Untersuchung von HelpDesk-Systemen der Kommunikationsbranche

Dissertation
zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)
vorgelegt der

Fakultät für Informatik und Automatisierung
der Technischen Universität Ilmenau

von

Diplom-Ingenieur Jörg Dieter Benze

Gutachter:
Univ. Prof. Dr.-Ing. habil. Dietrich Reschke
Univ. Prof. Dr.-Ing. habil. Wolfgang Fengler
Prof. Dr. rer. nat. Gerhard Raffius

Tag der Einreichung: 16. Dezember 2004
Tag der wissenschaftlichen Aussprache: 6. Oktober 2005

urn:nbn:de:gbv:ilm1-2005000171

Kurzfassung

Für aufgetretene Probleme in der IT-Infrastruktur ist die schnelle Verfügbarkeit von technischen Lösungen für Unternehmen wichtig. Die Beseitigung der Probleme wird durch HelpDesk-Systeme (Arbeitsorganisationsform) sichergestellt. Zur IT-Unterstützung eines HelpDesk-Systems werden unabhängig voneinander arbeitende IT-Systeme wie Trouble-Ticket Systeme, Netzwerkmanagementsysteme und Netzwerkdokumentationssysteme eingesetzt. Gegenstand dieser Arbeit ist eine Weiterentwicklung und Integration dieser IT-Unterstützungssysteme.

Das Konzept der Trouble-Ticket Systeme wird zu flexibel anpaßbaren Informationsverwaltungs- und Informationsaustauschsystemen weiterentwickelt. Hierzu wurde ein generisches Modell einer HelpDesk-Applikation entwickelt. Kern des generischen Modells ist ein zustandsraumbasiertes Steuerungssystem, welches eine neue Art der Steuerung, Regelung und Überwachung von Informationsströmen in HelpDesk-Systemen darstellt. Der Nachweis der Anwendbarkeit des generischen Modells ist durch eine prototypische Implementierung erbracht worden.
Da alle in der Arbeit analysierten Trouble-Ticket Systeme auf einer Client-Server Architektur basieren, gilt es festzustellen, ob eine andere Architekturform Vorteile zum Aufbau einer HelpDesk-Applikation aufweist. Exemplarisch wurde hierzu ein Architekturkonzept für eine HelpDesk-Applikation auf Basis mobiler Agenten entworfen. Dieses bietet Vorteile bei der Ausnutzung der Netzwerkressourcen und weist ein hohes Entwicklungspotential von Adaptions- und Automatisierungsmöglichkeiten auf, welches weit über dem heutiger Client-Server basierter Systeme liegt.

Für das Netzwerkmanagement werden Umbrella-Management-Systeme eingesetzt, die nur eine Überwachung ermöglichen. Aufgrund ihrer Architektur können sie kein automatisches Netzwerkmanagement leisten.
Grundvoraussetzung für die Automatisierung des Netzwerkmangements ist die Netzwerkdokumentation. In dieser Arbeit ist ein Schichtenmodell der Netzwerkdokumentation entstanden, welches das Netzwerks in funktionale Ebenen einteilt, wie sie für eine Netzwerkdokumentation benötigt werden. Diese neue Art der Netzwerkdokumentation ermöglicht die Abbildung aktueller Betriebszustandsdaten in das Netzwerkmodell. Durch die Verknüpfung der Topologieinformationen mit den Betriebszustandsdaten konnten neue Ansätze zum Aufbau modellbasierter Impact- und Root-Cause Analysen entwickelt werden.
Für den Aufbau eines automatischen Netzwerkmanagements ist in dieser Arbeit ein Architekturkonzept für ein aktives Netzwerkmanagement geschaffen worden. Hier ist im Gegensatz zu bestehenden Systemen die Anzahl der automatisch bearbeitbaren Fehlerfälle nicht beschränkt.

Abstract

The fast availability of technical solutions is important to businesses facing problems occurred in the IT infrastructure. The removal of these problems can be ensured by means of helpdesk systems (labor organization). The helpdesk system is supported by independently operating IT systems such as trouble ticket systems, network management systems, and network documentation systems. The subject of this study is the further development and integration of these IT support systems.

The concept of the trouble ticket systems is further developed into flexibly customizable information management and information exchange systems. A generic model of a helpdesk application has been developed for this purpose. The core of the generic model is a state-space-based control system which represents a new form of controlling, regulating, and monitoring information flows in helpdesk systems. The applicability of the generic model has been proved by means of a prototype implementation.
Since all trouble ticket systems analyzed in the scope of the study are based on a client-server architecture, possible advantages of using another type of architecture for setting up a helpdesk application have to be verified. As an example, an architectural concept has been designed for a helpdesk application based on mobile agents. This concept offers advantages regarding the utilization of the network resources. Its high potential for the development of adaptation and automation options by far exceeds the one of current client-server-based systems.

Umbrella management systems are used for network management. They merely provide monitoring features. Due to their architecture, they cannot perform automated network management.
The basic requirement for the automation of network management is the network documentation. In the scope of this study, a layer-model of the network documentation has been created. This model divides the network into the functional layers as required for a network documentation. This new type of network documentation permits the integration of up-to-date operating state data into the network model. By linking the topology information with the operating state data, new approaches for creating model-based impact and root cause analyses have been developed.
For the purpose of setting up automated network management, an architectural concept for active network management has been created in the scope of this study. In contrast to existing systems, this concept does not limit the number of error cases that can be processed automatically.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Teil I

Einleitung

Kapitel 1

Einleitung

1.1 Umfeld

Seitdem Computer und Netzwerke in Unternehmen Verwendung finden, ist ein Benutzer-Support für diese Systeme erforderlich. Die schnelle Verfügbarkeit von technischen Lösungen für aufgetretene Probleme in der IT-Infrastruktur ist für Unternehmen wichtig, da sie heute existentiell vom störungsfreien Betrieb ihrer Daten- und Kommunikationsnetzwerke abhängig sind. Störungen bzw. Unterbrechungen in der Verfügbarkeit dieser Netzwerke sind heute gleichbedeutend mit der Unterbrechung bzw. Einschränkung von Geschäftsabläufen, was zu Verlusten an Zeit und Geld führt.

Benutzer-Support bedeutet technische Lösungen für nicht technisch orientierte Benutzer der Systeme zur Verfügung zu stellen und allgemeinverständlich zu kommunizieren. Die Erarbeitung von Lösungen für Nichtfachleute ist fachgebietsunabhängig. Daher überrascht es nicht, daß ausgehend von der IT-Branche eine Supportindustrie am Entstehen bzw. Wachsen ist. Hierdurch wird anwendbares Fachwissen durch die Supportdienstleister zu einer handelbaren Ware. Es ermöglicht den Kunden ohne lange Vorlaufzeiten und hohe Initialisierungskosten schnell an qualitativ hochwertiges Wissen zu kommen [Bruton97].

Fachwissen alleine reicht zur Erbringung von Supportdienstleistungen nicht aus. Der Supportdienstleister muß über eine entsprechende soziale Kompetenz zur Wissensvermittlung verfügen. Er sollte in der Lage sein, komplexe Sachverhalte dem Nutzer, der kein Fachmann in dem Fachgebiet der Supportdienstleistung ist, verständlich zu erläutern. Diese Supportfähigkeit ist unabhängig vom Fachgebiet. Ebenso ist das Know-how, wie Supportdienstleistungen erbracht werden können, unabhängig vom Fachgebiet [Bruton97].

Zur Erbringung von Supportdienstleistungen werden heute Supportdienstleistungsorganisationen eingesetzt, die als Untergruppierung eines Unternehmens bzw. einer Organisation oder als selbständiges Unternehmen existieren. Das Ziel der Supportdienstleistungsorganisation ist einerseits, dem Nutzer einen optimalen Service zu bieten und andererseits den größtmöglichen Gewinn auf lange Sicht zu erzielen [Wöhe00]. Hierzu wird von der Supportdienstleistungsorganisation mindestens ein HelpDesk-System eingesetzt. Der Be-

griff “HelpDesk-System” wird in Publikationen von Unternehmen und Organisationen mit vielfachen Bedeutungsvarianten verwendet. Deshalb wird an dieser Stelle eine Definition versucht, wie der Begriff “HelpDesk-System” zu verstehen ist, was durch die bewußte Schreibweise des Begriffs mit einem großen “D” zum Ausdruck kommen soll.

> **Definition: HelpDesk-System**
>
> Unter einem HelpDesk-System ist eine Form der Arbeitsorganisation zu verstehen, die kurzfristig Unterstützung in Form von Dienstleistungen und Sachwerten erbringt, die in einem abgegrenzten Fokus liegt. Das HelpDesk-System besteht aus qualifizierten Mitarbeitern (sog. **Supportdienstleistungseinheit**) und EDV-Systemen zur Unterstützung der Tätigkeit der Mitarbeiter (sog. **HelpDesk-Applikationen**).

Eine Supportdienstleistungsorganisation setzt mindestens ein HelpDesk-System ein, kann aber auch mehrere HelpDesk-Systeme mit sehr unterschiedlichen Ausrichtungen einsetzen (siehe Abbildung 1.1).

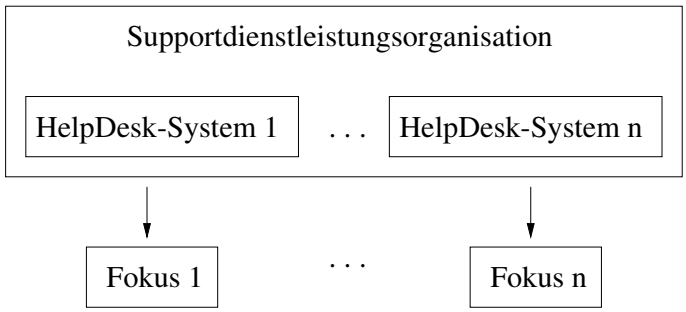


Abbildung 1.1: Aufbau einer Supportdienstleistungsorganisation

1.2 Motivation und Ziel der Arbeit

Motivation für diese Arbeit ist es, Verbesserungen in der **Verfügbarkeit von Telekommunikationsnetzwerken** zu erreichen. Hierbei soll die Zeitdauer vom Auftreten einer Störung bis zu deren Behebung verkürzt werden. Dies kann auf zwei Arten geschehen:

1. Verbesserung der Informationslage und der Kommunikation von Mitarbeitern im HelpDesk-System, welches durch eine Verbesserung der EDV-Unterstützung von Arbeitsprozessen erreicht wird.

2. Automatisierung bzw. Teilautomatisierung der Entstörung von Netzwerkfehlern.

Hierdurch soll eine **Steigerung der Zuverlässigkeit** bzw. eine **Verkürzung der Serviceunterbrechungen** erreicht werden.

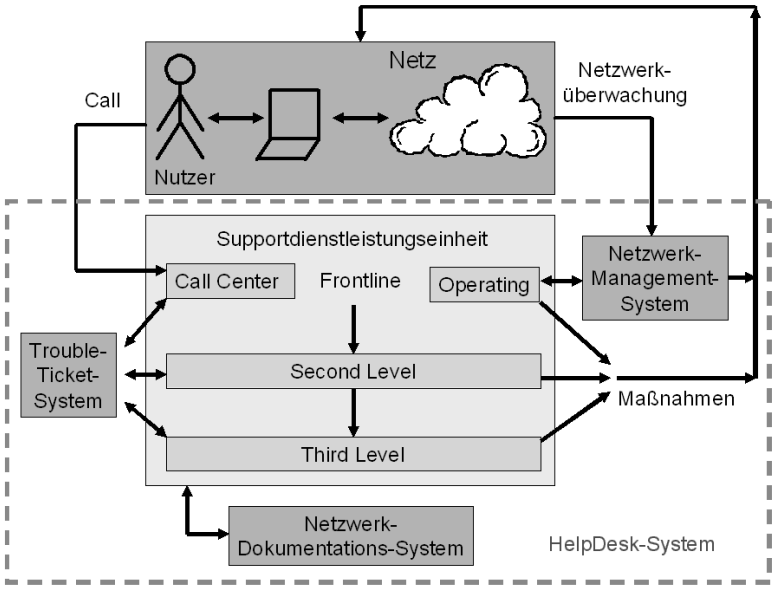


Abbildung 1.2: Darstellung des Fokuses der Arbeit

Für das HelpDesk-System bestehen prinzipiell **zwei** Möglichkeiten von einer Störung im Netzwerk Kenntnis zu erlangen (siehe Abbildung 1.2). Die **erste Möglichkeit** ist, daß ein Nutzer die Störung per Call an das Call Center meldet. Die **zweite Möglichkeit** besteht in der automatischen Überwachung der Komponenten des Netzwerks durch ein Netzwerkmanagementsystem, welches Alarmmeldungen generiert, wenn Abweichungen von einem vorher definierten Sollzustand festgestellt werden.

Die auflaufenden Meldungen, die Symptome für Störungen darstellen, werden in der Supportdienstleistungseinheit bearbeitet. Zur Unterstützung der Arbeitsprozesse kommen **drei Arten von technischen Systemen** zum Einsatz. Den **Schwerpunkt der Arbeit** bildet deren Untersuchung und Weiterentwicklung. Sie verfügen vielfach über Automatismen zur teil- oder vollautomatischen Ausführung von Arbeitsschritten.

- **Trouble-Ticket Systeme** zur Verwaltung, Steuerung und Dokumentation des Stands der Arbeitsprozesse.
- **Netzwerkmanagementsysteme**, die Informationen über den Betriebszustand des Netzwerks liefern, die Fernsteuerung von Komponenten ermöglichen und bei der Fehlererkennung unterstützen.
- **Netzwerkdokumentationssysteme** zur Dokumentation der Netzwerkinfrastruktur und der Konfiguration des Netzwerks.

Betrachtet man o.g. Systeme im Einzelnen, so sind folgende **Unzulänglichkeiten** bzw. **Mängel im heutigen Vorgehen** festzustellen:

Trouble-Ticket Systeme verfügen über eine geringe Flexibiliät. Hieraus resultiert eher eine Anpassung der Arbeitsprozesse an die Erfordernisse des EDV-Systems als umgekehrt. Auch sind die Reaktionszeiten auf geänderte Anforderungen zu groß.

In dieser Arbeit werden Trouble-Ticket Systeme zu **Datenhaltungs- und Informationsaustauschsystemen** weiterentwickelt, die schnell und flexibel an Supportdienstleistungseinheiten angepaßt werden können.

Netzwerkmanagementsysteme sind auf eine Überwachung des Netzwerks ausgerichtet. Ihre Architektur ist auf die Sammlung und Aufbereitung von Informationen ausgelegt, um dem Operator eine gesamtheitliche Sicht auf den aktuellen Betriebszustand zu bieten. Netzwerkmanagementsysteme sind nur ansatzweise in der Lage, Störungen automatisch zu erkennen und automatisch zu entstören. Dies hat seinen Grund u.a. in der auf einer Überwachung ausgelegten Architektur.

In dieser Arbeit wird ein **Architekturkonzept** entwickelt, welches auf ein **aktives Entstörungsmanagement** ausgerichtet ist und eine Integration mit Datenhaltungs- und Informationsaustauschsystemen (s.o.) ermöglicht.

Netzwerkdokumentationssysteme beschreiben den Aufbau und die Konfiguration einer Netzwerkinfrastruktur; allerdings nicht deren aktuellen Betriebszustand. Diese Inventardaten sind eine Schlüsselvoraussetzung für viele Automatismen bei Trouble-Ticket Systemen und Netzwerkmanagementsystemen.

Um den Aufbau von Netzwerkdokumentationssystemen besser zu strukturieren, wird ein **Schichtenmodell der Netzwerkdokumentation** entwickelt.
Ferner werden Erweiterungen in einem Netzwerkmodell vorgenommen, die es ermöglichen, aktuelle Betriebszustände in der Netzwerkdokumentation darzustellen. Durch eine Verknüpfung der aktuellen Betriebszustandsdaten aus dem Netzwerkmanagementsystem mit den Inventardaten aus dem Netzwerkdokumentationssytem ergeben sich in dieser Arbeit neue Ansätze zur **Automatisierung** der **Fehlererkennung** (Root-Cause Analyse), der **Fehlerauswirkung** (Impact-Analyse) und der **Fehlerentstörung**.

1.3 Struktur der Arbeit

Die Arbeit untergliedert sich in mehrere Teile. **Teil II** beschreibt **technische Systeme zur Unterstützung der manuellen Entstörung von Netzwerken**. **Teil III** beschreibt **technische Systeme zur automatischen Entstörung von Netzwerken**. **Teil IV** faßt die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf weiter Entwicklungsansätze.

Teil II:

Kapitel 2 untersucht die gängigen am Markt verfügbaren Softwareprodukte für Trouble-Ticket Systeme, vergleicht und klassifiziert sie. Es werden Verfahren und Methoden zusammengetragen, mit denen HelpDesk-Systeme aufgebaut und betrieben werden. Die sich hieraus ergebenden Ansätze für Automatisierungen im HelpDesk-Bereich werden dargelegt.
Kapitel 3 beschreibt praktische Probleme beim Einsatz von HelpDesk-Applikationen und unternimmt eine Grenzziehung, bis zu der der Einsatz von EDV-Systemen praktisch möglich ist.
Kapitel 4 beschreibt die Entwicklung eines zustandraumbasierten Steuerungssystems zum Informationsaustausch. Der Entwicklungsschwerpunkt liegt auf einem Steuerungs- und Regelungsverfahren für die Kommunikationsbeziehungen. Der Nachweis der Anwendbarkeit wird durch eine prototypische Implementierung einer HelpDesk-Applikation erbracht.
Kapitel 5 beschreibt die Entwicklung einer verteilten Systemarchitektur zum Informationsaustausch, mit der das Datenverkehrsaufkommen der verfügbaren Leitungsbandbreite angepaßt werden kann.
Kapitel 6 faßt die Ergebnisse von Teil II zusammen.

Teil III:

Kapitel 7 diskutiert **Ansätze zur Automatisierung des Netzwerkmanagements** in heterogenen Telekommunikationsnetzwerken. Schwerpunkt des Kapitels ist der Aufbau eines **Architekturkonzepts für ein automatisches Fehlermanagementsystem**. Ferner die Diskussion von Ansätzen zur Realisierung von Teilkomponenten der Architektur.

Teil IV:

Kapitel 8 faßt die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf weitere Entwicklungsansätze im Bereich der HelpDesk-System Technologie.

Teil II

Technische Systeme zur Unterstützung der manuellen Entstörung von Netzwerken

Kapitel 2

Analyse verfügbarer HelpDesk-Applikationen

In einem Unternehmen findet eine Vielzahl von Applikationen zur Unterstützung der Geschäftsabläufe und der Kommunikation Verwendung. Im wesentlichen sind die zur Unterstützung der Geschäftsabläufe verwendeten Applikationen in drei Gruppen von Anwendungsschwerpunkten einzuteilen, wie in Abbildung 2.1 gezeigt.

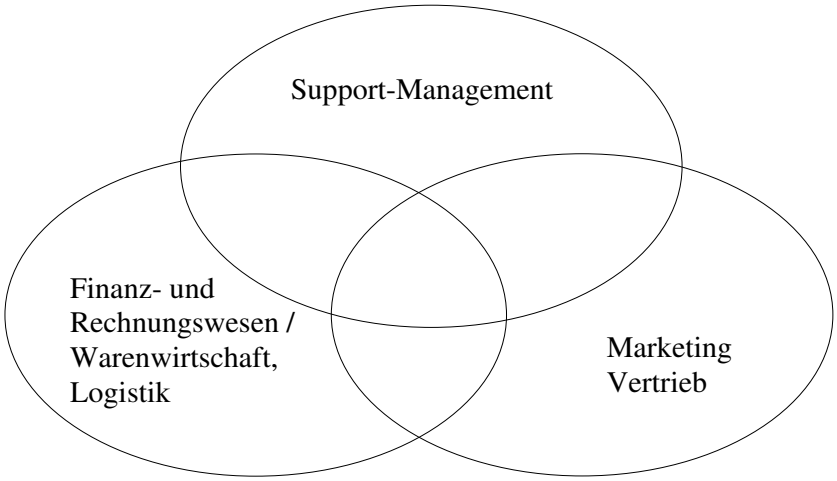


Abbildung 2.1: Anwendungsschwerpunkte von Applikationen zur Unterstützung von Geschäftsabläufen

Finanz- und Rechnungswesen / Warenwirtschaft, Logistik In diese Kategorie fallen Applikationen, die auf eine Unterstützung von Geschäftsabläufen mit überwiegend betriebswirtschaftlichem Aspekt spezialisiert sind (Lagerverwaltung, Versand, Einkauf, Finanzwirtschaft).

