliegenden, ursprünglichen Ursachen mit absoluter Sicherheit ermittelt.

Ohne das Ziel der kausalen Sicht einer Eventkorrelation zu verletzen kann man jetzt eine Vereinfachung des Kausalitätsgraphen vornehmen, indem man folgende **Bereinigungsoperationen** auf ihn durchführt:
(a) Jeder Zyklus wird zu einem Knoten verdichtet,
(b) Jedes e ∈ U ∩ W (sogenannte **indirekte Wirkungen / Beobachtungen**) wird aus dem Graphen entfernt, wobei man den Vorgänger von e direkt auf seinen Nachfolger zeigen läßt.
Der nun erzeugte, bereinigte Kausalitätsgraph ist ein einfacher, n-partiter Graph geworden, wobei n die Maximallänge eines Pfades von einer Ursache zu einer Beobachtung ist. Ein Pfad in diesem Graphen wird hiermit **Kausalitätskette** genannt.

Der bisher beschriebene Kausalitätsgraph entspricht aber immer noch nicht der Forderung nach einer strengen Zuordnung aus der Definition 1, da die Kausalitätsbeziehung noch eine **Unschärfe** - als Gegenbegriff zur Sicherheit - in ihrer Definition (e1 könnte e2 auslösen) beinhaltet. Diese Unschärfe in der Kausalitätsbeziehung stellt eines der größten Schwierigkeiten bei der Konzeption eines Eventmanagements dar. Da aber meistens weitere beschreibende Informationen über den Kontext, in dem ein Event auftritt, vorliegen, wurden ein Konzept entwickelt um diese Unschärfe zu beseitigen. Dieses Konzept basiert auf der Wahl einer Attributmenge um die Unschärfe von Kausalitätsketten berechnen, vergleichen und verknüpfen zu können. Um die weitere Vorgehensweise für verschiedene Attributmengen einheitlich beschreiben zu können, wird folgendes Modell eingeführt:

Definition 4.2: Sei < A, *, + > ein Halbring mit folgenden Eigenschaften:
- A ist eine Attributmenge für die Kausalitätsunschärfe, d.h. jedes Attribut a von A stellt ein Maß für die Unschärfe einer Kausalitätsbeziehung dar,
- ≤ ist eine partielle Ordnung,
- ≤ gilt auf A, d.h. daß $a_1 \leq a_2$, mit a_1 und a_2 aus A, folgendes bedeutet: „Eine mit a_1 markierte Kausalitätsbeziehung ist höchstens so sicher wie eine mit a_2 markierte Kausalitätsbeziehung“,
- * ist eine sogenannte **Katenation**, d.h. eine Operation, die alle Attribute für die einzelnen Kausalitäten einer Kausalitätskette miteinander verknüpft und somit ein sogenanntes Gesamtattribut für die Kausalitätskette berechnet. Dieses Gesamtattribut gibt dann ein Maß für die Unschärfe des Auftretens des letzten Events in der Kausalitätskette, wenn das erste Events in der Kausalitätskette aufgetreten ist, an,
- + ist eine sogenannte **Kombination**, d.h. eine Operation, die die Gesamtattribute mehrerer Kausalitätsketten miteinander verknüpft. Insbesondere wenn die Kausalitätsketten das gleiche Event e am Ende haben, liefert die Kombination ein Maß für die Unschärfe des allgemeinen Auftretens von e.

Ein 3-Tupel (G, < A, *, + >, ϕ) mit G Kausalitätsgraph, < A, *, + > wie oben und ϕ als Mappingfunktion von der Kantenmenge von G auf A (ϕ ordnet jeder Kausalitätsbeziehung in G ein Attribut in A zu) wird **Kausalitätsmodell** genannt.

Im Sinne von Definition 4.2 werden alle Kausalitätsbeziehungen entlang einer Kausalitätskette, die bei einer ursprünglichen Ursache u beginnt und bei einer reinen Wirkung w endet, zu einer sogenannten **Korrelationsbeziehung**, in Zeichen u ⇒ w, zusammengefasst. Wenn man nun in einem gegebenen Kausalitätsgraph G alle

Kausalitätsbeziehungen nach vorherigen Beschreibung durch Korrelationsbeziehungen ersetzt, so entsteht ein bipartiter Graph, dessen Kanten nur von der Partition der ursprünglichen Ursachen in die Partition reinen Wirkungen gerichtet sind. Ein solcher Graph G^K wird hiermit **Korrelationsgraph** genannt. Durch das Ersetzen von G mit seinem entsprechenden G^K entsteht aus einem Kausalitätsmodell ein sogennantes **Korrelationsmodell**.

Folglich kann man mittels eines Korrelationsmodells das Ziel der kausalen Sicht einer Eventkorrelation erreichen, indem man für jede reine Wirkung w und die entsprechende ursprüngliche Ursache u durch Katenation die Unschärfe einer Kausalitätskette von u nach w und durch anschließende Kombination die Unschärfe aller Kausalitätsketten von u nach w berechnet.

Die Wahl der Attributmenge A führt bezüglich eines Korrelationsmodells zu verschiedenen **Korrelationsarten**, die eine Gliederung der kausalen Sicht darstellen. Im folgenden werden einige Korrelationsarten angegeben:

- Die **deterministische** Korrelationsart:
  Es wird eine zweiwertige Attributmenge D = {0, 1} angenommen, deren Elemente sich auf die Kausalitätsbeziehung als sicher existent (= 1) oder nicht existent (= 0) auswirken kann. Die Ordnung $0 \leq 1$ besagt daß eine nicht existente Kausalitätsbeziehung höchstens so sicher ist wie eine sicher existente Kausalitätsbeziehung. Die Katenation wird durch das boolesche $\wedge$ und die Kombination durch das boolesche $\vee$ realisiert.
- Die **probabilistische** Korrelationsart:
  Die Attributmenge wird als P = [0,1] gesetzt, wobei jedes Attribut eine vorher untersuchte Wahrscheinlichkeit darstellt. $\leq$ ist die gewöhnliche „kleiner-gleich"-Beziehung und * ist die gewöhnliche Multiplikation[1] der reellen Zahlen. Die Kombination l_1+l_2 wird folgendermaßen[1] berechnet: $1-(1-a_1)(1-a_2)$.
- Die **temporale** Korrelationsart:
  $A = IR_+$ und stellt Zeiteinheiten a mit der Bedeutung „die Kausalitätsbeziehung erfolgt sicher innerhalb a Zeiteinheiten, d.h. das nächste Event tritt sicher nach a Zeiteinheiten auf" dar. Die partielle Ordnung $\leq$ gilt analog zur vorherigen Korrelationsart. * ist die gewöhnliche Addition und + ist das gewöhnliche Minimum der reellen Zahlen.

Durch Bildung eines Kartesischen Produktes von Attributmengen werden kombinierte Korrelationsarten erzeugt, die ein gegebenes System besser beschreiben könnten.

Folgender Satz:

> „Nach der Wahl einer Korrelationsart liegt ein bestimmtes Korrelationsmodell vor, also ist ein Korrelationsgraph gegeben so, daß zu jedem auftretenden Event seine Ursache bestimmt werden kann."

wäre die logische Folge der obigen Darstellungen. Jedoch ist er meistens falsch, weil die kausale Sicht nur eine idealisierte Untersuchung eines Eventkataloges darstellt - ohne zu berücksichtigen, daß alleine zum Aufbau des Kausalitätsgraphen eines Systems alle Events mindestens ein paar Mal erzeugt werden müßten um auf eine Bestimmung der Kausalitätsbeziehungen hoffen zu können.

Die kausale Sicht dient wegen der Durchführung einiger Kausalitätsuntersuchungen lediglich der Wahl einer Korrelationsart und dem Verständnis der Event-Zusammenhänge. Sie stellt den tatsächlichen Mechanismus, der für das Auftreten aller Events zuständig ist, dar und wird von einem Korrelator nur aproximiert.

4.3.2 Operationale Sicht

Die operationale Sicht setzt einen tatsächlichen Mechanismus, der für das Auftreten aller Events zuständig ist, voraus. Nichtlernende Korrelatoren gehen von einer ausreichend guten Näherung dieses bestimmten Mechanismus aus und lernende versuchen ihre Sicht darauf zu verbessern. Alle Korrelatoren haben aber als primäre Aufgabe die Eventverarbeitung der real aufgetretenen Events. Diese Verarbeitung geschieht aufgrund festge-

[1] Unter Voraussetzung der Attribut-Unabhängigkeit

legter Korrelationstechniken, deren Wahl von der kausalen Sicht und von den architektonischen Gegebenheiten beeinflußt wird.

In den nächstfolgenden Kapiteln werden vier Korrelationstechniken vorgestellt und kurz analysiert. Diese Korrelationstechniken können miteinander kombiniert werden, was zur Erzeugung neuer, hybriden Korrelationstechniken führt. Es wird allerdings eine mögliche Einteilung nach Kätker, siehe [KÄTPAT] gegeben.

4.3.2.1 Model Based - Korrelationstechniken

Model Based - Korrelationstechniken [JAK95] verwenden moderne Expertensysteme um eine Korrelation zu realisieren. Dabei greifen diese Korrelationstechniken auf eine Wissensbasis zurück, die dynamisches Strukturwissen der Komponenten und deren Verhaltensweisen enthält, und benutzt eine Regelbasis, die aus heuristisch erzeugten Korrelationsabläufen besteht. Demnach zeichnen sich diese Korrelationstechniken durch eine hohe Dynamik und eine einfache Handhabung kleiner Regelbasen aus. Allerdings verfügen sie über eine beschränkte Leistung und unterstützen sowohl die Parallelität als auch die Aktualisierung der Regelbasis in vernachlässigbarem Maß. Aufgrund dieser Tatsachen werden die Model Based - Korrelationstechniken vorwiegend auf device-level und als Bindeglied verschiedener Korrelationstechniken eingesetzt.

Diese Korrelationstechniken verwenden für die Wissensbasis beispielsweise:
- Objektorientierte Modelle, um hierarchisches und vererbbares Strukturwissen der Komponenten darzustellen,
- Zustandsautomaten, für die Beschreibung der Verhaltensweisen von Komponenten,

und setzen für die Regelbasis folgende Konzepte ein:
- Regelformulierungen, um Korrelationsabläufe oder Fehlerfortpflanzungen zu charakterisieren,
- Probabilistisches Schließen, um Hypothesen von Fehlerfortpflanzungen zu testen.

4.3.2.2 Fault Propagation - Korrelationstechniken

Diese Korrelationstechniken [KLI95] sind sehr optimistisch, da sie von einer realen kausalen Sicht, also von einer vollständig erfaßten Menge der Ursachen und Beobachtungen einer Fehlerfortpflanzung, ausgehen. Dies ist in der Praxis nur selten möglich, und zwar dann wenn ein kompletter Kausalitätsgraph erstellt werden konnte. In diesem Fall sind diese Korrelationstechniken zur Entwicklung sehr schnellen und sicheren Algorithmen für die Fehlerlokalisierung fähig. Dies ist kein überraschendes Ergebnis, da sie im Gegensatz zu den anderen Korrelationstechniken den Aufwand zur Ermittlung bestehender Kausalitäten nicht mehr betreiben müssen. Falls Komponenten-Dynamik auftritt, werden diese Korrelationstechniken vor einem schwer zu bewältigenden Problem gestellt.

Bekannte Vertreter dieser Korrelationstechniken sind:
- Das Coding-Decoding Konzept
  Aufbau eines sogenannten Codebooks, das die Information der Korrelationsbeziehungen in Form einer Matrix speichert. Beim Auftreten eines Events Suche eines maximalen Matchings der Korrelationsbeziehungen im Codebook,
- Das Schnittmengen Konzept
  Für jedes Event werden Mengen von möglichen Ursachen gebildet, die beim Auftreten geschnitten werden.

4.3.2.3 Model Traversing - Korrelationstechniken

Hier steht das Nachvollziehen der Fehlerfortpflanzung zur Laufzeit mittels einem Abhängigkeitsgraphen der Netzkomponenten im Mittelpunkt [JORD93]. Beim Auftreten eines Events, der als aktueller Event bezeichnet wird, werden von der sendenden Netzkomponente ausgehend die Pfade im Abhängigkeitsgraphen zurückverfolgt bis eine Fehlerlokalisierung stattfindet. Dabei werden frühere Events, deren zugehörigen Netzkomponenten bei der Pfadrückverfolgung besucht werden, mit dem aktuellen Event in Verbindung gesetzt und ana-

lysiert. Die Vorteile dieser Korrelationstechniken liegen in der potentiellen Parallelität, in der Dynamik und der hohen Leistungsfähigkeit. Bei einer komplexen Fehlerfortpflanzung, d.h. für einen vorausgesetzten Kausalitätsgraph der Events mit unübersichtlicher Pfadstruktur, oder bei einem schwer zu ermittelnden Abhängigkeitsgraphen erreichen auch Model Traversing - Korrelationstechniken ihre Grenzen. Der Nachteil bezüglich der komplexen Fehlerfortpflanzung wird hiermit als Mangel an **Flexibilität** bezeichnet.

Für die Abhängigkeiten zwischen den Netzkomponenten dienen z.B.
- Reale schichtbezogene Protokollverbindungen,
- Virtuelle schichtbezogene Verbindungen,

während bei der Feststellung des Bezuges zum aktuellen Event folgende Heuristiken beispielsweise analysiert werden könnten:
- Die Schicht,
- Der Eventtyp,
- Eventuell vorhandene Information über den Kontext, in dem das Event aufgetreten ist.

4.3.2.4 Case Based - Korrelationstechniken

Das Prinzip der Case Based - Korrelationstechniken [LEW93], [NDO93] ist der Vergleich des aktuellen Korrelationsproblems mit anderen ähnlichen Korrelationsprobleme, die früher gelöst wurden. Es wird keine Fehlerlokalisierung sondern eine Fehlerdiagnose angestrebt, die weniger auf eine Fehlerfortfpflanzung als auf eine Analyse der Umstände, in denen ein Event aufgetreten ist, basiert. Da diese Korrelationstechniken lernende Systeme darstellen, benötigen sie eine Trainingphase bevor sie vernünftig eingesetzt werden können. Die Hauptnachteile dabei sind keine akzeptable Dynamik und die Komplexität der Realisierung einer Ähnlichkeitsrelation für Korrelationsprobleme.

Die wichtigsten Vetreter der Case Based - Korrelationstechniken sind:
- Neuronale Netze

  Jedes Event wird einem Neuron im *input layer* und jede resultierende Ursache einem Neuron im *output layer* zugeordnet. Das Training findet durch *backpropagation* statt,
- Verteilte Fuzzy Agenten

  Events werden als externe Stimuli in einem Fuzzy-Agenten empfangen und aufgrund einer *Fuzzy Knowledge Base* und einer *Fuzzy Inference Engine* nach den Prinzipien der Fuzzy-Logik verarbeitet.

4.3.2.5 Zusammenfassung

Dieses Kapitel erstellt eine tabellarische Übersicht der Erkenntnisse über die verschiedenen Korrelationstechniken, so daß ein direkter Vergleich möglich ist. Allerdings werden dabei auch Aspekte einer groben Beurteilung, die Netz-spezifische Einsatzgebiete als Folge der Vor- und Nachteile vorschlägt, dargestellt (siehe Spalte „Empfohlenes Einsatzgebiet“). Die folgende Tabelle 4.1 ist nach Kätker [KÄTPAT] verfaßt worden:

Korrelations-techniken	Vorteile	Nachteile	Empfohlenes Einsatzgebiet
Model Based	Flexibilität, Dynamik	Leistungsfähigkeit, Parallelität, Integration	Netzkomponenten, also Geräte
Fault Propagation	hohe Leistungsfähigkeit, hohe Sicherheit	Vollständig erfaßter Kausalitätsgraph, Dynamik	Eingeschränkte Netzbereiche mit geringer Eventanzahl
Model Traversing	Leistungsfähigkeit, Parallelität, Dynamik	Flexibilität	Netz als geschichteter Graph
Case Based	Lernend, Integration, Flexibilität	Training, Dynamik, Ähnlichkeitsmaß der Korrelationsprobleme	Unklar

Tabelle 4.1: Vergleich der Korrelationstechniken

Bemerkung 4.1: Der Bereich, der Dienste, Anwendungen und Systemkomponenten enthält, wird bei keiner Korrelationstechnik als Einsatzgebiet empfohlen. Dies liegt daran, daß dieser Bereich eine komplexe, hierarchische Struktur der Objekte und ihrer Abhängigkeiten aufweist und keine momentan bekannte Korrelationstechnik für die Bewältigung dieses Problems geeignet ist. Es wird lediglich - bei gegebenen Abhängigkeitsgraph der o.g. Objekte - ein Durchlaufen dieses Graphen als vager Ansatz einer Korrelationstechnik vorgeschlagen.

4.4 Korrelationswahl

Aus den in Kapitel 4.3 vorgestellten grundlegenden Korrelationskonzepte folgt für die Konstruktion eines Korrelators, daß die Frage der Korrelationsart und -technik im Vorfeld geklärt werden sollte. Dies ist der Gegenstand des aktuellen Kapitels. Eine festgelegte Kombination einer Korrelationsart und einer Korrelationssicht wird als eine Korrelationsstrategie bezeichnet.

4.4.1 Wahl der Korrelationsart

Als Folge der kausalen Sicht werden nun einige Überlegungen zur Wahl der Korrelationsart vorgenommen. Durch Untersuchung der Eventkataloge erhält man eine Gesamtanzahl von ca. 1500 zu berücksichtigenden Events im Cluster. Eine vollständige Analyse der Kausalitätsbeziehungen dieser Events wäre unmöglich, da der entsprechende Korrelationsraum das Rechenvermögen der heutzutage bekannten Systeme sprengen würde. Noch schwieriger wäre es jede Kausalitätsbeziehung statistisch zu untersuchen, so daß eine Wahrscheinlichkeitsangabe möglich wird. Daraus folgt, daß eine probabilistische Korrelationsart, bei der jeder Kausalitätsbeziehung eine Wahrscheinlichkeit zugeordnet wird, für die Events im Cluster nicht sinnvoll wäre.

Eine Tendenz der Kausalitätsbeziehungen für Mylex-, NT- und MSCS-Events läßt sich allerdings durch Szenario-Betrachtungen (z.B. wie in Kapitel 4) testweise ermitteln. Als Ergebnis wird eine Menge von relativ sicheren und relativ unsicheren Kausalitätsbeziehungen von Cluster-Events registriert. Dies führt zu einer Berechtigung der Wahl einer deterministischen Korrelationsart mit einer Attributmenge D.

Eine Analyse der Eventstrukturen hat sowohl für Mylex- als auch für NT- und MSCS-Events eine zuverlässige Protokollierung der Auftrittszeiten ergeben. Dadurch entsteht die Möglichkeit für relativ sichere Kausalitätsbeziehungen - also in Kooperation mit der deterministischen Korrelationsart - ein Zeitintervall anzugeben, in dem sie erfolgen. Das bedeutet, daß eine temporale Korrelationsart mit der Attributmenge T zur Verfügung steht.

Die obigen Aussagen zusammenfassend wird nun als kausale Sicht des MSWEM eine **deterministisch-temporale Korrelationsart** mit DxT als Attributmenge festgelegt. Das Kartesische Produkt DxT als Attributmenge ist folgendermaßen zu verstehen: „Jeder Kausalitätsbeziehung wird zugleich eine Existenzbeschreibung $d \in \{$„Ja“(= 1), „Nein“(= 0) $\}$ und ein Zeitintervall t, in dem sie falls d = „Ja“ erfolgt, zugeordnet“. Demzufolge geht der MSWEM von einem deterministischen, mit Zeitattributen behafteten, Kausalitätsgraphen G als tatsächlicher Verursacher von Fehlerfortpflanzungen aus. Dieser Kausalitätsgraph G hat folgende Eigenschaften:

- Entlang eines Pfades P der Länge i werden die einzelnen d_1 bis d_i konjugiert und die einzelnen t_1 bis t_i addiert, also $d_P = d_1 \wedge ... \wedge d_i$ und $t_P = t_1 + ... + t_i$,
- Beim Kreuzen von n Pfade P_1 bis P_n in einem Knoten K werden die einzelnen d_{P1} bis d_{Pn} vereinigt und von den einzelnen t_{P1} bis t_{Pn} wird das Minimum als Ergebnis gesetzt. In Zeichen heißt das also $d_K = d_{P1} \vee ... \vee d_{Pn}$ und $t_K = \text{Min}\{ t_{P1},..., t_{Pn} \}$.

4.4.2 Wahl der Korrelationstechnik

Für die Wahl der Korrelationstechnik als Folge der operationalen Sicht spielen die speziellen Ziele des MSWEM eine wichtige Rolle. Dabei werden die oben beschriebene kausale Sicht und das Cluster als gegeben betrachtet.

Die Dynamik und das Fail-Over als spezielle Ziele des MSWEM führen bezüglich des hier vorausgesetzten Kausalitätsgraphen EG zu einer relativ hoch anzunehmenden Strukturkomplexität der Pfade. Das bedeutet, daß **Flexibilität** - so wie im Kapitel 5.2.2.3 vorweggenommen - an dieser Stelle unbedingt erforderlich ist.

Das spezielle Ziel 4, nämlich die Verfügbarkeit, impliziert eine pragmatische Vorgehensweise damit es realisiert werden kann. Das wirkt sich durch zwei weitere Anforderungen aus:

- **Fehlerlokalisierung vor Fehlerdiagnose**, d.h. daß primär das Auffinden des Fehlers und nicht seine Erklärung behandelt wird,
- **Leistungsfähigkeit**, da bei einem langsamen Vorgehen weniger Chancen bestehen den Ausfall zu verhindern als bei einem schnellem Vorgehen. Hierbei wird angemerkt, daß **Parallelität** der Leistungssteigerung dient.

Eine Zusammenfassung der speziellen Ziele und Anforderungen, die ab jetzt als **Korrelationsanforderungen** des MSWEM bezeichnet werden, gibt nun folgende Tabelle:

Spezielle Ziele	**Anforderungen und ihre Abkürzungen**
Dynamik	$\Rightarrow$ Flexibilität (A1)
Fail-Over	$\Rightarrow$ Flexibilität (A1)
Verfügbarkeit	$\Rightarrow$ Fehlerlokalisierung vor Fehlerdiagnose (A2) $\Rightarrow$ Leistungsfähigkeit (A3) $\supset$ Parallelität (A4)

Tabelle 4.2: Korrelationsanforderungen des MSWEM

In diesem Zusammenhang wird festgestellt, daß keine der im vorherigen Kapitel vorgestellten Korrelationstechniken (siehe Tabelle 5.1) ein perfektes Matching der dargestellten Anforderungen liefert. Zudem fällt - mit Bezug auf Bemerkung 5.x - die Abwesenheit einer Korrelationstechnik für den Systembereich auf. Dabei wird unter Systembereich die Ebene der Dienste, Anwendungen und Systemkomponenten verstanden. Da die MSWEM-Behandlung dieser Objekte im Cluster sehr wichtig ist, wird hiermit eine letzte Anforderung zur Berücksichtigung des Systembereiches (A5) an die in Tabelle 5.2 aufgezählten Korrelationsanforderungen angehängt.

Diese Erkenntnisse führen zu der Notwendigkeit, eine Kombination von Korrelationstechniken für den MSWEM einzusetzen. Es stellt sich die Frage einer sinnvollen Kombination. Zur Beantwortung dieser Frage werden zunächst einige Überlegungen angestellt:

- Case Based - Korrelationstechniken werden wegen der Behandlung der Fehlerdiagnose und der Anforderung A1 für den MSWEM als unbrauchbar eingestuft,
- Model Based - Korrelationstechniken fehlt die für die Anforderung A3 nötige Leistungsfähigkeit,
- Model Traversing - Korrelationstechniken verfügen nicht über Flexibilität, was in A1 gefordert wird. Allerdings ,
- Eine Kombination Model Based / Model Traversing würde bis auf A5 allen Korrelationsanforderungen genügen,
- Fault Propagation - Korrelationstechniken sind nur für geringe Eventanzahlen in einem Kausalitätsgraphen rentabel.

Die Betrachtung der obigen Überlegungen führt zu folgendem MSWEM-orientierten Lösungsansatz, der eine hybride Korrelationstechnik ist:

> Eine Erweiterung des aus Netzkomponenten aufgebauten Model Traversing - Abhängigkeitsgraphen auf Komponenten würde die Objekte des Systembereiches samt ihrer Abhängigkeiten integrieren. Da alle Knoten des so erzeugten Abhängigkeitsgraphen Event erzeugende Komponenten sind, wird dieser Graph Eventquellengraph (EQG) genannt. Damit wären bis auf die Flexibilität alle Anforderungen erfüllt.

Diese Lücke kann aber durch Hinzunahme von Wissen und darauf bezogene Regeln behoben werden. Das Integrieren dieser typischen Model Based - Konzepte könnte wiederum die Leistungsfähigkeit und damit die Verfügbarkeit negativ beeinflussen.

Um dem vorzubeugen wird eine auf Verfügbarkeit ausgerichtete Klasseneinteilung der Events vorgenommen. Eine anschließende Graphverknüpfung von EG und EQG, die Eventklassen berücksichtigt, erzeugt einen neuen Graphen, der Eventklassengraph (EKG) bezeichnet wird. Wegen der Verknüpfung ist der EKG kausal aufgebaut und enthält beträchtlich weniger Knoten = Eventklassen als der EG. Dadurch ist im EKG die Anwendung einer schnellen Fault Propagation - Korrelationstechnik möglich, was den negativen Einfluß der Model Based - Konzepte zumindest ausgleicht.