Vertrieb, Marketing In diese Kategorie fallen Applikationen, die eine Unterstützung bei den Kunden- und Lieferantenbeziehungen bieten. Hierzu zählen z.B. Vertriebsinformationssysteme oder Contact-Management-Systeme.

Support-Management In diese Kategorie fallen Applikationen zur Unterstützung der operativen Dienstleistungsaktivitäten. Dies schließt sowohl interne, als auch externe Dienstleistungen ein. Beispiele sind der Support für die eigenen Mitarbeiter, sowie der Support für vertriebene Produkte bzw. produktunabhängige Dienstleistungsaktivitäten (z.B. Gebäudereinigung).

In die Untersuchung der verfügbaren Applikationen wurden primär Produkte aus der Kategorie **Support-Management** einbezogen, die zur Unterstützung der Geschäftsabläufe in HelpDesk-Systemen Verwendung finden. Da die Trennung der Anwendungsschwerpunkte in Abbildung 2.1 fließende Grenzen aufweist, wurden auch Applikationen aufgenommen, die nicht im Bereich des Support-Managements liegen, aber eine Lösung angegeben haben. Als Datenquelle dienten die Produktbeschreibungen der Hersteller, die über das Internet bzw. auf Anfrage verfügbar waren. Ferner wurde die Datengrundlage durch Befragung der Hersteller nach einem standardisierten Fragebogen auf der CeBIT erweitert.

2.1 Untersuchung und Bewertung von Softwareprodukten zum Aufbau von HelpDesk-Applikationen

2.1.1 Kriterien

Um eine Bewertung der Produkte vorzunehmen werden Kriterien festgelegt, die dieser Abschnitt aufzeigt und diskutiert. Wie die Kriterien für einen speziellen Anwendungsfall zu gewichten sind, um ein geeignetes Produkt auszuwählen, hängt von den jeweiligen Anforderungen ab.

Architektur

Mit der Wahl der Architektur eines Systems werden die grundlegenden Eigenschaften des Systems festgelegt. So zeichnet sich das Peer-to-Peer Architekturmodell (P2P) durch Eigenschaften wie Robustheit, Fehlertoleranz oder Skalierbarkeit aus. Das Client-Server Architekturmodell bietet Vorteile bei der Sicherung des Datenbestands und dem Schutz vor unautorisierten Zugriffen. Alle hier untersuchten Applikationen basieren auf einer Client-Server Architektur. Andere Architekturformen wie P2P oder Agentensysteme sind bei den untersuchten Softwareprodukten nicht zu finden.

Betriebssystem

Bei der Wahl des Betriebssystems für eine HelpDesk-Applikation ist zwischen dem **Serverbetriebssystem** und dem **Clientbetriebssystem** zu unterscheiden.

Für das **Serverbetriebssystem** erlauben einige Softwareprodukte die Wahl zwischen einem Windows- und einem UNIX- bzw. Linux-System. Manche setzen explizit eine Windows- oder Unix-Plattform voraus. In den Fällen, in denen man wählen kann, sind verschiedene Aspekte zu berücksichtigen.
Eine **Windows-Plattform** ist in der Anschaffung hinsichtlich der Kosten für die Serverhardware günstiger als eine UNIX-Plattform. Allerdings ist zu beachten, daß für spezielle Features eines windowsbasierten Systems (z.B. Remote-Administration) zusätzliche Softwarekomponenten angeschafft werden müssen, die ein Unix-System bereits enthält bzw. die kostenlos im Internet verfügbar sind (z.B. GNU-Tools). Dies kann den anfänglichen Preisvorteil einer Windows-Hardware zunichte machen. Im Falle der oben angesprochenen Remote-Administration ist zu bedenken, daß ein windowsbasiertes System durch seine graphische Oberfläche deutlich mehr Bandbreite benötigt als ein konsolenorientiertes unixbasiertes System. Der Mangel an Bandbreite kann sich bei der Datenübertragung im Weitverkehrsbereich bei weltweit verteilten Servern und bei der Remote-Administration über einen Mobilfunkkanal mit 9,6 kbit/s als Hindernis erweisen. Die Nutzung von HSCSD mit bis zu 57,6 kbit/s und GPRS mit bis zu 171,2 kbit/s kann zur Verbesserung beitragen.
Nach der Installation eines Softwareupdates der HelpDesk-Applikation ist bei windowsbasierten Systemen eventuell der Neustart des Systems erforderlich. Wird hierbei der Bootvorgang einer remote administrierten Maschine nicht vollständig durchlaufen besteht die Gefahr, den Remote-Zugang zu verlieren. Ist am Aufstellungsort kein Fachpersonal vorhanden entstehen durch die Dienstreise eines Administrators Kosten. Bei unixbasierten Systemen ist die Gefahr gering, den Remote-Zugang zu verlieren, da nach einem Softwareupdate nur der Service der HelpDesk-Applikation neu gestartet werden muß.
Unixbasierte Systeme weisen eine höhere Stabilität auf als windowsbasierte Systeme. In [Ct8/2000] wird die Verfügbarkeit von Webservern unter WindowsNT und unter SunSolaris untersucht. Hierbei sind die Webserver unter WindowsNT um den Faktor 4 bei der Verfügbarkeit schlechter als die SunSolaris basierten Systeme. Bei einem Ausfall ist die Zeitdauer der Unterbrechung bei den WindowsNT-Systemen um den Faktor 2 länger als bei den SunSolaris-Systemen.
Werden große Rechenleistungen benötigt, sind unixbasierte Systeme besser nach oben skalierbar als windowsbasierte Systeme. Die Schulungskosten, die zum Aufbau des Know-hows hinsichtlich der Administration eines windows- oder unixbasierten Systems entstehen, sind erfahrungsgemäß gleich hoch.

Fazit:
Wird die HelpDesk-Applikation werktags zu den normalen Arbeitszeiten benötigt, so ist ein kostengünstiges windowsbasiertes Serversystem gut geeignet. Die Benutzeranzahl sollte zwei Stellen nicht überschreiten. Wird die HelpDesk-Applikation weltweit 24 Stunden / 365 Tage mit mehreren Server auf verschiedenen Kontinenten betrieben, so ist ein unixbasiertes Serversystem die bessere Wahl. Hier sind die höheren Anschaffungskosten vernachlässigbar gegenüber den Verlusten bei einem Systemstillstand und den Reisekosten. Durch den Einsatz von Linux ist eine Kostenreduzierung bei den Hardwarekomponenten möglich.

Die meisten der untersuchten Softwareprodukte verwenden als Benutzerinterface herstellerproprietäre Clients. Als **Clientbetriebssystem** erfordern diese ein Windows-System, manche können optional auf einem UNIX- bzw. Linux-System betrieben werden. Einige der untersuchten Produkte verwenden einen Java-Client, der die Anwesenheit einer Java Virtual Maschine auf der Client-Maschine erfordert.

Datenbank

Die Wahl des Datenbanksystems ist durch die Wahl der Serverplattform oder der HelpDesk-Applikation festgelegt, da manche Helpdesk-Produkte hier enge Vorgaben machen. So werden z.B. microsofteigene Standards wie ODBC zur Datenbankanbindung von einigen HelpDesk-Applikation verwendet. Dies geht zu Lasten der Plattformunabhängigkeit, da die in der Unix Welt existierenden Datenbankanbindungen hinsichtlich der Unterstützung von Microsoftstandards unvollständig sind. Besser ist hier eine HelpDesk-Applikation mit plattformunabhängigen Datenbankschnittstellen (wie z.B. SQL über IP). Hier bleibt das Problem der vielen SQL-Dialekte. Da viele Softwareprodukte entsprechende Anpassungen hinsichtlich der Datenbankanbindung haben, ist die Plattformunabhängigkeit und Skalierbarkeit eher möglich als bei microsofteigenen Standards. Abbildung 2.2 verdeutlicht die Aspekte, die bei der Datenbankauswahl zu berücksichtigen sind.

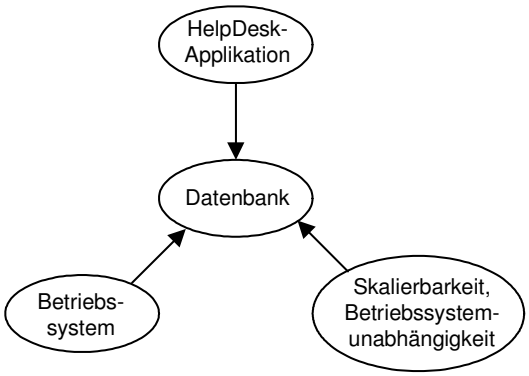


Abbildung 2.2: Aspekte zur Berücksichtigung der Datenbankauswahl

Fazit:
Möglichst die Datenbank wählen, die die meisten Kommunikationsschnittstellen bietet, eine hohe Stabilität im Betrieb aufweist und über Betriebssystemgrenzen hinweg skalierbar ist, d.h. Datenübernahme aus der Windows-Version des Datenbankservers hin zur UNIX-Version des Datenbankservers sollte problemlos möglich sein.

Webschnittstelle

Neben einem Benutzerinterface auf Basis eines herstellerproprietären Clients bieten einige Softwareprodukte ein optionales Web-Interface an. Der Funktionsumfang des Web-

Interfaces ist vielfach geringer als der des Clients, erspart aber den Installations- und Konfigurationsaufwand der Clientsoftware. Darüberhinaus gibt es Softwareprodukte, die vollständig webbasiert arbeiten. Hierbei ist zu bedenken, daß webbasierte HelpDesk-Applikationen eine geringere Performance aus Nutzersicht aufweisen als clientbasierte.

Fazit:
Möglichst die HelpDesk-Software wählen, die einen Nutzerzugang über ein Client- und Web-Interface ermöglicht. Hierbei sollte der Funktionsumfang des Web-Interfaces weit an den des Clients heranreichen.

Datenverteilung

Einige Softwareprodukte ermöglichen den Mehrserverbetrieb. Dies kann einerseits zum Aufbau eines ausfallredundanten Systems genutzt werden, andererseits ermöglicht es den Betrieb von Servern an verschiedenen Standorten (z.B. verschiedenen Kontinenten), um die Performance der HelpDesk-Applikation an den Standorten zu steigern. Abbildung 2.3 zeigt ein Beispiel mit zwei gekoppelten Servern.

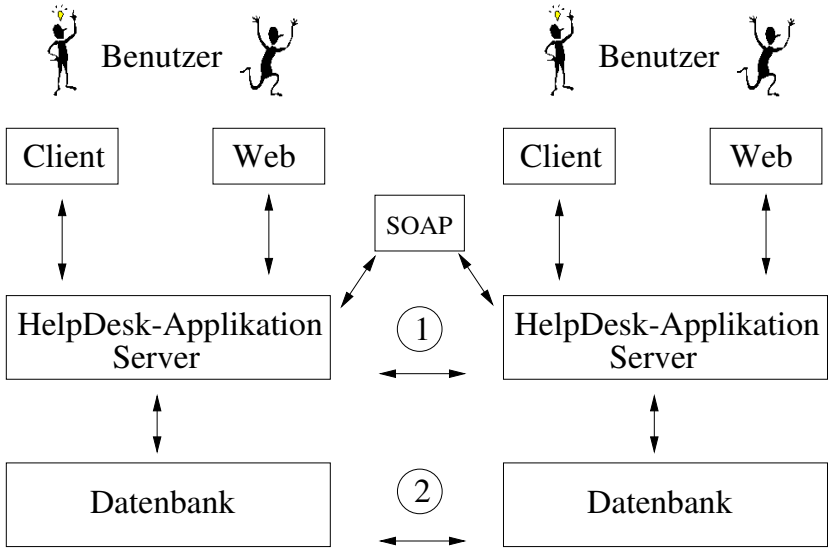


Abbildung 2.3: Optionen zur Datenverteilung in einem Mehrserverbetrieb

Das Kommunikationsinterface zum Benutzer hin wird durch einen proprietären **Client** realisiert bzw. steht ein **Web-Modul** zur Verfügung. Der **HelpDesk-Applikations Server** bildet die Kommunikationsschnittstelle für den Client und verwaltet die Daten in einer **Datenbank**.
Zur Datenübertragung bzw. zum Datenabgleich zwischen den am Verbund beteiligten Servern bieten die Helpdesk-Produkte zwei Optionen an.
Option 1 in Abbildung 2.3 gleicht die Daten über den HelpDesk-Applikations Server ab. Dieses Verfahren hat den Vorteil, daß Regelwerke erstellt werden können, die bestimmen,

wann welche Daten übertragen bzw. abgeglichen werden. Hierdurch eignet sich das Verfahren für den Datenabgleich zwischen Servern, die sich auf verschiedenen Kontinenten befinden, da über das Abgleichregelwerk auf die geringen Bandbreiten im Weitverkehrsbereich Rücksicht genommen werden kann. **Option 2** in Abbildung 2.3 gleicht die Daten auf Datenbankebene ohne die Beteiligung des HelpDesk-Applikations Servers ab. Diese Variante findet beim Aufbau ausfallredundanter Systeme bzw. bei Systemen mit Load Balancing zwischen den Servern Anwendung, da im hausinternen Bereich genügend Bandbreite zur Datenbankspiegelung zur Verfügung steht. Komplexe Datenabgleichregeln, wie sie in Option 1 verwendet werden, sind in Option 2 auch möglich, aber deren Einsatz nicht üblich.
Als weitere Möglichkeit zur Datenverteilung bieten einige HelpDesk-Applikations Server eine **SOAP**-Schnittstelle (Simple Object Access Protocol) an. Hierüber kann eine Datenverteilung zwischen HelpDesk-Applikationen unterschiedlicher Hersteller realisiert werden. Diese Art der Datenverteilung wird die o.g. zukünftig ablösen.

Fazit:
Nach Möglichkeit eine Helpdesk-Software wählen, die einen Datenabgleich nach Option 1 ermöglicht und über eine SOAP-Schnittstelle verfügt. Einerseits kann mittels dem Regelwerk im HelpDesk-Applikations Server weitgehender Einfluß auf den Datenabgleich genommen werden, andererseits können verschiedene Datenbanken verwendet werden.

Netzwerkarchitektur

Alle untersuchten Softwareprodukte benötigen eine Netzwerkinfrastruktur, die transparente Netzwerkverbindungen zwischen dem Client und dem Applikations-Server, zwischen dem Applikations-Server und der Datenbank und (beim Aufbau einer verteilten Serverarchitektur) zwischen den gekoppelten Applikations-Servern bzw. den gekoppelten Datenbanken ermöglichen. Wünschenswert ist, daß die Produkte die Netzwerkinfrastruktur minimal belasten und eine Datenkomprimierung und -verschlüsselung bieten.

Anpassung

Struktur und Workflow Ein weiterer Punkt zur Beurteilung einer HelpDesk-Applikation sind die Möglichkeiten zur Anpassung an die Erfordernisse der Supportdienstleistungsorganisation, d.h. primär an die Struktur und den Workflow. Die Systeme sind mit einem Standardworkflow bzw. einer Standardkonfiguration ausgestattet. Ferner ist zu berücksichtigen, ob das System graphisch orientiert oder skriptbasiert arbeitet, um die Struktur und den Workflow zu beschreiben. Graphisch orientierte Verfahren ermöglichen in der Regel Prozesse schneller zu erstellen bzw. abzuändern; allerdings kann die graphische Notation bei großen Implementierungen an Übersichtlichkeit verlieren, wenn Strukturierungsmöglichkeiten fehlen. Hier ist der Einsatz skriptbasierter Verfahren oftmals günstiger, da sich große Systeme damit besser strukturieren lassen.

Anpassungsmöglichkeiten durch den Benutzer Ein weiterer Bewertungspunkt ist der Einfluß des Benutzers auf das Design und das Layout der Benutzeroberfläche, da hiervon der Grad der Akzeptanz der HelpDesk-Applikation durch den Endanwender abhängt. Der Administrator sollte die Möglichkeit haben, eine Führung des Benutzers durch die Oberfläche zu erstellen. Einem Anwender, der nicht über das Administratorprivileg verfügt, sollten einfache Anpassungen seiner Oberfläche ermöglicht werden (z.B. große Schriften für Anwender mit Sehschwäche). Diese Einstellungen sollten so gespeichert werden, daß der Benutzer an jedem Client, an dem er sich anmeldet, seine persönliche Oberfläche bekommt.

Anpassungsunterstützung durch Werkzeuge Im Rahmen der Anpassung der HelpDesk-Applikation an die Supportdienstleistungsorganisation sind nicht nur Veränderungen am Aussehen, sondern auch an der logischen Struktur vorzunehmen. Diese Veränderungen werden von einer privilegierten Person vorgenommen. Hierfür enthalten die untersuchten Softwareprodukte Werkzeuge, deren Funktionalität sich in drei Gruppen einteilen läßt:

- Werkzeuge zur Abbildung der Struktur und der Geschäftsprozesse.
- Werkzeuge zur Gestaltung der Benutzeroberfläche.
- Werkzeuge zur Gestaltung der Datenbankstrukturen und zum Verwalten der Datenbestände.

Bei den Gruppen handelt es sich um eine Klassifikation der benötigten Funktionalitätsbereiche. Diese Funktionalitätsbereiche werden in einem Tool zusammengefaßt; häufig verschwimmen aber die Grenzen zwischen den Werkzeugen.

Schnittstellen und Programmierbarkeit Bei der Bewertung eines Produkts ist zu berücksichtigen welche Schnittstellen dieses bietet. Besondere Beachtung gilt dem Punkt, auf welche Art und Weise Daten in das System importiert und aus dem System exportiert werden können. Als Nachteil ist anzusehen, daß die Funktionalität, die ein Produkt enthält, nicht auf ein anderes übertragen werden kann, da es hier keine einheitlichen Schnittstellen gibt. Verwendet man Produkt A und paßt dies an seine Erfordernisse an, ist es nicht möglich auf Produkt B umzusteigen ohne die Anpassung nochmal in Produkt B vorzunehmen. Es existiert innerhalb der untersuchten Produkte kein Verfahren zur Beschreibung der Struktur und des Workflows, welches eine Portierbarkeit zwischen den Produkten ermöglichen würde.
Ferner ist irgendwann der Punkt erreicht, an dem gewünschte Funktionalitäten nicht mehr mit den verfügbaren Werkzeugen realisiert werden können. An dieser Stelle ist interessant, inwiefern eine Anpassung auf der Ebene der Programmierung erfolgen kann. Einige Produkte bieten Schnittstellen zu Programmiersprachen wie VisualBasic, Perl, C, C++ und Java. Hierdurch wird es möglich mittels selbst geschriebenen Codes die Funktionalität flexibel zu erweitern. Fehlen Schnittstellen zu Programmiersprachen können tiefgreifende Veränderungen an der Funktionalität nur vom Hersteller vorgenommen werden.

Auswertungsfunktionen und Statistiken Schließlich sind die Möglichkeiten festzustellen, statistische Auswertungen der Datenbestände zu liefern. Bei einigen Produkten sind vordefinierte Abfragen vorhanden. Interessant ist die Frage, ob eigene Auswertungen erstellt werden können. Dies ist in den Fällen wichtig, in denen neue Funktionalitäten das System erweitern und über diese Erweiterungen Statistiken zu erstellen sind.

2.1.2 Vergleich und Klassifikation

Eine Vergleichsübersicht der untersuchten HelpDesk-Applikationen zeigt Abbildung 2.4. Auf der linken Seite sind die Produkte von oben nach unten aufgetragen; unten von links nach rechts die Features, die von der Summe der Produkte unterstützt werden. In der Matrix kennzeichnet ein Punkt, daß das Produkt das Feature unterstützt bzw. enthält. Mangels Datengrundlage können nicht alle Kriterien in der Auswertung berücksichtigt werden. Daher beschränkt sich die Auswertung auf folgende Gruppen von Features:

Systemvoraussetzungen Server Diese Gruppe enthält die Systemvoraussetzung bzw. das Serverbetriebssystem.

Systemvoraussetzungen Client In dieser Gruppe sind die clientseitigen Systemvoraussetzungen aufgelistet.

Datenbank Diese Gruppe enthält die Datenbanksysteme, die von den Produkten unterstützt bzw. vorausgesetzt werden.

Web-Schnittstelle Hier ist ersichtlich, welche Benutzerinterfaces für die Anwender verfügbar sind. Möglich sind ein herstellerproprietärer Client, dies in Kombination mit einem optionalen Web-Modul oder ein ausschließlicher Web-Zugang.

Datenverteilung In dieser Gruppe wird aufgelistet, nach welchem Verfahren die Produkte in einer Mehrserver-Architektur eine Datenverteilung zwischen den am Verbund beteiligten Servern vornehmen. **Datenverteilung über Datenbank** meint, daß die Datenbanken der beteiligten Server per Replikation auf Datenbankebene konsistent gehalten werden (siehe Abbildung 2.3 (Seite 15), Option 2). **Datenverteilung über Applikation** bedeutet, daß die Applikation die Verteilung der Daten zwischen den darunterliegenden Datenbanken übernimmt (siehe Abbildung 2.3 (Seite 15), Option 1). Beide Modelle arbeiten online. **Datenverteilung über Client** sagt aus, daß die Datenbestände zu fest definierten Zeitpunkten per Ex- und Import abgeglichen werden, d.h. der Datenabgleich erfolgt nicht online. **Lastverteilung über die Applikation** bedeutet, daß in einer Mehrserverarchitektur ein Load-Balancing zwischen den beteiligten Maschinen realisiert wird, wobei jede Maschine die gleichen Services bietet. Bei einer **Standortverteilung über die Applikation** wird eine Datenübertragung zwischen den am Verbund beteiligten Servern nach einem Regelwerk vorgenommen. Dieses Feature ist Voraussetzung zum Aufbau eines weltweiten Serververbunds, da nicht alle Daten zu allen Standorten übertragen werden müssen.

Abbildung 2.4: Vergleichsübersicht der untersuchten HelpDesk-Applikationen

Anpassungsmöglichkeiten Bei den Anpassungsmöglichkeiten sind folgende Arten zu unterscheiden. **Anpassung der Datenbankstruktur** meint, daß es dem Anwender ermöglicht wird, die Datenbankstruktur nach seinen Wünschen abzuändern. **Anpassung der Clientstruktur** erlaubt es dem Endanwender, persönliche Präferenzen (wie Schriftgröße oder Farbeinstellungen) einzustellen. Ferner ist interessant, ob der Anwender den enthaltenen **Workflow** an seine Gegebenheiten anpassen kann.

Betrachtet man die Art und Weise der **technischen Realisierung**, sind bei den untersuchten Produkten **drei Tendenzen** erkennbar.
Die **erste Tendenz** ist die Bereitstellung eines Baukastensystems, aus dem der Anwender sich neben einer Basiskomponente weitere optionale Komponenten gemäß seiner Bedürfnisse auswählen kann. Die Hersteller von Baukastensystemen stellen sicher, daß die verschiedenen Komponenten zueinander passen. Allerdings handelt es sich bei den Baukastensystemen um herstellerproprietäre Lösungen, so daß Baukastensysteme verschiedener Hersteller bzw. Module daraus nicht miteinander kombiniert werden können.
Die **zweite Tendenz** besteht in ganzen Unternehmenslösungen, die auch Module mit Funktionalitäten einer HelpDesk-Applikation enthalten. Aufgrund der weitläufigen Ausrichtung dieser Produkte spielt die Funktionalität des Helpdesk-Moduls vielfach eine untergeordnete Rolle.
Die **dritte Tendenz** ist die Einbindung in am Markt weit verbreitete Systeme wie z.B. Windows und Lotus Notes. Ziel ist es, keine weitere Clientsoftware auf der Anwendermaschine zu installieren.

Die untersuchten Softwareprodukte zum Aufbau von **HelpDesk-Applikationen** lassen sich in **zwei Produktausrichtungen klassifizieren** (siehe Abbildung 2.4, Seite 19).

Out-of-the-Box Lösungen

Out-of-the-Box Lösungen sind auf bestimmte Anwendungsfälle optimierte HelpDesk-Applikationen wie z.B. den Support einer internen EDV-Struktur. Vorteil dieser Art von Produkten ist die schnelle Integrierbarkeit in die Geschäftsprozesse, wenn der Zielfokus des Produkts hierauf ausgerichtet ist. HelpDesk-Applikationen dieser Art sind für Standardanwendungsfälle in kleineren Supportdienstleistungsorganisationen mit einfachen Strukturen geeignet.

Entwicklungs- und Betriebsumgebung

Entwicklungs- und Betriebsumgebungen dienen zum Aufbau und Betrieb von individuellen HelpDesk-Applikationen. Hierdurch ist eine Adaptierbarkeit an die Struktur und den Workflow der Supportdienstleistungsorganisation gegeben. Diese Anpassung erfordert aber einen gewissen Aufwand und umfangreiche Kenntnisse in der Thematik bei den Mitarbeitern.

2.1.3 Fazit

Abschließend ist festzustellen, daß die Hersteller von HelpDesk-Applikationen zwei Marktsegmente abdecken. Dies ist zum einen der Bereich der "Kleinanwender", die wenig flexible, aber schnell integrierbare **Out-of-the-Box Lösungen** auf Windowsbasis bekommen. Auf der anderen Seite stehen die "Großanwender", deren gehobene Ansprüche nach Flexibilität und Skalierbarkeit mit **Entwicklungs- und Betriebsumgebungen** bedient werden. Am unteren Ende kommen hierfür High-End Windows-Systemen zum Einsatz. Am oberen Ende findet man UNIX-Systeme.

Die HelpDesk-Applikationen lassen sich in die Kategorien "Out-of-the-Box Lösungen" und "Entwicklungs- und Betriebsumgebung" einteilen (s.o.). Nur letztere sind im Rahmen dieser Arbeit weiter von Interesse, da sie weitgehende Adaptionsmöglichkeiten bieten.

Auffällig ist, daß alle untersuchten HelpDesk-Applikationen auf einer **Client-Server Architektur** basieren. Andere Architekturformen (z.B. Peer-to-Peer oder Agentensysteme) sind nicht zu finden.

Die Funktionalität einer HelpDesk-Applikation, die mittels einer Entwicklungs- und Betriebsumgebung erstellt wurde, kann nicht von einem Produkt A in ein Produkt B übertragen werden. Die **Portierung der Funktionalität** ist heute nicht möglich, da kein generelles Verfahren zur maschinenlesbaren Beschreibung der Funktionalität einer HelpDesk-Applikation existiert.

2.2 Eigene Anforderungen

Aus den praktischen Erfahrungen des Betriebs eines Trouble-Ticket Systems heraus muß eine HelpDesk-Applikation folgende Anforderungen erfüllen:

- Die HelpDesk-Applikation unterstützt die Abwicklung von Aufgaben in einer vorgegebenen Zeitspanne.
- Sie gewährleistet, daß Nachrichten zwischen zwei Nutzern in einer vorgegebenen Zeitspanne übertragen werden.
- Das Workflow-Regelwerk der HelpDesk-Applikation ist schnell an neue Anforderungen anpaßbar.
- Datenverluste im laufenden Betrieb treten nicht auf. Eine vollständige Datenwiederherstellung nach Systemausfällen ist sicherzustellen.
- Die Datenbestände sind für alle Nutzer jederzeit erreich- und modifizierbar.
- Suchabfragen zur Filterung des Datenbestands nach vorgegebenen Kriterien sind jederzeit möglich und führen mit kurzen Antwortzeiten zu Resultaten.

2.3 Gegenwärtige Verfahren für das Supportmanagement

Zur Darstellung eines umfassenden Bildes der gegenwärtigen Verfahren für das Supportmanagement wird einerseits auf die Literatur zu diesem Themengebiet und andererseits auf meine Erfahrung aus mehrjähriger Tätigkeit als Entwickler von HelpDesk-Systemen zurückgegriffen. Die gegenwärtigen Verfahren lassen sich unterteilen in Verfahren zur **Strukturierung eines HelpDesk-Systems** und Verfahren zur **Workflowkontrolle**.

2.3.1 Strukturierung eines HelpDesk-Systems

Nach [Czegel98] ist die typische Struktur eines HelpDesk-Systems durch eine **Dreiteilung** in **Frontline**, **Second-Level** und **Third-Level** gekennzeichnet (siehe Abbildung 2.5).

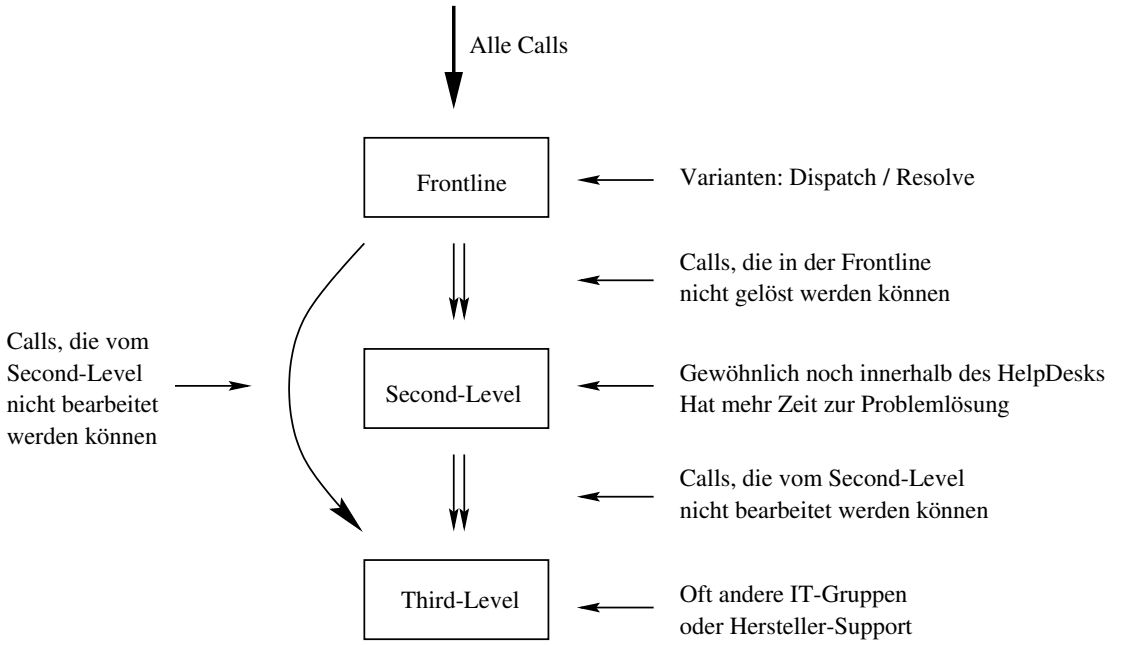


Abbildung 2.5: Typische Struktur eines HelpDesk-Systems

Frontline

Die Frontline ist die Anlaufstelle für den Nutzer des HelpDesk-Systems, d.h. hier werden die Calls entgegengenommen. Ein einzelner Kontaktpunkt bietet den Vorteil, daß der Nutzer das Problem nicht diagnostizieren muß, um zu entscheiden, welchen Teil des Supports er anruft [Bruton97]. Ferner ist die Erstellung von Statistiken an einem einzelnen Kontaktpunkt einfacher. Die Frontline kann auf zwei Arten organisiert sein. Bei einer **Dispatch-Frontline** wird der Call entgegengenommen und sofort an den Second-Level zur Bearbeitung weitergeleitet. Bei einer **Resolve-Frontline** wird der Call entgegenge-

nommen und in einer vertretbaren Zeitspanne versucht, eine Lösung für den Call zu finden. Nur wenn dies nicht möglich ist, wird der Call an den Second-Level zur Bearbeitung übergeben. Dispatch-Frontline und Resolve-Frontline werden auch als **Call-Annahme** und **First-Level Support** bezeichnet [Wald99].

Second-Level

Im Second-Level findet man Mitarbeiter mit tiefgreifenden Systemkenntnissen, deren Aufgabe die Bearbeitung der weitergeleiteten Calls ist. Hierfür steht mehr Zeit zur Verfügung als an der Resolve-Frontline bzw. dem First-Level-Support. Der Second-Level ist in mehrere Fachgebiete unterteilt, die zur Bearbeitung verschiedener Arten von Calls dienen.

Third-Level

Der Third-Level ist in der Regel nicht im gleichen Unternehmen bzw. in der gleichen Abteilung wie die Frontline und der Second-Level angesiedelt. Hier handelt es sich um externe IT-Gruppen oder den Support eines Herstellers, der von Second-Level kontaktiert wird. Der Third-Level-Support kann auch die Entwicklungsabteilung eines Unternehmens sein, wobei die vorgelagerten Level die Entwicklungsabteilung vor den Kundenanfragen abschirmen.

Eine **EDV-Unterstützung** zur **Automatisierung von Verfahren** im HelpDesk (siehe nächsten Abschnitt) sollte die drei Level abbilden. Zur Abbildung von **Fachgebieten** müssen die Mitarbeiter **Gruppen** zuzuordnen sein. Die Gruppen ihrerseits müssen dem Level zuortbar sein.

2.3.2 Verfahren zur Workflowkontrolle

Der Workflow eines HelpDesk-Systems ist durch die Arbeitsvorgänge gekennzeichnet, die zwischen der Annahme des Calls und dem Abschluß der Bearbeitung liegen. Es ist einzusehen, daß der HelpDesk nur eine **begrenzte Zahl von Calls pro Zeiteinheit** bearbeiten kann. Abbildung 2.6 zeigt nach [Czegel98] eine mögliche Verteilung der Calls in Abhängigkeit der Tageszeit.

Aus wirtschaftlichen Gründen ist der Workload des HelpDesk-Systems auf einem gleichen Niveau zu halten. Tagesspitzen können aus diesem Grund nicht sofort abgefangen werden. Ein erhöhtes Call-Aufkommen macht es erforderlich, die Bearbeitung einiger Calls zurückzustellen, wobei für diese die Reihenfolge der Bearbeitung festgelegt werden muß. Hierzu stehen verschiedene **Verfahren zur Workloadkontrolle** bereit (siehe Abbildung 2.7), die **zwei Strategien** verfolgen. Einerseits wird versucht, die Bearbeitung von Calls abzulehnen (**Call-Vermeidung**) und andererseits die Bearbeitung der angenommenen Calls voranzutreiben (**Call-Beschleunigung**).

- Verfahren zur Call-Vermeidung:
  - Vermeidung von Calls

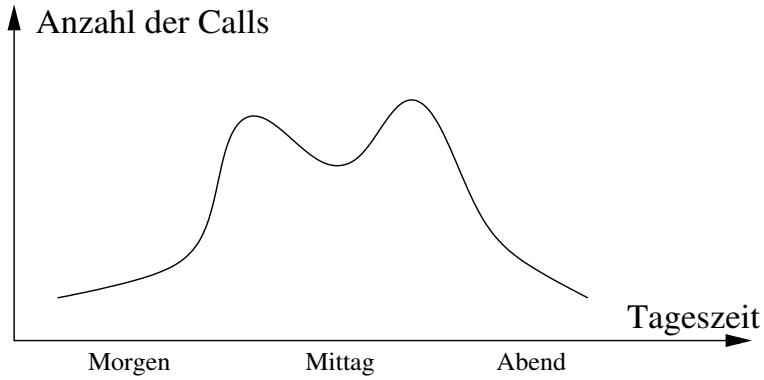


Abbildung 2.6: Typische Verteilung der Anzahl der Calls in Abhängigkeit der Tageszeit

  - Call-Annahme Verfahren
  - First-Level-Support

- Verfahren zur Call-Beschleunigung:
  - Weiterleitung von Calls
  - Priorisierung
  - Eskalation
  - Problemkontrolle

Die Verfahren stellen sich im Detail wie folgt dar:

Vermeidung von Calls

Jeder Call, der die Frontline erreicht, belegt Ressourcen. Aus diesem Grund ist der erste Schritt zur Reduktion des Workloads die Vermeidung von Calls auf die Frontline. Hierzu werden den Nutzern im Internet "Frequently Asked Questions" (FAQs), "Tips und Tricks" bzw. eine Supportdatenbank zur Verfügung gestellt, in der Antworten auf häufig gestellte Fragen oder bekannte Probleme publiziert werden [Czegel98].

Call-Annahme Verfahren

Eine weitere Möglichkeit zur Reduktion des Workloads bietet die Call-Annahme. Hier werden nur Calls zur Bearbeitung angenommen, die eine vorgegebenen Menge von Kriterien erfüllen. Gründe für die Ablehnung sind z.B.:

- Der Call liegt nicht im Zuständigkeitsbereich des HelpDesk-Systems.
- Der Caller konnte nicht identifiziert werden.
- Die Kostendeckung für die Bearbeitung des Calls ist nicht vorhanden.

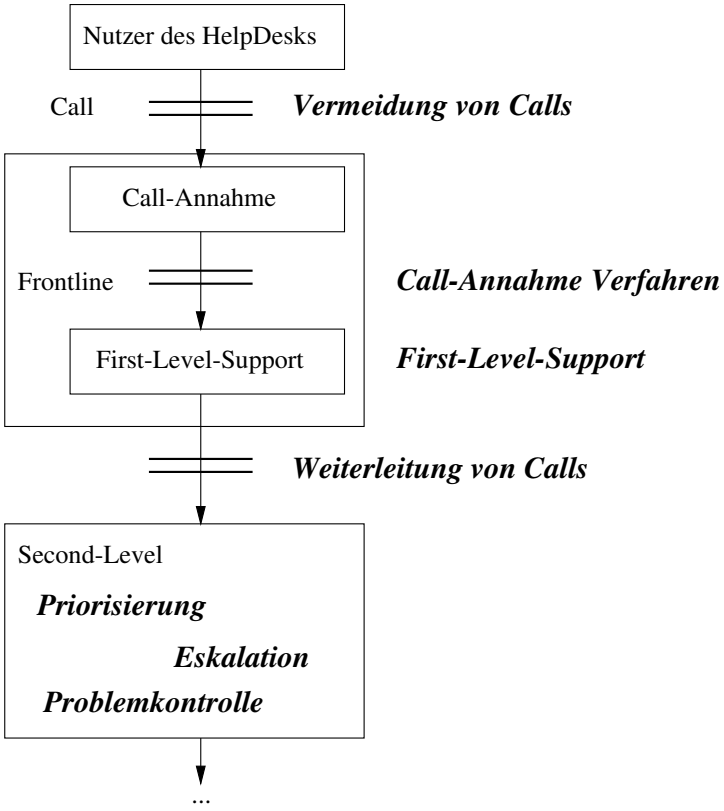


Abbildung 2.7: Verfahren zur Workloadkontrolle

- Eine Lösung für den Call ist in den FAQs (s.o.) publiziert.

Die Entscheidung für oder gegen eine Call-Annahme könnte unter Berücksichtigung o.g. Kriterien **automatisiert** von einer **EDV-Unterstützung** getroffen werden.

First-Level-Support

Ziel des First-Level-Supports ist es, durch einen Lösungsversuch den Call abschließend zu bearbeiten. Hierdurch wird eine Reduktion der Call-Anzahl erreicht, die an den Second-Level weitergeleitet werden.

Weiterleitung von Calls

Kann ein Call an der Frontline bzw. dem First-Level Support nicht gelöst werden, so erfolgt die Weiterleitung an den Second-Level. Hierbei ist zu berücksichten, daß der Mitarbeiter einerseits über das notwendige Fachwissen verfügt und andererseits Zeit zur termingerechten Bearbeitung hat.

Nach [Wald99] stehen als Kriterien für die **Weiterleitungs-Entscheidung** nur die Daten aus der Call-Annahme und des First-Level-Supports zur Verfügung. Auf Basis dieser Informationen erfolgt eine **Klassifikation** der Calls, die Grundlage für die Weiterleitungs-Entscheidung ist.
Die Qualität der Weiterleitungs-Entscheidung hat einen starken Einfluß auf die Geschwindigkeit der Call-Bearbeitung, da aus folgenden Gründen bei falscher Entscheidung die Bearbeitung des Calls länger dauert:

- Ein falsch weitergeleiteter Call wird von dem Mitarbeiter, der kein Spezialist für diese Art von Calls ist, in der Regel erkannt. Er wird diesen Call aufgrund praktischer Erfahrung eher bearbeiten als zurückweisen. Durch den Mangel an Know-how entsteht bei der Bearbeitung ein Zeitverzug.