Die obige Korrelationstechnik wird MSWEM-Korrelationstechnik benannt und liegt den folgenden Ausführungen zugrunde. Die Wissensbasis wird bei der Konstruktion des MSWEM (siehe nächstes Kapitel) implizit aufgebaut und die Regelbasis tritt in Form von Korrelationsregeln, die auf die Wissensbasis zurückgreifen, auf.

4.5 Konstruktion des MSWEM

Dieses Kapitel beschreibt die Konstruktion der Abhängigkeits- und Kausalitätsgraphen, die von der MSWEM-Korrelationstechnik gefordert werden. Dabei wird implizit eine Wissensbasis aufgebaut, die in der Knoten- und Kanteninformation der Graphen enthalten ist. Dadurch werden also allgemeine Informations-Strukturen aufgestellt, die die Grundlage für die späteren Korrelationsregeln darstellen.

4.5.1 Kausalitätsgraph für Events (*EG*)

Hier wird der aus der kausalen Sicht resultierende Kausalitätsgraph für Cluster-Events (EG) vorgestellt. Weil der MSWEM kein lernendes System ist, bleibt der EG im Clusterbetrieb eine konstante Aproximation des realen Kausalitätsgraphen G, der für die Event-Fortpflanzung eigentlich verantwortlich ist.

Der EG stellt ein erster Versuch dar, das erlangte Wissen über die Events in einer Informationsstruktur zu fassen. Er wurde als Kausalitätsgraph modelliert, da sich bei der Wahl der Korrelationsart die Kausalität zwischen Events durch die Untersuchung der Dokumentation festgestellt wurde.

Die Attribute der Graphknoten und die verschiedenen Arten von Graphkanten stellen Elemente der Wissensbasis dar, auf die später die Korrelationsregeln zurückgreifen.

4.5.1.1 Graphknoten im EG

Die Graphknoten im EG stellen reale Events, die nach Kapitel „Systematisierung der Events“ erfaßt wurden, dar. Dabei trägt jedes Event im EG eine Information, die einerseits bei der Erstellung der Eventkataloge erfaßt wurde und andererseits Laufzeit-abhängig ist, mit sich. Diese Information wird nun - von ihrer Erfassung abhängig - in Form von Attributen der Graphknoten vorgestellt.

Attribute aus den Eventkatalogen

- Eventquelle (q)
  Dieses Attribut bezeichnet die Eventquelle q gemäß den Eventkatalogen. Dieses Attribut ist besonders wichtig, da es die Verbindungsstelle zwischen dem EG und dem EQG darstellt. Hierbei wird festgestellt, daß einige Events verschiedene optionale Eventquellen aufweisen (siehe Eventkatalog der NT-Netz-Events). Die Behandlung dieses Problems erfolgt allerdings in späteren Kapiteln.

- Eventinterpretation (i)

Die Eventinterpretation i wird zunächst durch die im Kapitel „Gruppierung nach der Eventinterpretation“ beschriebene Gruppierung vorgegeben. Allerdings wird an dieser Stelle zur Erreichung des speziellen Zieles *Dynamik* eine Erweiterung der Eventinterpretation notwendig. Diese Erweiterung geschieht durch Hinzunahme folgender zwei Interpretationswerte in die Menge I der Eventinterpretationen:

- Softwarekomponente wird eingefügt (SK_ADD),
- Softwarekomponente wird gelöscht (SK_DEL).

Die neuen Eventinterpretationen beziehen sich nur auf Softwarekomponenten, da das Einfügen bzw. Löschen von Operationen - wie im Kapitel EQG beschrieben wird - von der Dynamik der initiierenden Softwarekomponenten abhängt.

- Eventverarbeitung (v)
  Dieses Attribut wird von der Eventverarbeitung v, die bei der Systematisierung der Events vorgeschlagen wurde, übernommen.

Laufzeit-Attribute

- Sendezeit
  Dieses Attribut zeigt den Zeitpunkt t an, zu dem das Event aufgetreten ist. Wegen der gewählten Korrelationsart und der damit verbundenen Attributmenge DxT ist t eine posititve reelle Zahl. Demnach wird die Sortierung einer gegebenen Menge von Events bezüglich ihres Auftretens durch die Sortierung ihrer Sendezeiten ermöglicht. Da die Sendezeit den Events, also Graphknoten, zugeteilt wird, erübrigt sich das temporale Attribut bei den Graphkanten des EG.

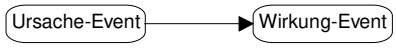
Bemerkungen:

- Unter Berücksichtigung der Annahme ARA1, die eine korrekte Konfiguration voraussetzt, wird hier bezüglich der erweiterten Eventinterpretation gefordert, daß keine nicht dynamische Softwarekomponente eingefügt oder gelöscht wird,
- Die im Anhang B sich befindenden Eventkataloge integrieren die erweiterte Eventinterpretation,
- Die Sendezeit wird als Laufzeit-Attribut erst bei den Korrelationsregeln verwendet.

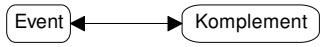
4.5.1.2 Graphkanten im EG

Ein reiner Kausalitätsgraph würde bei gegebener Korrelationsart jeder Kante die entsprechenden Attribute zuordnen, würde diese Kanten aber als Kausalitätsbeziehungen nach der kausalen Sicht betrachten und z.B. keine Schleifen zulassen. Allerdings gehen durch diese einfachen Kausalitätsbetrachtungen semantische Inhalte von Eventbeziehungen verloren. Um dieses Wissen nicht zu verlieren werden hier mehrere Arten von Kausalitätsbeziehungen zugelassen:

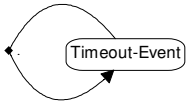
- Die „kausal“-Beziehung
  Diese Beziehung stellt die klassische Kausalitätsbeziehung dar, d.h. sie geht von einem sogenannten Ursache-Event zu einem Wirkung-Event,



- Die „komplement“-Beziehung
  Dadurch werden zwei Events mit gegenteiliger Aussage dargestellt. Die Darstellung (siehe rechtes Bild) deutet auf eine in beide Richtungen mögliche Kausalität hin - jedoch ist diese Beziehung meistens nur in einer Richtung möglich. Die Richtung wird durch Betrachtung der Ein- und Ausgangskanten an jedem beteiligten Knoten ersichtlich und/oder erfolgt aufgrund der Kenntnis eines Verwaltungseingriffes, der diese Beziehung bedingte.
  Falls z.B. eine RAID-Platte ausfällt und ein manueller Austausch dieser Platte stattfindet, wird nach dem Ausbau der alten Platte das Event „GONE“ und nach dem Einbau der neuen Platte das Event „FOUND“ gesendet. Diese beiden Events sind komplementär,

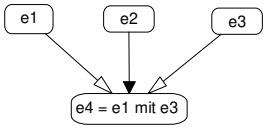


- Die „repetitive“-Beziehung



  Wenn das Auftreten eines gegebenen Events sich wiederholt, da es z.B. durch einen Timeout in seiner Eventquelle erzeugt wird, dann entsteht eine sogenannte „Eigenkausalität“, die im reinen Kausalitätsgraphen nicht modelliert ist. Ein solches Phänomen kann der Ursprung eines Eventstorms sein und ist deswegen sehr wichtig. Im EG wird dies über eine repetitive-Beziehung, die eine Schleife darstellt, realisiert,

- Die „mit“-Beziehung



  Falls ein Event nur aufgrund des konjugierten Auftretens anderer Events erfolgen kann, liegt eine sogenannte „mit“-Beziehung vor. Dabei werden die daran beteiligten Kausalitäten mit weißen Pfeilspitzen markiert (siehe Bild).

4.5.1.3 Analyse des EG

Dieses Kapitel setzt sich die Analyse eines Muster-EG für die Physikalischen Laufwerke (PhysDev) des RAIDs als Ziel. In Kapitel 2.1 wurde eine grobe Funktionsbeschreibung des Mylex-RAIDs gegeben, die als Grundlage des hier in Abbildung 4.1 vorgestellten Kausalitätsgraphen dient:

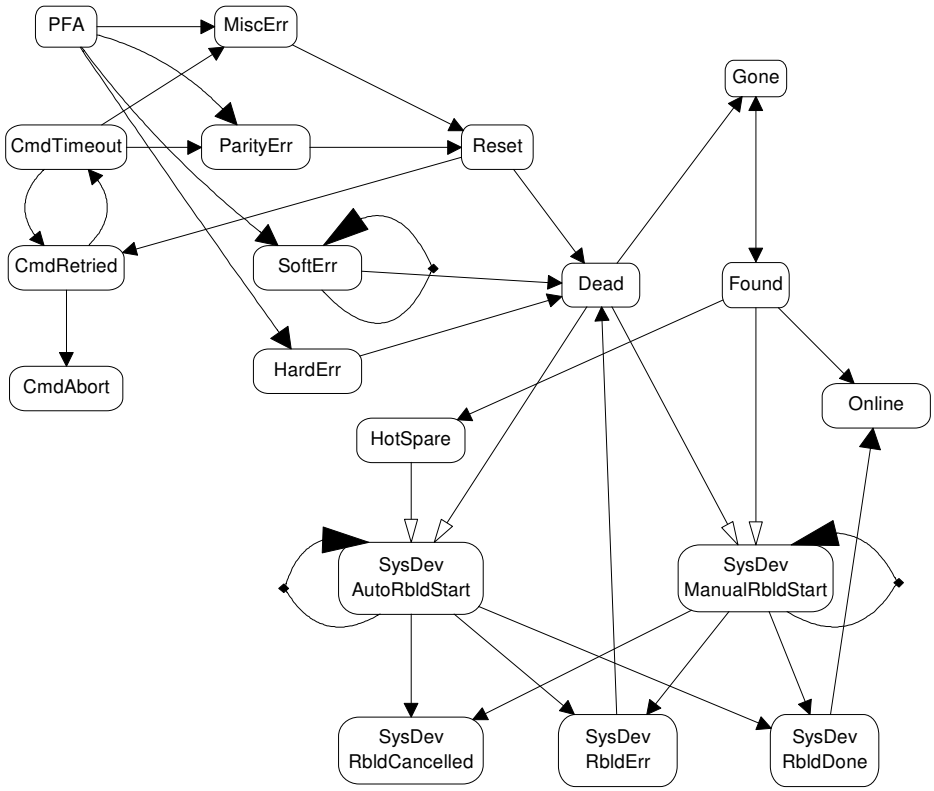


Abbildung. 4.1: Kausalitätsgraph für die RAID-Platten

Alle SysDev-Events wurden explizit als solche gekennzeichnet. Ihre Mitberücksichtigung resultiert aus der Annahme, daß die „SysDev“-Events die „PhysDev“-Events ersetzten können, und aus der Tatsache, daß in diesem Muster-EG alle Aspekte der PhysDev-Events dargestellt werden sollten.

Eine statische Betrachtung des obigen Kausalitätsgraphen führt zu folgenden Feststellungen:

- *P(redictive)F(ailure)A(nalysis)* ist die (theoretische) ursprüngliche Ursache aller Events,

- *CmdAbort*, *SysDev_RbldCancelled* und *Online* sind reine Beobachtungen,
- *CmdRetried* und *CmdTimeout* bilden eine Schleife,
- Es gibt mehrere Zyklen, wie z.B.: *ParityErr*, *Reset*, *CmdRetried*, *CmdTimeout*, *ParityErr*,
- *Gone* und *Found* sind komplementär, wobei *Found* nach *Gone* folgt, da *Found* nur Ausgangs- und *Gone* nur Eingangskanten hat,
- *SoftErr*, *SysDev_AutoRbldStart* und *SysDev_ManualRbldStart* haben repetitive-Beziehungen.

Bemerkungen:

- Redundanz-Informationen (im Sinne von Duplikate) werden im EG über die Eventquelle gespeichert aber nicht angezeigt,
- Der EG ist ein Geflecht von Eventabhängigkeiten, die Inter- und Intra-Eventquellen verlaufen.
- Es wird keine Laufzeit-Information angezeigt.

4.5.2 Abhängigkeitsgraph für Eventquellen (*EQG*)

Einerseits enthält der EG keine direkte Information über die Abhängigkeiten der Eventquellen und eine derartige Struktur ist für die spätere Korrelation wichtig, da sie die fehlerhafte Eventquelle „schnell findet“. Andererseits beinhaltet der EG keine zufriedenstellende Modellierung der „quasi-variablen“-Events, da deren Eventquelle erst zur Laufzeit instanziiert wird. Demzufolge wird ein Graph konstruiert, der die Abhängigkeiten der Eventquellen direkt modelliert.

Der Eventquellengraph stellt die Basis für eine Model Traversing Korrelationstechnik, die den Systembereich mitberücksichtigt, dar. Der EQG speichert durch seinen Aufbau das Wissen, das über die Struktur der Softwarekomponeneten im Cluster vorhanden ist, und zusätzlich besitzt er die Fähigkeit, wichtige Operationen wie z.B. Failover und Informationen, wie z.B. Komponenten-Dynamik und Redundanz zu veranschaulichen.

Erweiterung des Abhängigkeits-Begriffes

Im Hinblick auf die Realisierung des speziellen Ziels *Failover* wird hier eine besondere Art von Abhängigkeiten eingeführt, nämlich die sogenannten **Failover-Abhängigkeiten**. Diese Abhängigkeiten bestehen zwischen Softwarekomponenten, die redundant sind. Dabei wird den Failover-Abhängigkeiten, die tatsächlich bestehen, der Zustand *aktiv* und den Failover-Abhängigkeiten, die redundant bestehen, der Zustand *inaktiv* zugewiesen.

Beispielsweise wird im RAID einem Logischen Laufwerk im Online-Zustand eine *aktive* Failover-Abhängigkeit zu einem Physikalischen Laufwerk im Online-Zustand und eine *inaktive* Failover-Abhängigkeit zu einem Physikalischen Laufwerk im Standby-Zustand zugewiesen.

Der Zustand einer Failover-Abhängigkeit wird durch eine Failover-Operation (siehe spezielles Ziel *Failover*) gesetzt.

Da Failover-Abhängigkeiten nicht nur Beziehungen zwischen Eventquellen darstellen, sondern auch zustandsbehaftet sind, werden sie im EQG mit einer dualen Existenz als Graphknoten und Graphkanten versehen (näheres dazu siehe in den nächsten Kapiteln).

4.5.2.1 Graphknoten im EQG

Das Attribut Evebntquelle für
Die Graphknoten im Eventquellengraph sind zunächst alle möglichen Eventquellen, die bei einer Gruppierung nach der Eventquelle herausgearbeitet werden. Dabei bezieht sich die vorliegende Untersuchung beispielhaft auf Eventquellen, die gemäß den Kapiteln „Gruppierung nach der Eventquelle“ systematisiert wurden.

Das herausgearbeitete Wissen über Eventquellen wird im EQG mittels Attribute der Graphknoten folgendermaßen gespeichert:

Attribute

- Momentaner Zustand der Eventquelle:
  Um ein besseres Verständnis über das Auftreten eines Events zu erlangen, wäre es wichtig zu wissen in welchen Zustand sich eine Eventquelle kurz vor dem Senden befindet. Dafür wird eine zur Eventinterpretation (siehe Kapitel 3.2.3) analoge Gruppierung der Eventquellen nach ihrem Verfügbarkeitszustand eingeführt. Diese Gruppierung hat eine Menge von momentanen Quellenzuständen MQZ = {SK_NV, SK_UV, SK_DNV, SK_DUV, SK_V, SK_SZ, O_NV, O_UV, O_DNV, O_DUV, O_V, O_M, O_E und O_SZ} als Wertemenge. Jeder Eventquelle q wird dann ein momentaner Zustand mqz aus MQZ zugeordnet.

- Bereich der Eventquelle:
  Die Frage nach dem geeigneten Empfänger eines verarbeiteten Events beschäftigt jeden Eventkorrelator. Zur Lösung dieser Frage werden hier beispielsweise Administratoren für drei Verwaltungsebenen des Clusters angenommenen. Das Wissen über die enthaltende Verwaltungsebene wird den Eventquellen in Form eines „Bereichs"-Parameters mitgegeben. Folgende Tabelle schildert die vom MSWEM konfigurierten Verwaltungsebenen bzw. Administratoren und deren zugeordneten Bereich-Werte:

Verwaltungsebene / Administrator	Bereich
Mylex RAID-System bzw. RAID-Administratoren	MLX
NT-PCs und ihr Netzverhalten, bzw. Netz- und System-Administratoren	NT
MSCS und die in MSCS-Gruppen gefaßten Client-Anwendungen bzw. MSCS- und Anwendungs-Administratoren	MSCS/APP

Tabelle 4.3 : MSWEM – Bereiche

  Client-Anwendungen werden hier als Teil des MSCS/APP-Bereiches konfiguriert, da sie laut Annahme als MSCS-Gruppen konfiguriert sind und damit primär der MSCS-Verwaltung unterliegen. Sie sind wegen ihrer besonderen Wichtigkeit im Cluster zusätzlich an dieser Stelle aufgeführt.
  Die Bereich-Einteilung stellt eine Partition dar und kann durch eine Konfiguration des Clusters oder sogar vom Konfigurationsmanagement - falls unterstützt - gesteuert werden.

Einerseits wurden am Anfang dieses Kapitels die Graphknoten nur als Eventquellen modelliert. Andererseits ist jedes Graphelement, das eine Eingangs- oder Ausgangskante enthält, als Graphknoten zu verstehen. In diesem Zusammenhang werden folgende Überlegungen angestellt:
- Eine Failover-Abhängigkeit ist zustandbehaftet (*aktiv*/*inaktiv*),
- Es gibt Eventquellen, die den Zustand einer Failover-Abhängigkeit beeinflussen,
- Demzufolge hat eine Failover-Abhängigkeit Eingangskanten.

Also sind Failover-Abhängigkeiten, trotz ihres Beziehungs-Charakters, Graphknoten im EQG. Allerdings beinhalten sie eine andere Art von Information als die Eventquellen und haben demnach nur folgendes

Attribut

- Momentaner Zustand der Failover-Abhängigkeit:
  Wie im vorherigen Kapitel geschildert, kann eine Failover-Abhängigkeit *aktiv* (kurz: A_A) oder *inaktiv* (kurz: A_I) sein. Demzufolge wird einer Failover-Abhängigkeit ein momentaner Zustand aus der Menge MFAZ = {A_A, A_I} zugeordnet.

Zusammenfassend wird nun folgendes vorgenommen:
- Failover-Abhängigkeiten und Eventquellen werden als Graphknoten des EQG zugelassen,
- Es wird ein gemeinsames Attribut „Momentaner Zustand der Graphknoten" über die vereinigte Wertemenge MZ = MQZ ∪ MFAZ gebildet.

Damit sind die Graphknoten im EQG und ihre mitgeführten Informationen vollständig beschrieben.

4.5.2.2 Graphkanten im EQG

Die Graphkanten im EQG stellen Beziehungen zwischen Softwarekomponenten, Operationen und Failover-Abhängigkeiten dar. Dabei wird aufgrund der aus den Eventkatalogen und aus der Beschreibung des PC-Clusters synthetisierten Informationen eine Grobeinteilung in

- Beziehungen zwischen Softwarekomponenten,
- Beziehungen zwischen Softwarekomponenten und Operationen

und

- Failover-Abhängigkeiten als Beziehungen

vorgenommen.

Wenn eine Beziehung von Graphknoten k1 nach Graphknoten k2 besteht, werden k1 **als beziehender Knoten** und k2 als **bezogener Knoten** bezeichnet.

Im folgenden wird eine Beschreibung dieser Beziehungen und ihre bildliche Symbolisierung angegeben.

Beziehungen zwischen Softwarekomponenten

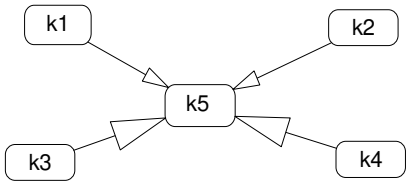
Die Beziehungen, die zwischen Softwarekomponenten auftreten können, sind dreierlei:

- „braucht"-Beziehung

  Diese Beziehungen zeigen eine direkte Abhängigkeit zwischen Softwarekomponenten bezüglich Verfügbarkeit an. Dabei würde z.B. ein Ausfall der Softwarekomponente einen Ausfall der Softwarekomponente k2 bewirken (siehe nebenstehende Abbildung). Mehrere in einem Graphknoten antreffende *braucht*-Beziehungen sind disjunktiv zu betrachten,

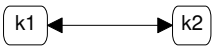
- „und"-Beziehung

  Die *und*-Beziehung stellt mehrere konjugierte *braucht*-Beziehungen dar. D.h. daß die Verfügbarkeit des bezogenen Graphknoten nur durch eine Änderung der Verfügbarkeit aller beziehender Knoten beeinträchtigt wird. Es können auf einem Graphknoten mehrere Klassen von *und*-Beziehungen einwirken, so daß intraklassen-Konjunktion und interklassen-Disjunktion stattfindet. Dies bedeutet, daß entweder ein Ausfall von (k1 und k2) oder von (k3 und k4) den Ausfall von k5 in der nebenstehenden Abbildung verursachen könnte. Verschiedene Klassen von *und*-Beziehungen werden bildlich durch unterschiedliche Größen der Pfeilspitzen gekennzeichnet,



- „redundant"-Beziehung

  Wie es schon aus dem Namen erkenntlich wird, deutet diese Beziehung zwei redundante Softwarekomponenten an. Oft gehen von Softwarekomponenten, die mit der *redundant*-Beziehung verknüpft sind, *und*-Beziehungen zu einem anderen, bezogenen Knoten. Dies beruht auf der Tatsache, daß nur der Ausfall aller redundanten Softwarekomponenten den Ausfall des bezogenen Knoten bewirken würde.



Bemerkungen:

- Es wurde festgestellt, daß Failover-Operationen analog zu Softwarekomponenten in *braucht*-Beziehungen als beziehende oder bezogene Knoten und in *und*-Beziehungen als bezogene Knoten involviert werden können,
- Normal-Operationen kommen höchstens in *braucht*-Beziehungen als beziehende Knoten vor.

Beziehungen zwischen Softwarekomponenten und Operationen

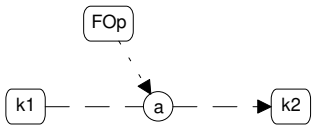
Bei der Systematisierung der Events wurden nur Operationen mit einer auslösenden Softwarekomponente festgestellt. Folgende Beziehung stellt das Ergebnis dieser Feststellung dar:

- „init"-Beziehung

  Die *init*-Beziehung zeigt an, daß eine Softwarekomponente Sk - auch initiierende Softwarekomponente genannt - eine Operation Op anstoßen kann. Sie ist wichtig um die Operation bei einem Einfügen bzw. Löschen der initiierenden Softwarekomponente dementsprechend in den bzw. aus dem EQG einzufügen bzw. zu löschen.

Failover-Abhängigkeiten als Beziehungen

Da Failover-Abhängigkeiten sowohl Graphknoten als auch Graphkanten darstellen, stellen sie als Graphkante eine besondere Beziehung dar. Demnach werden sie gesondert gekennzeichnet und enthalten im Linienverlauf einen Graphknoten, der den aktuellen Zustand anzeigt, und auf dem sich eine Failover-Operation beziehen kann. In der rechten Abbildung besteht eine Failover-Abhängigkeit zwischen den Graphknoten k1 und k2 und sie ist von der Failover-Operation FOp aktiv = a gesetzt worden.



4.5.2.3 Analyse des EQG

Dieses Kapitel hat die Beschreibung eines EQG als Ziel. Dabei wird beispielhaft der EQG für den MSCS/APP-Bereich unter Berücksichtigung der Annahmen aus Kapitel 5.6.2 aufgebaut und anschließend werden die darauf möglichen Strukturänderungen analysiert. Dabei stellen der Aufbau eine Wissensrepräsentation und die Strukturänderungen Laufzeitaspekte dar.

Aufbau

Der Aufbau des o.g. Muster-EQG wird in Abbildung 4.3 graphisch dargestellt. Dabei beziehen sich die in diesem Abschnitt enthaltenen Namen zwischen Anführungszeichen auf Notationen in der Abbildung. Weiterhin werden allgemein in Abbildungen Softwarekomponenten als Rechtecke, Operationen als Wolken und Failover-Abhängigkeiten als Kreise dargestellt.

1. Es wird von einem Cluster mit folgender Konfiguration ausgegangen:
   - Das Cluster besteht aus zwei Nodes, nämlich „Node1" und „Node2",
   - Ein Cluster-internes Netz „Net1_Int" und ein Cluster-Netz „Net2_Ext" zur Kommunikation mit der Außenwelt sind vorhanden,
   - „Node1" ist über die Netzwerkkarten „NIC11" an „Net1_Int" und „NIC12" an „Net2_Ext" angebunden,
   - „Node2" ist über die Netzwerkkarten „NIC21" an „Net1_Int" und „NIC22" an „Net2_Ext" angebunden,
   - Die Anwendungen „WWW-Server" bzw. „SQL-Server" sind als Cluster-Gruppen „GrWWW" bzw. „GrSQLS" konfiguriert. Dabei verfügen beide Gruppen über entsprechende Ressourcen der folgenden Ressource-Typen

     IP-Adresse „ResT_IP,
     Netz-Name „ResT_NetN",
     Fault-Tolerant Disk „ResT_FTD"
     Generic Application „ResT_GApp".