- Fehlgeleitete Calls werden spätestens dann erkannt, wenn Wächter anschlagen, die die Bearbeitungszeit überwachen (siehe Abschnitt *Eskalationen*, Seite 30). Die Wächter alarmieren Mitarbeiter, die den fehlgeleiteten Call richtig weiterleiten. Der entstandene Zeitverlust ist nicht mehr einzuholen.

Zu kritisieren ist, daß die Qualität der Weiterleitungs-Entscheidung im hohen Maß von weichen Faktoren (beispielsweise dem Wissensstand und der Tagesform der Mitarbeiter in der Frontline) abhängig ist. Da die Weiterleitungs-Entscheidung einen starken Einfluß auf die Geschwindigkeit der Call-Bearbeitung hat, sollte die Qualität nicht von weichen Faktoren abhängen. Wünschenswert ist ein hohes und gleichbleibendes Qualitätsniveau.

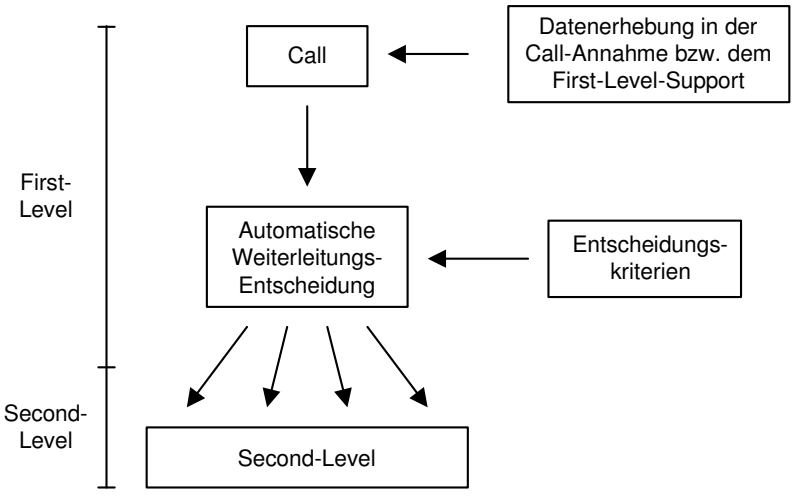


Abbildung 2.8: Forderung nach einer Automatisierung der Weiterleitungs-Entscheidung

Dies kann mit einer **Automatisierung der Weiterleitungs-Entscheidung** erreicht werden (siehe Abbildung 2.8). Ausgehend von den Daten der Call-Annahme und dem First-Level-Support können auf Basis vorgegebener Kriterien Weiterleitungs-Entscheidungen

getroffen werden, die eine gleichbleibende Qualität aufweisen. Eine Automatisierung der Weiterleitungs-Entscheidung bietet den Vorteil, daß die Entscheidungskriterien schnell und kurzfristig änderbar sind, was dem HelpDesk-System eine schnelle Reaktion auf sich ändernde Nutzeranforderungen ermöglicht.

Priorisierung

In der **Frontline** werden die Calls in der Reihenfolge des Eingangs bearbeitet. Nach der Weiterleitung der Calls zum **Second-Level** ist man nicht mehr gezwungen diese Reihenfolge einzuhalten. Die Calls werden priorisiert und in der Reihenfolge ihrer Priorität bearbeitet. Bei der Priorisierung wird nach [Bruton97] zwischen der **Problem-Priorisierung** und der **Nutzer-Priorisierung** unterschieden (vgl. Abbildung 2.9):

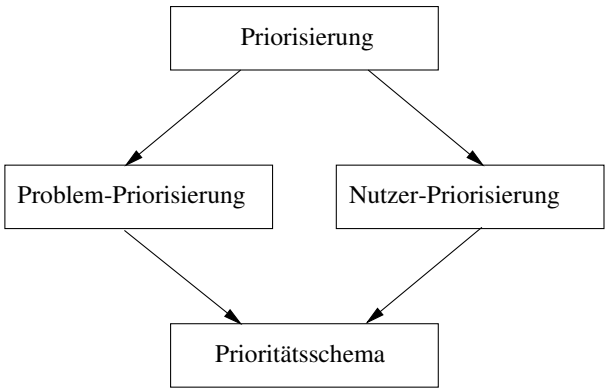


Abbildung 2.9: Priorisierung von Calls

- **Problem-Priorisierung**
  Nach [Czegel98] definiert sich die Problem-Priorisierung durch die Auswirkung eines Problems auf den Geschäftsablauf. Hierdurch werden die Calls monetär bewertet, so daß der mit dem höchsten Wert die höchste Priorität bekommt.

- **Nutzer-Priorisierung**
  Mit der Nutzer-Priorisierung wird die Bedeutung eines Nutzers für das HelpDesk-System berücksichtigt. Als Beispiele sind hier der Anteil eines Nutzers am Umsatz des HelpDesk-Systems oder der Einfluß von ausgezeichneten Einzelpersonen auf das HelpDesk-System zu nennen.

Als Ansatz für ein Prioritätsschema auf Basis der Problem- und Nutzerpriorisierung bietet sich eine Portfoliedarstellung an (vgl. Abbildung 2.9).

Nach [Czegel98] ist zu empfehlen, die Calls in verschiedene Kategorien (d.h. Problem-Kategorien und Nutzer-Kategorien) zu klassifizieren (vgl. Abschnitt *Weiterleitung von Calls*, Seite 25ff) und für jede Kategorie ein eigenes Prioritätsschema zu erstellen. Aufgrund

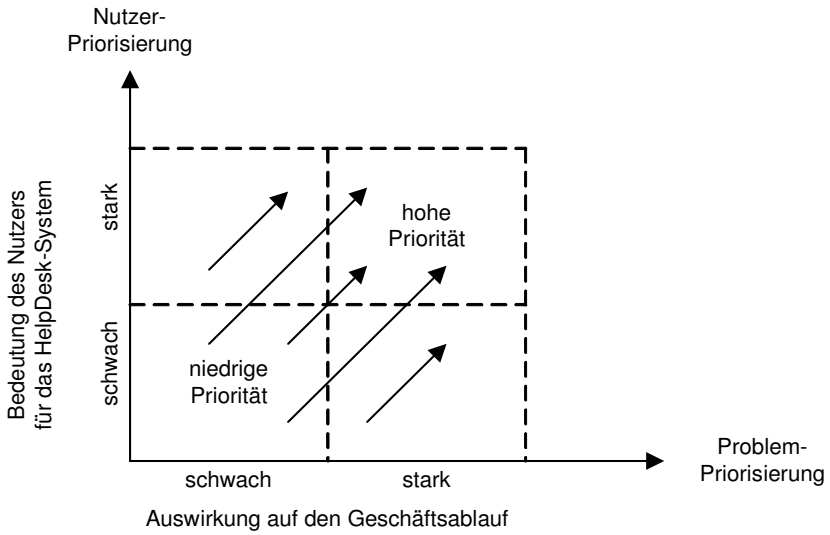


Abbildung 2.10: Ansatz für eine Priorisierung auf Basis einer Portfoliodarstellung

praktischer Erfahrung sollte die Priorisierung auf max. fünf Prioritätsstufen beschränkt werden. Nachfolgend wird für diese Vorgehensweise das Beispiel in [Czegel98] wiedergegeben, das den Support einer internen EDV-Infrastruktur zum Ziel hat. Zur Einteilung der Calls, die im Second-Level zu bearbeiten sind, werden nachfolgende **Kategorien** definiert:

Problem
: Ein Problem ist eine Unterbrechung bzw. eine Beeinträchtigung des Geschäftsprozesses, das seine Ursache in der Hardware oder Software hat.

Arbeit
: Geplante oder ungeplante Arbeit zur Beseitigung ernster Probleme bzw. ungeplante Arbeit zur Lösung wiederkehrender Probleme, die die Ressourcen zu stark beanspruchen.

Fragen
: Fragen, die an den Support gestellt werden und sich mit der Bedienung und der Verfügbarkeit von Software- und/oder Hardwarekomponenten im Netzwerk beschäftigen.

Anforderungen
: Anforderungen sind die Inanspruchnahme von Serviceleistungen des Supports. Dies sind z.B. Unterweisungen in der Bedienung von Software-Produkten oder der Aufbau neuer PC-Arbeitsplätze.

Ein aus dieser Einteilung resultierendes Prioritätsschema ist in Tabelle 2.1 wiedergegeben.

Priorität	Problem	Arbeit	Fragen	Anforderungen
1	Kritische Komponenten ausgefallen; Auswirkung auf den Geschäftsablauf.	—	—	—
2	Kritische Komponenten schwach; Auswirkung auf den Geschäftsablauf.	—	Kritische Nutzer können nicht arbeiten; Auswirkung auf den Geschäftsablauf	Notfall-Anforderung; Auswirkung auf den Geschäftsablauf.
3	Einige, nichtkritische Komponenten ausgefallen oder schwach; keine Auswirkungen auf den Geschäftsablauf	Geplante oder ungeplante Arbeit zur Beseitigung ernster Probleme bzw. ungeplante Arbeit zur Lösung wiederkehrender Probleme, die die HelpDesk Resourcen zu stark beanspruchen.	—	Alle anderen Anforderungen, die der HelpDesk in diesem Bereich unterstützt.
4	Einzelne, nichtkritische Komponenten defekt oder schwach. Keine Auswirkung auf den Geschäftsablauf.	—	—	—
5	Kleine oder unwichtige Probleme, die kosmetischen Charakter haben.	—	Alles andere	—

Tabelle 2.1: Beispielhaftes Prioritätsschema für ein HelpDesk-System zum Support einer internen EDV-Infrastruktur (nach [Czegel98])

Generell ist eine **Automatisierung der Priorisierung** anzustreben. Voraussetzung hierzu ist die Erstellung eines Prioritätsschemas, vergleichbar mit dem Beispiel in Tabelle 2.1, welches auf Basis der Daten eines Calls diesem eine Priorität zuordnet. Hierdurch wird einerseits eine konsistente und für den Nutzer nachvollziehbare Priorisierung gewährleistet. Andererseits können Änderungen im Prioritätsschema schnell in die Praxis umgesetzt werden.

Eskalationen

Eine Eskalation bezeichnet die **Erhöhung der Dringlichkeit** der Bearbeitung eines Calls. Dies erfolgt zu einem Zeitpunkt (**Eskalationszeitpunkt**) an dem absehbar ist, daß mit den aktuellen Ressourcen die Bearbeitung des Calls nicht zum vorgegebenen Termin abgeschlossen werden kann. Als Ereignis zum Auslösen einer Eskalation wird fast ausschließlich die Überschreitung eines Zeitlimits bei der Bearbeitung des Calls verwendet. Die Wahl des Zeitlimits erfolgt in Abhängigkeit der Priorität des Calls, d.h. je höher die Priorität, desto kürzer das Zeitlimit.
Eskalationen dienen einerseits der Abwendung von negativen Auswirkungen der Terminüberschreitung (z.B. Konventionalstrafe). Andererseits dienen sie der Beschleunigung der Call-Bearbeitung und somit der Reduktion der Calls, die im HelpDesk bearbeitet werden. Man unterscheidet **zwei Arten** von Eskalationen:

- Bei der **Lösungseskalation** wird die Beschleunigung der Call-Bearbeitung durch einen Zuwachs an Know-how erreicht. Die Bearbeitung des Calls wird von einem Mitarbeiter mit niedrigem Skill-Level zu einem mit hohem Skill-Level übertagen.
- Die **Autoritäteneskalation** dient zur Information von Mitarbeitern mit Leitungsfunktionen. Die Beschleunigung der Call-Bearbeitung wird durch Managementmaßnahmen erreicht.

Calls können mehrfach eskalieren. Die Eskalationsanzahl wird von einem Eskalationszähler dokumentiert. Der Eskalationszähler wird auch **Eskalationsstufe** genannt, d.h. ein Call der n mal eskaliert ist befindet sich auf der Eskalationsstufe n (siehe Abbildung 2.11).
Für den Bereich der Eskalationen ist eine **Automatisierung des Eskalationsmanagements** anzustreben, welche o.g. Eskalationsarten berücksichtigt. Ein EDV-unterstütztes Eskalationsmanagement könnte sicherstellen, daß die vorgegebenen Eskalationsverfahren wie geplant ablaufen und alle an einem Vorgang beteiligten Personen über diesen informiert sind.

Problemkontrolle

Der maximale Workload eines HelpDesk-Systems ist aufgrund der verfügbaren Ressourcen begrenzt. In einer Überlastsituation können Calls nicht termingerecht bearbeitet werden. Nach [Czegel98] und [Czegel99] besteht die Gefahr, in den **Out-Of-Control-Zyklus** zu geraten (siehe Abbildung 2.12), der durch eine Rückkoppelung eine ständige Überlastsituation herbeiführt. Dies bewirkt ein **Zusammenbruch des Aufgabenmanagements**, was

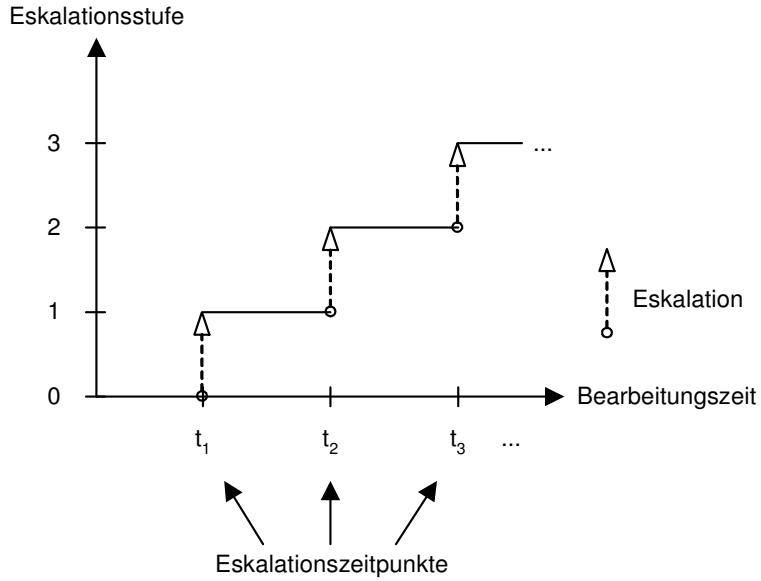


Abbildung 2.11: Eskalationsstufe eines Calls in Abhängigkeit der Bearbeitungszeit

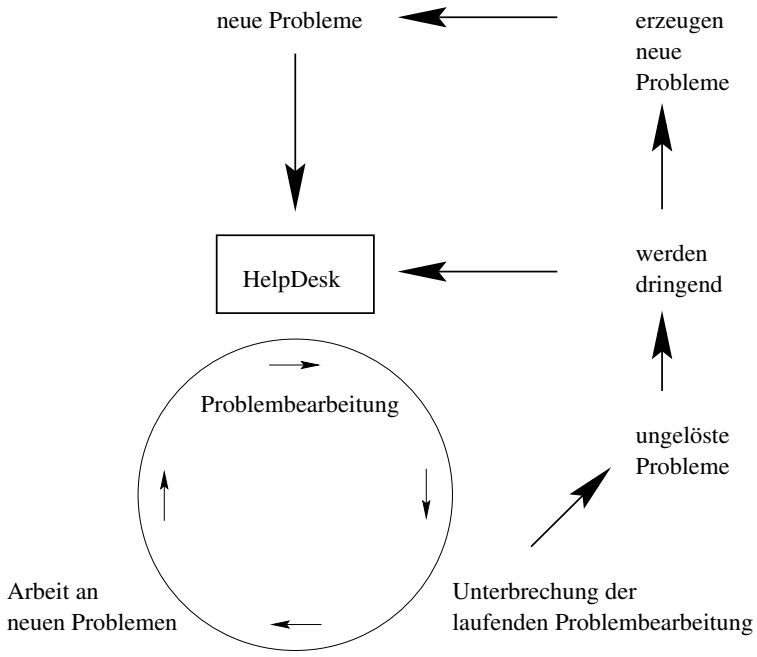


Abbildung 2.12: Out-Of-Control-Zyklus [Czegel98][Czegel99]

die termingerechte Bearbeitung von Calls unmöglich macht. **Ziel der Problemkontrolle** ist die Vermeidung des Out-Of-Control-Zykluses.

Der Out-Of-Control-Zyklus entsteht, wenn bei einer hohen Call-Anzahl im Second Level weiterhin viele Calls von der Frontline in den Second Level weitergeleitet werden. Die Entscheidung über die Bearbeitungsreihenfolge wird mittels des Prioritätsschemas (vgl. Abschnitt *Priorisierung*, Seite 27) getroffen. Hierbei kann die Bearbeitung niedrig priorisierter Calls durch später eintreffende höher priorisierte Calls unterbrochen werden. Dies führt dazu, daß die Menge unbearbeiteter Calls größer wird. Die ungelösten Probleme werden mit der Zeit dringend und verursachen neue Probleme, die neue Calls erzeugen. Durch diese Rückkoppelung wird der hohe Workload zusätzlich erhöht, was zum Zusammenbruch des Aufgabenmanagements führt.

Um die Kontrolle über den Workload wiederherzustellen, muß die Anzahl der Calls kurzfristig reduziert werden. Dies darf nicht durch Verwerfen von Calls geschehen. Die Call-Bearbeitung muß im Rahmen der mit den Nutzern vereinbarten SLAs (Service Level Agreements) erfolgen, um negative Auswirkungen beim Bruch der SLA für den HelpDesk zu vermeiden (z.B. Konventionalstrafe). Hierzu wird das Verfahren der **geschichteten Problemkontrolle** angewendet.
Abbildung 2.13 zeigt die prinzipielle Vorgehensweise, um für einen Call eine Lösungsstrategie zu entwickeln. Ausgehend von der Problembeschreibung bzw. dem Call erfolgt eine **Ursachenidentifikation**, die die Basis für eine **Lösungsstrategieentwicklung** ist. Nach [Czegel99] können beide Vorgänge auf verschiedenen Ebenen (engl. Level) abgewickelt werden, wobei die Level ein Maß für den Bearbeitungsaufwand sind. Die Ursachenidentifikation kann auf dem **Surface-Level** (oberflächlich), dem **Mid-Level** (mittlere Stufe) oder dem **Root-Level** (Wurzel-Stufe) erfolgen. Die Lösungsstrategieentwicklung kann auf den Leveln **Kurzzeit-Strategie**, **mittelfristige Strategie** oder **Langzeit-Strategie** vorgenommen werden. Abbildung 2.14 zeigt den Zusammenhang zwischen dem Ursachenidentifikations-Level und dem Lösungsstrategie-Level.

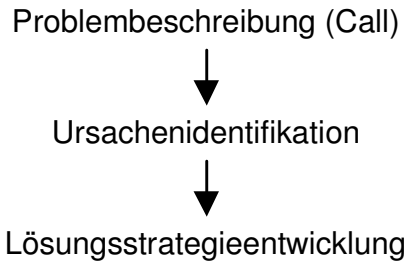


Abbildung 2.13: Prinzipielle Vorgehensweise zur Findung einer Lösungsstrategie

Im Normalbetrieb wird die Ursachenidentifikation auf dem Root-Level vorgenommen, da nur eine Langzeit-Lösungsstrategie eine dauerhafte Problembeseitigung sicherstellt.
Um die Call-Anzahl im HelpDesk zu verringern kann die Ursachenidentifikation auf Mid- oder Surface-Level betrieben werden. Durch diese Reduzierung des Bearbeitungsaufwands

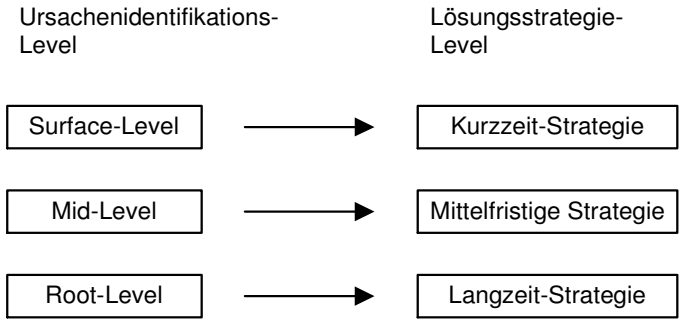


Abbildung 2.14: Zusammenhang zwischen dem Ursachenidentifikations-Level und dem Lösungsstrategie-Level

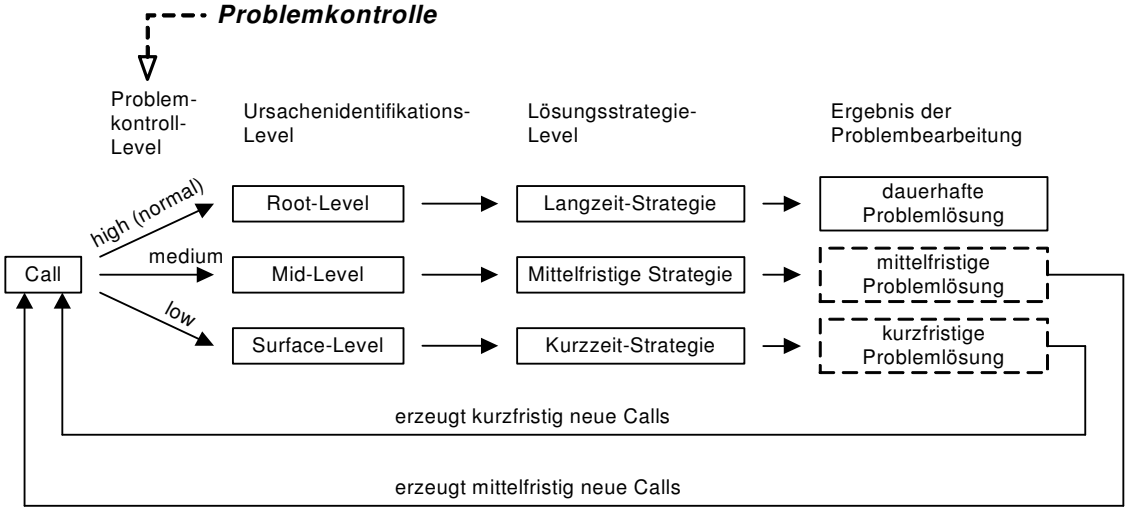


Abbildung 2.15: Funktionsweise der Problemkontrolle

läßt sich die Call-Bearbeitung beschleunigen. Der Vorteil der geschichteten Problemkontrolle (Veränderung des Problemkontroll-Levels, siehe Abbildung 2.15) liegt in der schnellen Reduktion der Call-Anzahl ohne die vereinbarten SLAs zu brechen. Allerdings erzeugen kurz- und mittelfristige Problemlösungen kurz- und mittelfristig neue Calls.

Fazit:
Die Problemkontrollstrategie dient zum **Abfangen von Workloadspitzen**, da sie keine dauerhafte Problemlösung schafft. Sie verschiebt die Problembearbeitung auf einen späteren Zeitpunkt, ohne die SLAs gegenüber den Nutzern zu brechen.

Eine **Automatisierung der geschichteten Problemkontrolle** wird schwer realisierbar sein, da die Festlegung der Ursachenidentifikations- und Lösungsstrategie-Level von der Problembeschreibung (Call) abhängig ist. Somit müssen die Levels für jeden Call indivi-

duell festgelegt werden. Diese Festlegung der Levels erfordert heute viel Berufserfahrung. Einfacher wäre eine Festlegung der Levels für Problemklassen. Da die Calls bei der Weiterleitung von der Frontline zum Second-Level zum Treffen der Weiterleitungs-Entscheidung klassifiziert werden (vgl. Abschnitt *Weiterleitung von Calls*, Seite 25), ist hier eine Festlegung von Ursachenidentifikations- und Lösungsstrategieleveln für jede Problemklasse möglich, was Basis für einen Automatisierungsansatz ist.

2.3.3 Fazit

Die gegenwärtigen **Verfahren für das Supportmanagement** unterteilen sich in Verfahren zur Strukturierung des HelpDesks und Verfahren zur Workflowkontrolle.
Die **Strukturierung** ist durch eine **Dreiteilung** in **Frontline**, **Second-Level** und **Third-Level** gekennzeichnet.
Die **Verfahren zur Workflowkontrolle** sind auf eine **Reduktion des Call-Bearbeitungsaufwands** ausgelegt. Sie unterteilen sich in **Verfahren zur Vermeidung der Call-Bearbeitung** und **Verfahren zur Beschleunigung der Call-Bearbeitung**.
Eine **EDV-technische Unterstützung der Verfahren** für das Supportmanagement ist anzustreben, wobei jedes Verfahren verschiedene **Ansatzmöglichkeiten für eine Automatisierung** (wie oben gezeigt) bietet.

2.4 Schwachstellen heutiger HelpDesk-Applikationen

Ein wichtiger Aspekt bei der Betrachtung der Schwachstellen heutiger HelpDesk-Applikationen ist, daß es kein generisches Modell zur Beschreibung der Struktur und der Funktionsweise einer HelpDesk-Applikation gibt. Für Beschreibungen dieser Art stehen nur die herstellerproprietären Applikationsmodelle der jeweiligen Entwicklungs- und Betriebsumgebung zur Verfügung. Da diese zueinander inkompatibel sind, erfordert die Portierung einer HelpDesk-Applikation von einem Produkt zu einem anderen neben der Neuimplementierung auch einen neuen Entwurf des Applikationsmodells. Ein generisches Modell, welches die Struktur und die Funktionsweise einer HelpDesk-Applikation allgemein beschreibt und zur Erstellung eines Applikationsmodells nutzbar ist, könnte letzteres verhindern.
Betrachtet man die Adaptionsgeschwindigkeit, mit der sich neue Anforderungen an eine HelpDesk-Applikation realisieren lassen, so kann diese gering sein, da die produktspezifischen Applikationsmodelle nur Modellierungen auf rechnernaher Ebene erlauben, die Anforderungen aber auf Geschäftslogikebene gestellt werden. Ein generisches Modell, welches einerseits eine Modellierung auf Geschäftslogikebene ermöglicht und andererseits die Erstellung eines rechnernahen produktspezifischen Applikationsmodells erleichtert, wäre von Vorteil.

Kapitel 3

Praktische Probleme beim Einsatz von HelpDesk-Applikationen

Ein HelpDesk-System stellt seinen Nutzern Materialien und angewandtes Wissen in einem begrenzten Wissensgebiet (Fokus) zur Verfügung. Die internen Arbeitsabläufe sind beim Einsatz von HelpDesk-Applikationen zur Unterstützung der Geschäftsprozesse durch den Dualismus Mensch – Maschine geprägt. Aus diesem Grund ist eine ganzheitliche Betrachtung, also die soziale und technische Sicht auf ein HelpDesk-System notwendig, um dessen Funktionsweise zu verstehen. Vor diesem Hintergrund ist folgende Frage hinsichtlich des Einsatzes von HelpDesk-Applikationen in HelpDesk-Systemen interessant.
Wo liegen die Grenzen der Einsatzmöglichkeiten von HelpDesk-Applikationen bzw. was kann mit ihnen maximal erreicht werden?

Zur Beantwortung der Frage nach den Grenzen der Einsatzmöglichkeiten von HelpDesk-Applikationen betrachtet man die Definition der Begriffe “Signal”, “Zeichen”, “Daten”, “Nachricht”, “Information” und “Wissen” (siehe Abbildung 3.1).

Signal
: Die Übertragung oder Speicherung von Zeichen geschieht unter Verwendung geeigneter physikalischer Signale (Spannungssignale, Stromsignale, Lichtsignale), die an das jeweilige physikalische Übertragungs- bzw. Speichermedium angepaßt sind. Die Umsetzung von Zeichen in ein Signal erfolgt durch das Codealphabet der verwendeten Codierungsmethode.

Zeichen
: Zeichen dienen zur Kommunikation zwischen einer Quelle und einer Senke. Der Zeichenvorrat, d.h. die Menge der Zeichen, die zur Kommunikation verwendet werden, bezeichnet man als Alphabet.

Daten
: Daten dienen zur Darstellung von Nachrichten und bestehen aus einer bestimmten

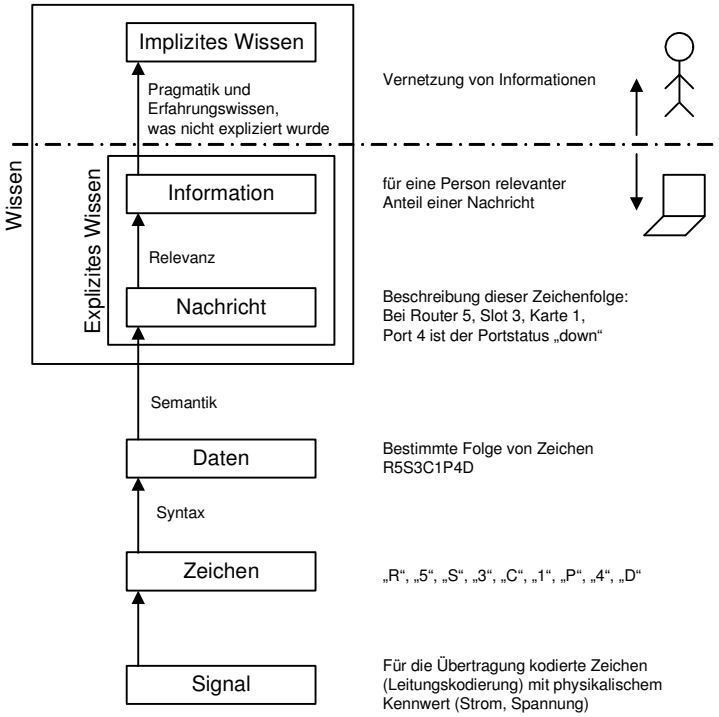


Abbildung 3.1: Zusammenhang zwischen den Begriffen “Signal”, “Zeichen”, “Daten”, “Nachricht”, “Information” und “Wissen”.

Folge von Zeichen. Der Aufbau der Zeichenfolge unterliegt formalen Regeln, die Syntax genannt werden.

Nachricht
: Eine Nachricht ergibt sich aus der Beschreibung einer Zeichenfolge. Die Beschreibung der Bedeutung einer Zeichenfolge geschieht mittels der Semantik.

Information und Wissen
: Eine Information ist der für eine Person relevante Anteil einer Nachricht. Eine Beschreibung zur Abgrenzung der Begriffe “Information” und “Wissen” findet man in [Stelzer03], die hier zitiert wird:

> Eine Möglichkeit, die Begriffe Wissen und Information voneinander abzugrenzen, besteht darin, den Begriff Wissen anhand des Explikationsgrades in explizites und implizites Wissen zu unterteilen. Explizites Wissen läßt sich speichern, verarbeiten und übertragen. Implizites Wissen muß abgebildet werden, bevor es einer Verarbeitung zugänglich wird. Während des Prozesses der Explizierung wird dem impliziten Wissen eine Form bzw. Gestalt gegeben, indem das implizite Wissen z. B. mit Hilfe einer natürlichen oder einer maschinenverarbeitbaren Sprache ausgedrückt wird. Dieser

Prozeß der Abbildung, des “In-eine-Form-Gießens” bzw. der Gestaltgebung, wird im Lateinischen mit “informare” bezeichnet. Der Begriff “Informatio” bezeichnet demnach ein Abbild, eine in eine (sprachliche) Form gegossene bzw. explizierte Vorstellung, einen Begriff. Demnach ist Information eine Teilmenge des Wissens. Information bezeichnet den expliziten Teil des Wissens. Dieser Zusammenhang ist in Abbildung 3.2 dargestellt ([Stelzer03], Seite 8).

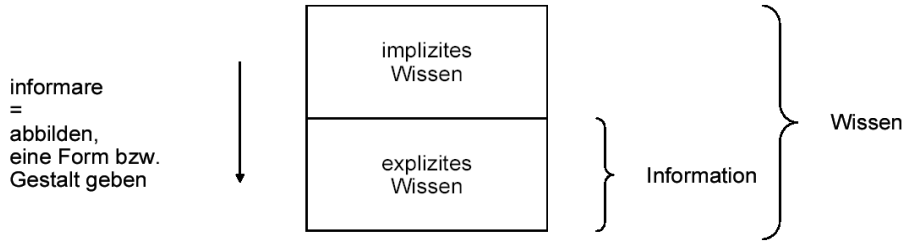


Abbildung 3.2: Information und implizites Wissen (aus [Stelzer03], Seite 9)

Betrachtet man die Beziehungen zwischen den Begriffen “Signal”, “Zeichen”, “Daten”, “Nachricht”, “Information” und “Wissen” nach Abbildung 3.1, so läßt sich eine **Grenze** ziehen, bis zu der der Einsatz von EDV-Systemen (HelpDesk-Applikationen) zur Unterstützung der Geschäftsprozesse in einem HelpDesk-System sinnvoll ist. Diese **Grenze liegt zwischen dem impliziten Wissen und dem expliziten Wissen**.
Die direkte Nutzung des **impliziten Wissens** ist für ein EDV-System nicht möglich. Implizites Wissen bedarf der Explizierung, damit es vom EDV-System genutzt werden kann. Der Vorgang der Explizierung kann durch das EDV-System unterstützt werden. Als Beispiel können strukturierte Eingabemasken (sog. Wizards) angeführt werden, die den Nutzer Schritt für Schritt abfragen. Hierbei kann das EDV-System Hilfen zur Beantwortung der Fragen geben.
Explizites Wissen kann theoretisch in vielfältiger Form durch ein EDV-System genutzt werden. Praktisch unterliegt dies jedoch Einschränkungen, da für viele Anwendungsfälle keine Verfahren zur Wissensnutzung in bezug auf den Anwendungsfall existieren.
Festzustellen ist, daß EDV-Systeme mit Techniken zur Speicherung, Übersetzung und Verarbeitung von Nachrichten und Informationen (d.h. explizites Wissen) helfen können. Festzustellen ist auch, daß zur Bearbeitung einer Störung bzw. zur Findung einer Lösung für ein Netzwerkproblem das implizite Wissen eines Operators erforderlich ist. Aus diesem Grund liegt der **Schwerpunkt bei der Entwicklung von HelpDesk-Applikationen** auf der Verwaltung der Artefakte mit explizitem Wissen (= Informationen) und der bedarfsgerechten Bereitstellung für die Mitarbeiter des HelpDesk-Systems. Ferner sollte die HelpDesk-Applikation den Informationsaustausch (d.h. Austausch von Artefakten mit explizitem Wissen) unter den Mitarbeitern durch Steuerung, Regelung und Überwachung

der Informationsströme unterstützen, so daß diese über die bestmögliche Informationslage im Hinblick auf Ihre Tätigkeit verfügen. Im speziellen sind bei der Weiterentwicklung von HelpDesk-Applikationen folgende Punkte zu berücksichtigen:

- Die HelpDesk-Applikation sollte unabhängig vom Fokus des HelpDesk-Systems eingesetzt werden können, d.h. fokusspezifische Sachverhalte sind Bestandteil der Parametrisierung.
- Die HelpDesk-Applikation sollte über eine hohe Flexibilität hinsichtlich der Anpassung an die Struktur und den Workflow des HelpDesk-Systems verfügen.

Vor diesem Hintergrund stellen die zwei nachfolgenden Kapitel **Konzepte zum Aufbau von HelpDesk-Applikationen** vor, die einerseits als **Informationsverwaltungssystem** und andererseits als **Informationsaustauschsystem** für die Mitarbeiter des HelpDesks bereit stehen. Eines der vorgestellten Konzepte ist exemplarisch implementiert worden, um praktische Erfahrung mit diesem Ansatz zu sammeln.

Kapitel 4

Entwicklung eines zustandsraumbasierten Steuerungssystems zum Informationsaustausch

4.1 Ansatz zur Konzeptentwicklung

Abbildung 4.1 zeigt den Ansatz zur Konzeptentwicklung. Die Hersteller von **Entwicklungs- und Betriebsumgebungen** zur Erstellung einer **HelpDesk-Applikation** (vgl. Abschnitt 2.1.3, Seite 21) stellen dem Anwender ein Modell zur Verfügung. Mit diesem Modell beschreibt der Anwender seine Applikation und implementiert diese. Man nennt das vorgegebene Modell **Applikationsmodell**.

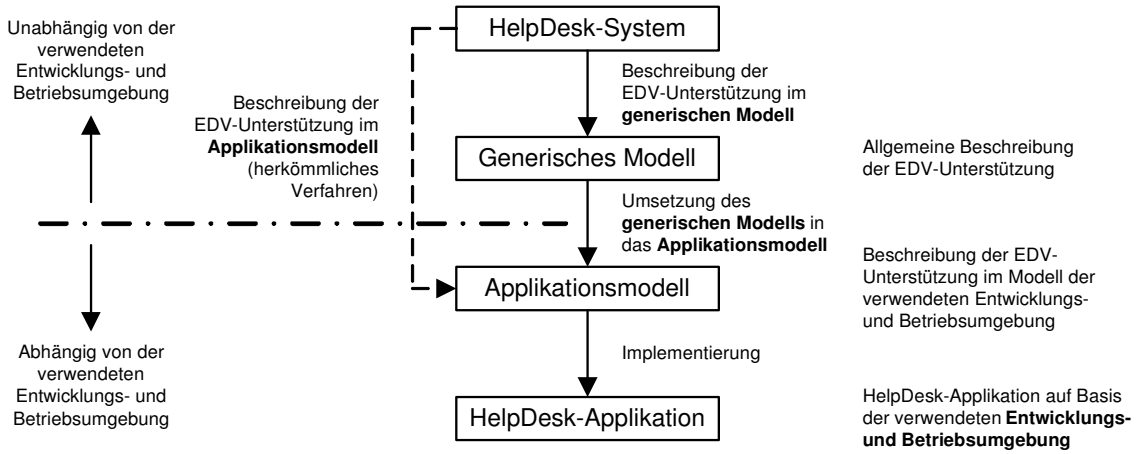


Abbildung 4.1: Ansatz zur Konzeptentwicklung

Die Beschreibung der HelpDesk-Applikation im Applikationsmodell hat den Nachteil, daß sie herstellerabhängig ist. Auch eignen sich die verfügbaren Applikationsmodelle nur schwer zur Beschreibung eines Informationsverwaltungs- und Informationsaustauschsystems (vgl. Kapitel 3), da die Modellierungsmöglichkeiten auf einer zu niedrigen Abstraktionsebene liegen.
Abhilfe schafft die Einführung eines **generischen Modells**, mit dem eine herstellerunabhängige Beschreibung der EDV-Unterstützung für ein HelpDesk-System möglich wird. **Schwerpunkt des Kapitels ist der Entwurf eines generischen Modells** zur Beschreibung einer HelpDesk-Applikation für die Informationsverwaltung und -verteilung im HelpDesk-System. Dieses gliedert sich in folgende Konzepte auf.

4.2 Daten- und Informationskonzept

Jeder Supportbereich eines Unternehmens muß Daten und Informationen im HelpDesk zur Abwicklung der Geschäftsprozesse vorhalten. Das generische Modell kennt zur Modellierung von Daten und Informationen **Dateneinheiten** und **Workfloweinheiten**, die wie folgt spezifiziert sind.

Dateneinheit
: Dateneinheiten sind Einheiten, auf die die Mitarbeiter des HelpDesk-Systems in Abhängigkeit ihrer Schreib- und Leserechte Zugriff haben. Dateneinheiten dienen als Informationsquellen, die wie ein Lexikon zum Nachschlagen verwendet werden (z.B. Telefonverzeichnis des Unternehmens, Adressverzeichnis der Unternehmensstandorte, technische Beschreibungen etc.). Dateneinheiten beschreiben keine Arbeitsvorgänge, die mittels eines Workflows von einem Bearbeiter zu einem anderen Bearbeiter wandern.