2. Es besteht keine Graphkante zwischen einem Node oder einer NIC und einem Netz, da diese Softwarekomponenten nur den Zugang zum Netz beeinflussen können, nicht aber das Netz an sich.

3. Die beiden Netze haben verschiedene Auswirkungen wegen ihres unterschiedlichen Charakters (Cluster-intern / Cluster-extern). Demzufolge sind auch die entsprechenden NICs beziehende Knoten für verschiedene Eventquellen.

4. Zur Modellierung der Sicht eines Cluster-Clients auf Client-Anwendungen wird eine künstliche Softwarekomponente namens „Anwendung“ eingebaut. Diese Softwarekomponente wird primär von „Net2_Ext“, von den NICs zum „Net2_Ext“ und vom „Cluster“ beeinflußt, da diese Softwarekomponenten die Kommunikation mit der Außenwelt darstellen.

5. Die Failover-Abhängigkeiten, die eigentlich zwischen Nodes und Gruppen bestehen, werden zwischen Nodes und entsprechende Ressourcen gebündelt dargestellt, so daß sie sich wegen den Beziehungen zwischen Ressourcen und Gruppen schließlich auch auf die Gruppe auswirken. Dies erfolgt aufgrund der Annahme, daß nur MoveGroup-Failover-Operationen im Cluster berücksichtigt werden, und aufgrund der Tatsache, daß eine Failover-Abhängigkeit zwischen Nodes und Gruppen zu einem Verlust des Failover-Aspektes der entsprechenden Ressourcen führen würde (siehe Abbildung 4.2). Dabei werden die Zustände der so dargestellten Failover-Abhängigkeiten trotzdem von der MoveGroup-Operation, „MvGr“ bezeichnet, gesteuert.

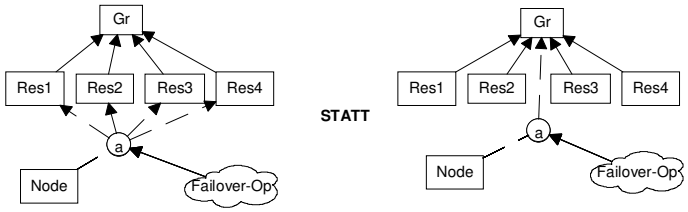


Abbildung 4.2: Darstellung der Failover-Abhängigkeiten

In der Abbildung 4.3 sind „GrWWW“ bzw. „GrSQLS“ und ihre entsprechenden Ressourcen beispielsweise auf „Node1“ bzw. auf „Node2“ aktiv.

6. Die Quorum-Ressource wird im Sinne der vorherigen Überlegungen auch mit Failover-Abhängigkeiten versehen.
   In der Abbildung 4.3 ist die Quorum-Ressource „QUORUM“ auf „Node1“ aktiv.

Strukturänderungen

Redundante Softwarekomponenten im Cluster sind:

Mylex: Lüfter, Temperaturregler, Stromversorgungen, Physikalische Laufwerke oder Controller,
MSCS: Nodes und Netze.

Es wird bemerkt, daß der EQG für jede redundante Softwarekomponente die einzelnen Knoten enthält.

Die Strukturänderungen des EQG realisieren die Anforderungen Dynamik und Failover des MSWEM. Sie beinhalten das Löschen, das Einfügen und das Failover von Graphknoten.

1. Löschen einer Softwarekomponente
   Das Löschen einer redundanten Softwarekomponente aus dem EQG ist gleichbedeutend zu einem Eliminieren aus der Cluster-Sicht. Dabei werden alle Ein- und Ausgangskanten im EQG mitgelöscht. Falls eine der gelöschten Ein- oder Ausgangskanten eine Failover-Abhängigkeit war, müssen alle ihre Eingangskanten - wegen ihrer dualen Knoten-/Kanten-Existenz - wiederum gelöscht werden.

2. Einfügen einer Softwarekomponente
   Beim Einfügen einer Softwarekomponente werden ihre Abhängigkeiten bezüglich der vorhandenen Softwarekomponenten und Operationen gesetzt. Dabei muß auf *und*-Beziehungen und auf *redundant*-Beziehungen geachtet werden: Beim Vorkommen einer *redundant*-Beziehung kann man immer auf das Vorkommen einer *und*-Beziehung zu einem bezogenen Knoten schließen. Wichtig ist auch, daß die Failover-Abhängigkeiten für Ressourcen mit Bezug zur „MoveGroup“-Operation richtig gesetzt werden.

3. Behandlung von Failover
   Wenn eine Gruppe „gemovt“ wird, müssen ihre „alten“ Failover-Abhängigkeiten als inaktiv und die „neuen“ Failover-Abhängigkeiten als aktiv markiert weden.

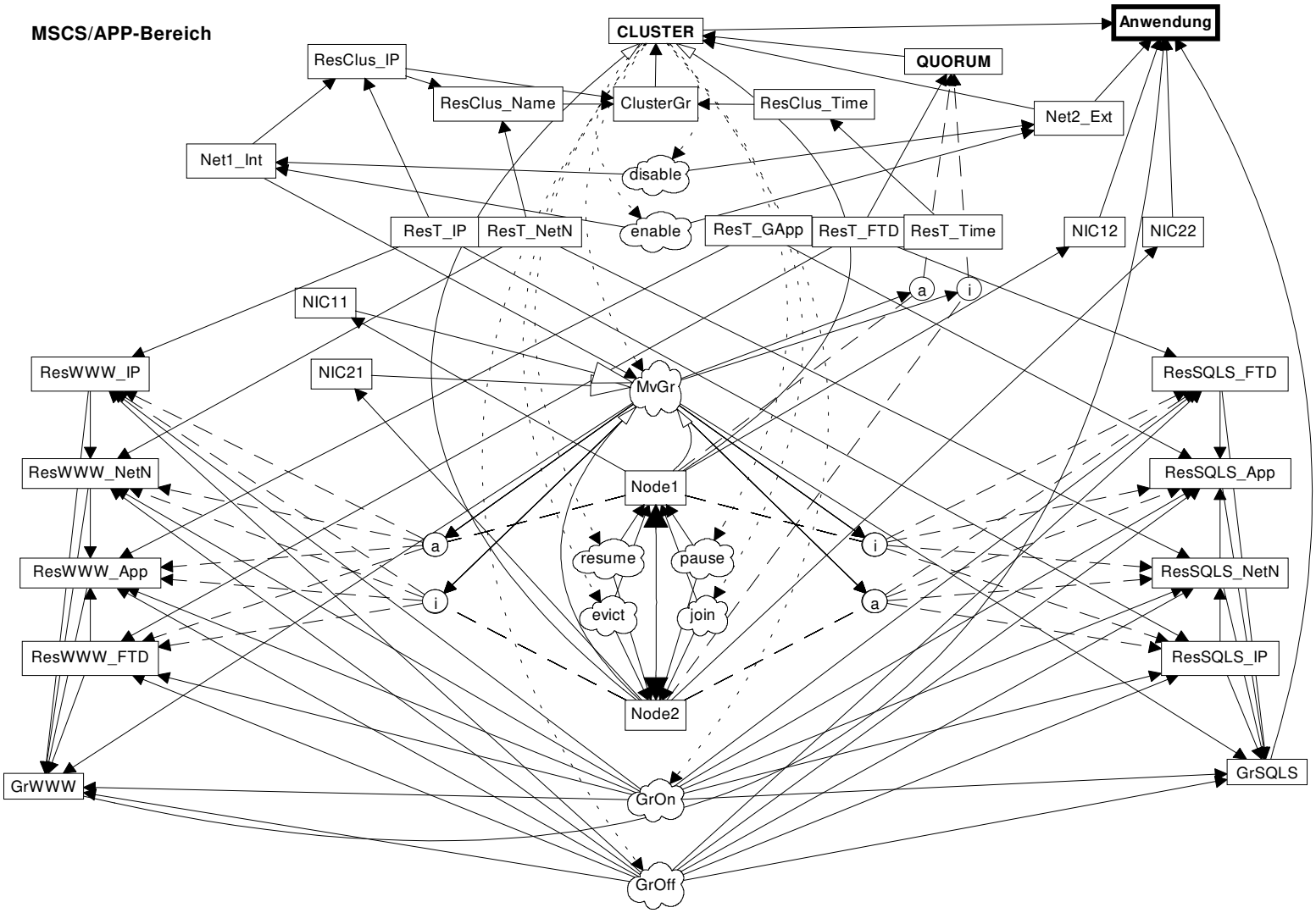


Abbildung 4.3: Abhängigkeitsgraph der Eventquellen im MSCS/APP-Bereich

4.5.3 Eventklassen

Als Ergebnis des EG und des EQG liegt eine vollständige, aber komplexe Struktur von Event-Abhängigkeiten vor. Die zu erstellende Korrelation ist - nur als Fehlerlokalisierung gesehen - eine Suche auf diese Struktur. Demzufolge stellt sich die Frage, wie eine solche Strukur in eine einfachere, möglichst hierarchische Form darstellbar ist. Als Lösung wird hier ein Konzept von Eventklassen vorgeschlagen, deren Abhängigkeits-Graph eine leicht zu durchsuchende Struktur darstellen soll.

4.5.3.1 Erzeugung der Eventklassen

Die Konzeption der Eventklassen bezieht sich hauptsächlich auf die schnelle Fehlerlokalisierung. Da sich die Fehlerlokalisierung auf eine fehlerhafte Eventquelle bezieht, gilt folgendes:

> Eventklassen werden als zusammengefaßte Verfügbarkeitsinformationen über Eventquellen - durch Events angezeigt - verstanden. Eine Eventklasse ist also die Menge aller Events einer Eventquelle, die einen drohenden oder tatsächlichen Fehlerzustand anzeigen, und hat damit eine sogenannte „höchstwahrscheinlich nicht verfügbar"-Interpretation (**HNV**). Damit beziehen sie sich auch auf ein Mißerfolg einer Operationsausführung.

In diesem Sinne gilt folgende Abbildung von Eventinterpretationen auf die Eventklasseninterpretation, die in Tabelle 4.4 dargestellt wird:

Eventinterpretationen	Eventklasseninterpretation
SK_DUV	HNV
SK_DNV	HNV
SK_UV	HNV
SK_NV	HNV
O_DUV	HNV
O_DNV	HNV
O_UV	HNV
O_NV	HNV
O_M	HNV

Tabelle 4.4: Eventlkassen

Zu jeder Eventquelle gibt es eine entsprechende Eventklasse. Es folgt nun die Frage, wie Events den Eventklassen zugeordnet werden. Dafür wird der Event-Eventquelle-Zusammenhang betrachtet und es wird festgestellt, daß manchmal (siehe Anhang B2) zwei Eventquellen einem Event zugeordnet werden. Allerdings hat sich bei der Untersuchung der Event-Dokumentation herausgestellt, daß dies nur eine Laufzeit-abhängige Erscheinung ist. D.h. zu einem gewissen Zeitpunkt kann die Zuordnung des Events zu seiner Eventquelle eindeutig ermittelt werden. Demzufolge wird einem Event während der Korrelation eine eindeutige Eventklasse zugeteilt.

*4.5.3.2 Aufbau des Graphen für Eventklassen (***EKG***)*

Im folgenden Kapitel wird auf die Klassendefinition aufbauend und vom EQG ausgehend ein Graph der Eventklassen (EKG) konstruiert. Dieser Graph wird im Hinblick auf die Korrelation ausgerichtet, ist demnach Laufzeit-orientiert, und sollte möglichst hierarchisch sein. Dafür werden einerseits alle redundanten Informationen oder für die Verfügbarkeit irrelevante Daten aus dem EKG so gelöscht, daß sie in einer indirekten Form noch zum Abfragen bereit stehen. Andererseits werden Klassen aufgenommen, die keine Eventquellen darstellen, um einige Zusammenhänge beizubehalten, die durch den EQG verlorengegangen sind.

In diesem Sinne läuft die Konstruktion von EQG nach EKG in fünf Phasen ab:

- Phase 1.:
  Als erstes findet eine Zusammenfassung redundanter Graphknoten (Duplikate) und eine Elimination der Verwaltungs-Operationen (siehe 4.6) statt. Ersteres steht noch durch das Eventquellen-Attribut der Events zur Verfügung und Zweiteres ist allgemein unwichtig. Die Eventquelle „ResTyp“ für Ressourcen-Typen wird dabei trotzdem nicht gelöscht, da sie einen wichtigen Zusammenhang zwischen DLL und Ressourcen darstellt. Außerdem werden die Failover-Abhängigkeiten im Hinblick auf Phase 2. nicht angezeigt. Es wird eine Klasse „Client/Server“ künstlich eingebaut, die den Netz-Kommunikations-Aspekt eines Systems darstellt. Demnach bündelt diese Klasse die Abhängigkeiten der Kommunikationsdienste und wirkt sich auf die Nodes aus.
  Die Abbildung 4.3 zeigt das Ergebnis der Phase 1. Dabei werden die drei Bereiche entsprechend eingeteilt. Die fett gedruckten Pfeile stellen Bereichsübergänge dar; die halbfett umrahmten Rechtecke sind die Cluster-Dienste und die restlichen Notationen sind wie gehabt.

- Phase 2.:
  In der zweiten Phase werden die Failover-Operationen eliminiert, wobei all ihre Eingangskanten auf alle bezogenen Knoten umgesetzt und gestrichelt angezeigt werden. Dies hat ein Tuning zur Folge und wird durch die Tatsache motiviert, daß eine Failover-Operation an sich selten eine Fehlerursache ist. Abbildung 4.4 stellt das Ergebnis nach dieser Elimination dar.

- Phase 3.:
  Als nächstes findet eine Zusammenfassung der PC-modellierenden Klassen statt. Diese sind: PDC-System, System und Client/Server. Da diese Klassen verschiedene Aspekte eines PCs symbolisieren und ebenfalls im Hinblick auf ein Tuning, ist dieser Schritt vorgenommen worden. In Abbildung 4.5 ist das Ergebnis dieser Phase angegeben.

- Phase 4.:
  Dies ist ein entscheidender Schritt zur Reduktion der Kantenanzahl. Die Elimination redundanter Kanten bedeutet, daß wenn zwischen zwei Knoten eine direkte Kante und ein weiterer Pfad besteht, die direkte Kante gelöscht werden kann. So z.B. kann zwischen dem Cluster-Dienst „Clus_Svc“ und dem Dienst-Controller „Svc_Ctrl“ die Kante gelöscht werden, da zwischen „Clus_Svc“ und dem Cluster-Netz-Dienst „Clus_Net_Svc“ und zwischen„Clus_Net_Svc“ und „Svc_Ctrl“ jeweils eine Kante besteht. Dieser Schritt wird durch die Annahme, daß sich ein Fehler im Graphen gleichmäßig verbreitet, begründet. Der Ergebnisgraph wird in Abbildung 4.6 aufgezeigt.

- Phase 5.:
  Die letzte Phase bezweckt einen möglichst hierarchischen Aufbau, der wegen Phase 4. jetzt durchführbar ist. Dafür werden von der „Wurzel“ aus, nämlich die künstliche Klasse „Anwendung“ nach unten alle Nachfolger der 1. Generation in einer Ebene einsortiert, anschließend wird - vom nächsten Nachfolger aus - die Prozedur wiederholt usw. Falls dabei eine Klasse zu mehreren Ebenen gehören könnte, so wird sie meistens in die oberste eingestuft, damit sie von der Fehlerlokalisierung (siehe Kapitel 4.6.1) so schnell wie möglich entdeckt wird. Die Ungenauigkeit dieses Kriteriums („meistens“) wird durch das erlangte und hier eingebaute Wissen über die PC-Cluster-Architektur begründet. So z.B. befindet sich die Klasse „Svc_Ctrl“ auf Ebene 6 und nicht auf Ebene „0“ (unter „Afd_Svc“), da sie nicht nur ein Nachfolger der Klasse „Afd_Svc“ (Ebene 1), sondern auch von der Klasse„ClusDisk_Svc“ (Ebene 7) ist. Ein Beispiel für eine Ausnahme vom Kriterium wäre die Klasse „Net“, die wegen ihrer Zugehörigkeit zum NT-Bereich auf Ebene 6 (Vorgänger „Net_Ext“) statt 7 (Vorgänger „Net_Int“) lokalisiert wird. Einige horizontale Abhängigkeiten, wie z.B. „PC“ nach „Net“ mußten ebenfalls wegen der PC-Cluster-Architektur in der angegebenen Form modelliert werden. Durch diese letzte Phase wurde eine Hierarchisierung des EKGs vorgenommen, die für die weiteren Korrelations-Untersuchungen eine wichtige Priorisierung festlegt.
  Der endgültige, „fast-hierarchische“ EKG wird in der Abbildung 4.8 zusammen mit den Hierarchie-Ebenen angegeben.

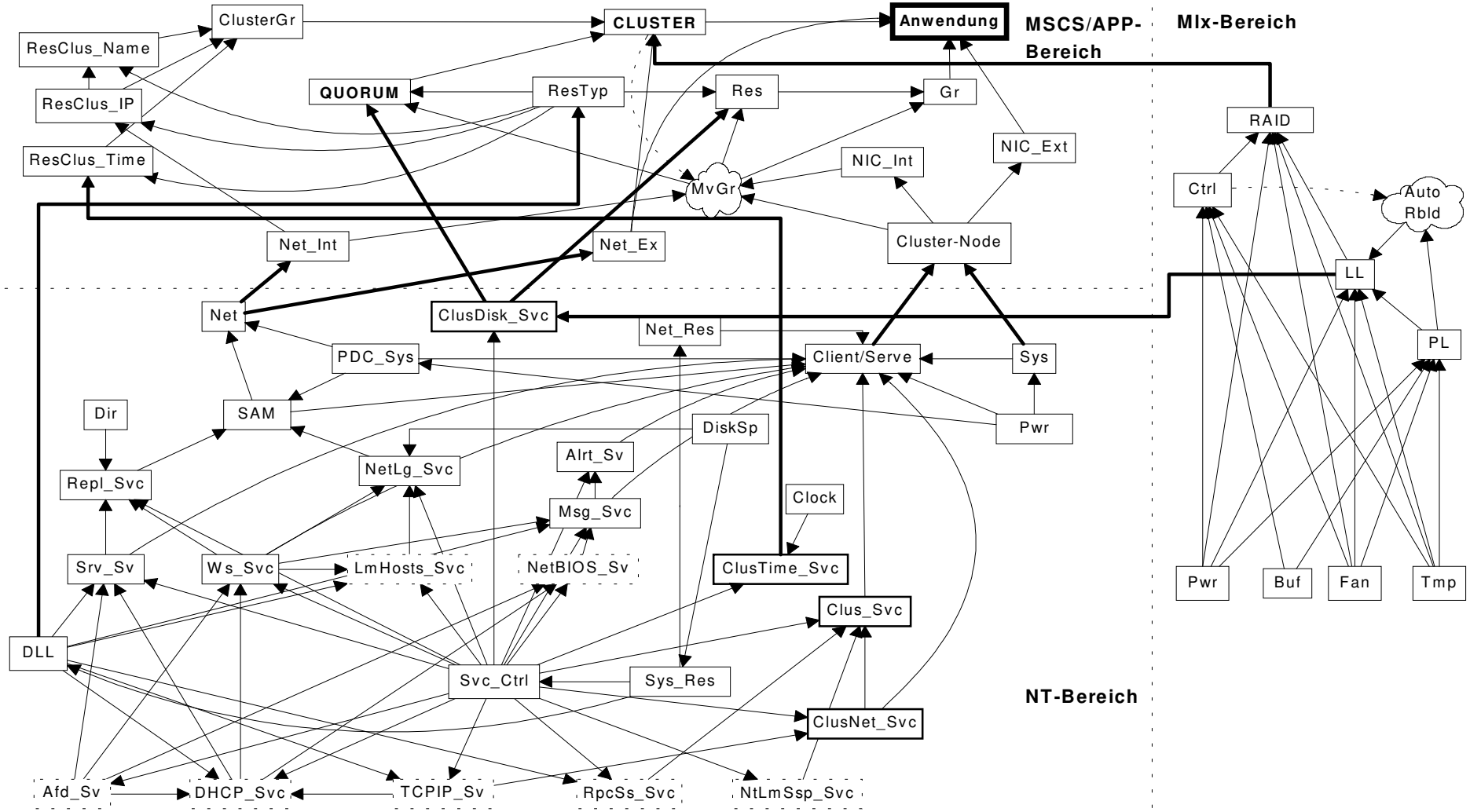


Abbildung 4.4: Phase 1 der EKG-Konstruktion

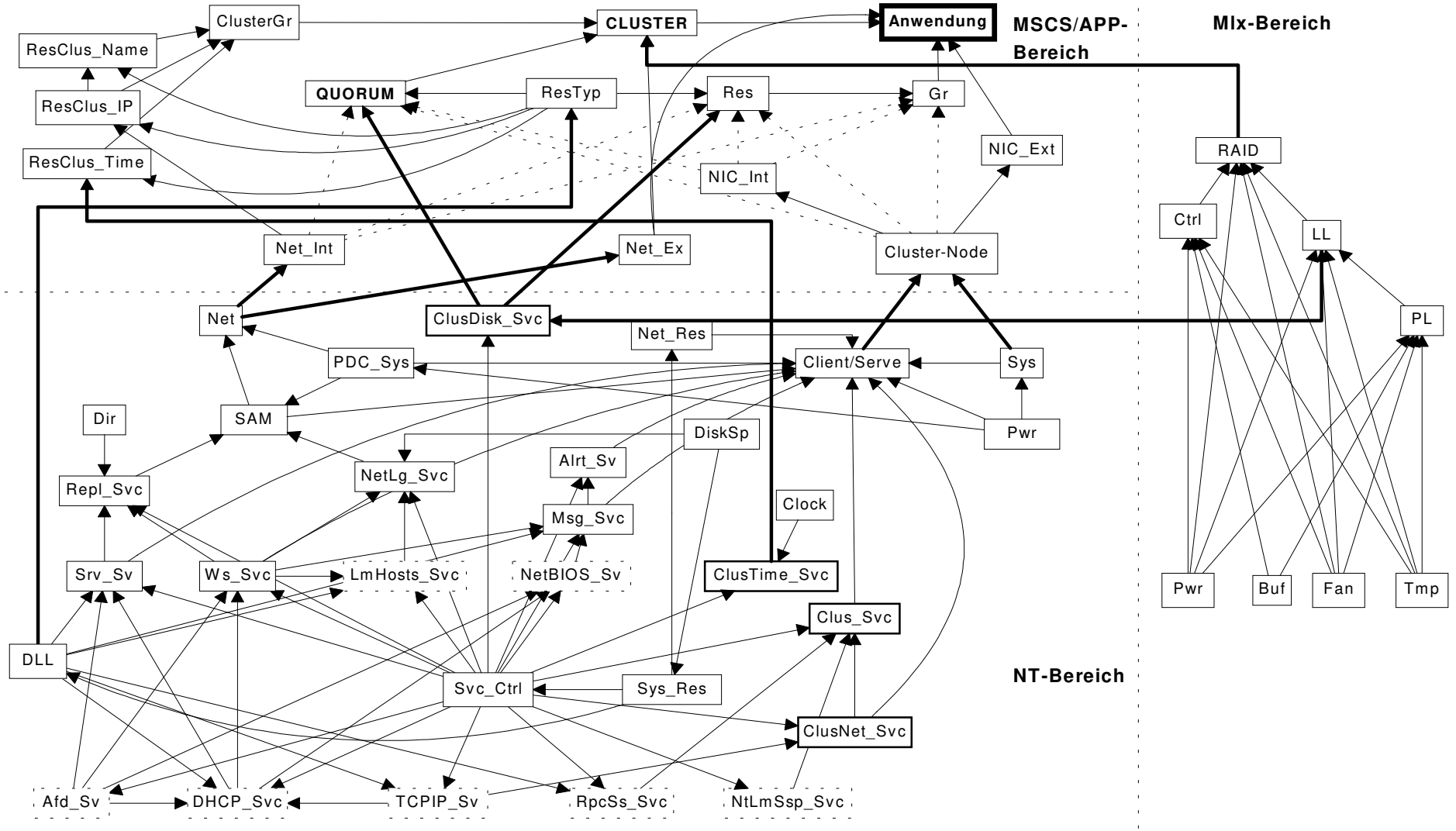


Abbildung 4.5: Phase 2 der EKG-Konstruktion

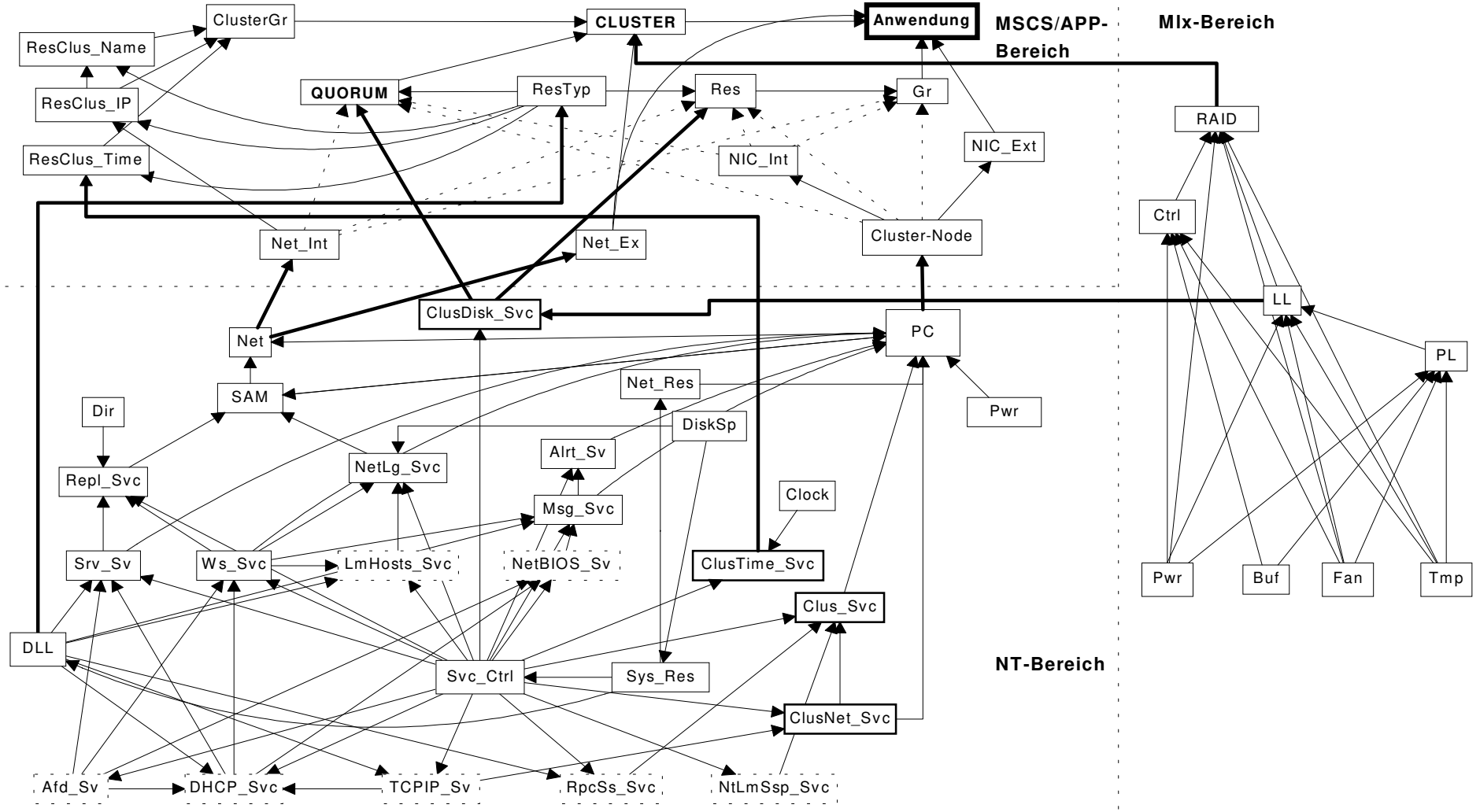


Abbildung 4.6: Phase 3 der EKG-Konstruktion

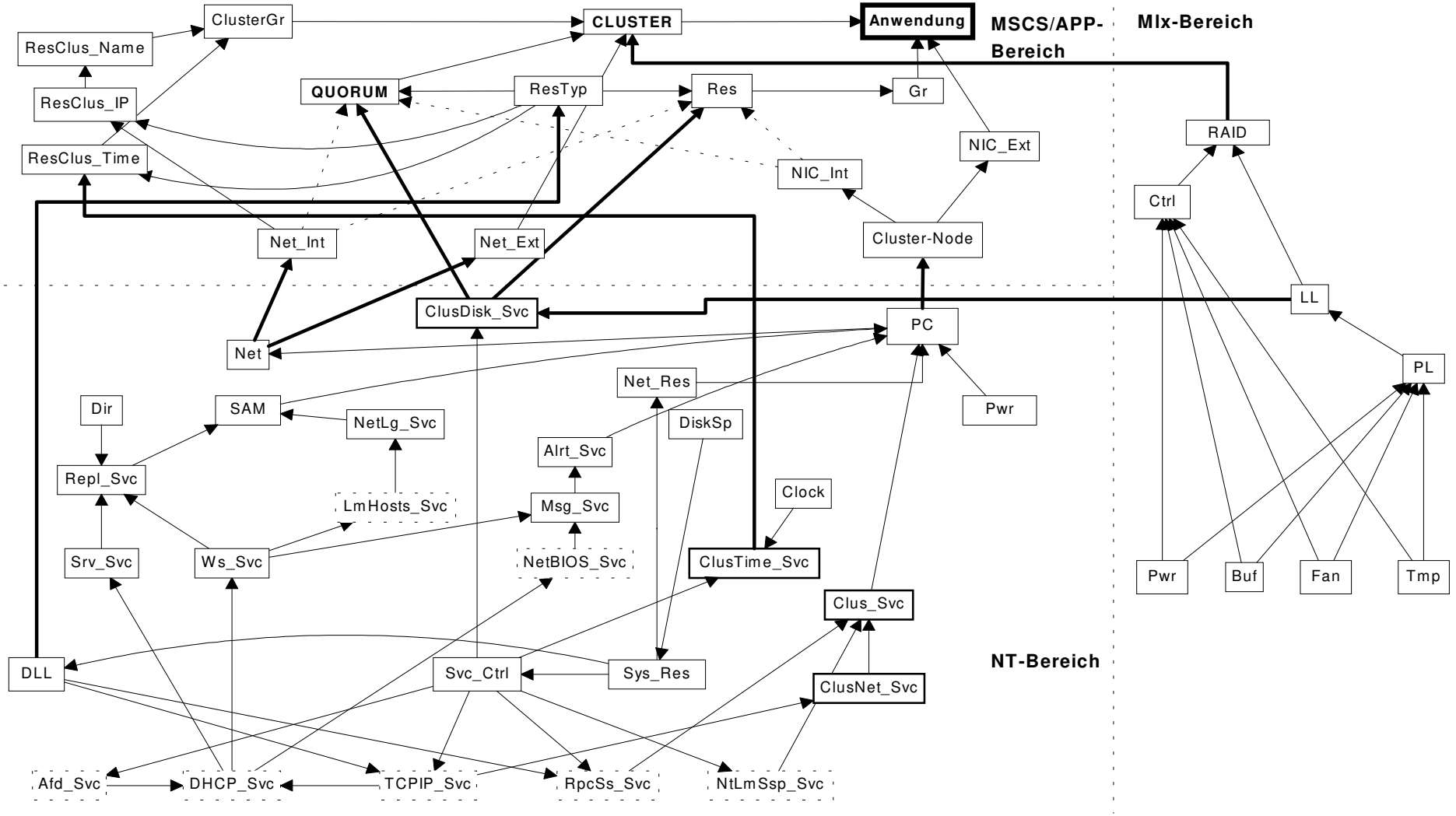


Abbildung 4.7: Phase 4 der EKG-Konstruktion

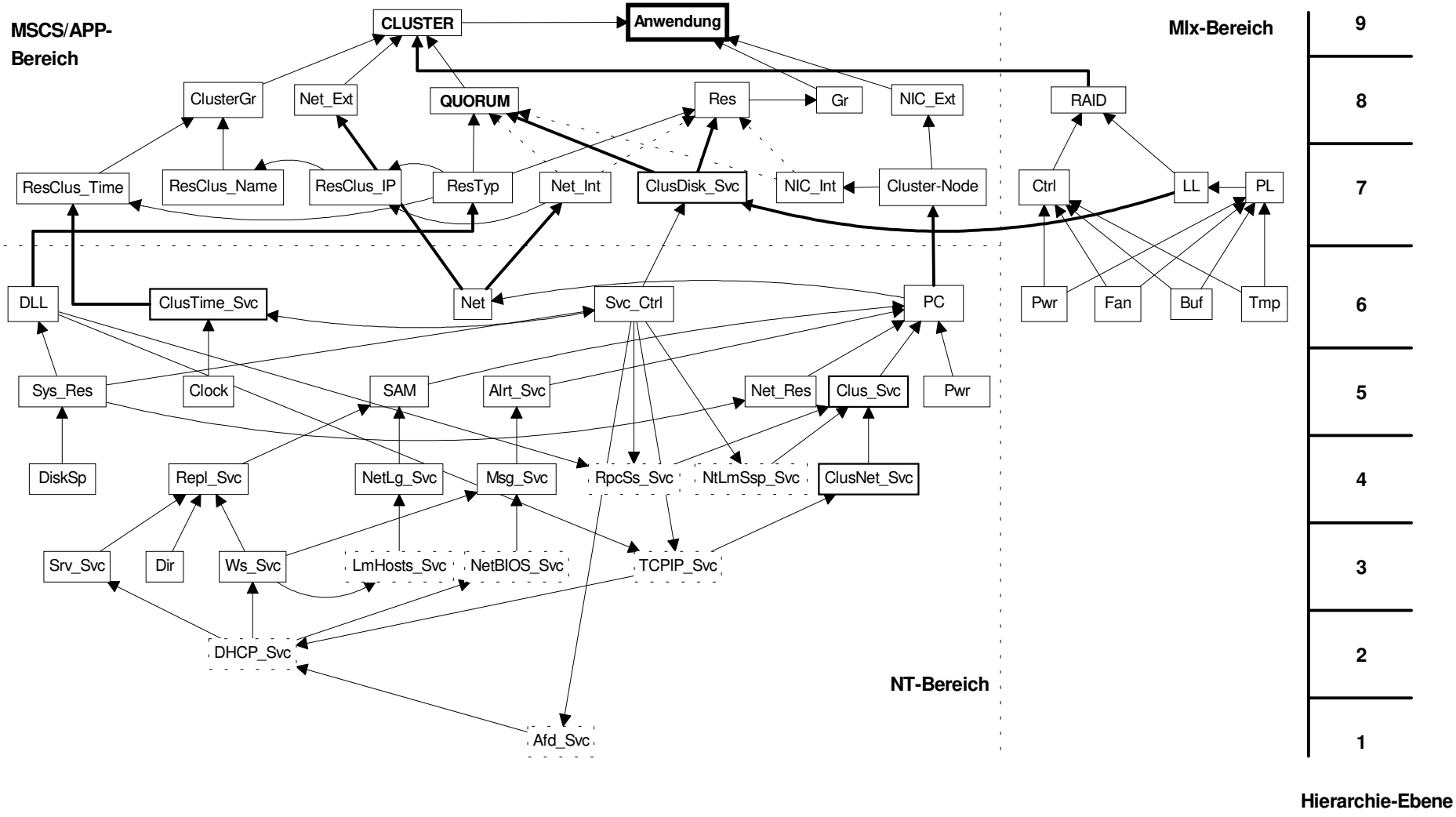


Abbildung 4.8: Phase 5 der EKG-Konstruktion

4.6 Korrelation im MSWEM

Dieses Kapitel setzt sich als Ziel die Korrelationsregeln des MSWEM, die auf die im Kapitel 4.5 angegebene Informationsstruktur zurückgreifen, aufzubauen. Dabei werden das Wissen, das in den Eventkatalogen herausgearbeitet wurde, und Laufzeitaspekte berücksichtigt.

Zu Beginn werden folgende Überlegungen angestellt:

- Operationen, die nur von Administratoren vorgenommen werden können, werden bewust veranlaßt und gesondert überwacht. D.h. eine einfache Weiterleitung reicht aus um den entsprechenden Administrator vom Erfolgszustand zu unterrichten und eine Einbeziehung in die Korrelation ist nicht notwendig. Diese Operationen werden **Verwaltungs-Operationen** genannt und sind folgende:
  Im MSCS/APP-Bereich: *Evict*,
  Join,
  Pause,
  Resume,
  Disable,
  Enable.
  Im MLX-Bereich: *ExpandCapacity*,
  ManualRebuild.
  Demzufolge bleiben für eine Korrelationsbehandlung noch folgende Operationen übrig:
  Failover-Operationen: *MoveGroup* im MSCS/APP-Bereich,
  AutoRebuild im MLX-Bereich.
  Normal-Operationen: *UPS_ShutDown* im NT-Bereich,
  Init im MLX-Bereich.

- Eine Fehlerfortpflanzung im EKG oder im EG kann wegen dem kausalen Charakter dieser Graphen keine Lücken aufweisen. D.h. insbesondere, daß z.B. entlang eines Pfades k1 $\rightarrow$ k2 $\rightarrow$ k3 das Auftreten von k1 und k3 das Auftreten von k2 impliziert.

- Dem Auftreten eines Events liegt immer eine eindeutige Ursache zugrunde. Eine Fehlerfortpflanzung im EG findet konsekutiv statt - also es gibt keine Sprünge entlang eines Pfades. Dies wird durch die Kausalität des EG begründet.

- Zwischen zwei komplementäre Events kann es keine weiteren Beziehungen geben.

Da ein hierarchischer interklassen-Aufbau durch den EKG möglich war, wird eine

Laufzeit-Modellierung

Der MSWEM verwaltet die verarbeiteten Events in sogenannte **Zeit-Pfad-Stacks**. Diese Stacks werden für den EG erzeugt und sind zeitlich geordnet, da die neuen Events - falls sie noch nicht im Stack enthalten sind - immer oben auf den Stack gepusht werden. Die Zeit-Pfad-Stacks sind Pfad-bezogen, d.h. daß es für jeden einfachen Pfad einen Stack gibt und daß bei jeder Pfadverzweigung neue Stacks erzeugt werden - die allerdings an den alten Stack entsprechend der Verzweigung angeknüpft sind. Da die genaue Realisierung der Zeit-Pfad-Stacks das Vorhaben dieser Diplomarbeit überschreitet, wird hier eine derartige Laufzeitstruktur vorausgesetzt ohne darauf im Detail einzugehen.

4.6.1 Korrelations-Operationen des MSWEM

Information

Die Information ist eine einfache MSWEM-Operation, die weniger wichtige Events so kennzeichnet und an das Management-System weitergereicht. Sie wird nur im Sinne einer vollständigen Betrachtung mitberücksichtigt.

Meldung

Im Gegensatz zur Information ist die Meldung wichtig, da sie die Administratoren von Events, auf die er reagieren muß, in Kenntnis setzt. Diese Operation enthält drei Parameter:

- Text beinhaltet den Meldungstext, der meistens gleich dem des Events ist,
- Administrator gibt den zu empfangenden Administrator, abhängig vom Bereich, an,
- Typ zeigt wie schwerwiegend die Meldung ist. Dabei enthält MSWEM folgende Meldungstypen:

Meldungstyp	Bedeutung
Katastrophe	Schwerwiegender Ausfall
Fehler	Ausfall
Warnung	Drohung
Information	Kein Ausfall

Dabei wird bei einer Katastrophe zuerst die Meldung ohne detaillierte Fehlerlokalisierung gesendet, damit der Administrator sofort agieren kann. Anschließend findet dann eine Analyse statt, deren Ergebnisse in einer folgenden Meldung vom Typ „Bezug“ mitgeteilt werden.

Filterung

Die Filterung ist eines der mächtigsten Korrelations-Operationen, da sie aufgrund vorbestimmter Kriterien Events von einer Weiterleitung ausschließt. Der MSWEM unterscheidet drei Arten von Filterung, die im folgenden angegeben werden.

- Filterung nach der Eventberücksichtigung:
  Diese Filterung folgt zwangsläufig aus der Tatsache, das die nicht berücksichtigten Events gar nicht verarbeitet werden. Die Information dazu erhält der MSWEM durch Auswertung der Eventkataloge.

- Filterung nach den Annahmen:
  Aufgrund der in Kapitel 5.2 dargestellten Annahmen findet hier eine zur Eventberücksichtigung analoge Filterung statt.

- Filterung nach der Verwaltung:
  Alle Events, die von Verwaltungs-Operationen ausgelöst werden, sind bewußt von Administratoren eingeleitet worden und müssen nicht korreliert werden. Dabei handelt es sich nicht nur um Events, die Verwaltungs-Operationen als Eventquelle haben, sondern auch um Folgeevents die Zustände von Softwarekomponenten anzeigen. Das bedeutet, daß z.B. die Events „NODE_DELETED“ oder „NODE_STATUS_JOINING“ gefiltert werden.

- Filterung nach dem Kontext:
  Diese Filterung erfolgt aufgrund der Verarbeitung durch den MSWEM und ist - wie der Name schon andeutet - kontextabhängig. Meistens wird sie aufgrund schon verarbeiteter Events angewendet.

Kompression

Die Kompression stellt eine einmalige Weiterleitung sich wiederholender Events dar. Diese Events können durch Timeouts oder durch Verarbeitungsstand-Anzeigen - siehe *repetitiv*-Beziehung im Kapitel 4.5.1.2 - Eventstorms verursachen.

Priorisierung

Die Priorisierung ist eine wichtige Korrelations-Operation, die Events nach gewissen Kriterien Gewichtungen so zuordnet, daß eine weitere Verarbeitung erleichtert wird. Im MSWEM werden folgende Priorisierungen angewendet:

- Gewichtung bezüglich Verfügbarkeit G_V:

In diesem Kontext wird diese Art von Priorisierung als Gewichtung bezüglich eines potentiellen Cluster- oder Anwendungsausfalls verstanden.
Falls die Events eine HNV-Eventinterpretation haben, wird aufgrund des hierarchischen Aufbaus des EKG jedem Graphknoten eine G_V gleich seiner Hierarchieebene zugewiesen. So z.B. hätten die Klassen „Anwendung“ und „Cluster“ die G_V 9, die Klasse „Cluster-Node“ hätte die G_V 7 und die Klasse „Ws_Svc“ hätte die G_V 3.
Events mit der Eventinterpretation SK_V oder SK_SZ, die also nicht im EKG berücksichtigt werden, erhalten eine G_V gleich 0.

Je höher die G_V ist, desto höher ist also die Wahrscheinlichkeit eines schwerwiegenden Ausfalls.

- Gewichtung bezüglich Redundanz G_R:
  Eine andere Art von Priorisierung muß bezüglich redundanter Softwarekomponenten vorgenommen werden. Hier wird die Anzahl n_A noch verfügbarer redundanter Softwarekomponenten in Bezug zur Gesamtzahl n der konfigurierten Softwarekomponenten folgendermaßen gesetzt:
  $G_R = 1 - (n_A - 1) / n$.
  Diese Berechnungsweise garantiert:
  Eine Gewichtung höher als 1, falls keine redundante Softwarekomponente mehr verfügbar ist,
  Eine Gewichtung gleich 1, falls nur eine redundante Softwarekomponente noch verfügbar ist,
  Eine stufenweise sinkende Gewichtung mit der Erhöhung von n_A
  Im den späteren Algorithmen wird in dieser Arbeit als Schwellwert für n redundante Softwarekomponenten 0,5 gesetzt, was einen Ausfall von n/2 dieser Softwarekomponenten bedeutet. Dieser Schwellwert sollte für RAID- Softwarekomponenten wegen der 2-Redundanz (also 2 redundante Softwarekomponenten) der enthaltenen Softwarekomponenten konstant bleiben. Für MSCS- Softwarekomponenten kann er - je nach Node-Anzahl - entsprechend gesetzt werden. Da z.Z. nur zwei Nodes vom MSCS unterstützt werden und eine Konfiguration von einem Cluster-internen und einem Cluster-externen Netz vorausgesetzt wird, ist der 0,5 Schwellwert auch für die MSCS- Softwarekomponenten vertretbar.

- Gewichtung bezüglich Eventinterpretation:
  Die Eventinterpretation liefert aufgrund ihrer Gruppierungswerte eine steigende Reihenfolge deren Gewichtungen. Diese Reihenfolge ist nur für Softwarekomponenten gegeben und zwar folgendermaßen:
  $(SK_SZ = 0) < (SK_V = 1) < (SK_DUV = 2) < (SK_DNV = 3) \leq (SK_UV = 3) < (SK_NV = 4)$.
  Dies ist durch die Gruppierung nach der Eventinterpretation motiviert.

Suppression

Falls ein Event als Folge gemeldeter Events auftritt, so wird seine Meldung zurückgehalten. Allerdings wird das Event für eine spätere Verarbeitung gespeichert.

Fehlerlokalisierung im EKG

Fehlerlokalisierung im EKG sucht unter Hinzunahme einer „IsAlive“-Operation, die den Momentanen Zustand der Graphknoten abfragt, die fehlerverursachende Eventquelle. Dies geschieht zunächst durch Lokalisierung der Eventklasse und anschließende Suche einer potentiell verursachenden Failover-Operation. Dabei werden von einem Knoten ausgehend zunächst die Knoten seiner 1. unteren Ebene bis zum ersten gestörten Knoten nacheinander besucht, dann wird von diesem Knoten ausgehend der Vorgang wiederholt, usw. Unter 1. Ebene eines Knotens wird nicht etwa eine Hierarchieebene im EKG sondern die Menge aller direkt (nach unten) verbundenen Knoten verstanden.
Durch Verwaltung einer Rekonstruktionsliste und einer globalen Bereichübergangsliste stellt diese Operation eingebaute Features für eine sichere Lokalisierung und für eine gezielte Weiterleitung zur Verfügung.

`Fehlerlokalisierung_im_EKG` (IN: e; OUT: Rekonstruktionsliste; OUT: Bereichübergangsliste) {

Gehe von gegebener Klasse aus zum 1. Knoten in der 1. unteren Ebene

`do until` (eine Ebene hat nur Alive-Klassen) oder (keine Klassen mehr vorhanden)

```
        Führe IsAlive auf entsprechende Eventquelle aus
        if nicht Alive then
                Merke Knoten
                if (kein entsprechendes Event auf ein Stack) then
                        Merke diesen in Rekonstruktionsliste
                        Aktualisiere entsprechenden Stack
                end if
                Gehe zum 1. Knoten in der 1. unteren Ebene weiter
        else
                Gehe horizontal weiter
        end if
        Merke Bereichsübergänge
        Merke versteckte Failover-Operationen über punktierte Kanten (siehe EKG-Phase 3)
loop

if (eine Ursache unmittelbar nach einer versteckten Failover-Operation) then
        if (versteckte Failover-Operation hat ein Problem) then
                Setze Failover-Operation als Ursache
        end if
end if
}
```

Korrelation im EG

Durch die Fehlerlokalisierung im EKG wurde eine schnelle Suche der Eventklassen-Fehlerursache vorgestellt. Was geschieht aber, wenn nicht nur HNV-Events empfangen werden oder wenn innerhalb einer Eventklasse mehrere HNV-Events vorhanden sind?

Die Korrelation im EG ist eine zusammengesetzte Operation des MSWEM, die eine Lösung der obigen Frage darstellt. Sie bezieht sich demnach auf Teilbereiche des EG, die Events einer bestimmten Softwarekomponente enthalten, und funktioniert gemäß folgendem Algorithmus:

```
Korrelation_im_EG (IN: e ) {

Finde Zuordnung zur Eventquelle und speichere „Bereich“ und „Momentaner Zustand der Graphknoten“
if Bereichübergangsliste ≠ ∅ then speichere die Listenwerte in „Bereich“ mit

Gehe alle direkten Ursachen durch              {

    if Antreffen von repetitiv-Beziehungen then Kompression
    if Antreffen von komplement-Beziehungen then
          Lösche entsprechenden Stack
          Meldung(„“, Bereich, Information)
    if Antreffen von kausal-Verzweigungen then
        Gewichtung bezüglich Eventinterpretation G_E
        Suche direkte Ursachen, die in Stacks enthalten sind
        if direkte Ursachen in Stacks then                          // d.h.: ALTES EVENT
            for each direkte Ursache u in Stack s do
                if e in s then
                    Suppression              // Zyklus oder Schleife
                else
                    Filterung bezüglich Kontext
            end do
            if mehr als 1 direkte Ursache in Stacks then
                Bilde neuen Stack s und setze die entsprechenden Verknüpfungen
            end if
            pushe e auf s
        else if G_E ≥ 3 then        // also e = SK_NV oder SK_UV oder SK_DNV // d.h.: NEUES EVENT
```

```
                Gewichtung bezüglich Verfügbarkeit G_V
                if Eventquelle = „PC“ then
                    if Eventquelle = „PDC_Sys“ then          // Ausfall des PDC
                        Meldung(„“, Bereich, „Katastrophe“)
                    end if
                end if
                if redundante Softwarekomponente then
                    Gewichtung bezüglich Redundanz G_R
                    if G_R > 0,5 then Meldung(„“, Bereich, „Katastrophe“)
                    else
                        G = (G_E + G_V ) * G_R               // Bilde zusammengesetzten Schwellwert
                        Meldung(„“, Bereich, G-abhängig) // z.B. ab 11 = (4+7)*1 Katastrophe
                    end if
                else
                    G = (G_E + G_V )                         // Bilde zusammengesetzten Schwellwert
                    Meldung(„“, Bereich, G-abhängig)         // z.B. ab 11 = (4+7) Katastrophe
                end if
                Erzeuge neuen Stack s
                Pushe e auf s
            else if G_E = 2 then        // also e = SK_DUV
                Gewichtung bezüglich Verfügbarkeit G_V
                if Eventquelle = „PC“ then
                    if Eventquelle = „PDC_Sys“ then          // Ausfall-Drohung des PDC
                        Meldung(„“, Bereich, „Fehler“)
                    end if
                end if
                Meldung(„“, Bereich, Warnung)
                Erzeuge neuen Stack s
                Pushe e auf s
            else                        // Auch neues Event - aber unwichtig, da höchstwahrscheinlich
                Suppression             // nur durch eine frühzeitige Löschung von Stacks als neu empfunden
            endif
        if Antreffen von mit-Verzweigungen then
            if nicht alle direkten Ursachen in Stacks enthalten sind then
                Merke diese in Rekonstruktionsliste
                Suppression
            else
                if G_E = 3 then
                        Meldung(„“, Bereich, Fehler)
                else if G_E = 2
                        Meldung(„“, Bereich, Warnung)
                else
                        Suppression
                end if
                Bilde neuen Stack und setze die entsprechenden Verknüpfungen
            endif
}
}
```

4.6.2 Korrelations-Algorithmus des MSWEM

Dieses Kapitel stellt einen Korrelations-Algorithmus vor, der auf die Korrelations-Operationen im Kapitel 4.6.1 aufbaut. Dieser Korrelations-Algorithmus beschreibt das Vorgehen des MSWEM beim Empfang eines zu berücksichtigenden Events.