Workfloweinheit
: Workfloweinheiten sind Einheiten, die durch einen Workflow von einem Bearbeiter zum nächsten übertragen werden. Sie durchlaufen somit einen Lebenszyklus, vergleichbar einer Umlaufmappe. Hierbei ist ein Workflow durch folgende **vier Merkmale** gekennzeichnet:

  - Anfang
  - Ende
  - n Bearbeiter (mit $n \geq 1$)
  - zeitlich begrenzte Bearbeitungszeit

  Der "Lebensweg" einer Workfloweinheit wird in dieser mitprotokolliert. Hierdurch kann der Weg und der Status des Arbeitsvorgangs, der durch die Workfloweinheit beschrieben wird, überwacht werden. Nach Abschluß des Durchlaufs wird die Workfloweinheit in die "Ablage" gelegt und wandelt sich somit in eine Dateneinheit.

4.3 Area-Konzept

Mit dem Area-Konzept wird das Ziel einer **Datenkapselung** verfolgt, um mehrere Supportbereiche physikalisch auf einem Server zu betreiben, die Daten der Supportbereiche aber logisch zu trennen, um die **Vertraulichkeit der Daten** gegenüber anderen Supportbereichen sicherzustellen. Diese logische Trennung ist bei vielen Entwicklungs- und Betriebsumgebungen nur schwer möglich, d.h. im Applikationsmodell erfolgt in der Regel keine logische Trennung der Datenbestände. Andererseits wird durch die Area eine **Workload-Management-Domain** gebildet innerhalb der die zu bearbeitenden Arbeitsvorgänge koordiniert werden. Abbildung 4.2 zeigt den Aufbau einer Area.

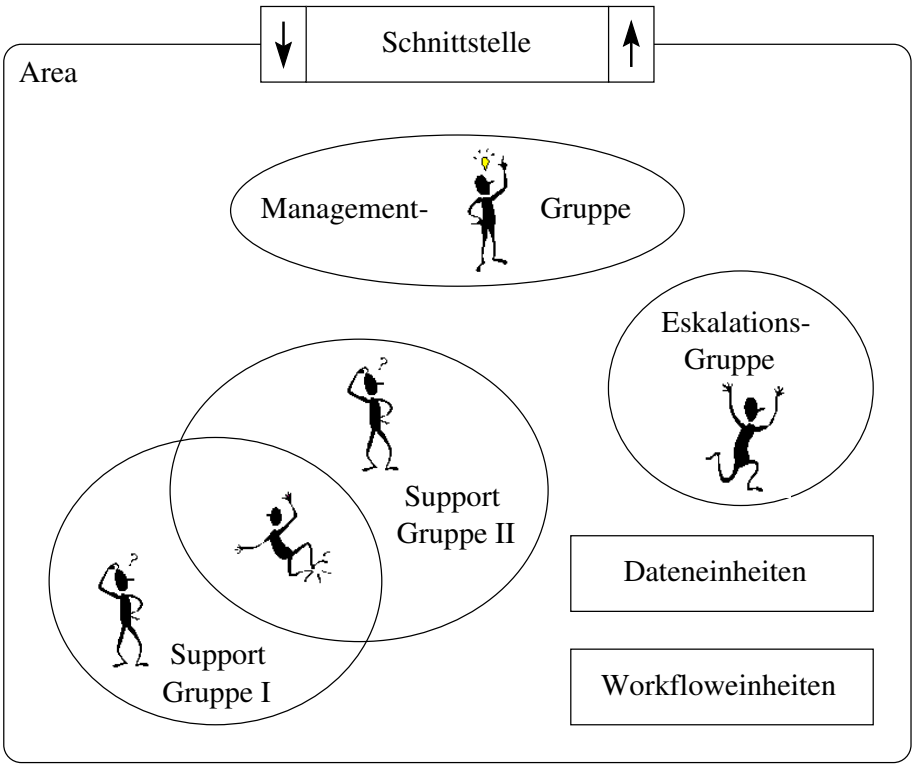


Abbildung 4.2: Aufbau einer Area

Die **Struktur des Area-Konzepts** teilt sich auf die Komponenten **User**, **Group** und **Area** auf, die nachfolgend beschrieben werden:

User

Ein User ist die Repräsentation eines Mitarbeiters einer Supportdienstleistungseinheit in dem System. Dem User sind alle Daten des Mitarbeiters, die im Hinblick auf den Fokus des HelpDesks interessant sind, zugeordnet (z.B. Telefonnummer, Mailadresse, etc.). Ein User hat nur das Recht sich an dem System anzumelden. Alle weiteren Rechte werden ihm über die Gruppenzugehörigkeit gegeben.

Group

Eine Group verfügt über gewisse **Zugriffsrechte** auf die Daten- und Workfloweinheiten. Alle Mitglieder einer Group (User) verfügen kraft ihrer Mitgliedschaft über die Rechte der Group. Ein User kann Mitglied in mehreren Groups sein. Er verfügt dann über die Summe der Rechte der einzelnen Groups. Eine Group existiert nur innerhalb einer Area und es gibt keine areaübergreifenden Groups. Eine Group kann nur Rechte auf Daten- und Workfloweinheiten der Area vergeben zu der sie gehört. Alle Groups einer Area existieren gleichberechtigt nebeneinander in der Area. Es gibt **zwei Arten von Groups**, die Primary-Groups und die Secondary-Groups. Die **Primary-Groups** dienen zur Strukturierung der User. Ein User kann nur Mitglied in einer Primary-Group sein. Über eine Primary-Group kann eine Menge von Usern, die Mitglieder der Primary-Group sind, angesprochen werden. Die **Secondary-Groups** dienen zur Vergabe von Rechten auf Daten- und Workfloweinheiten. Ein User kann Mitglied in mehreren Secondary-Groups sein. Über eine Secondary-Group kann keine Gruppe von Usern angesprochen werden.

Area

Eine Area bildet nach außen einen abgeschlossenen Bereich, der neben Usern und Gruppen eigene Daten- und Workfloweinheiten enthält. Eine Area bildet eine **Workload-Domain**, d.h. alle Arbeitsvorgänge, die innerhalb einer Area zu bearbeiten sind, werden durch die Mitglieder der Management-Gruppe dieser Area koordiniert und überwacht.

Die Area verfügt über eine **Schnittstelle zur Außenwelt**, über die Arbeitsvorgänge, die von der Außenwelt an die Area gestellt werden, übergeben werden können. Über diese Schnittstelle übermittelt die Area an einen außenstehenden Auftraggeber Statusmeldungen bzgl. des Auftrags oder die Fertigmeldung.

Neue Arbeitsvorgänge, die innerhalb der Area entstehen, können von jedem User der Area eröffnet werden.

Jede Area verfügt mindestens über eine Gruppe, die **Management-Gruppe**. Aufgabe der Mitglieder der Management-Gruppe ist die Kontrolle und Steuerung der in der Area zur Bearbeitung anstehenden Jobs. Die Mitglieder der Management-Gruppe nehmen innerhalb der Area das **Workload-Management** vor.

Alle neu zu erledigenden Arbeitsvorgänge werden prinzipiell an die Management-Gruppe einer Area weitergeleitet. Die Mitglieder der Management-Gruppe entscheiden, von welcher Gruppe und von welchem User der Arbeitsauftrag bearbeitet wird, wobei sie den neuen Arbeitsauftrag an die entsprechende Gruppe bzw. den entsprechenden User zur Bearbeitung weiterleiten.

Dies bedeutet nicht, daß alle Arbeitsvorgänge der Area, die durch die Management-Gruppe überwacht und kontrolliert werden, von ihren Mitgliedern bearbeitet werden müssen. **Routineentscheidungen** können zur Entlastung der Mitarbeiter von der HelpDesk-Applikation **automatisch** nach einem vorgegebenen **Regelwerk** getrof-

fen werden und nur Sachverhalte, für die es keine Regel gibt (d.h. Ausnahmen) sind von den Mitarbeitern zu entscheiden.

Da durch oben beschriebene Verfahren die Management-Gruppe eine Schlüsselfunktion innerhalb der Area ausübt, ergibt sich, daß die kleinste mögliche Area mindestens eine Gruppe, d.h. die Management-Gruppe, enthält und mindestens ein User Mitglied in der Area und in der Area Mitglied der Management-Gruppe ist. Die Area verfügt über je eine Datenbank zur Verwaltung der Dateneinheiten und der Workfloweinheiten.

4.4 Rechte-Konzept

Nicht jeder Mitarbeiter darf uneingeschränkten Zugriff auf alle Daten- und Workfloweinheiten haben. Hieraus ergibt sich die Notwendigkeit eines Systems zur **Rechtevergabe**.

Auf Daten- und Workfloweinheiten einer Area können nur User zugreifen, die Mitglied der Area sind. Ein direkter Zugriff von außen über die Schnittstelle der Area ist nicht möglich. Das Recht eines Users ist darauf beschränkt, sich am System anzumelden. Durch die Anmeldung erhält er keine Rechte auf die Daten- und Workfloweinheiten der Area.
Nur Gruppen besitzen Rechte auf Daten- und Workfloweinheiten, die sie an ihre Mitglieder weitergeben. Ist ein User Mitglied in mehreren Gruppen, so verfügt er über die Summe der Rechte, die ihm durch die Mitgliedschaften gegeben werden.

Diese Art der Rechtevergabe, d.h. die Rechtevergabe an die Gruppen und nicht an die Nutzer zu binden, hat folgende Vorteile. Einerseits ermöglicht es, User schnell und flexibel Rechte zuzuteilen, andererseits muß bei der Erstellung einer neuen Gruppe nur einmal verifiziert werden, ob das Systemverhalten hinsichtlich der Rechte das gewünschte Verhalten aufweist.
Bei kommerziellen Entwicklungs- und Betriebsumgebungen, die zur Implementierung eingesetzt werden, ist die Rechtevergabe auch über Gruppen anstatt über Nutzer geregelt. Allerdings kennen diese Applikationen keine Area und keine Datenkapselung der Daten- und Workfloweinheiten in der Area, wodurch sich mit einer zunehmenden Systemgröße und Mitarbeiteranzahl die Unübersichtlichkeit erhöht. Durch das Fehlen einer Area ist es bei Verwendung einer kommerziellen Entwicklungs- und Betriebsumgebung nur unzureichend möglich, die Daten und Strukturen mehrerer Support-Einheiten getrennt parallel auf einem Server ablaufen zu lassen.

4.5 Token-Container-Konzept

4.5.1 Erstellung eines Daten- und Informationsmodells für einen Workflowprozeß

Das Token-Container-Konzept verfolgt das Ziel, alle Workflowprozesse durch einen Mechanismus zu steuern und zu überwachen. Dies soll unabhängig von der Art der zu transportierenden Daten möglich sein. Abbildung 4.3 zeigt die Eigenschaften eines Workflowprozesses.

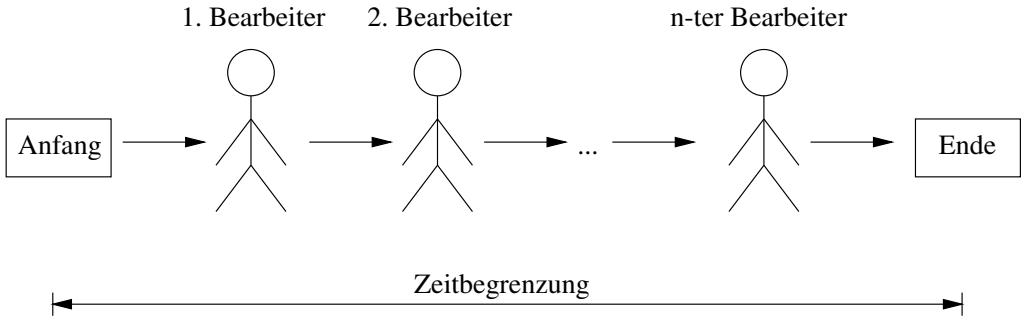


Abbildung 4.3: Eigenschaften eines Workflowprozesses

Die Workfloweinheit wird zukünftig **Job** genannt. Ein Job transportiert Daten zwischen den am Workflowprozess beteiligten Bearbeitern. Abbildung 4.4 zeigt den datentechnischen Aufbau eines Jobs.

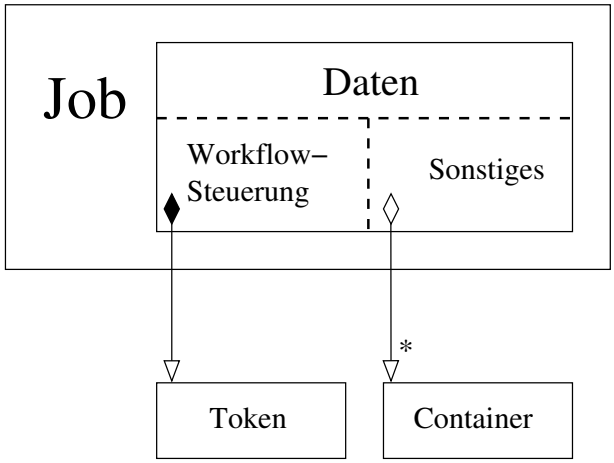


Abbildung 4.4: Datentechnischer Aufbau eines Jobs

Ein Job enthält alle Daten, die für einen Arbeitsauftrag relevant sind. Ein Job ist als elektronische Form einer Laufmappe anzusehen, die von Bearbeiter zu Bearbeiter weitergeleitet wird und in der alle Zettel für einen Arbeitsauftrag eingeheftet sind.

Die Steuerung des Jobs hängt von den Daten ab, die er beinhaltet. Diese können in die Kategorien "für die Workflow-Steuerung relevant" und "für die Workflow-Steuerung nicht relevant" klassifiziert werden. Um die Softwarestrukturen für die Steuerung des Workflows einheitlich zu verwenden, wird der Job durch die Elemente **Token** und **Container** repräsentiert. Der Token enthält alle **für die Workflow-Steuerung relevanten Daten** und der Container alle **für die Workflow-Steuerung irrelevanten Daten**. Aus diesem Grund ist der **Token systemweit einheitlich**.

Das **Token-Container-Konzept** hat den **Vorteil**, daß die **Softwarekomponente zur Steuerung des Workflows** nur einmal erstellt werden muß und **für viele Arten von Workflowprozessen (Containern)** verwendet werden kann. Dieser Ansatz steht im Gegensatz zur Vorgehensweise der kommerziellen Entwicklungs- und Betriebsumgebungen, die für einen Job keine Aufteilung in Token und Container vornimmt und somit für jede Art von Job die Neuimplementierung der Workflowlogik erforderlich macht.

4.5.2 Steuerung, Regelung und Überwachung der Informationsströme

Ein Job bzw. eine Workfloweinheit hat zu jedem Zeitpunkt einen bestimmten Zustand, der sich in den Werten einer endlichen Anzahl von Daten q_1 bis q_n mit $n \geq 1$ des Tokens widerspiegelt. Abbildung 4.5 verdeutlicht dies.

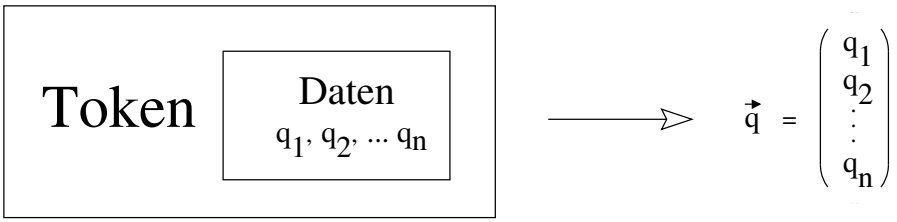


Abbildung 4.5: Repräsentation der Daten eines Tokens

Jedes Datum besitzt einen eigenen Wertebereich, wobei aus Erfahrung der Wertebereich eine endliche Anzahl von Werten enthält. Den Zustand eines Jobs ist als Vektor $\vec{q} = [q_1, q_2, ..., q_n]^T$ in einem durch die n Einheitsvektoren $\vec{e}_n$ gebildeten Zustandsraum darstellbar. Abbildung 4.6 zeigt dies beispielhaft für den Fall $\vec{q} = [b, k]^T$ mit $q_1 \epsilon \{a, b, c\}$ und $q_2 \epsilon \{j, k, l\}$. Aufgrund der endlichen Wertebereiche der Daten besitzt der Zustandsraum nur eine endliche Menge von möglichen Zuständen.

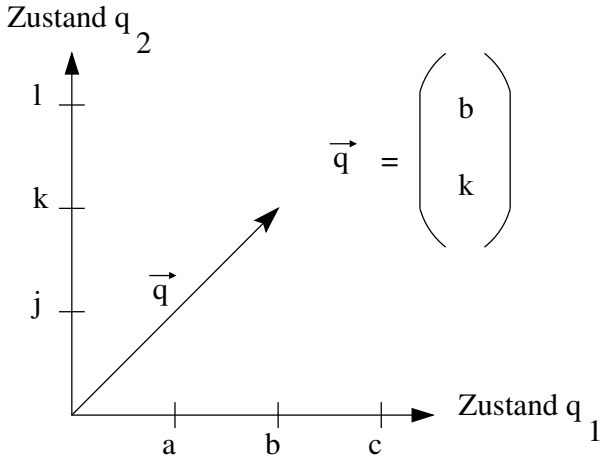


Abbildung 4.6: Darstellung eines Jobs im Zustandsraum

Der Zustand eines Jobs repräsentiert sich in der Position des zum Job gehörenden Vektors. Eine Änderung am Zustand des Jobs bewirkt eine Änderung der Position des korrespondierenden Vektors. Die Änderung wird durch einen Transitionsvektor $\vec{q}$ zwischen dem Ausgangszustand $\vec{q}$ und dem neuen Zustand $\vec{q}\,'$ beschrieben, wie Abbildung 4.7 zeigt.

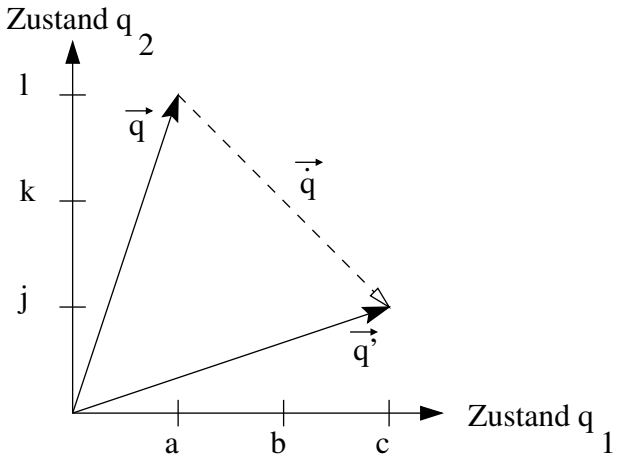


Abbildung 4.7: Zustandsübergang des Jobs durch einen Transitionsvektor

Vor dem Hintergrund der o.g. vier Merkmale eines Workflowprozesses läßt sich der **"Lebenslauf" eines Jobs im Zustandsraum** durch einen **Anfangsvektor**, einen **Endvektor** und einer **Folge von Zustandsübergängen** zwischen dem Anfangs- und dem Endvektor darstellen. Abbildung 4.8 verdeutlicht dies.

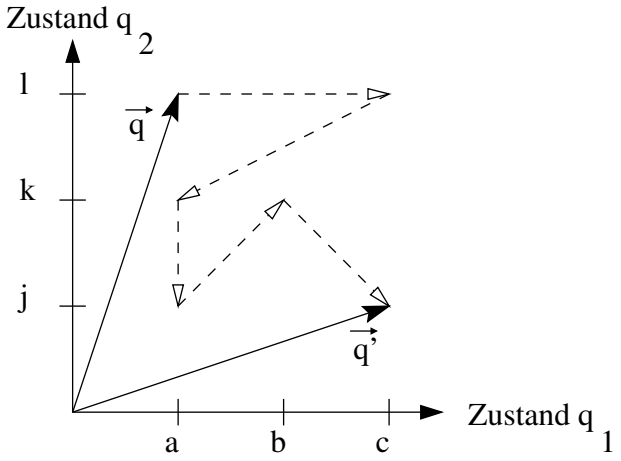


Abbildung 4.8: Lebenslauf eines Jobs im Zustandsraum

Eine **Workflow-Steuerung, -Regelung und -Überwachung** kann betrieben werden, indem der Weg des Jobzustandsvektors im Zustandsraum verfolgt wird. Bei der **Workflow-Steuerung** wird die Lage des Anfangs- und des Endvektors, sowie die Abfolge

der Transitionsvektoren fest vorgegeben. Der Job muß explizit diesen vorgegebenen Weg durchlaufen, ohne ihn verlassen zu können. Bei der **Workflow-Überwachung** werden Bereiche im Zustandsraum definiert, die unkritische Zustände der Jobzustandsvektoren im Betriebsablauf beschreiben und in denen sich die Jobzustandsvektoren frei bewegen können. Erfolgt auf einem Jobzustandsvektor eine Zustandstransition, die ihn aus dem Bereich der unkritischen Zustände verschiebt, führt das zu einer Warnmeldung. Bei einer **Workflow-Regelung** werden neben der Warnmeldung eine oder mehrere Transitionen veranlaßt, die den Zustandsvektor wieder in den Bereich der unkritischen Zustände zurückbewegen. Abbildung 4.9 verdeutlicht dies.