Da im EKG die Operationen herausgefiltert wurden, behandelt der Korrelations-Algorithmus diese gesondert. Dabei wird auf spezielles Wissen aus den Eventkatalogen zurückgegriffen. Z.B. gilt:

- O_V wird als „Operation gestartet“ interpretiert
- Failover-Operationen haben keine O_HNV Events.

Die Failover-Operationen werden teilweise unterschiedlich betrachtet. Dies liegt an der Tatsache, daß Move-Group unmittelbar den Zustand der Gruppe, also der Client-Anwendung, beeinflussen kann, während ein Fehler von AutoRebuild sich erst beim zusätzlichen Ausfall des Redundanz-Laufwerks bemerkbar macht.

```
do while WAHR
{
    Empfange Eventmenge;

    Filtere nach Eventberücksichtigung;
    Information;

    Filtere nach Annahmen;
    Information in Bereichs-abhängige Log-Dateien;

    Filtere nach Verwaltung;
    Speichere Redundanzinformationen, falls vorhanden;
    Lösche Stacks, falls vorhandene komplementär-Events;
    Information in Bereichs-abhängige Log-Dateien;

    Gewichtung der Events in der Eventmenge bezüglich Verfügbarkeit und Redundanz;
    Berechne G = G_V * G_R für jedes Event
    Finde G_max als Maximum von G

    if (mehrere Events mit gleicher G_max) then Gewichtung dieser bezüglich Eventinterpretation

    Bearbeite Event e mit höchster Gewichtung zuerst:

    case of Eventquelle
        1. Eventquelle = Failover-Operation
                if Eventinterpretation = O_V then Suppression        // Gestartet
                if Eventinterpretation = O_E then                    // Erfolg
                        Meldung („“, MSCS/APP oder MLX, Information)
                end if
                if Eventinterpretation = O_M then                    // Schwerwiegender Ausfall droht!
                        if Eventquelle = MoveGroup then
                                Meldung(„“,MSCS/APP, Katastrophe)
                                Finde die direkt verursachende Softwarekomponente sk
                                setze Eventquelle = sk
                                go to 4.
                        else
                                Gewichtung des LL bezüglich Redundanz
                                if G_R = 1 then
                                        Meldung („“, MSCS/APP + MLX, Fehler)
                                        Erzeuge „MLXEV_SYSDEV_CRITICAL“
                                        go to 4
                                else
                                        Meldung („“, MLX, Warnung)
                                end if
                        end if
                end if
        2. Eventquelle = UPS_SDown
                if Eventinterpretation = O_V oder Eventinterpretation = O_M then
```

```
                        if (Gruppe aktiv auf Node) then
                                Meldung(„“,MSCS/APP und MLX, Katastrophe)
                        else
                                Gewichtung bezüglich Redundanz
                                if (G_R > 0,5)
                                        Meldung(„“,MSCS/APP, Katastrophe)
                                else
                                        Meldung(„“,MSCS/APP, Fehler)
                                end if
                        end if
                end if
                if Eventinterpretation = O_E then
                        Gewichtung bezüglich Redundanz
                        if (G_R > 0,5)
                                Meldung(„“,MSCS/APP, Katastrophe)
                        else
                                Meldung(„“,MSCS/APP, Fehler)
                        end if
                        Erzeuge „CLUSTER_NODE_DOWN“
                        go to 4.
        3. Eventquelle = check
                if Eventinterpretation = O_V then   Suppression
                if Eventinterpretation = O_E then   Meldung(„“,MLX, Information)
                if Eventinterpretation = O_M oder Eventinterpretation = O_DUV then
                        Gewichtung des LL bezüglich Redundanz
                        if G_R = 1 then
                                Meldung („“, MSCS/APP + MLX, Warnung)
                        else
                                Meldung („“, MLX, Warnung)
                        end if
        4. Eventquelle = Softwarekomponente
                if Eventinterpretation = SK_V
                        Speichere eventuelle Redundanzinformationen
                        Korrelation_im_EG(e)
                if Eventinterpretation = SK_HNV then
                        Fehlerlokalisierung im EKG
                        Korrelation im EG
   end case
}
```

Fazit:

- Der MSWEM nimmt keine Aktionen zur Fehlerbehebung vor (siehe Ausblick: Ausbau zum TTS),
- Eine Fehlerdiagnose findet nur implizit, in einer einfachen Form statt. Dies ist der Fall, da eine primäre Orientierung auf die Fehlerlokalisierung stattgefunden hat,
- Es werden spezielle Korrelations-Operationen aufgestellt.

5. Realisierungsaspekte des MSWEM

Dieses Kapitel stellt einige Aspekte zur Realisierung des MSWEM vor.

5.1 Vergleich mit regelbasierten Systemen

Dieses Kapitel hat als Ziel den MSWEM mit regelbasierten Systemen, als momentan bekannteste und verbreitetste Eventmanager, zu vergleichen. Dazu wird ein Ansatz zur Modellierung des MSWEM als regelbasiertes System gegeben. Es wird eine allgemeine Prolog-Syntax mit deklarativem Sinn verwendet. Dieser Ansatz versteht sich als eine kuze Prädikat- und Regel-Angabe, ohne daß auf Überlegungen zur Datenstruktur, Prolog-Programmierstil o.ä. Details eingegangen wird.

Im Hinblick auf die Modellierung als Regelbasis werden zuerst die Aktionen festgelegt. Diese sind:

- Vernachlässigung (e)
- Speichern_in_Liste (L, e)
- Abfragen_aus_Liste (L, e)
- Lösche_Liste (L)
- Erzeuge_Liste (L)
- Melden_als_Katastrophe (e, Administrator)
- Melden_als_Fehler (e, Administrator)
- Melden_als_Warnung (e, Administrator)
- Melden_als_Information (e, Administrator)

Dabei entsprechen die Listen den in Kapitel 4 erwähnten Stacks, die aufgetretene Events speichern.

Mit sk wird eine Softwarekomponente und mit e ein Event bezeichnet

Entsprechend den Gruppierungen (siehe Systematisierung, Kapitel 3) werden zweistellige Prädikate über ein Event e und dem jeweiligen Gruppierungswert aufgestellt. So z.B. bedeuten:

Berücksichtigung (e, N)	:	e wird nicht berücksichtigt,
Eventquelle (e, Cluster-Node1)	:	Die Eventquelle von e ist der Cluster-Node1, usw.

Für die vorgenommenen Priorisierungen/Gewichtungen können zweistellige Prädikate über das Event e und dem Gewichtungswert definiert werden:

GR(sk,1)	bedeutet, daß die Priorisierung nach der Redundanz für sk gleich 1 ist,
GV(sk,6)	bedeutet, daß die Priorisierung nach der Verfügbarkeit für sk gleich 6 ist,
GE(e,6)	bedeutet, daß die Priorisierung nach der Verfügbarkeit für e gleich 4, also SK_NV ist.

Seien alle Attribute der Events im EG und der Eventquellen im EQG zunächst analog zu den obigen Definitionen als zweistellige Prädikate gegeben. Seien alle Kantenarten, die zwei Knoten im EG oder im EQG verbinden, - bis auf *repetitiv*- und *mit*-Beziehungen im EG und *und*-Beziehungen im EQG - als zweistellige Prädikate modelliert.

Der Bereich einer Softwarekomponente, dessen Event e auftgetreten ist, wird durch das Prädikat Bereich(e) gegeben. Das Logische Laufwerk, worauf sich eine Auto-Rebuild-Operation beim Auftreten eines Events e bezieht soll durch Auto-Rebuild(e, LL) gegeben werden.

Folgende Regeln können jetzt beispielhaft angegeben werden:

- Vernachlässigung :- Berücksichtigung (e, N)
  Event e wird nicht berücksichtigt und demnach vernachlässigt,

- Melden_als_Katastrophe (e, Bereich(e)) :- GR(e,1), Eventquelle(e, Cluster-Node),

Wenn sich noch ein Node im Cluster befindet, und es wird ein Ausfall-Event über ihn empfangen, dann wird ain Katastrophen-Alarm ausgelöst,

- Melden_als_Information (e1, Bereich(e1)), Lösche_Liste (L) :- komplementBeziehung (e1, e2)
  Lösche die Events in der entsprechenden Liste L und melde als Information, wenn angetroffenes Event e1 ein sich in der Liste L befindende Event e2 per *komplement*-Beziehung matcht,

- Melden_als_Fehler (e1, Bereich(e1)) :- Eventquelle(e, Auto-Rebuild), Eventinterpretation(e, O_M), GR(Auto-Rebuild (e, LL) ,1),
  Melde als Fehler, wenn ein Auto-Rebuild eines Logischen Laufwerks LL scheitert und die Redundanz des LL gleich 1 ist. Dabei bezieht sich die Redundanz eines LL auf die Anzahl der Standby-PL, worüber es noch verfügen könnte.

Diese Beispielskette könnte noch komplexer und umfangreicher gestaltet werden. Jedoch wurden die Grundmechanismen damit aufgezeigt und deswegen wird an dieser Stelle die Beispielskette abgebrochen.

5.2 Integration weiterer Treiber und Anwendungen

Bei der Konzeption des MSWEM in Kapitel 4 wurde versucht die Konstruktion des Korrelators anhand einiger Beispielen aufzuzeigen. Dabei wurden auch Strukturänderungen der Eventquellen, wie das Einfügen oder das Löschen von Softwarekomponen, im EQG berücksichtigt (siehe Kapitel 4.5.2.3). Allerdings wurde eine Beschreibung zur Einbindung weiterer Softwarekomponenten in den EG und in den Korrelationsmechanismus nicht angegeben. Aufgrund dieser Überlegungen bezweckt dieses Kapitel eine allgemeine Anleitung zur Integration zusätzlicher Eventquellen in den MSWEM. Dabei wird exemplarisch auf Treiber und Anwendungen Bezug genommen.

Als gemeinsames Vorgehen werden zunächst folgende Schritte angegeben:

- Die Menge aller Komponenten-Events muß erfaßt werden,
- Eine zu Kapitel 3 entsprechende Systematisierung sollte vorgenommen werden - d.h. daß die Eventquelle, falls die neue Komponente keine Event-sendende Unterkomponenten hat, nicht mehr erfaßt werden muß,
- Betrachtung sinnvoller Annahmen, um die Eventmenge - nach Filterung der Eventberücksichtigung - zu minimieren,
- Die erzeugte Eventmenge sollte nun für eine Integration in den EG folgendermaßen untersucht werden:
  - Feststellung der kausal-Beziehungen,
  - Feststellung der mit-Beziehungen,
  - Feststellung der repetititv-Beziehungen,
  - Feststellung der komplement-Beziehungen.

Dadurch ist jetzt die Eventmenge der neuen Komponente in sich vollständig untersucht. Die weiteren Schritte beinhalten eine Einbindung in die Umgebung und werden getrennt behandelt, da sich Treiber und Anwendungen diesbezüglich unterscheiden.

5.2.1 Integration weiterer Treiber

Treiber werden in NT ähnlich verwaltet wie Dienste. Demzufolge sollten bei der Integration weiterer Treiber in den MSWEM folgende zusätzliche Schritte vorgenommen werden:

A) Einbindung in den NT-Bereich (und EQG)
   Das bedeutet, daß die Abhängigkeitsbeziehungen des neuen Treibers und all seinen Unterkomponenten auf der NT-System-Ebene klargestellt werden sollten. Dabei werden einerseits Abhängigkeiten zu vorhandenen MSWEM-Eventquellen hin erzeugt und andererseits werden Abhängigkeiten von MSWEM-Eventquellen geschaffen. Als Dienste sind Treiber jedenfalls vom Dienst-Controller und

damit von System Ressourcen und vom Plattenplatz als MSWEM-Eventquellen abhängig (siehe Abbildung 4.8).

B) Integration in den EKG

Durch die unter A) in Erfahrung gebrachten Abhängigkeiten kann der neue Treiber nun als Eventklasse in den EKG eingebunden werden. Da in keiner Konstruktionsphase des EKG NT-Dienste eliminiert werden, ist das Vorgehen zulässig. Falls der Treiber Operationen oder andere Unterkomponenten hat, die nicht gefilterte Events (durch die Systematisierung) senden können, müssen diese entsprechend ihrer unter A) erfaßten Abhängigkeiten in den EKG eingebracht werden. Dabei müssen die fünf Phasen der Hierarchisierung vorgenommen werden.

C) Integration in die Korrelation

Wegen der Integration in den EKG und der Integration in den EG hat der Treiber jetzt alle Voraussetzungen, um in einer Korrelations-Operation des MSWEM eingebunden zu werden. Da er als Dienst keinen Sonderfall im Algorithmus, sollte nur noch untersucht werden, ob vielleicht eine seiner Unterkomponenten einer besonderen Behandlung bedarf.

5.2.2 Integration weiterer Anwendungen

Die zu integrierenden Anwendungen werden als Cluster-aware Anwendungen vorausgesetzt. Diese sind Anwendungen, die als Cluster-Ressource modelliert werden könen. Dazu werden folgende Voraussetzungen benötigt:

- Die Anwendung kann einem Ressourcen-Typ zugeordnet werden,
- Die Anwendung verfügt über einen Mechanismus, der ihre Daten und Ausführungszustände bei einem unerwarteten Herunterfahren der Anwendung sichert. Dies wird im Hinblick auf ein Failover gefordert.

Bei der Integration weiterer Anwendungen in den MSWEM sollten folgende zusätzliche Schritte vorgenommen werden:

A) Einbindung in den MSCS/APP-Bereich (und EQG)

Hierbei werden die Anwendungen als MSCS-Gruppen modelliert. Die der Anwendung entsprechenden MSCS-Ressource wird vom Typ „GenericApplication“ konfiguriert. An dieser Stelle muß man die Failover-Abhängigkeiten zu Nodes, Netze und NICs, und alle anderen Abhängigkeiten zu ClusterDisk-Dienste setzen. Dabei sollten eventuelle Unterkomponenten, die nach der Systematisierung noch zu berücksichtigende Events senden, mit einbezogen werden. Nicht zu vergessen sind die Abhängigkeiten zwischen der „MoveGroup“-Operation und den Failover-Abhängigkeiten.

B) Integration in den EKG

Durch die Einbindung in den EQG und der Integration in den EG kann die Anwendung jetzt samt ihrer Unterkomponenten - nach Durchlaufen der fünf Phasen in den EKG integriert werden.

C) Integration in die Korrelation

Diese Integration läuft analog zu den Treibern ab.

5.3 Integration mit dem Statusmanagement

Ein weiterer Aspekt der Realisierung des MSWEM ist die Integration mit dem Statusmanagement. Dabei beziehen sich die folgenden Ausführungen ausschließlich auf das Statusmanagement, das in einer vorangehenden Diplomarbeit hinsichtlich des Managements eines PC-Clusters erstellt wurde und das ab jetzt einfach als „Statusmanagement“ referenziert wird. Da das Statusmanagement SNMP-basierend aufgebaut wurde, wird implizit durch diese Integration die Ausrichtung des MSWEM auf das Internet-Management ersichtlich.

5.3.1 Zustands-Aspekte der Integration

Die Zustands-Aspekte bei der Integration mit dem Statusmanagement beziehen sich hauptsächlich auf eine Analyse der sinnvoll zu übernehmender Zustände und eine anschließende Interpretations-Umsetzung dieser.

Übernahme

Basierend auf die Konzeption der Cluster-MIB, der RHost-MIB als Erweiterung der Host-MIB und der VHost-MIB erreicht das Statusmanagement eine erste Zustandsbestimmung der Ressourcen über Anwendung der „Instrumentierungen betroffener Agenten“. Dabei werden unter Ressourcen die Instantiierungen aller MIB-Gruppen der o.g. MIBs verstanden. Da diese Gruppen entweder implizit oder explizit die im EQG definierten Bereiche MLX, NT und MSCS/APP enthalten, wird eine Teilmengen-Beziehung der im MSWEM modellierten Softwarekomponenten zu den Statusmanagement-Ressourcen ersichtlich. Demnach bezieht sich im weiteren Verlauf des Kapitels 5.3 der Begriff „Ressource“ auch auf die entsprechende „Softwarekomponente“. Folgende Tabelle zeigt eine dementsprechende Zuordnung der MSWEM-Bereiche zu den MIB-Gruppen auf:

MIB	MIB-Gruppe/n	MSWEM-Bereich
Cluster-MIB	Cluster Info Group, Cluster Applicationgroups, Cluster HW Infrastructure Group	MSCS/APP
RHost-MIB	Erweiterungs-Gruppen	MSCS/APP
VHost-MIB	Alle Gruppen	MSCS/APP
Cluster-MIB	Cluster Disks Group	MLX
Host-MIB	„OS-Layer“ und „PC-Layer“	NT

Tabelle 5.1 : Zuordnung der MSWEM-Bereiche zu MIB-Gruppen

Bemerkung 5.1: Die Host-MIB sollte noch um die proprietären NT-MIBs erweitert werden, da sonst Probleme bei der Deduktion der NT-Dienst-Abhängigkeiten entstehen könnten.

Jeder Ressource wird im Statusmanagement eine Menge sogenannter „nativer Zustände“, die vom Hersteller bestimmt werden, zugewiesen. Demnach können diese Zustände in gewissem Sinne als „brauchbarer“ Meldungstext eines Events betrachtet werden.

Anschließend wird vom Statusmanagement eine „Generische Ressourcen-Zustandsdefinition“ eingeführt, die u.a. eine Einteilung in „Internen Zuständen“ und „Externen Zuständen“ gibt und in etwa einer MSWEM-Eventinterpretation entspricht. Da die „Externen Zustände“ einer gegebenen Ressource eine zusätzliche Sicht der abhängigen Ressource auf die gegebene darstellen, und weil ein derartiger Zusammenhang durch den EQG im MSWEM modelliert ist, werden die „Externen Zuständen“ für eine Übernahme nicht weiter betrachtet.

Die „Inneren Arbeitsphasen“ als Teil der „Internen Zustände“ werden im Statusmanagement und demzufolge auch bezüglich einer Übernahme nicht betrachtet. Also kommen zunächst nur noch die restlichen „Internen Zustände“ für eine Übernahme in Frage.

Umsetzung

Die o.g. restlichen „Internen Zustände“ werden im Statusmanagement in Gruppen von Fehler- und Warnungszustände eingeteilt, die jeweils eine weitere Gliederung beinhalten. Dabei besitzt jede Gruppe einen Zustand „not-applicable“, der sich eigentlich auf einen „Externen Zustand“ bezieht und demzufolge hier nicht weiter betrachtet wird (siehe Übernahme). Die übrigen Zustände und ihre Erklärung werden in der Tabelle 5.x2 angegeben:

Gruppe	Zustand	Bedeutung
Fehlerzustand	**ok**	Die Ressource ist fehlerfrei
Fehlerzustand	**error-recovery**	Die Ressource führt eine Fehlerkorrektur durch; Fehler hat nach außen keine Wirkung, da Fehlertolerante Ressource

Fehlerzustand	**non-recoverable-error**	Ausfall
Warnungszustand	**no-redundancy-warning**	Der nächste Fehler könnte einen Ausfall verursachen (nur bei fehlertoleranten Ressourcen)
Warnungszustand	**non-recoverable-error-warning**	Die Ressource wird demnächst ausfallen

Tabelle 5.2: Übernommene Zustände aus dem Statusmanagement

Aus der Tabelle 5.2 ist eine Einordnung der Statusmanagement-Zustände in eine MSWEM-Eventinterpretation bezüglich Verfügbarkeit einfach durchzuführen. Diese Einordung wird in folgender Tabelle beschrieben:

Statusmanagement-Zustand	**MSWEM-Eventinterpretation**
ok	SK_V
error-recovery	SK_DUV, wegen potentiellem Problem bei der Fehlerkorrektur
non-recoverable-error	SK_NV
no-redundancy-warning	SK_DNV
non-recoverable-error-warning	SK_DNV

Tabelle 5.3: Übernommene Zustände aus dem Statusmanagement

Eine weitere Betrachtung der Statusmanagement-Zustände findet hier aus folgenden Gründen nicht statt:

1. Als nächste Schritte werden im Statusmanagement die Einführung von Abbildungen und Abhängigkeitsbetrachtungen durchgeführt. Die Abbildungen entsprechen entweder einer Systematisierung nach der Eventinterpretation, da sie „nativ“ → „generisch“ durchführen, oder sie beinhalten Abhängigkeitsbetrachtungen. Diese werden hier generell für eine Übernahme bzw. Umsetzung nicht berücksichtigt, da sie im MSWEM vom EQG hinreichend modelliert sind.
2. Weitere Betrachtungen im Statusmanagement („Enthaltenseins-Beziehungen“, „Zusammenfassung der Komponenten-Zustands-Bewertungen“) basieren entweder auf Abhängigkeiten (siehe 1.) oder auf eine unterschiedliche Priorisierung im Vergleich zum MSWEM und sind damit „inkompatibel“.

5.3.2 Management-Aspekte der Integration

Die o.a. Zustandsinformationen, die der MSWEM vom Statusmanagement übernimmt (siehe Tabelle 5.x3), sind zur Laufzeit je nach Ressource auf einen Agenten der entsprechenden MIB, im folgenden Statusmanagement-Agenten genannt, verfügbar. Es stellt sich nun die Frage welche Rolle der MSWEM in einem SNMP-basierenden Management spielt und wie und wann er mit den entsprechenden Statusmanagement-Agenten kommuniziert.

Manager-Agenten-Rolle

Die Rolle des MSWEM im Rahmen der SNMP-basierenden Kommunikation mit dem Statusmanagement ist einfach festzustellen, da er in diesem Kontext Zustandsinformationen von den Statusmanagement-Agenten benötigt. Also handelt es sich bei einer derartigen Kommunikation um eine eindeutige Manager-Rolle des MSWEM.

Der MSWEM sollte durch eine Cluster-Gruppe des Typs „Generic Service“ realisiert werden, damit er einerseits Failover-fähig ist und andererseits als NT-Dienst über eine zusätzliche Kontrolle durch den Service Controll Manager verfügt. Allerdings impliziert diese Realisierung wiederum eine Agenten-Rolle des MSWEM bezüglich des Statusmanagements, da der Statusmanagement-Agent die Cluster-Gruppen (über „Cluster Applicationgroups“ der Cluster-MIB) und alle NT-Dienste (über „OS-Layer“ der Host-MIB) bezüglich ihrer Zustände direkt oder indirekt abfragt.

Interaktion

Es bleibt noch zu klären welche Interaktions-Variante gewählt wird.

Dabei sollen die Senkung der Netzlast und die Leistungssteigerung Ziele dieser Wahl sein. Eine SNMPv2-Unterstützung wird vorausgesetzt.

Dazu wird festgestellt, daß einerseits der MSWEM bei jeder Ausführung der Korrelations-Operation „Fehlerlokalisierung im EKG“ eine „Umgebungs-Abfrage“ benötigt, um den Momentanen Zustand der Graphknoten in Erfahrung zu bringen (siehe „IsAlive“-Operation), und andererseits, daß die generischen Statusmanagement-Zustände einen zusammenfassenden Charakter der „nativen Zustände“ haben. Da in diesem Kontext eine „Umgebungs-Abfrage“ seitens MSWEM häufiger vorkommen kann als eine Änderung eines generischen Statusmanagement-Zustandes, erscheint folgende Überlegung - im Hinblick auf die Senkung der Netzlast - durchaus sinnvoll:

> Bei der SNMP-basierenden Kommunikation des MSWEM mit einem Statusmanagement-Agenten ist eine Trap-Interaktion für jede Änderung eines generischen Statusmanagement-Zustandes seitens Statusmanagement-Agent zu wählen. Dies trägt zur Senkung der Netzlast und zur Leistungssteigerung des MSWEM bei.

Im Hinblick auf die Leistungssteigerung wäre es sinnvoll dem MSWEM eine eigene Umgebungs-Datenbank UDB beizufügen, in der für jede Softwarekomponente ein Datensatz mit dem Momentanen Zustand des Graphknotens und der letzten Zustands-Änderung gespeichert wird. Dabei könnten die von den Statusmanagement-Agenten gesendeten `snmpV2-traps` beim Empfang vom MSWEM folgendermaßen umgesetzt werden:

- Lokalisiere über MIB-Gruppe im sendenden Agenten die Ressource, übersetze sie in Softwarekomponente und gehe zum entsprechendem UDB-Datensatz,
- Finde über den empfangenen `snmpTrapOID.0` die Zustandsänderung des generischen Zustandes heraus und übersetze den neuen Zustand in eine Eventinterpretation gemäß Tabelle 5.x3. Aktualisiere dementsprechend den Momentanen Zustand des Graphknotens in der UDB,
- Aktualisiere aufgrund des empfangenen `sysUpTime.0` die entsprechende letzte Zustands-Änderung in der UDB.

Damit ist bei einer „Fehlerlokalisierung im EKG“ statt einer Management-Operation („IsAlive“) nur noch ein Zugriff auf eine integrierte Datenbank für den MSWEM notwendig, was seine Leistung beträchtlich steigern kann.