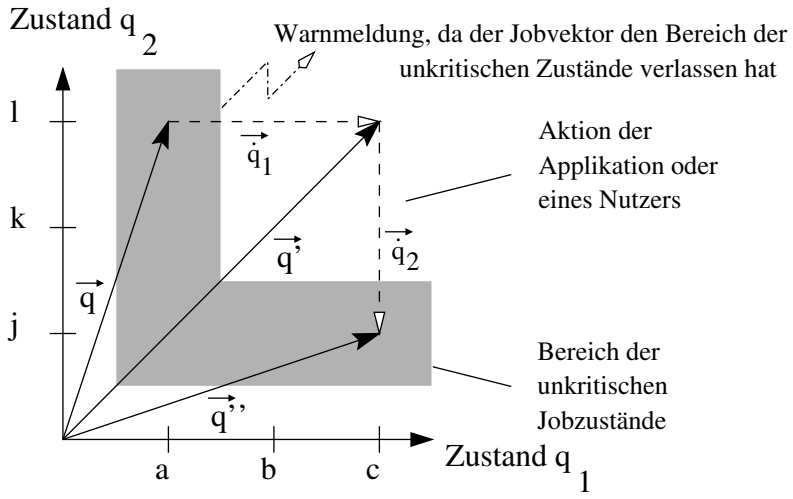


Abbildung 4.9: Workflow-Steuerung, -Regelung und -Überwachung

Hierdurch ergeben sich zwei Arten der Workflow-Regelung (siehe Abbildung 4.10). Bei der ersten Art der Workflow-Regelung unternimmt die Applikation selbständig Maßnahmen, um den Jobzustandsvektor mittels entsprechender Transitionen zurück in den gewünschten Bereich zu verschieben. Bei der zweiten Art der Workflow-Regelung ist die Warnmeldung der Applikation an einen Nutzer der Anlaß, daß der Nutzer seinerseits Maßnahmen ergreift, um den Jobzustandsvektor in den gewünschten Bereich zurückzubefördern. Die Zustandsregelung durch die Applikation ist zuverlässiger, weshalb man beim Systemdesign darauf achten sollte, betriebskritische Sachverhalte von der Applikation regeln zu lassen.

4.5.3 Zustandsregler zur Workflowkontrolle

Ein Zustandsregler zur Kontrolle der Informationsströme läßt sich unter Verwendung eines endlichen deterministischen Automaten nach Abbildung 4.11 wie folgt aufbauen:
Der endliche deterministische Automat wird aus dem Sechstupel $A = (I, O, Q, \delta, \vec{q_0}, F)$ gebildet [Class98], für das gilt:

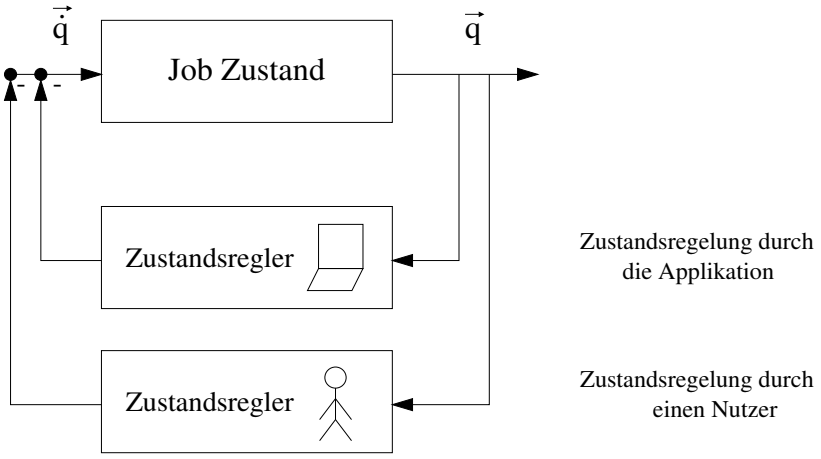


Abbildung 4.10: Zustandsregelung von Jobs

- I und O sind das Ein- bzw. Ausgabealphabet.
- Q ist eine endliche, nichtleere Menge von Zuständen bzw. Zustandsvektoren $\vec{q}$.
- δ ist eine Abbildung von der Menge der Paare ($\vec{q}$, i) bestehend aus einem Zustandsvektor $\vec{q}$ und einem Eingabezeichen $i \in I$ in die Menge der Paare ($\vec{q'}$, o) bestehend aus einem Folgezustand $\vec{q'}$ und einem Ausgabezeichen $o \in O$, wobei jedem Paar ($\vec{q}$, i) genau ein Paar ($\vec{q'}$, o) durch δ zugeordnet wird.
- $\vec{q_0} \in Q$ ist der Anfangszustand.
- $F \subseteq Q$ ist die Menge der Endzustände.

Der Zustand eines Jobs ist durch den Zustandsvektor $\vec{q}$ des Tokens beschrieben, der wie folgt definiert ist:

$$\vec{q} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{pmatrix}$$

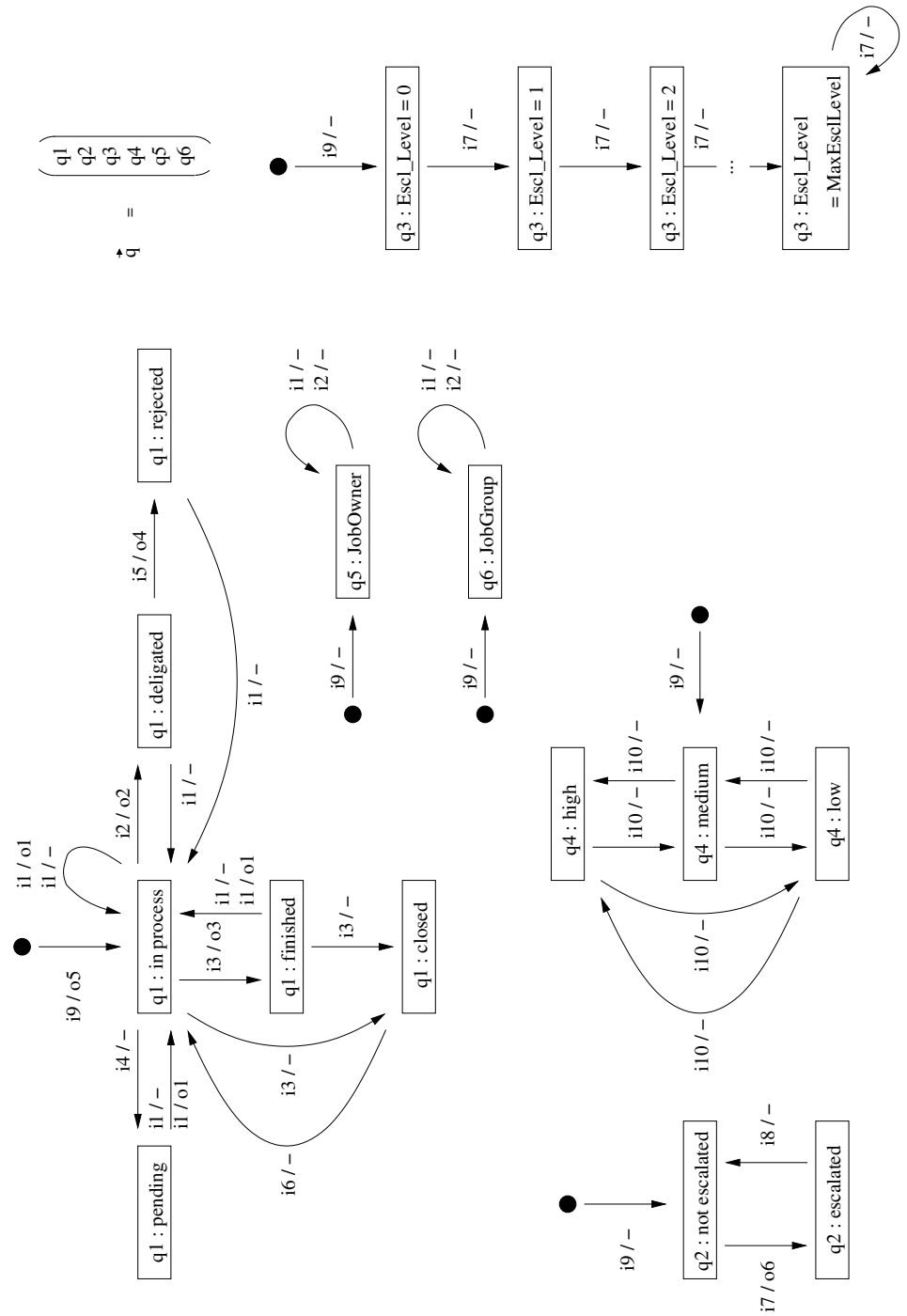


Abbildung 4.11: Zustandsregler zur Kontrolle der Informationsströme auf Basis eines endlichen deterministischen Automaten

Die Komponenten des Vektors $\vec{q}$ sind wie folgt definiert:

Status: q_1 mit $q_1 \in Q_1 = \{$in process, deligated, rejected, finished, pending, closed$\}$

Der Status q_1 beschreibt den aktuellen Bearbeitungszustand des Jobs, wobei nachfolgende Elemente der Menge Q_1 folgende Bedeutung haben:

$q_1 =$ in process
Der Job ist in Bearbeitung. Es ist kein kritischer Betriebszustand vorhanden.

$q_1 =$ deligated
Der Job wurde von einem User (Quelle) zu einem anderen User (Senke) weitergeleitet, der Empfänger hat den Empfang noch nicht bestätigt.

$q_1 =$ rejected
Der Empfang eines weitergeleiteten Jobs ist innerhalb der vorgegebenen Zeit Δt_{accept} nicht vom Empfänger bestätigt worden.

$q_1 =$ finished
Die Bearbeitung eines Jobs ist abgeschlossen. Diese Tatsache ist noch nicht bestätigt worden.

$q_1 =$ pending
Der Job ist nicht in Bearbeitung, da auf die Rückmeldung eines Dritten (z.B. Kunde, Lieferant etc.) gewartet wird.

$q_1 =$ closed
Der Job ist geschlossen und belastet keine Ressourcen innerhalb des HelpDesks.

Escalation: q_2 mit $q_2 \in Q_2 = \{$escalated, not escalated$\}$

$q_2 =$ escalated
Der Job ist eskaliert, die Eskalation ist noch nicht bearbeitet.

$q_2 =$ not escalated
Der Job ist nicht eskaliert.

Escalation-Level: q_3 mit $q_3 \in Q_3 = \{0, 1, 2, ..., MaxEscalationLevel\}$
und $MaxEscalationLevel \in N_0^+$

Zähler für die Eskalationsstufe, auf der sich der Job befindet.

Priorität: q_4 mit $q_4 \in Q_4 = \{$low, medium, high$\}$

$q_4 =$ low
Der Job hat eine niedrige Priorität.

$q_4 =$ medium
Der Job hat eine mittlere Priorität.

$q_4 =$ high
Der Job hat eine hohe Priorität.

Job-Owner: q_5 mit $q_5 \in Q_5 = Q_{User}$ = Menge der User des Systems

User, der aktueller Bearbeiter des Jobs ist.

Job-Group: q_6 mit $q_6 \in Q_6 = Q_{PrimaryGroup}$ = Menge der Primary-Groups des Systems

Primary-Group des Users, der aktueller Bearbeiter des Jobs ist.

Für den Anfangszustand $\vec{q_0}$ gilt:

$$\vec{q_0} = \begin{pmatrix} q_1 = in\ process \\ q_2 = not\ escalated \\ q_3 = 0 \\ q_4 = medium \\ q_5 = User \\ q_6 = PrimaryGroup \end{pmatrix}$$

Für die Menge der Endzustände F gilt: $F = Q$

Das Eingabealphabet $I = (i_1,\ i_2,\ i_3,\ i_4,\ i_5,\ i_6,\ i_7,\ i_8,\ i_9,\ i_{10},)$, wobei für jede Eingabe optionale Parameter angegeben werden können, wird wie folgt gebildet:

i_1: Eingabe “akzeptieren”

Der User akzeptiert einen an ihn weitergeleiteten Job bzw. bearbeitet einen Job, dessen Besitzer er ist oder dessen Besitzer er durch diesen Vorgang wird.

i_2: Eingabe “weiterleiten”

Der User leitet einen Job an einen anderen weiter.

i_3: Eingabe “schließen”

Der User schließt einen Job.

i_4: Eingabe “parken”

Der User parkt einen Job, da er den Job derzeit nicht bearbeiten kann, weil er auf die Rückmeldung eines Dritten (z.B. eines Kunden, Lieferanten etc.) wartet.

i_5: Eingabe “TimeOutAkzeptieren”

Ein an einen User weitergeleiteter Job wurde nicht innerhalb einer Zeitspanne Δt_{accept} akzeptiert bzw. bestätigt.

i_6: Eingabe “reopen”

Ein geschlossener Job wird wieder geöffnet.

i_7: Eingabe “TimeOutBearbeitung”

Die für den Job vorgesehene Bearbeitungszeit $\Delta t_{escalation}$ ist überschritten.

i_8: Eingabe “Eskalation bearbeiten”
Ein User bearbeitet einen eskalierten Job.

i_9: Eingabe “Job anlegen”
Ein User legt einen neuen Job an.

i_{10}: Eingabe “Priorität verändern”
Ein User verändert die Priorität des Jobs.

Das Ausgabealphabet $O = (o_1, o_2, o_3, o_4, o_5, o_6)$ wird wie folgt gebildet:

o_1: Mitteilung “Job übernommen”
Mitteilung an den Besitzer eines Jobs, wenn der Besitz an diesem von einem anderen User übernommen wird.

o_2: Mitteilung “Job bearbeiten”
Mitteilung, einen Job zu bearbeiten.

o_3: Mitteilung “Job schließen”
Mitteilung, daß ein Job geschlossen werden kann.

o_4: Mitteilung “Job nicht akzeptiert”
Mitteilung, daß ein Job nicht innerhalb einer Zeitspanne Δt_{accept} nach Weiterleitung vom Empfänger akzeptiert wurde.

o_5: Mitteilung “Job-Nummer”
Mitteilung der neuen Job-Nummer an den User, der Ersteller dieses neuen Jobs ist.

o_6: Mitteilung “Eskalation bearbeiten”
Mitteilung an einen User, der für einen eskalierten Job Eskalationsbearbeiter ist, den eskalierten Job zu bearbeiten.

Die Abbildungsfunktion δ ist gemäß nachfolgender Tabelle definiert:

Zustand $\vec{q}$	Eingabe i	Folgezustand $\vec{q'}$	Ausgabe o
q_1 = in process	$i_1 : akzeptieren$	q_1’ = in process	o_1 : *Job übernommen* (falls der User, der akzeptiert, nicht der Besitzer des Jobs ist, sonst keine Mitteilung)
q_1 = in process	$i_2 : weiterleiten$	q_1’ = deligated	o_2 :*Job bearbeiten* (an Empfänger)
q_1 = deligated	$i_1 : akzeptieren$	q_1’ = in process	—

q_1 = deligated	i_5 : *TimeOut-Akzeptieren*	q_1' = rejected	o_4 : *Job nicht akzeptiert* (an den Sender des Jobs und an die Management-Gruppe)
q_1 = rejected	i_1 : *akzeptieren*	q_1' = in process	—
q_1 = in process	i_3 : *schließen*	q_1' = finished (falls der User keine Berechtigung zum Schließen hat) q_1' = closed (falls der User die Berechtigung zum Schließen hat)	o_3 :*Job schließen* (an die Management-Gruppe, falls der User keine Berechtigung zum Schließen hat, sonst keine Mitteilung)
q_1 = closed	i_6 : *reopen*	q_1' = in process (falls der User die Berechtigung zum Reopen hat) q_1' = closed (falls der User keine Berechtigung zum Reopen hat)	—
q_1 = finished	i_3 : *schließen*	q_1' = closed (falls der User die Berechtigung zum Schließen hat)	—
q_1 = finished	i_1 : *akzeptieren*	q_1' = in process	o_1 : *Job übernommen* (falls der User, der akzeptiert, nicht der Besitzer des Jobs ist, sonst keine Mitteilung)
q_1 = in process	i_4 : *parken*	q_1' = pending	—

q_1 = pending	i_1 : *akzeptieren*	q_1' = in process	o_1 : *Job übernommen* (falls der User, der akzeptiert, nicht der Besitzer des Jobs ist, sonst keine Mitteilung)
q_1 = —	i_9 :*Job anlegen*	q_1' = in process	o_5 : *Job-Nummer* (an den Ersteller des Jobs)
q_2 = —	i_9 :*Job anlegen*	q_2' = not escalated	—
q_2 = not escalated	i_7 : *TimeOut-Bearbeitung*	q_2' = escalated	o_6 : *Eskalation bearbeiten* (an den zuständigen Eskalationsbearbeiter)
q_2 = escalated	i_8 : *Eskalation bearbeiten*	q_2' = not escalated	—
q_3 = —	i_9 :*Job anlegen*	q_3' = 0	—
q_3 = *Escl-Level*	i_7 : *TimeOut-Bearbeitung*	q_3' = q_3+1 (für q_3< $MaxEsclLevel$) q_3' = $MaxEsclLevel$ (für q_3 = $MaxEsclLevel$)	—
q_4 = —	i_9 :*Job anlegen*	q_4' = medium	—
q_4 = low	i_{10} : *Priorität verändern*	q_4' = medium	—
q_4 = low	i_{10} : *Priorität verändern*	q_4' = high	—
q_4 = medium	i_{10} : *Priorität verändern*	q_4' = low	—
q_4 = medium	i_{10} : *Priorität verändern*	q_4' = high	—

q_4 = high	i_{10} : *Priorität verändern*	q_4' = low	—
q_4 = high	i_{10} : *Priorität verändern*	q_4' = medium	—
q_5 = —	i_9 :*Job anlegen*	q_5' = User (der den Job angelegt hat)	—
q_5 = User	i_1 : *akzeptieren*	q_5' = User	—
q_5 = User	i_2 : *weiterleiten*	q_5' = Empfänger	—
q_6 = —	i_9 :*Job anlegen*	q_6' = Primary-Group des Users, der den Job angelegt hat	—
q_6 = Primary-Group	i_1 : *akzeptieren*	q_6' = Primary-Group des Users	—
q_6 = Primary-Group	i_2 : *weiterleiten*	q_6' = Primary-Group des Empfängers	—

Tabelle 4.1: δ-Funktion des Zustandsreglers zur Workflowkontrolle

Der oben beschriebene endliche deterministische Automat stellt aus Sicht der User nachfolgende **Basisoperationen** auf einem Job zur Verfügung:

Job anlegen

Die Basisoperation *Job anlegen* führt den User zunächst durch die Eingabemaske des Containers, um die Informationen abzufragen, die zur Eröffnung des Jobs benötigt werden. Nach Abschluß der Eingabe der Daten legt der User den Job mit der Eingabe i_9*:anlegen* an, wobei der Zustandsvektor $\vec{q}$ des neuen Jobs auf den Anfangswert $\vec{q_0}$ gesetzt wird und der User Besitzer des neuen Jobs wird. Im Gegenzug erhält der User mit der Mitteilung o_5*:Job Nummer* vom System die Job-Nummer. Abbildung 4.12 verdeutlicht dies.

Job bearbeiten

Tätigkeiten, die ein User für einen Job erledigt, sind in diesem zu protokollieren. Es kann nur der User Änderungen im Job vermerken, der Besitzer des Jobs ist. Ein Job hat nur einen Besitzer. Ein User trägt diese Änderungen in den Job (d.h. in das Token bzw. in den Container) ein und akzeptiert diese Änderung mit der Eingabe i_1*:akzeptieren*, um sie im System zu speichern. Abbildung 4.13 zeigt dies.