Bemerkungen: Ein mögliches Problem dieser Modellierung wäre, daß während eines UDB-Zugriffs durch den MSWEM der Zustand der Datenbank nicht aktualisiert werden könnte, falls zwischenzeitlich ein Trap vom Statusmanagement-Agenten eintrifft. Damit hätte der MSWEM mit einem veralteten Zustand korreliert. Da aber die UDB-Struktur besonders einfach ist (nur eine Tabelle mit drei Feldern) kann die Zeit für einen UDB-Zugriff vernachlässigt werden.

Allerdings ist das Problem einer UDB-Inkonsistenz durch lange Transportzeiten oder durch Verlust eines Traps deutlich schwieriger. Dies könnte durch eine Retrieval-Interaktion (`get, get-next, get-bulk`) des MSWEM mit dem Statusmanagement-Agenten oder direkt mit einem MO behoben werden - was aber die Netzlast erhöhen würde.

6. Abschließende Bemerkungen

Das vorliegende Kapitel stellt eine Synthese der vorliegenden Diplomarbeit dar. In diesem Sinne wird eine rückwirkende Analyse der vorgenommenen Bearbeitung, der erlangten Erkenntnisse und der aufgetretenen Probleme unterbreitet. Ferner werden mögliche weiterführende Betrachtungen im Ausblick angegeben.

6.1 Zusammenfassung und Erkenntnisse

Die Überlegungen in der Einleitung gaben zunächst eine Zielsetzung der Konzeption eines Eventmanagements für ein Cluster vor. Daraufhin wurde eine strukturierte Vorgehensweise zur Erreichung der Ziele konzipiert, die dann in der Tat umgesetzt wurde:

Es wurde eine Einteilung des Clusters nach Softwarekomponenten als funktionale Einheiten, die in einer komplexen Abhängigkeitsstruktur angeordnet sind und die als Eventquellen fungieren können, durchgeführt. Die Softwarekomponenten wurden aufgrund ihrer Hardwareabhängigkeit in drei Klassen partitioniert: RAID-, Betriebssystem-, Cluster-Software. Aufgrund der exemplarischen Betrachtung eines Microsoft Cluster Servers kombiniert mit einem Mylex RAID-Controller fand eine Konkretisierung der Softwarekomponenten-Klassen statt. Im Hinblick auf spätere Eventinformationen wurde eine Beschreibung der Cluster-Architektur für jede dieser Klassen durchgeführt.

Eine umfassende Analyse der Events im Cluster, abhängig von der Softwarekomponenten-Klasse, ergibt eine Systematisierung dieser nach Berücksichtigung, Quelle, Interpretation und Verarbeitung. Dabei wird festgestellt, daß eine Operation als Eventquelle fungieren kann und daß die Verarbeitung mit der Interpretation eng verbunden ist. Das Ergebnis der Systematisierung ist in Eventkatalogen gefaßt worden, wobei sich der NT-Eventkatalog wegen der hohen Eventanzahl auf eine sinnvoll gewählte Teilmenge der Events beschränkt. Dennoch bleibt die Anzahl der Ereignisse so groß, daß eine Eventmanagement erforderlich ist.

Aufgrund der Cluster-Architektur und der Event-Systematisierung ergeben sich einerseits spezielle Ziele des MSWEM, wie z.B. Berücksichtigung von Failover und Komponenten-Dynamik, die über die Ziele eines allgemeinen Eventmanagement hinausgehen. Andererseits werden allgemeine Ziele, beispielsweise die Minimierung der Eventanzahl, konkretisiert.
Es folgt eine erste Reduktion der Kombinationsmöglichkeiten der Events, die durch sinnvolle Annahmen ermöglicht wird. Weitere Realisierungen der Ziele sind nur noch durch Korrelation erreichbar.
Demzufolge werden existierende Korrelationsansätze untersucht. Als Ergebnis wird eine hybride Korrelationsstrategie zur Korrelation entwickelt.
Die Informationsstruktur der Korrelationsstrategie erfordert einen kausalen Eventgraphen -EG-, einen Abhängigkeitsgraphen der Eventquellen -EQG- und einen aus den ersten beiden resultierenden, hierarchischen Eventklassengraphen -EKG.
Der Verarbeitungsmechanismus der Korrelationsstrategie enthält nicht nur gewöhnliche Korrelationsoperationen sondern stellt auch zwei spezielle Operationen, nämlich „Fehlerlokalisierung im EKG“ und „Korrelation im EG“ auf.

Als Vergleich mit den momentan verbreitetsten Korrelatoren werden Ansätze zur regelbasierten Formalisierung des MSWEM in Prolog aufgezeigt.
Im Hinblick auf Erweiterbarkeit des MSWEM werden Anleitungen zur Einbindung weiterer Treiber und Anwendungen aufgestellt.
Für eine SNMP basierende Realisierung wird die Integration mit dem vorhandenen, MIB-basierten Statusmanagement untersucht.

Fazit: Es konnte eine geeignete Korrelationsstrategie gefunden werden, die die Events im Cluster hinreichend verdichtet.

6.2 Bemerkungen zur Informationsbeschaffung

In diesem Kapitel wird eine Diskussion der bei der Erstellung der Diplomarbeit aufgetretenen Probleme vorgestellt. Diese Probleme sind im Wesentlichen auf die Informationsbeschaffung bezüglich der Bedeutung von Events zurückzuführen.

Wie schon im Kapitel 3 aufgezeigt wurde, ist für einige Events keine ausreichende Information, die für ein Eventmanagement notwendig wäre, erhältlich. Diese Tatsache kann, bis auf einige Ausnahmen die Microsoft betreffen, nicht auf fehlende Dokumentation beruhen, da alle Mittel - bis hin zu kommerziellen Supportanfragen an den Herstellern - ohne Erfolg erschöpft wurden.

Um die Ursache dieser Probleme zu untersuchen, werden folgende Überlegungen angestellt:
- Eine ausreichende Event-Information im Hinblick auf die Erfassung von einem Eventmanagement setzt eine Ursachenbeschreibung voraus, die eine Berücksichtigung der Dynamik enthält. Z.B. wären Zustandsübergangsbeschreibungen für Ursachen eines Management-Events oder Interaktionsdiagramme für Ursachen von Protokollverletzungs-Events notwendig.
- Die verfügbaren Produktdokumentationen stellen aber im Wesentlichen nur statische Auflistungen von Schnittstellen-Operationen zur Verfügung. Z.B. wurden die Mylex-Events einer Header-Datei, die mit dem Controller-Treiber geliefert wird, entnommen.

Aufgrund dieser beiden Überlegungen wird die Ursache der Probleme bei der Informationsbeschaffung ersichtlich.

6.3 Ausblick

An dieser Stelle werden abschließend folgende weiterführende Aspekte des MSWEM angegeben:

- Die beabsichtigte Ausrichtung des MSCS auf Load-Balancing könnte ein zusätzliches spezielles Ziel hinsichtlich einer entsprechenden Unterstützung für den MSWEM darstellen. Dabei wäre zur Realisierung dieses Zieles eine Weiterentwicklung des MSWEM primär durch eine Verfeinerung der Priorisierungs-Operationen im Verarbeitungsmechanismus vorstellbar.

- Um eine leistungsfähigere Fehlerdiagnose beizusteuern, wäre die Integration einer Case Based Korrelationstechnik in die Korrelationsstrategie zu untersuchen. Dabei wäre ein erster Schritt die Einführung einer sogenannten „Cluster-Historie“ als Grundlage des Erfahrungsbezuges.

- Im Hinblick auf eine steigende Leistungsfähigkeit der Hardware und insbesondere des Intra-Cluster-Netzes könnten immer mehr Komponenten in die Cluster-Architektur aufgenommen werden, was zu einer Erhöhung der Unübersichtlichkeit in der Organisation führen kann. Demnach wäre ein Ausbau des MSWEM zu einem Trouble Ticket System (TTS) eine mögliche Lösung. Es wird bemerkt, daß bei einem derartiger Ausbau der MSWEM durch seine Konzeption die Filter- und Diagnose-Module, sowie die Wissensdatenbank in wesentlichen Zügen zur Verfügung stellt.

- Durch die Erkenntnis in Kapitel 6.2 versteht sich diese Diplomarbeit auch als Ansporn für / Aufruf an Softwarehersteller ihre Produktdokumentationen um den dynamischen Charakter zu erweitern. Dies würde durch ein einfacheres tiefgründigeres Verständnis der Fehlerursachen u.a. die Anwendung einer hybriden probabilistisch-temporalen Korrelationsart (L = P x T) ermöglichen und damit zu einer erheblichen Leistungssteigerung eines Eventmanagements beitragen.

- Die prototypische Realisierung mit einem der verbreiteten Regelbasierten Systeme, wie z.B. Transview Event Center (EC), wäre unter Zuhilfenahme des in Kapitel 5.1 aufgezeigten Ansatzes von Interesse.

Literaturverzeichnis

Bemerkung: Ein '*' vor einer Literaturangabe bedeutet, daß die Quelle nicht öffentlich zugänglich ist.

[BAU97] U. Baumgarten; *Betriebssysteme*, Vorlesung TU München, WS 1996/97

[BRA86] I. Bratko, *Prolog Programmierung für künstliche Intelligenz*, Addison Wesley Verlag, 1986

[BUC96] H. Buckel, *Windows NT,* Franzis-Verlag, 1996

[DMTF96] DMTF; Common Information Model (CIM), Dezember 1996

[GÖR93] G. Görtz; *Einführung in die künstliche Intelligenz*, Addison Wesley Verlag, 1993

[GRO93] J. Grove, *Windows NT - Antworten*, MS Press Deutschland, 1993

[HEGRN] H.-G. Hegering; *Rechnernetze I*, Vorlesung TU München, WS 1996/97

[HEGNSM] H.-G. Hegering; *Netz- und Systemmanagement*, Vorlesung TU München, SS 1997

[HEGAB93] H.-G. Hegering, S.Abeck; *Integriertes Netz- und Systemmanagement*, Addison Wesley Verlag, 1993

[JAK95] G. Jakobson, M. Weissman; *Real-time telecommunication network management: extending event management with temporal constraints*, ISINM 1995

[JORD93] J.F. Jordaan, M.E. Paterok; *Event Correlation in Heterogeneous Networks Using the OSI Management Framework*, IFIP INM III, 1993

[KÄTPAT] S. Kätker, M. Paterok; *Fault Isolation and Event Correlation for Integrated Fault Management*, IBM European Networking Center, Heidelberg

[KLI95] S. Kliger, S. Yemini, Y. Yemini, D. Oshie, S. Stolfo; *A coding approach to event correlation*, Proc. 4th IFIP / IEEE, Santa Barbara, USA, 1995

[KUP96] M. Kuppinger, *MS Windows NT im Netzwerk, Version 4,* Microsoft Press Deutschland, 1996

[KUR97] J. Kuri; *Der mit dem Wolf tanzt ...,* c't Heft 7, 1997

[LEW93] L. Lewis; *A Case-Based Reasoning Approach to the Resolution of Faults in Communication Networks*, Proc. ISINM, 1993,

[LOU95] S. Louis, R. D. Burris; *Management Issues for High Performance Storage Systems*, Juli 1995
http://www.computer.org/conferen/mss95/louis/louis.htm

[MSDN97] *Microsoft *Developer Network* - Library, CD, April 1997

[TECHNET97] * Microsoft *TechNet* - CD, August 1997

[SDK97] * Microsoft *Software Development Kit* - CD, April 1997

[DDK97] * Microsoft *Driver Development Kit* - CD, April 1997

[RESKIT97] * Microsoft *Resource Kit - CD,* Juli 1997

[TECHREF97] Microsoft *Technical Reference* - Handbücher, 1997

[MSCLMSG] *Microsoft Cluster Server ; *clusmsg.h* Header-Datei,

[MSCLERR] *Microsoft Cluster Server ; *cluserr.h* Header-Datei,

[MSCLAPI] *Microsoft Cluster Server ; *clusapi.h* Header-Datei,

[MIN96] M. Minasi, C. Anderson, E. Cregon*; Mastering Windows NT Server 4.0*, Sybex Inc. 1996, 2.Auflage

[MLXGAM] *Mylex ; *gamapi.h* Header File

[MLX92] Mylex, *DAC 960 SX Family User Guide*, 1992
ftp://206.216.93.9/pubs/dac960/
ftp://206.216.93.9/pubs/scsi2scs/

[MLX96] *Mylex; *GAM Server API Document*, 1996

[NAY97] J, Naydek*; Konzeption eines Statusmanagements für ein PC-Cluster*, Diplomarbeit, LMU München - MNM

[NDO93] T.D. Ndousse, Distributed Fuzzy Agents: *A Framework for Intelligent Network Monitoring*,

[PFI95] G. F. Pfister; *In Search Of Clusters*, Prentice Hall, 1995

[SIM95] R. J. Simon, *Windows 95, Windows NT API bible, Band 1 und 2*, Markt & Technik Buch- und Software-Verlag (SAMS), 1996

[ROS96] M. T. Rose; *The Simple Book*, Prentice Hall, 1996

[SCHÜ] D. Schürth,W. Nejdl, R. Hager; *Fault Management of Infrastructure Networks*, TU Aachen

[SCH97] A. Schweikard; *Wissensbasierte Systeme*, Vorlesung TU München, WS 1996/97

[SNICC96] *SNI; *TransView Control Center /-Agent*, Benutzerhandbuch , März 1996

[SNIMIB97] *SNI Augsburg; *NTCluster.MIB*, November 1997

[STA94] R. Standke; *Windows NT und Windows NT Advanced Server*, IWT-Verlag, 1994

[TAN 92] A. S. Tanenbaum, *Computer-Netzwerke* , Wolfram's Verlag, 1992

[TAN 96] A. S. Tanenbaum, *Moderne Betriebssysteme* , Prentice Hall, 1996

[TIV97] Tivoli; *TME 10 Enterprise Console*, 1997

[WRI97] J. Wright; *Clustering Technologies for Windows NT Servers* ; Datapro, 1997
http://www.datapro.com/

[WYA93] A. L. Wyatt, *Windows NT,*Osborne McGraw-Hill, 1993

Abbildungsverzeichnis

ABBILDUNG 1.1: VORGEHENSWEISE 6
ABBILDUNG 2.1: MÖGLICHE HARDWARE-ARCHITEKTUR EINES PC-CLUSTERS 8
ABBILDUNG 2.2: NT-NETZ-ARCHITEKTUR 11
ABBILDUNG 2.3: REGISTRY-STRUKTUR 12
ABBILDUNG 2.4: WORSTATION-DIENST 13
ABBILDUNG 2.5: SERVER-DIENST 14
ABBILDUNG 2.6: DIENST-ABHÄNGIGKEITEN 15
ABBILDUNG 2.7: AUTHENTIFIKATION BEI BENUTZER-ANMELDUNGEN 16
ABBILDUNG 2.8: CLUSTER-DIENST-ABHÄNGIGKEITEN 18
ABBILDUNG 3.1: NT-EVENTARTEN 35
ABBILDUNG. 4.1: KAUSALITÄTSGRAPH FÜR DIE RAID-PLATTEN 55
ABBILDUNG 4.2: DARSTELLUNG DER FAILOVER-ABHÄNGIGKEITEN 60
ABBILDUNG 4.3: ABHÄNGIGKEITSGRAPH DER EVENTQUELLEN IM MSCS/APP-BEREICH 61
ABBILDUNG 4.4: PHASE 1 DER EKG-KONSTRUKTION 64
ABBILDUNG 4.5: PHASE 2 DER EKG-KONSTRUKTION 65
ABBILDUNG 4.6: PHASE 3 DER EKG-KONSTRUKTION 66
ABBILDUNG 4.7: PHASE 4 DER EKG-KONSTRUKTION 67
ABBILDUNG 4.8: PHASE 5 DER EKG-KONSTRUKTION 68

Tabellenverzeichnis

TABELLE 2.1: HARDWARE-WAHL 8
TABELLE 2.2: K-UND E-PARTITIONIERUNG 9
TABELLE 2.3: DIENST-GRUPPEN 14
TABELLE 2.4: DIENST-GRUPPEN-ABHÄNGIGKEITEN 15
TABELLE 2.5: DIENST-ABHÄNGIGKEITEN 15
TABELLE 3.1: ERKLÄRUNG DER EVENTTYPEN 30
TABELLE 3.2: REGISTRY-PARAMETER DER EVENTQUELLEN 31
TABELLE 3.2: ÜBERSICHT DER CLUSTER-ERRORS 43
TABELLE 3.3: ZUSAMMENFASSUNG DER SYSTEMATISIERUNG 43
TABELLE 4.1: VERGLEICH DER KORRELATIONSTECHNIKEN 50
TABELLE 4.2: KORRELATIONSANFORDERUNGEN DES MSWEM 52
TABELLE 4.3 : MSWEM - BEREICHE 57
TABELLE 4.4: EVENTLKASSEN 62
TABELLE 5.1 : ZUORDNUNG DER MSWEM-BEREICHE ZU MIB-GRUPPEN 79
TABELLE 5.2: ÜBERNOMMENE ZUSTÄNDE AUS DEM STATUSMANAGEMENT 80
TABELLE 5.3: ÜBERNOMMENE ZUSTÄNDE AUS DEM STATUSMANAGEMENT 80
TABELLE A.1.1: VOM DAC960SX UNTERSTÜTZTE RAID-LEVELS 87
TABELLE : ÜBERSICHT DER MYLEX - EVENTS 91
TABELLE : ÜBERSICHT DER NT-NETZ-EVENTS 99

Anhang A: Wichtige Details zur Beschreibung des Clusters

A.1 Details zum Mylex SCSI-RAID

Hintergrundwissen für die Event-Beschreibung

Mit dem Einsatz eines Mylex SCSI-RAID-Systems hat man sich für einen **D**isk **A**rray **C**ontroller 960SX entschieden, der verschiedene SCSI-Formate und RAID-Speicherungsarten (RAID-Levels) unterstützen kann. Da die unterschiedlichen SCSI-Formate eventtechnisch nicht von Belang sind, wird ein kurzer Überblick der verschiedenen RAID-Levels angegeben. Dazu ein paar

Vorbemerkungen:

- In einem Mylex-RAID können mehrere (höchstens 8) Platten zu einer **Gruppe** zusammengefaßt werden. Dabei können höchstens 8 Gruppen gebildet werden,
- Der Einfachheit halber wird die Größe X der Platten in einer Gruppe als konstant angenommen,
- (**MLXVB3**) Die **maximale Nutzung** eines RAID-Levels bezieht sich auf den maximalen Auslastungsgrad A einer Gruppe, der das Verhältnis der Gesamtgröße der nichtredundanten Platten zu der Gesamtgröße aller Platten in einer Gruppe darstellt.

Nun sind wir in der Lage folgende Mylex RAID-Level Tabelle anzugeben:

RAID-Levels	Erklärung	Max. Nutzung	Ausfall-sicherheit
JBOD (Mylex: 7)	Bedeutet „Just a Bunch Of Disks“ und beinhaltet keine spezielle RAID-Verwaltung, d.h. der Controller hat keine Anwendung.	100%	Nein
0	Ein sequentieller Datenblock wird auf alle Platten einer Gruppe homogen verteilt. Diese Technik wird **Striping** genannt. Dabei definiert man die **Stripe-Größe** als die Datenlänge auf einer Platte.	100%	Nein
1	Zu jeder Platte gibt es eine Platte gleichen Inhalts. Diese Technik wird **Mirroring** genannt.	50%	Ja
0+1 (Mylex: 6)	Stellt eine Kombination von Striping und Mirroring dar.	50%	Ja
3	Kombination von Striping und redundanter Block-Parität - **Parity** genannt. Dabei wird die Parity-Information auf einer Platte gespeichert.	100%-X	Ja
5	Analog zu RAID-Level 3 und zusätzlich wird auch die Parity-Information homogen auf alle Platten einer Gruppe verteilt.	100%-X	Ja

Tabelle A.1.1: Vom DAC960SX unterstützte RAID-Levels

Aus der vorangehenden Tabelle können folgende **Erkenntnisse** ableitetet werden:

- Mylex RAID-Level 5 hat im Vergleich zu den RAID-Levels 1, 3 und 6 eine höhere maximale Nutzung und unterstützt im Gegensatz zu den RAID-Levels 0 und 7 Ausfallsicherheit,
- Hohe Werte für die Stripe-Größe eignen sich für großen, sequentiellen Datentransfer und kleine Werte eignen sich eher für kleinen, nicht sequentiellen Datentransfer. D.h. für gegebenen Clusteranwendungsprofil gibt es eine optimale Stripe-Größe.

Die gewonnenen Erkenntnisse führen nun zur folgenden Annahme:

- (**MLXA1**): Der RAID-Level wird auf 5 festgesetzt wobei die Stripe-Größe - einmal beliebig gewählt - konstant bleibt.

Im folgenden werden wir einige Charakteristika des DAC960SX beschreiben, die den Annahmen (**A1**) und (**MXLA1**) genügen. Dabei werden Ablaufzusammenhänge geschildert und spezifische Begriffe, die mit fetten Buchstaben gekennzeichnet werden, eingeführt.

DAC960SX-Charakteristika:

- (**MLX1**) Der Controller kann 5 SCSI-Kanäle a 6 physikalische Platten (physical drives - **PHYSDEV**) unterstützen, d.h. insgesamt 30 Platten. Dabei kann man einige PHYSDEVs zu einer logischen Platte (**L**ogical **UN**it, system drive - **SYSDEV**) so zusammenfassen, daß man höchstens 8 SYSDEVs erzeugt;

- (**MLX2**) Ein stromversorgtes, gruppenzugehörendes und richtig funktionierendes PHYSDEV befindet sich im Zustand **ONLINE**;
- (**MLX3**) Automatische Erkennung von PHYSDEV-Fehlern durch Kommando-Timeouts;
- (**MLX4**) Beim Erkennen eines beliebigen Plattenfehlers (**MISC_ERROR**) oder eines Paritätsfehlers (**PARITY_ERROR**) wird das PHYSDEV automatisch zurückgesetzt (**RESET**) und es findet eine Befehlswiederholung statt;
- (**MLX5**) Wenn dies wiederholt scheitert oder wenn ein achtmaliger **SOFT_ERROR** oder ein einmaliger **HARD_ERROR** stattfindet, wird das PHYSDEV in den **DEAD**-Zustand gesetzt. Außerdem kann der DEAD-Zustand durch Abwesenheit oder durch nicht vorhandene Stromversorgung eines PHYSDEVs erreicht werden;
- (**MLX6**) Falls das RAID über mindestens ein stromversorgtes, keiner Gruppe gehörendes PHYSDEV (**STANDBY / HOTSPARE** genannt) verfügt, findet ein automatischer, benutzertransparenter Wiederaufbau (**REBUILD**) dieses PHYSDEVs (und implizit der zugeordneten SYSDEVs) statt. Dabei spricht man von einem Übergang des PHYSDEV-Zustands von STANDBY nach REBUILD. Falls das RAID über kein STANDBY (mehr) verfügt, muß man das ausgefallene PHYSDEV im laufenden RAID-Betrieb austauschen (**Hot Swap**) und einen manuellen REBUILD anstoßen;
- (**MLX7**) Währendessen befinden sich die dem ausgefallenen PHYSDEV zugeordneten SYSDEVs entweder im **CRITICAL**-Zustand (falls keine weitere zugeordnete PHYSDEVs ausgefallen sind) oder im **OFFLINE**-Zustand (sonst);
- (**MLX8**) Ein SYSDEV, dessen zugeordneten PHYSDEVs ausnahmslos ONLINE sind, wird als **ONLINE** bezeichnet;
- (**MLX9**) Der Controller verfügt über einen eingebauten Mechanismus zur Vorhersage von PHYSDEV-Fehlern, der sich „predictive failure analysis“ (**PFA**) nennt;
- (**MLX10**) Die „EXPANDCAPACITY“-Events beziehen sich höchstwahrscheinlich* auf eine Kompressionsoperation auf ein PHYSDEV (und implizit auf die zugeordneten SYSDEVs);
- (**MLX11**) Die „COMMAND“-Events beziehen sich auf PHYSDEV-Kommandos und wirken sich auf {MLX3, MLX4, MLX5} aus;
- (**MLX12**) Der „REQSENSE“-Event zeigt an, daß der Controller zusätzliche PHYSDEV-Information über die sogenannten „sense bytes“ abfragen muß;
- (**MLX13**) Die „CHECK“-Events beziehen sich auf eine Konsistenzüberprüfung der redundanten Parity-Information eines SYSDEVs und wirken sich direkt auf MLX4 aus. Dabei beziehen sich in diesem Kontext die „...SYSDEV_FAILED“- und „...PHYSDEV_FAILED“-Events höchstwahrscheinlich* auf das Ergebnis des CHECKs bzgl. SYSDEVs und PHYSDEVs;
- (**MLX14**) Die „INIT“-Events beziehen sich auf die Initialisierung eines SYSDEVs, die höchstwahrscheinlich* nach dem „FOUND“-Event eines SYSDEVs dessen Zuordnungen zu den PHYSDEVs setzt;
- (**MLX15**) Die SCSI-Kanäle zwischen Controller und Hosts heißen Host-Kanäle und die SCSI-Kanäle zwischen Controller und Platten heißen Platten-Kanäle
- (**MLX16**) Wenn der Controller ersetzt wird (ist im laufenden RAID-Betrieb möglich und wird **HOT PLUG** genannt), wird der neue Controller automatisch gleich konfiguriert. Ein Eingreifen ist nur beim Ändern der RAID-Größe (Gesamtgröße des zur Verfügung stehenden Speichers im RAID) erforderlich;
- (**MLX17**) Der Controller kann dual/parallel zu einem zweiten, identischen Controller betrieben werden (**dual-active-configuration**), wenn beide gleiche Anzahl an Host- und Platten-Kanäle (wobei die letzteren zusätzlich noch 1 zu 1 verbunden sein müssen) und gleiche Anzahl an PHYSDEVs haben. In dieser Konfiguration kann jeder Controller ein Fehler seines dualen Controllers erkennen und automatisch dessen Funktionen übernehmen (**fail-over**)- dies steigert natürlich die Redundanz des Speichersystems erheblich. Wenn der fehlerhafte Controller ersetzt/repariert wurde, wird ihm vom „Überlebenden“ automatisch die Kontrolle über die eigenen Funktionen zurück übertragen (**fail-back**);
- (**MLX18**) Jeder Controller verfügt über einen eigenen **Cache** (verschiedene Arten), der im Fall einer dual-active-configuration nicht zur Leistungssteigerung sondern auch zur (intercontroller) redundanten Speicherung von Schreib-Daten dient;
- (**MLX19**) Der „STATE_CHG_TABLE_FULL“-Event zeigt an, daß die im Cache gehaltene Tabelle

zur Speicherung der Statusunterschiede in den PHYSDEVs voll ist;

- (**MLX20**) Die Lüfter- (**FAN**), Stromversorgungs- (**FMTPOWER**) und Temperatur- (**FMTHEAT**) Events beziehen sich auf physikalische Einflußgrößen, die sensortechnisch gemessen werden, Schwellwertüberwachungen unterliegen und für den RAID-Betrieb unabdingbar sind.

Mit dem DAC960SX wird von Mylex ein RAID-Management-Tool, der **G**lobal **A**rray **M**anager (kurz: GAM), geliefert. GAM kann in Managementplattformen wie DMI/LANDesk oder HP OpenView integriert werden und verfügt über eine graphische Benutzeroberfläche (GUI), die u.a. auch für Windows NT implementiert wurde. GAM bietet dem RAID-Benutzer eine sehr einfache, generische Schnittstelle an - die IOCTL. Diese ist auch zu Windows NT kompatibel.

Informations-Struktur „dga_event_info_t“

```
/* GAM Event Record Format definition      - begin*/

typedef struct dga_event_info
{       u32bits  ei_ErrorCode;              /* Non zero if data is not valid */
        u32bits  ei_EventSeqNo;             /* IO: Event Sequence Number */
        u32bits  ei_EventCode;              /* Event Code value */
        u32bits  ei_EventTime;              /* Time Stamp for this event */
        u08bits  ei_ControllerNo;           /* Controller Number of the event */
        u08bits  ei_ChannelNo;              /* Channel number of the event */
        u08bits  ei_TargetID;               /* Target ID of event */
        u08bits  ei_LunID;                  /* Logical unit/Logical Drive of event*/
        u08bits  ei_FamilyControllerNo;     /* Controller number in Family */
        u08bits  ei_ControllerType;         /* Controller type eg. DAC960PD */
        u08bits  ei_DevType;                /* Device Type (SCSI, Logical, etc.) */
        u08bits  ei_OSType;                 /* Operating System Type */
        u32bits  ei_Reserved1;
        u32bits  ei_Reserved2;
} dga_event_info_t;

/* GAM Event Record Format definition      - end*/

#define  dga_event_info_s       sizeof(dga_event_info_t)
#define  ei_SysDevNo            ei_LunID
#define  ei_RaidLevel           ei_DevType
#define  ei_CabinetNo           ei_TargetID
#define  ei_FanNo               ei_LunID
#define  ei_PowerUnitNo         ei_LunID
#define  ei_HeatSensorNoei_LunID
```

Anhang B:Details zur Systematisierung der Events

B.1 Eventkatalog der Mylex - Events

ID	Meldungstext	Berücksichtigung	Quelle	Interpretation	Verarbeitung
01	MLXEV_PHYSDEV_ONLINE	J	PL	SK_V	M
02	MLXEV_PHYSDEV_HOTSPARE	J	PL	SK_UV	VSM
03	MLXEV_PHYSDEV_HARD_ERROR	J	PL	SK_DNV	VSM

04	MLXEV_PHYSDEV_PFA	J	PL	SK_DUV	VSM
05	MLXEV_PHYSDEV_AUTO_REBUILD_START	J	PL_Rbld	O_V	M
06	MLXEV_PHYSDEV_MANUAL_REBUILD_START	J	PL_Rbld	O_V	M
07	MLXEV_PHYSDEV_REBUILD_DONE	J	PL_Rbld	O_E	M
08	MLXEV_PHYSDEV_REBUILD_CANCELED	J	PL_Rbld	O_DUV	VSM
09	MLXEV_PHYSDEV_REBUILD_ERROR	J	PL_Rbld	O_M	VSM
10	MLXEV_PHYSDEV_REBUILD_NEWDEV_FAILED	N			
11	MLXEV_PHYSDEV_REBUILD_SYSDEV_FAILED	N			
12	MLXEV_PHYSDEV_DEAD	J	PL	SK_NV	VSM
13	MLXEV_PHYSDEV_FOUND	J	PL	SK_ADD	VSM
14	MLXEV_PHYSDEV_GONE	J	PL	SK_DEL	VM
15	MLXEV_PHYSDEV_UNCONFIGURED	J	PL	SK_NV	VSM
16	MLXEV_PHYSDEV_EXPANDCAPACITY_START	J	PL_Exc	O_V	M
17	MLXEV_PHYSDEV_EXPANDCAPACITY_DONE	J	PL_Exc	O_E	M
18	MLXEV_PHYSDEV_EXPANDCAPACITY_ERROR	J	PL_Exc	O_M	SM
19	MLXEV_PHYSDEV_COMMAND_TIMEOUT	J	PL	SK_DUV	VSM
20	MLXEV_PHYSDEV_COMMAND_ABORT	J	PL	SK_DUV	VSM
21	MLXEV_PHYSDEV_COMMAND_RETRIED	J	PL	SK_DUV	VSM
22	MLXEV_PHYSDEV_PARITY_ERROR	J	PL	SK_DNV	VSM
23	MLXEV_PHYSDEV_SOFT_ERROR	J	PL	SK_DNV	VSM
24	MLXEV_PHYSDEV_MISC_ERROR	J	PL	SK_DNV	VSM
25	MLXEV_PHYSDEV_RESET	J	PL	SK_DUV	VSM
26	MLXEV_PHYSDEV_ACTIVESPARE	N			
27	MLXEV_PHYSDEV_WARMSPARE	N			
28	MLXEV_PHYSDEV_REQSENSE	J	PL	SK_SZ	SM
29	MLXEV_SYSDEV_CHECK_START	J	LL_Chk	O_V	M
30	MLXEV_SYSDEV_CHECK_DONE	J	LL_Chk	O_E	M
31	MLXEV_SYSDEV_CHECK_CANCELED	J	LL_Chk	O_DUV	VSM
32	MLXEV_SYSDEV_CHECK_ERROR	J	LL_Chk	O_M	VSM
33	MLXEV_SYSDEV_CHECK_SYSDEV_FAILED	N			
34	MLXEV_SYSDEV_CHECK_PHYSDEV_FAILED	N			
35	MLXEV_SYSDEV_OFFLINE	J	LL	SK_NV	VSM
36	MLXEV_SYSDEV_CRITICAL	J	LL	SK_DNV	VSM
37	MLXEV_SYSDEV_ONLINE	J	LL	SK_V	M
38	MLXEV_SYSDEV_AUTO_REBUILD_START	J	LL_Rbld	O_V	M
39	MLXEV_SYSDEV_MANUAL_REBUILD_START	J	LL_Rbld	O_V	M
40	MLXEV_SYSDEV_REBUILD_DONE	J	LL_Rbld	O_E	M
41	MLXEV_SYSDEV_REBUILD_CANCELED	J	LL_Rbld	O_DUV	VSM
42	MLXEV_SYSDEV_REBUILD_ERROR	J	LL_Rbld	O_M	VSM
43	MLXEV_SYSDEV_REBUILD_NEWDEV_FAILED	N			
44	MLXEV_SYSDEV_REBUILD_SYSDEV_FAILED	N			
45	MLXEV_SYSDEV_INIT_STARTED	J	LL_Init	O_V	M
46	MLXEV_SYSDEV_INIT_DONE	J	LL_Init	O_E	M
47	MLXEV_SYSDEV_INIT_CANCELED	J	LL_Init	O_DUV	VSM
48	MLXEV_SYSDEV_INIT_FAILED	J	LL_Init	O_M	VSM
49	MLXEV_SYSDEV_FOUND	J	LL	SK_ADD	VSM
50	MLXEV_SYSDEV_GONE	J	LL	SK_DEL	VM
51	MLXEV_SYSDEV_EXPANDCAPACITY_START	J	LL_ExC	O_V	M
52	MLXEV_SYSDEV_EXPANDCAPACITY_DONE	J	LL_ExC	O_E	M
53	MLXEV_SYSDEV_EXPANDCAPACITY_ERROR	J	LL_ExC	O_M	SM
54	MLXEV_SYSDEV_BADBLOCK	J	LL	SK_DUV	VSM
55	MLXEV_SYSDEV_SIZECHANGED	N			
56	MLXEV_SYSDEV_TYPECHANGED	N			
57	MLXEV_SYSTEM_STARTED	J	R	SK_V	M

58	MLXEV_WRITEBACK_ERROR	J	Buf	O_SZ	SM
59	MLXEV_STATE_CHG_TABLE_FULL	J	Buf	O_DUV	VSM
60	MLXEV_ADAPTER_DEAD	J	Ctrl	SK_NV	VSKA
61	MLXEV_ADAPTER_RESET	J	Ctrl	SK_DUV	VSM
62	MLXEV_ADAPTER_FOUND	J	Ctrl	SK_ADD	VSM
63	MLXEV_ADAPTER_GONE	J	Ctrl	SK_DEL	VM
64	MLXEV_BBU_FOUND	N			
65	MLXEV_BBU_POWER_LOW	N			
66	MLXEV_BBU_POWER_OK	N			
67	MLXEV_FMTFAN_FAILED	J	Fan	SK_NV	VSM
68	MLXEV_FMTFAN_OK	J	Fan	SK_V	M
69	MLXEV_AEMI_FAN_FAILED	J	Fan	SK_NV	VSM
70	MLXEV_FMTFAN_NOTPRESENT	J	Fan	SK_NV	VSM
71	MLXEV_FMTPOWER_FAILED	J	Pwr	SK_NV	VSM
72	MLXEV_FMTPOWER_OK	J	Pwr	SK_V	M
73	MLXEV_AEMI_PWR_SUPPLY_FAILED	J	Pwr	SK_NV	VSM
74	MLXEV_FMTPOWER_NOTPRESENT	J	Pwr	SK_NV	VSM
75	MLXEV_FMTHEAT_BAD	J	Tmp	SK_UV	VSM
76	MLXEV_FMTHEAT_CRITICAL	J	Tmp	SK_DUV	VSM
77	MLXEV_FMTHEAT_OK	J	Tmp	SK_V	M
78	MLXEV_AEMI_OVER_TEMPERATURE	J	Tmp	SK_UV	VSM
79	MLXEV_FMTHEAT_NOTPRESENT	J	Tmp	SK_NV	VSM

Tabelle : Übersicht der Mylex - Events

B.2 Eventkatalog der NT - Events

ID	Meldungstext	Berücksichtigung	Quelle	Interpretation	Verarbeitung
2102	The workstation driver is not installed.	J	Config		
2104	An internal error occurred. The network cannot access a shared memory segment.	J	L_Sys	SK_UV	VSM
2105	A network resource shortage occurred.	J	Net_Res	SK_NV	VSM
2106	This operation is not supported on workstations.	N			
2114	The Server service is not started.	J	L_S_Svc	SK_NV	VSM
2115	The queue is empty.	N			
2116	The device or directory does not exist.	J	Config		
2118	The name has already been shared.	J	Config		
2119	The server is currently out of the requested resource.	J	L_Sys_Res	SK_NV	VSM
2121	Requested addition of items exceeds the maximum allowed.	J	Config		
2123	The API return buffer is too small.	J	Config		
2136	A general network error occurred.	J	Net	SK_NV	VSKA
2137	The Workstation service is in an inconsistent state. Restart the computer before restarting the Workstation service.	J	L_WS_Svc	SK_NV	VSM
2138	The Workstation service has not been started.	J	L_WS_Svc	SK_NV	VSM
2140	An internal Windows NT error occurred.	J	L_Sys	SK_NV	VSM
2141	The server is not configured for transactions.	J	Config		
2142	The requested API is not supported on the remote server.	J	Config		
2144	The computer name already exists on the network. Change it and restart the computer.	J	Config		
2146	The specified component could not be found in the configuration information.	J	Config		

2147	The specified parameter could not be found in the configuration information.	J	Config		
2150	The printer does not exist.	N			
2151	The print job does not exist.	N			
2180	The service database is locked.	J	Spec_Svc	SK_NV	VSM
2182	The requested service has already been started.	J	Config		
2184	The service has not been started.	J	Spec_Svc	SK_NV	VSM
2185	The service name is invalid.	J	Config		
2186	The service is not responding to the control function.	J	Spec_Svc	SK_DUV	VSM
2187	The service control is busy.	J	Spec_Svc	SK_DNV	VSM
2189	The service could not be controlled in its present state.	J	Spec_Svc	SK_DUV	VSM
2191	The requested pause or stop is not valid for this service.	J	Config		
2194	A thread for the new service could not be created.	J	L_Sys_Res	SK_NV	VSM
2202	The user name or group name parameter is invalid.	J	Config		
2203	The password parameter is invalid.	J	Config		
2211	The server is configured without a valid user path.	J	Config		
2219	The security database could not be found.	J	L_SAM	SK_NV	VSM
2220	The group name could not be found.	J	Config		
2221	The user name could not be found.	J	Config		
2222	The resource name could not be found.	J	Config		
2223	The group already exists.	J	Config		
2224	The user account already exists.	J	Config		
2226	This operation is only allowed on the primary domain controller of the domain.	J	Config		
2227	The security database has not been started.	J	L_SAM	SK_NV	VSM
2229	A disk I/O failure occurred.	N			
2233	Unable to add to the user accounts database session cache segment.	J	L_SAM	SK_UV	VSM
2234	This operation is not allowed on this special group.	J	Config		
2236	The user already belongs to this group.	J	Config		
2237	The user does not belong to this group.	J	Config		
2239	This user account has expired.	J	Config		
2240	The user is not allowed to log on from this workstation.	J	Config		
2241	The user is not allowed to log on at this time.	J	Config		
2242	The password of this user has expired.	J	Config		
2245	The password is shorter than required.	J	Config		
2246	The password of this user is too recent to change.	J	Config		
2247	The security database is corrupted.	J	L_SAM	SK_UV	VSM
2249	This replicant database is outdated; synchronization is required.	J	L_SAM	SK_UV	VSM
2250	The network connection could not be found.	J	Config		
2251	This asg_type is invalid.	N			
2270	The computer name could not be added as a message alias. The name may already exist on the network.	N			
2273	This message alias could not be found on the network.	N			
2276	This message alias already exists locally.	J	Config		
2277	The maximum number of added message aliases has been exceeded.	J	Config		
2278	The computer name could not be deleted.	J	Config		
2280	An error occurred in the domain message processor.	J	L_Msg_Svc	SK_NV	VSM
2281	The message was sent, but the recipient has paused the Messenger service.	J	R_Msg_Svc	SK_NV	VSM

2282	The message was sent but not received.	J	R_Msg_Svc / R_WS_Svc	SK_DUV / SK_DUV	SM / SM
2283	The message alias is currently in use. Try again later.	J	Config		
2285	The name is not on the local computer.	J	Config		
2289	The broadcast message was truncated.	J	Config		
2294	This is an invalid device name.	J	Config		
2297	A duplicate message alias exists on the network.	J	Config		
2298	This message alias will be deleted later.	J	Config		
2299	The message alias was not successfully deleted from all networks.	J	Config		
2310	This shared resource does not exist.	J	Config		
2311	This device is not shared.	J	Config		
2312	A session does not exist with that computer name.	J	Config		
2351	This computer name is invalid.	J	Config		
2357	Could not determine the type of input.	N			
2362	The buffer for types is not big enough.	N			
2378	This log file has changed between reads.	N			
2401	There are open files on the connection.	J	Config		
2403	This share name or password is invalid.	J	Config		
2404	The device is being accessed by an active process.	J	Config		
2432	An invalid or nonexistent alert name was raised.	J	L_Alrt_Svc	SK_DNV	VSM
2453	Could not find the domain controller for this domain.	J	PDC_NetLg_Svc	SK_NV	VSM
2457	This server's clock is not synchronized with the domain controller's clock.	J	L_Clock	SK_UV	VSM
2470	Try down-level (remote admin protocol) version of API instead.	N			
2481	The UPS service is not configured correctly.	J	Config		
2482	The UPS service could not access the specified Comm Port.	J	Config		
2483	The UPS indicated a line fail or low battery situation. Service not started.	J	Config		
2484	The UPS service failed to perform a system shutdown.	J	UPS_SDown	O_M	VSKA
2550	The browser service was configured with MaintainServer-List=No.	J	Config		
3020	A power failure was detected at name. The server has been paused.	J	L_Pwr	SK_NV	VSKA
3021	Power has been restored at name. The server is no longer paused.	J	L_Pwr	SK_V	VM
3022	The UPS service is starting shutdown at name due to low battery.	J	UPS_SDown	O_V	VSKA
3023	There is a problem with a configuration of user specified shutdown command file. The UPS service started anyway.	N			
3030	The server cannot export directory name, to client name. It is exported from another server.	N			
3031	The replication server could not update directory name from the source on name due to error code.	J	Spec_Dir	SK_UV	VSM
3032	Master name did not send an update notice for directory name at the expected time.	J	R_S_Svc / R_Repl_Svc	SK_NV / SK_NV	VSM / VSM
3034	The domain controller for domain name failed.	J	PDC_Sys	SK_NV	VSKA
3035	Failed to authenticate with name, a Windows NT Server for domain name.	J	L_NetLg_Svc	SK_UV	VSM
3038	Replicator could not access name on name due to system error code.	J	L_Sys	SK_DUV	VSM

3039	Replicator limit for files in a directory has been exceeded.	J	Config		
3040	Replicator limit for tree depth has been exceeded.	N			
3041	The replicator cannot update directory name. It has tree integrity and is the current directory for some process.	J	Config		
3042	Network error code occurred.	J	L_Repl_Svc	SK_NV	VSM
3045	System error code occurred.	J	L_Repl_Svc	SK_NV	VSM
3047	IMPORT path path cannot be found.	J	Config		
3049	Replicated data has changed in directory name.	J	Spec_Dir	SK_UV	VSM
3051	The Registry or the information you just typed includes an illegal value for text.	J	Config		
3054	A request for resource could not be satisfied.	J	L_Sys_Res	SK_NV	VSM
3056	A system error has occurred.	J	L_Sys	SK_DNV	VSM
3057	An internal consistency error has occurred.	N			
3060	The service did not respond to control and was stopped with the DosKillProc function.	J	Spec_Svc	SK_NV	VSM
3061	An error occurred when attempting to run the service program.	J	Spec_Svc	SK_NV	VSM
3062	The sub-service failed to start.	J	Config		
3064	There is a problem with the file.	J	Spec_Svc	SK_NV	VSM
3095	This Windows NT computer is configured as a member of a workgroup, not as a member of a domain. The Netlogon service does not need to run in this configuration.	J	Config		
3096	The Windows NT primary domain controller for this domain could not be located.	N			
3100	The operation failed because a network software error occurred.	N			
3106	An unexpected network control block (NCB) was received. The NCB is the data.	N			
3112	An illegal server message block (SMB) was received. The SMB is the data.	N			
3113	Initialization failed because the requested service name could not be started.	N			
3140	The service has stopped due to repeated consecutive occurrences of a network control block (NCB) error. The last bad NCB follows in raw data.	N			
3152	An illegal server message block (SMB) was received. The SMB is the data.	N	Identisch	zu	3112
3170	The Alerter service had a problem creating the list of alert recipients. The error code is code.	N			
3172	There was an error sending name the alert message - text. The error code is code.	J	Config		
3173	There was an error in creating or reading the alerter mailslot. The error code is code.	J	L_Alrt_Svc	SK_DNV	VSM
3206	The replicator cannot update directory name. It has tree integrity and is the current directory for some process.	N	Identisch	zu	3041
3207	The server cannot export directory name to client name. It is exported from another server.	J	Config		
3208	The replication server could not update directory name from the source on name due to error code.	N	Identisch	zu	3031
3209	Master name did not send an update notice for directory name at the expected time.	N	Identisch	zu	3032
3210	Failed to authenticate with name, a Windows NT Server for domain name.	N	Identisch	zu	3035
3212	Network error code occurred.	N	Identisch	zu	3042
3215	Unrecognized message received in mailslot.	N			

3216	System error code occurred.	N	Identisch	zu	3045
3218	IMPORT path path cannot be found.	N	Identisch	zu	3047
3222	Replicator could not access filename on name due to code system error.	N			
3223	The primary domain controller for domain name has apparently failed.	J	PDC_Sys	SK_NV	VSKA
3224	An error occurred while changing this computer's password.	J	Ch_CompPW	O_M	VSM
3226	An error occurred while synchronizing with primary domain controller name.	J	L_SAM	SK_UV	VSM
3230	A power failure was detected at the server.	J	L_Pwr	SK_NV	VSKA
3231	The UPS service performed server shutdown.	J	UPS_SDown	O_E	VSKA
3232	The UPS service did not complete execution of the user specified shutdown command file.	J	Config		
3236	The UPS service failed to execute a user specified shutdown command file filename. The error code is the data.	N			
3257	The system returned an unexpected error code. The error code is the data.	J	L_Sys	SK_NV	VSM
3408	The program cannot be used with this operating system.	J	Config		
3501	You used an invalid option.	J	Config		
3502	System error code has occurred.	N			
3503	The command contains an invalid number of arguments.	J	Config		
3504	The command completed with one or more errors.	J	Config		
3505	You used an option with an invalid value.	J	Config		
3506	The option name is unknown.	J	Config		
3507	Option name is ambiguous.	J	Config		
3510	A command was used with conflicting switches.	J	Config		
3511	Could not find subprogram name.	J	Config		
3513	More data is available than can be returned by Windows NT.	J	Spec_O	O_M	VSM
3515	This command can be used only on a Windows NT Server system.	N			
3521	The name service is not started.	N			
3523	The name service could not be started.	J	Spec_Svc	SK_NV	VSM
3527	The name service is stopping.	N			
3528	The name service could not be stopped.	J	Spec_Svc	SK_DUV	SM
3533	The service is starting or stopping. Please try again later.	N			
3534	The service did not report an error.	N			
3538	The name service failed to resume.	J	Spec_Svc	SK_DUV	SM
3539	The name service failed to pause.	J	Spec_Svc	SK_DUV	SM
3540	The name service continue is pending. text	N			
3541	The name service pause is pending. text	N			
3544	The name service has been started by another process and is pending. text	N			
3547	A service-specific error occurred: code.	J	Spec_Svc	SK_DNV	VSM
3679	An error occurred while reading your profile.	J	Config		
3689	The Workstation service is already running. Windows NT will ignore command options for the workstation.	J	Config		
3691	There are open files and/or incomplete directory searches pending on the connection to name.	N			
3710	An error occurred while opening the Help file.	J	Config		
3711	The Help file is empty.	J	Config		

3712	The Help file is corrupted.	J	Config		
3713	Could not find a primary domain controller for domain name.	J	PDC_Sys	SK_NV	VSM
3716	The device type is unknown.	N			
3719	A matching share could not be found so nothing was deleted.	J	Config		
3721	The password is invalid for name.	J	Config		
3722	An error occurred while sending a message to name.	J	Config		
3723	The password or user name is invalid for name.	J	Config		
3725	An error occurred when the share was deleted.	J	Config		
3726	The user name is invalid.	J	Config		
3727	The password is invalid.	J	Config		
3728	The passwords do not match.	J	Config		
3729	Your persistent connections were not all restored.	J	Config		
3730	This is not a valid computer name or domain name.	J	Config		
3732	Default permissions cannot be set for that resource.	N			
3734	A valid password was not entered.	J	Config		
3735	A valid name was not entered.	J	Config		
3736	The resource named cannot be shared.	J	Config		
3737	The permissions string contains invalid permissions.	N			
3738	You can only perform this operation on printers and communication devices.	N			
3742	Name is an invalid user or group name.	J	Config		
3743	The server is not configured for remote administration.	J	Config		
3753	User user name is not a member of group name.	N			
3754	User user name is already a member of group name.	N			
3755	There is no such user: user name.	J	Config		
3756	This is an invalid response.	J	Config		
3757	No valid response was provided.	J	Config		
3760	Date is not a recognized day of the week.	J	Config		
3761	The time range specified ends before it starts.	J	Config		
3762	Number is not a recognized hour.	J	Config		
3763	Number is not a valid specification for minutes.	J	Config		
3764	Time supplied is not exactly on the hour.	J	Config		
3765	12 and 24 hour time formats may not be mixed.	J	Config		
3766	Text is not a valid 12-hour suffix.	J	Config		
3767	An illegal date format has been supplied.	J	Config		
3768	An illegal day range has been supplied.	J	Config		
3769	An illegal time range has been supplied.	J	Config		
3770	Arguments to NET USER are invalid. Check the minimum password length and/or arguments supplied.	J	Config		
3771	The value for ENABLESCRIPT must be YES.	J	Config		
3773	An illegal country code has been supplied.	J	Config		
3774	The user was successfully created but could not be added to the USERS local group.	J	Usr_2_Grp	O_M	VSM
3775	The user context supplied is invalid.	J	Config		
3776	The dynamic-link library name could not be loaded, or an error occurred while trying to use it.	J	Spec_Dll	SK_NV	VSM
3777	Sending files is no longer supported.	J	Config		
3778	You may not specify paths for ADMIN$ and IPC$ shares.	J	Config		
3779	User or group name is already a member of local group name.	N			
3780	There is no such user or group: name.	J	Config		
3781	There is no such computer: computer name.	J	Config		
3782	The computer computer name already exists.	J	Config		

3790	The system could not find message: text.	J	Config		
3802	This schedule date is invalid.	J	Config		
3805	The Server service has not been started.	N			
3806	The AT job ID does not exist.	J	Config		
3809	The command line cannot exceed 259 characters.	J	Config		
3813	The AT schedule file was deleted.	J	Config		
3815	The AT command has timed-out. Please try again later.	J	Config		
3816	The minimum password age for user accounts cannot be greater than the maximum password age.	J	Config		
3817	You have specified a value that is incompatible with servers with down-level software. Please specify a lower value.	N			
3870	Computer name is not a valid computer name.	J	Config		
3871	Code is not a valid Windows NT message number.	J	Config		
3912	Could not locate a time-server.	J	Config		
3913	Could not find the primary domain controller for domain name.	N			
3915	The user's home directory could not be determined.	J	Config		
3916	The user's home directory has not been specified.	J	Config		
3917	The name specified for the user's home directory name is not a universal naming convention (UNC) name.	J	Config		
3932	Name is not a valid domain or workgroup name.	J	Config		
3951	You specified too many values for the name option.	J	Config		
3952	You entered an invalid value for the name option.	N			
3953	The syntax is incorrect.	J	Config		
3960	You specified an invalid file number.	J	Config		
3961	You specified an invalid print job number.	N			
3963	The user or group account specified cannot be found.	N			
4738	The share name entered is not accessible from an MS-DOS workstation. Are you sure you want to use this share name? name:	N			
5022	Printing errors occurred.	N			
5305	A network control block (NCB) command timed-out. The session may have terminated abnormally. The NCB is the data.	N			
5309	No resource was available in the network adapter. The network control block (NCB) request was refused. The NCB is the data.	N			
5317	The local session table is full. The network control block (NCB) request was refused. The NCB is the data.	N			
5334	There are too many network control block (NCB) commands outstanding. The NCB request was refused. The NCB is the data.	N			
5508	The security database full synchronization has been initiated by the server name.	N			
5509	Windows NT could not be started as configured. A previous working configuration was used instead.	J	Config		
5510	The exception 0xnumber occurred in the application application at location 0xnumber.	N			
5511	The server name claims to be a primary domain controller for the name domain. However, the security identifier on the server name in that domain and on the primary domain controller name don't match. One of the servers should be removed from the domain.	N			
5512	The server name and name both claim to be the primary domain controller for the name domain. One of the servers	J	PDC_Sys	SK_UV	VSKA

	should be demoted or removed from the domain.				
5513	The computer computer name connected to server name using the trust relationship to the name domain. However, the computer doesn't properly know the security identifier (SID) for the domain. Reestablish the trust relationship.	N			
5601	The password for this computer is not found in the local security database.	J	L_SAM	SK_UV	VSM
5602	An internal error occurred while accessing the computer's local or network security database.	J	L_SAM / Spec_SAM	SK_UV	VSM
5700	The Netlogon service could not initialize the replication data structures successfully. The service is terminated.	J	L_DiskSp	SK_NV	VSM
5701	The Netlogon service failed to update the domain trust list.	J	L_DiskSp / L_SAM	SK_NV / SK_UV	VSM / VSM
5702	The Netlogon service could not add the RPC interface. The service is terminated.	J	L_NetLg_Svc	SK_NV	VSM
5703	The Netlogon service could not read a mailslot message from name.	J	L_NetLg_Svc	SK_UV	VSM
5704	The Netlogon service failed to register the service with the service controller. The service is terminated.	J	L_Svc_Ctrl	SK_NV	VSM
5705	The change log cache maintained by the Netlogon service for database changes is corrupted. The Netlogon service is resetting the change log.	J	L_DiskSp / L_NetLg_Svc	SK_NV / SK_UV	VSM / VSM
5706	The Netlogon service could not create server share name.	J	L_S_Svc	SK_NV	VSM
5707	The down-level logon request for the user user name from name failed.	J	Config		
5708	The down-level logoff request for the user user name from name failed.	N			
5709	The Windows NT network logon request for the user name from computer name (via name) failed.	J	Config		
5710	The Windows NT network logoff request for the user name from computer name failed.	N			
5711	The partial synchronization request from the server name completed successfully. Number change(s) has (have) been returned to the caller.	N			
5712	The partial synchronization request from the server name failed.	J	Spec_Sys	SK_NV	VSM
5715	The partial synchronization replication of the name database from the primary domain controller name completed successfully. Number change(s) is (are) applied to the database.	N			
5716	The partial synchronization replication of the name database from the primary domain controller name failed.	J	Spec_Sys	SK_NV	VSM
5717	The full synchronization replication of the name database from the primary domain controller name completed successfully.	N			
5718	The full synchronization of the name database from the primary domain controller name failed.	J	Spec_Sys	SK_NV	VSM
5719	No Windows NT Server for the domain name is available.	J	Net	SK_NV	VSKA
5720	The session setup to the Windows NT Server name for the domain name failed because the computer computer name does not have a local security database account.	J	L_SAM	SK_UV	VSM
5721	The session setup to the Windows NT Server name for the domain name failed because the Windows NT Server does not have an account for the computer computer name.	J	Spec_SAM	SK_UV	VSM
5722	The session setup from the computer computer name failed to authenticate. The name of the account referenced in the	J	Spec_SAM	SK_UV	VSM

	security database is user name.				
5723	The session setup from the computer computer name failed because there is no trust account in the security database for this computer. The name of the account referenced in the security database is user name.	J	Spec_SAM	SK_UV	VSM
5724	Could not register control handler with service controller name.	J	Spec_Svc_Ctrl	SK_NV	VSM
5725	Could not set service status with service controller name.	J	Spec_Svc_Ctrl	SK_NV	VSM

Tabelle : Übersicht der NT-Netz-Events

B.3 Details zu den MSCS Events

ClusMsg.h **ANFANG**

```
/*++
Copyright (c) 1991-1996  Microsoft Corporation
Module Name:    clusmsg.h
Abstract:    This file contains the message definitions for the cluster service.
Author:    davidp 19-Sep-1996
Revision History:
Notes:    This file is generated from clusmsg.mc
--*/

#ifndef _CLUSMSG_H_
#define _CLUSMSG_H_

//
//  Values are 32 bit values layed out as follows:
//
//   3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//  +---+-+-+-----------------------+-------------------------------+
//  |Sev|C|R|     Facility          |               Code            |
//  +---+-+-+-----------------------+-------------------------------+
//
//  where
//
//      Sev - is the severity code
//
//          00 - Success
//          01 - Informational
//          10 - Warning
//          11 - Error
//
//      C - is the Customer code flag
//
//      R - is a reserved bit
//
//      Facility - is the facility code
//
//      Code - is the facility's status code
//
//
// Define the facility codes
```

```
//
//
// Define the severity codes
//
//
// MessageId: ERROR_NETWORK_NOT_FOUND
// MessageText:  The cluster network could not be found.
#define ERROR_NETWORK_NOT_FOUND          0x000013AAL
//
// MessageId: ERROR_NETWORK_NOT_AVAILABLE
// MessageText:  A cluster network is not available for this operation.
#define ERROR_NETWORK_NOT_AVAILABLE      0x000013ABL
//
// MessageId: ERROR_NODE_NOT_AVAILABLE
// MessageText:  A cluster node is not available for this operation.
#define ERROR_NODE_NOT_AVAILABLE         0x000013ACL

#endif // _CLUSMSG_H_
```

ClusMsg.h **ENDE**

Cluster.inf **ANFANG**

```
 [SourceDisksNames]
1=,,1

[SourceDisksFiles]
cluadmin.exe=1,,0,0
cluadmex.dll=1,,0,0
iisclex3.dll=1,,0,0
debugex.dll=1,,0,0
cluadmin.cnt=1,,0,0
cluadmin.hlp=1,,0,0
cluster.exe=1,,0,0
clusapi.dll=1,,0,0
clussprt.dll=1,,0,0
clusdisk.sys=1,,0,0
clusnet.sys=1,,0,0
wshclus.dll=1,,0,0
clussvc.exe=1,,0,0
clusprxy.exe=1,,0,0
timeserv.exe=1,,0,0
clusres.dll=1,,0,0
iisclus3.dll=1,,0,0
resutils.dll=1,,0,0
resrcmon.exe=1,,0,0
rpcltccl.dll=1,,0,0
rpcltscl.dll=1,,0,0
security.dll=1,,0,0
setup.exe=1,,0,0
clusdb=1,,0,0
clusdb.log=1,,0,0
cluster.inf=1,,0,0

[ClusterDriverFiles]
clusdisk.sys,,,1
clusnet.sys,,,1
```

```
[ClusterFiles]
clussvc.exe,,,1
clusprxy.exe,,,1
timeserv.exe,,,1
clusres.dll,,,1
iisclus3.dll,,,1
resrcmon.exe,,,1
rpcltccl.dll,,,1
rpcltscl.dll,,,1
wshclus.dll,,,1
setup.exe,,,1

[ClusterAdminFiles]
cluadmin.exe,,,1
cluadmin.cnt,,,1
cluadmin.hlp,,,1
cluadmex.dll,,,1
iisclex3.dll,,,1
debugex.dll,,,1
resutils.dll,,,1

[ClusterSystemFiles]
clussprt.dll,,,1

[ClusterAdminSystemFiles]
clusapi.dll,,,1
cluster.exe,,,1

[ClusterSecurityFiles]
security.dll,,,1

;
; These are some files that are created by the cluster service or otherwise
; not in a different section.
;
[ClusterUninstallFiles]
clusdb
clusdb.log
cluster.inf,,,1

[ClusDisk.Service]
DisplayName = %ClusDisk.SvcDesc%
ServiceType = 1         ;   SERVICE_KERNEL_DRIVER
StartType = 1           ;   SERVICE_SYSTEM_START
ErrorControl = 1        ;   SERVICE_ERROR_NORMAL
ServiceBinary  = %12%\clusdisk.sys
LoadOrderGroup = Filter

[ClusNet.Service]
DisplayName = %ClusNet.SvcDesc%
ServiceType = 1         ;   SERVICE_KERNEL_DRIVER
StartType = 3           ;   SERVICE_DEMAND_START
ErrorControl = 1        ;   SERVICE_ERROR_NORMAL
ServiceBinary  = %12%\clusnet.sys
LoadOrderGroup = Tdi
Dependencies = Tcpip
```

```
[Cluster.Service]
DisplayName = %Cluster.SvcDesc%
ServiceType = 0x10      ;   SERVICE_WIN32_OWN_PROCESS
StartType = 2           ;   SERVICE_AUTO_START
ErrorControl = 1        ;   SERVICE_ERROR_NORMAL
ServiceBinary = %CLUSTERTARGETDIR%\clusprxy.exe
Dependencies = ClusNet,RpcSs,NtLmSsp

[ClusSvc_EventLog_Inst]
AddReg=ClusSvc_EventLog_AddReg

[ClusSvc_EventLog_AddReg]
HKR,,EventMessageFile,0,"%CLUSTERTARGETDIR%\clussvc.exe"
HKR,,TypesSupported,0x00010001,7

[ClusSvc_EventLog_DelReg]
HKLM,System\CurrentControlSet\Services\EventLog\ClusSvc

;
; This is here in case the service controller gets wedged and can't delete
; the services.
;
[ClusSvc_Services_DelReg]
HKLM,System\CurrentControlSet\Services\ClusSvc
HKLM,System\CurrentControlSet\Services\ClusNet
HKLM,System\CurrentControlSet\Services\ClusDisk

[Time.Service]
DisplayName = %Time.SvcDesc%
ServiceType = 0x10      ;   SERVICE_WIN32_OWN_PROCESS
StartType = 3           ;   SERVICE_DEMAND_START
ErrorControl = 1        ;   SERVICE_ERROR_NORMAL
ServiceBinary = %CLUSTERTARGETDIR%\timeserv.exe


[GeneralRegEntries]
AddReg=General, ClusSvc, TimeSvc, ClusDisk, ClusNet, AdminRegEntries

[NewNodeRegEntries]
AddReg=General, ClusDisk, ClusNet, TimeSvc, AdminRegEntries

[ClusterRegEntries]
AddReg=ServiceRegEntries

[LocalQuorumRegEntries]
AddReg=LocalQuorumServiceRegEntries

[General]
;
; Enable cluster rpc transport
;
HKLM,Software\Microsoft\Rpc\ClientProtocols,ncadg_cluster,0,"rpcltccl.dll"
HKLM,Software\Microsoft\Rpc\ServerProtocols,ncadg_cluster,0,"rpcltscl.dll"

;
; Enable Uninstall
;
```

```
HKLM,Software\Microsoft\Windows\CurrentVersion\Uninstall\NT                    Clu-
sters,DisplayName,0,%ClusterProductName%
HKLM,Software\Microsoft\Windows\CurrentVersion\Uninstall\NT                    Clu-
sters,UninstallString,0,"%CLUSTERTARGETDIR%\setup -uninstall"


[ClusSvc]
HKLM,System\CurrentControlSet\Services\ClusSvc
HKLM,System\CurrentControlSet\Services\ClusSvc\Parameters

[TimeSvc]
HKLM,System\CurrentControlSet\Services\TimeServ
HKLM,System\CurrentControlSet\Services\TimeServ\Parameters

[ClusDisk]
;
;   ClusDisk
;
HKLM,System\CurrentControlSet\Services\ClusDisk,Tag,0x10001,4
HKLM,System\CurrentControlSet\Services\ClusDisk\Parameters
HKLM,System\CurrentControlSet\Services\ClusDisk\Parameters\Scsi

[ClusNet]
;
;   ClusNet
;
HKLM,System\CurrentControlSet\Services\ClusNet\Parameters
HKLM,System\CurrentControlSet\Services\ClusNet\Parameters\Winsock

[AdminRegEntries]
HKLM,Software\Classes\CLSID\{4EC90FB0-D0BB-11CF-B5EF-00A0C90AB505},,,"Cluster Administrator
Standard Extension"
HKLM,Software\Classes\CLSID\{4EC90FB0-D0BB-11CF-B5EF-
00A0C90AB505}\InProcServer32,,,"%CLUSTERTARGETDIR%\cluadmex.dll"
HKLM,Software\Classes\CLSID\{4EC90FB0-D0BB-11CF-B5EF-
00A0C90AB505}\InProcServer32,ThreadingModel,,"Apartment"
HKLM,Software\Classes\CLSID\{4EC90FB0-D0BB-11CF-B5EF-
00A0C90AB505}\ProgId,,,"CLUADMEX.Standard"
HKLM,Software\Classes\CLSID\{4EC90FB0-D0BB-11CF-B5EF-
00A0C90AB505}\VersionIndependentProgId,,,"CLUADMEX.Standard"
HKLM,Software\Classes\CLUADMEX.Standard,,,"Cluster Administrator Standard Extension"
HKLM,Software\Classes\CLUADMEX.Standard\CLSID,,,{4EC90FB0-D0BB-11CF-B5EF-
00A0C90AB505}
HKLM,Software\Classes\CLSID\{4EC90FC0-D0BB-11CF-B5EF-00A0C90AB505},,,"Cluster Administrator
IIS 3.0 Extension"
HKLM,Software\Classes\CLSID\{4EC90FC0-D0BB-11CF-B5EF-
00A0C90AB505}\InProcServer32,,,"%CLUSTERTARGETDIR%\iisclex3.dll"
HKLM,Software\Classes\CLSID\{4EC90FC0-D0BB-11CF-B5EF-
00A0C90AB505}\InProcServer32,ThreadingModel,,"Apartment"
HKLM,Software\Classes\CLSID\{4EC90FC0-D0BB-11CF-B5EF-
00A0C90AB505}\ProgId,,,"CLUADMEX.IIS"
HKLM,Software\Classes\CLSID\{4EC90FC0-D0BB-11CF-B5EF-
00A0C90AB505}\VersionIndependentProgId,,,"CLUADMEX.IIS"
HKLM,Software\Classes\CLUADMEX.IIS,,,"Cluster Administrator IIS 3.0 Extension"
HKLM,Software\Classes\CLUADMEX.IIS\CLSID,,,{4EC90FC0-D0BB-11CF-B5EF-00A0C90AB505}
HKLM,Software\Microsoft\Windows\CurrentVersion\App
Paths\CluAdmin.exe,,,"%CLUSTERTARGETDIR%\CluAdmin.exe"
HKCU,Software\Microsoft\Cluster Administrator
```

```
[ServiceRegEntries]
HKR,Nodes
HKR,Networks
HKR,NetworkInterfaces
HKR,Groups
HKR,Resources
HKR,ResourceTypes
HKR,Quorum

;
; Resource types
;
HKR,ResourceTypes\%GenApp.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%GenApp.TypeName%,Name,,%GenApp.LTypeName%
HKR,ResourceTypes\%GenApp.TypeName%,IsAlivePollInterval,0x10001,60000
HKR,ResourceTypes\%GenApp.TypeName%,LooksAlivePollInterval,0x10001,5000
HKR,ResourceTypes\%GenApp.TypeName%,AdminExtensions,0x10000,"{4EC90FB0-D0BB-11CF-B5EF-
00A0C90AB505}"

HKR,ResourceTypes\%GenSvc.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%GenSvc.TypeName%,Name,,%GenSvc.LTypeName%
HKR,ResourceTypes\%GenSvc.TypeName%,IsAlivePollInterval,0x10001,60000
HKR,ResourceTypes\%GenSvc.TypeName%,LooksAlivePollInterval,0x10001,5000
HKR,ResourceTypes\%GenSvc.TypeName%,AdminExtensions,0x10000,"{4EC90FB0-D0BB-11CF-B5EF-
00A0C90AB505}"

HKR,ResourceTypes\%NetName.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%NetName.TypeName%,Name,,%NetName.LTypeName%
HKR,ResourceTypes\%NetName.TypeName%,IsAlivePollInterval,0x10001,60000
HKR,ResourceTypes\%NetName.TypeName%,LooksAlivePollInterval,0x10001,5000
HKR,ResourceTypes\%NetName.TypeName%,AdminExtensions,0x10000,"{4EC90FB0-D0BB-11CF-
B5EF-00A0C90AB505}"

HKR,ResourceTypes\%PhysDisk.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%PhysDisk.TypeName%,Name,,%PhysDisk.LTypeName%
HKR,ResourceTypes\%PhysDisk.TypeName%,IsAlivePollInterval,0x10001,60000
HKR,ResourceTypes\%PhysDisk.TypeName%,LooksAlivePollInterval,0x10001,5000
HKR,ResourceTypes\%PhysDisk.TypeName%,AdminExtensions,0x10000,"{4EC90FB0-D0BB-11CF-
B5EF-00A0C90AB505}"

HKR,ResourceTypes\%PrtSplSvc.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%PrtSplSvc.TypeName%,Name,,%PrtSplSvc.LTypeName%
HKR,ResourceTypes\%PrtSplSvc.TypeName%,IsAlivePollInterval,0x10001,60000
HKR,ResourceTypes\%PrtSplSvc.TypeName%,LooksAlivePollInterval,0x10001,5000
HKR,ResourceTypes\%PrtSplSvc.TypeName%,AdminExtensions,0x10000,"{4EC90FB0-D0BB-11CF-
B5EF-00A0C90AB505}"

HKR,ResourceTypes\%FileShr.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%FileShr.TypeName%,Name,,%FileShr.LTypeName%
HKR,ResourceTypes\%FileShr.TypeName%,IsAlivePollInterval,0x10001,60000
HKR,ResourceTypes\%FileShr.TypeName%,LooksAlivePollInterval,0x10001,5000
HKR,ResourceTypes\%FileShr.TypeName%,AdminExtensions,0x10000,"{4EC90FB0-D0BB-11CF-B5EF-
00A0C90AB505}"

HKR,ResourceTypes\%IpAddr.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%IpAddr.TypeName%,Name,,%IpAddr.LTypeName%
```

```
HKR,ResourceTypes\%IpAddr.TypeName%,IsAlivePollInterval,0x10001,60000
HKR,ResourceTypes\%IpAddr.TypeName%,LooksAlivePollInterval,0x10001,5000
HKR,ResourceTypes\%IpAddr.TypeName%,AdminExtensions,0x10000,"{4EC90FB0-D0BB-11CF-B5EF-
00A0C90AB505}"

HKR,ResourceTypes\%TimeSvc.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%TimeSvc.TypeName%,Name,,%TimeSvc.LTypeName%
HKR,ResourceTypes\%TimeSvc.TypeName%,IsAlivePollInterval,0x10001,120000
HKR,ResourceTypes\%TimeSvc.TypeName%,LooksAlivePollInterval,0x10001,60000

;HKR,ResourceTypes\%DhcpSvc.TypeName%,DllName,,"clusres.dll"
;HKR,ResourceTypes\%DhcpSvc.TypeName%,Name,,%DhcpSvc.LTypeName%
;HKR,ResourceTypes\%DhcpSvc.TypeName%,IsAlivePollInterval,0x10001,60000
;HKR,ResourceTypes\%DhcpSvc.TypeName%,LooksAlivePollInterval,0x10001,5000
;HKR,ResourceTypes\%DhcpSvc.TypeName%,AdminExtensions,0x10000,"{4EC90FC0-D0BB-11CF-
B5EF-00A0C90AB505}"

HKR,ResourceTypes\%IIS.TypeName%,DllName,,"iisclus3.dll"
HKR,ResourceTypes\%IIS.TypeName%,Name,,%IIS.LTypeName%
HKR,ResourceTypes\%IIS.TypeName%,IsAlivePollInterval,0x10001,60000
HKR,ResourceTypes\%IIS.TypeName%,LooksAlivePollInterval,0x10001,5000
HKR,ResourceTypes\%IIS.TypeName%,AdminExtensions,0x10000,"{4EC90FC0-D0BB-11CF-B5EF-
00A0C90AB505}"

HKR,ResourceTypes\%MSMQ.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%MSMQ.TypeName%,Name,,%MSMQ.LTypeName%
HKR,ResourceTypes\%MSMQ.TypeName%,IsAlivePollInterval,0x10001,120000
HKR,ResourceTypes\%MSMQ.TypeName%,LooksAlivePollInterval,0x10001,60000

HKR,ResourceTypes\%MSDTC.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%MSDTC.TypeName%,Name,,%MSDTC.LTypeName%
HKR,ResourceTypes\%MSDTC.TypeName%,IsAlivePollInterval,0x10001,120000
HKR,ResourceTypes\%MSDTC.TypeName%,LooksAlivePollInterval,0x10001,60000

[LocalQuorumServiceRegEntries]
HKR,ResourceTypes\%LocalQuorum.TypeName%,DllName,,"clusres.dll"
HKR,ResourceTypes\%LocalQuorum.TypeName%,Name,,%LocalQuorum.TypeName%
HKR,ResourceTypes\%LocalQuorum.TypeName%,IsAlivePollInterval,0x10001,60000
HKR,ResourceTypes\%LocalQuorum.TypeName%,LooksAlivePollInterval,0x10001,5000

[Strings]
ClusterProductName="Microsoft Cluster Server"

;
; Resource Type Names (Non-Localizable)
;
GenApp.TypeName="Generic Application"
GenSvc.TypeName="Generic Service"
NetName.TypeName="Network Name"
PhysDisk.TypeName="Physical Disk"
PrtSplSvc.TypeName="Print Spooler"
FileShr.TypeName="File Share"
IpAddr.TypeName="IP Address"
TimeSvc.TypeName="Time Service"
IIS.TypeName="IIS Virtual Root"
LocalQuorum.TypeName="Local Quorum"
DhcpSvc.TypeName="DHCP Server"
MSMQ.TypeName="Microsoft Message Queue Server"
```

```
MSDTC.TypeName="Distributed Transaction Coordinator"

;
; Localizable Resource Type Names
;
GenApp.LTypeName="Generic Application"
GenSvc.LTypeName="Generic Service"
NetName.LTypeName="Network Name"
PhysDisk.LTypeName="Physical Disk"
PrtSplSvc.LTypeName="Print Spooler"
FileShr.LTypeName="File Share"
IpAddr.LTypeName="IP Address"
TimeSvc.LTypeName="Time Service"
IIS.LTypeName="IIS Virtual Root"
LocalQuorum.LTypeName="Local Quorum"
DhcpSvc.LTypeName="DHCP Server"
MSMQ.LTypeName="Microsoft Message Queue Server"
MSDTC.LTypeName="Distributed Transaction Coordinator"

;
; Service Names (descriptions)
;
Time.SvcDesc="Time Service"
Cluster.SvcDesc="Cluster Server"
ClusDisk.SvcDesc="Cluster Disk"
ClusNet.SvcDesc="Cluster Network"



CLUSTERTARGETDIR = "c:\winntse\cluster"
```

Cluster.inf **ENDE**