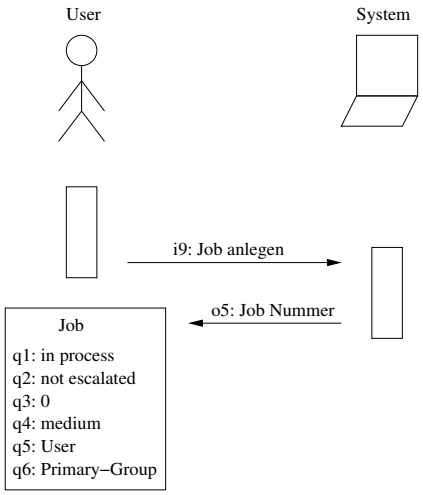


Abbildung 4.12: Erstellung eines Jobs

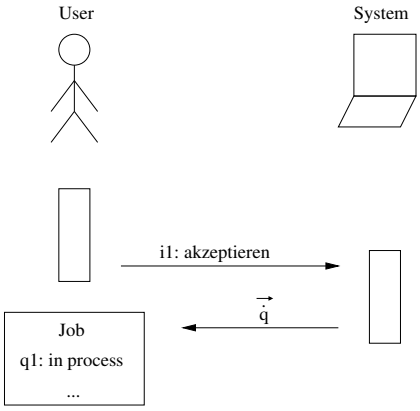


Abbildung 4.13: Bearbeitung eines Jobs

Job weiterleiten

Ein User kann einen Job an einen anderen weiterleiten. Für das Weiterleiten eines Jobs von User A nach User B gibt es eine definierte Übergabeprozedur (siehe Abbildung 4.14), die sicherstellt, daß zu jedem Zeitpunkt ein User für die Bearbeitung des Jobs zuständig ist. Jobs, die zu lange im Status q_1 = *deligated* stehen, wechseln nach der Zeit Δt_{accept} in den Status q_1 = *rejected*, wobei dieses Ereignis an den Absender des Jobs und an User mit Überwachungsfunktionalitäten durch die Mitteilung o_4*:Job nicht akzeptiert* gemeldet wird. Der Absender bzw. ein User mit Überwachungsfunktionalitäten muß dann den Job übernehmen oder einen anderen Bearbeiter hierfür suchen.

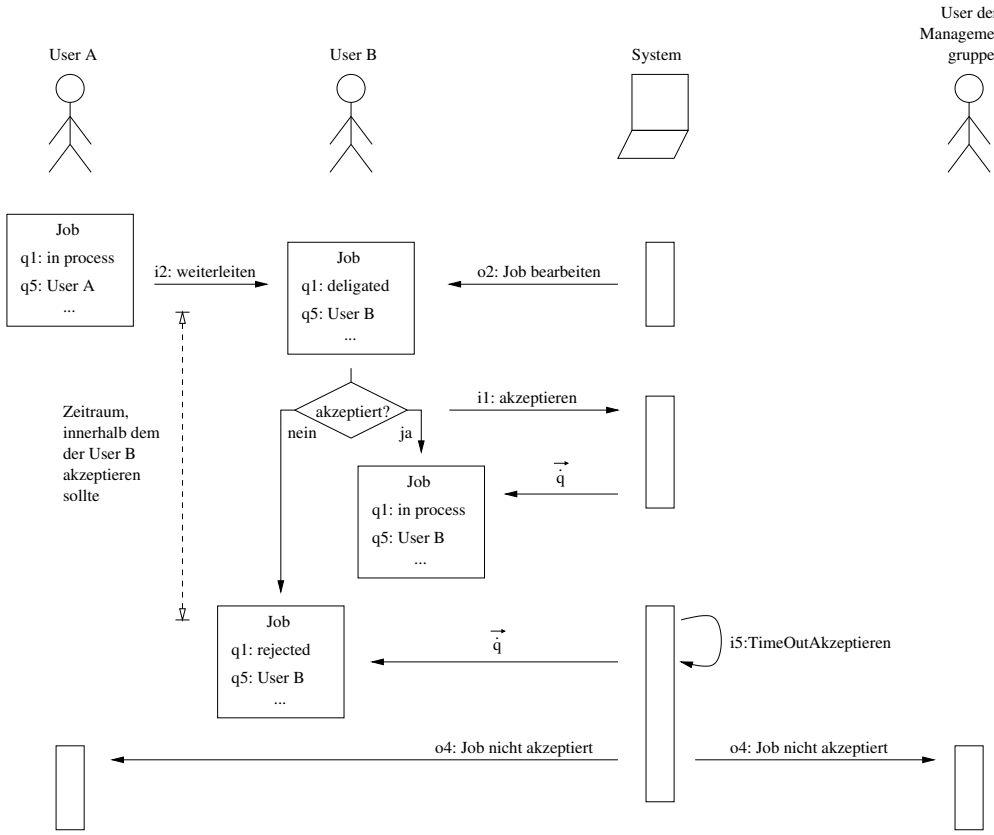


Abbildung 4.14: Weiterleitung eines Jobs

Job holen

Jeder User kann den Besitz an einen Job übernehmen. Der User, von dem der Job übernommen wurde, bekommt hierüber die Nachricht o_1*:Job übernommen* (siehe Abbildung 4.15).

Job parken

Ein User hat die Möglichkeit einen Job zu parken, wenn er derzeit nicht bearbeitet werden kann, da auf Rückmeldungen von Dritten (z.B. eines Kunden bzw. Lieferanten) gewartet wird. Durch das Parken eines Jobs wird die Eskalation nicht ausgesetzt. Das Parken von Jobs ist hinsichtlich der späteren statistischen Auswertung wichtig, da hierüber festgestellt werden kann, wie hoch der Bearbeitungsaufwand für die Supportdienstleistungsorganisation war.

Job schließen

Ist die Bearbeitung eines Jobs nach der Auffassung des aktuellen Besitzers abgeschlossen, dann kann dieser das Schließen des Jobs initiieren. Der Job wechselt in den Status q_1 = *finished*. Endgültig geschlossen werden kann ein Job nur von Usern mit

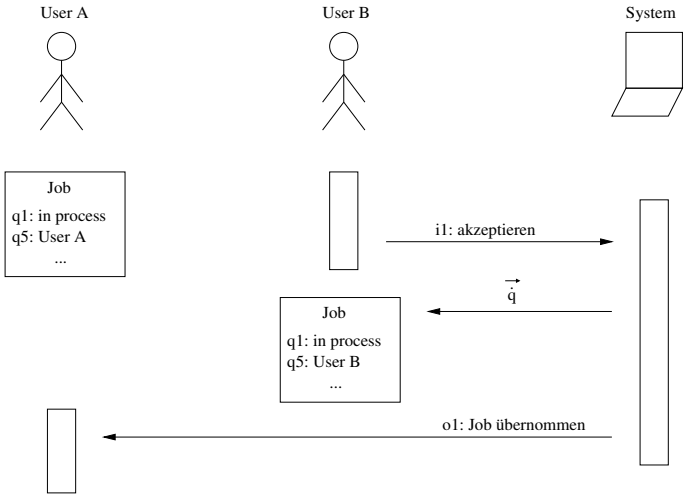


Abbildung 4.15: Übernahme eines Jobs

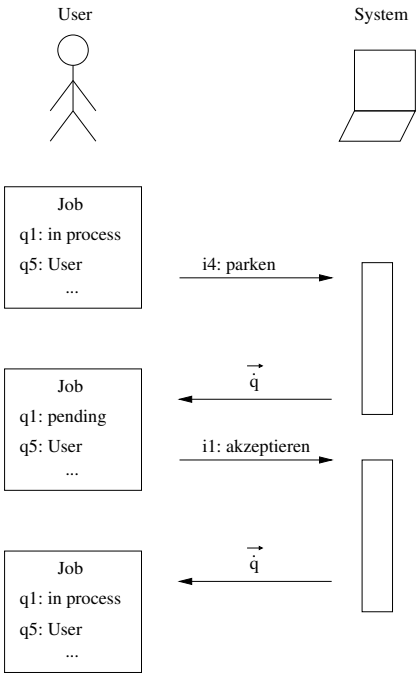


Abbildung 4.16: Parken eines Jobs

Überwachungsfunktionalitäten, die ihn in den Status $q_1 = \textit{closed}$ versetzen können (siehe Abbildung 4.17). Dieses Verfahren stellt sicher, daß Jobs nur dann geschlossen werden, wenn sie korrekt und vollständig bearbeitet worden sind.

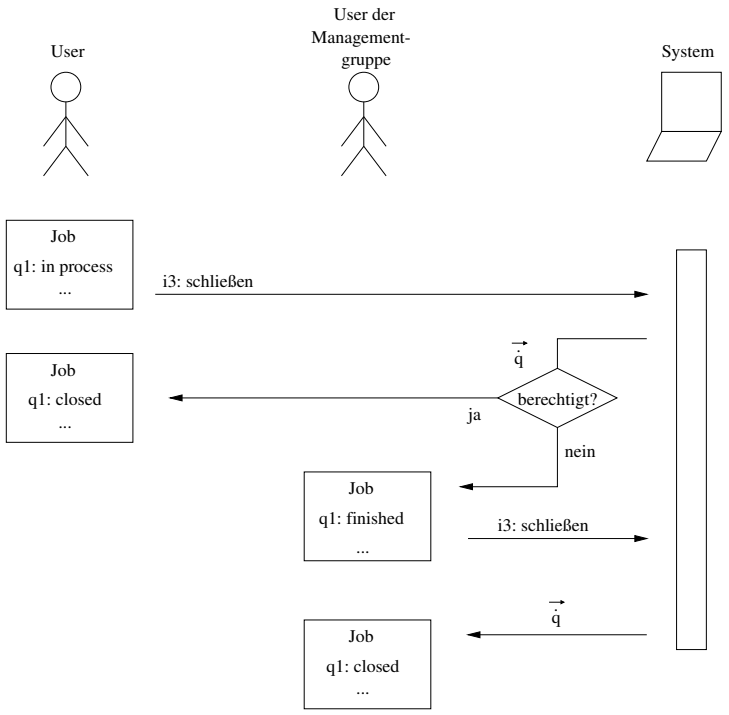


Abbildung 4.17: Schließen eines Jobs

4.5.4 Aufbau und Funktionsweise des Eskalationssystems

Die grundlegende Aufgabe des Eskalationssystems ist es, die vorgegebene Bearbeitungszeit von Jobs zu überwachen. Wird die Bearbeitungszeit eines Jobs überschritten, so wird dies Mitarbeitern mit Überwachungsfunktionalitäten zur Kenntnis gegeben. Die Abbildung 4.18 zeigt den Aufbau des Eskalationssystems und die Abbildung 4.19 zeigt den Ablauf der Basisoperation *Eskalation bearbeiten*.

Das Eskalationssystem umfaßt eine endliche Anzahl von Eskalationsstufen. Die maximale Eskalationsstufe ist der *MaxEskalationLevel*. Die aktuelle Eskalationsstufe eines Jobs ist in q_3 = *EscalationLevel* dokumentiert. Mit dem Anlegen eines Jobs (i_9*:anlegen*) wird diesem die Eskalationsstufe q_3 = *0* zugewiesen.

Die Eskalationsstufe eines Jobs wird bei jeder Eskalation um eins erhöht. Hat der Job die höchste Eskalationsstufe *MaxEscalationLevel* erreicht, dann eskaliert der Job immer wieder auf der höchsten Stufe, bis er geschlossen wird (d.h. q_1 = *closed*).

Eine Eskalation (bzw. die k-te Eskalation) erfolgt zum Zeitpunkt t_k, wenn die Zeitgrenze Δt_k überschritten wird (auf die Berechnung der t_k und Δt_k wird im Abschnitt 4.5.5, *Grundtypen von Eskalationsverfahren*, Seite 61, eingegangen).

Ist ein Job in einem Zustand $q_1 \neq$ *closed* und ist die im Job als Attribut hinterlegte nächste Eskalationszeit überschritten, so sendet das System dem für den Job zuständigen Eskalationsbearbeiter die Mitteilung o_6*:Eskalation bearbeiten*. Der Eskalationsbearbeiter

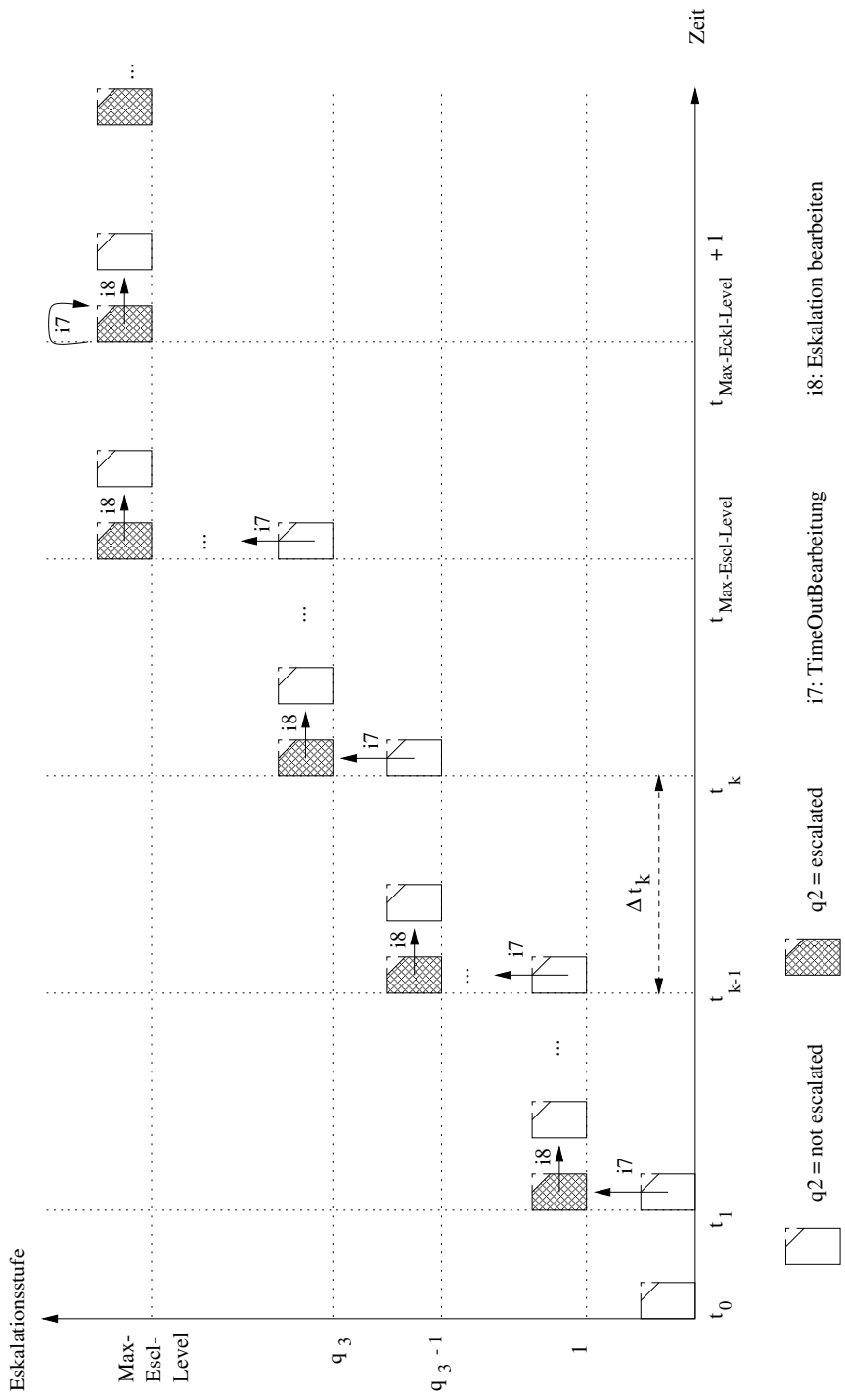


Abbildung 4.18: Aufbau und Funktionsweise des Eskalationssystems

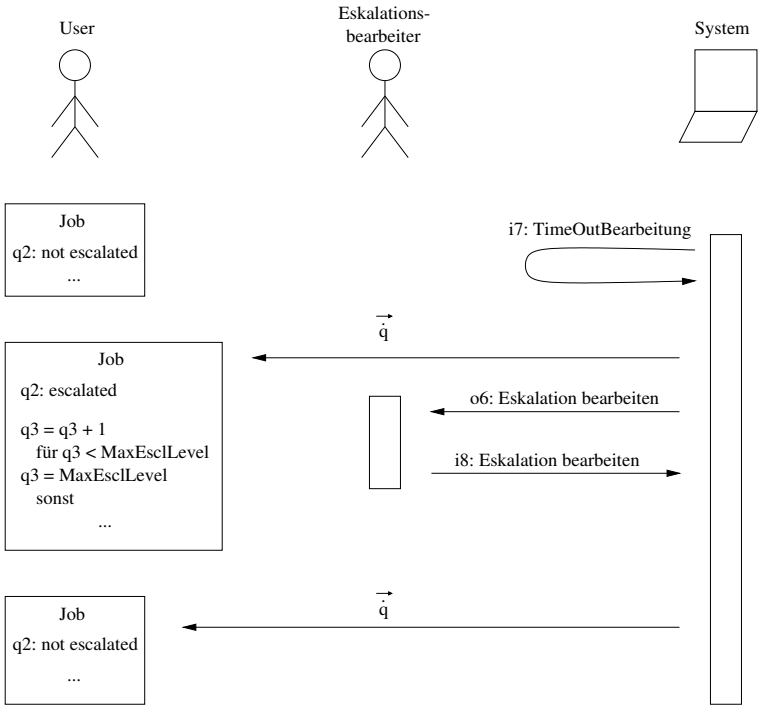


Abbildung 4.19: Ablauf des Eskalationsverfahrens

bearbeitet die Eskalation, d.h. er setzt q_2 des Jobs auf *not escalated* zurück. Vor dem Rücksetzen der Eskalation hat er die Möglichkeit, den nächsten Eskalationszeitpunkt beliebig festzusetzen oder den vom System vorgeschlagenen Standardwert t_{k+1} zu verwenden.

4.5.5 Grundtypen von Eskalationsverfahren

Das Eskalationssystem kennt zwei grundsätzliche Verfahren zur Festlegung des Eskalationsverhaltens, die in Kombination vier verschiedene Grundtypen von Eskalationsqueues ermöglichen (siehe Abbildung 4.20).

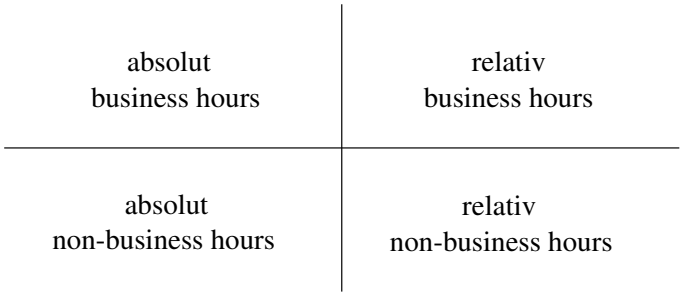


Abbildung 4.20: Die vier Grundtypen von Eskalationsqueues

Absolute Eskalation

Bei der absoluten Eskalation gilt für die Berechung des Zeitpunkts der nächsten Eskalation:

$$t_k = t_{k-1} + \Delta t_k$$

mit

$$\Delta t_k = \text{f(k, } q_5\text{:User, } q_6\text{:Primary-Group, } q_4\text{:Priorität)}$$

wobei die Funktion zur Bestimmung der Δt_k in einer Tabelle hinterlegt ist, durch die das Verhalten des Eskalationssystems konfiguriert werden kann.
Charakteristisch für die absolute Eskalation ist, daß die Eskalationsqueues eines Jobs eindeutig festgelegt sind. Dies bedeutet, daß die Eskalationszeitpunkte eines Jobs nicht verändert werden können. Die Berechnung des nächsten Eskalationszeitpunkts erfolgt hier nur bei der Bearbeitung der Eskalation, d.h. wenn ein User die Basisoperation *Eskalation bearbeiten* ausführt.

Relative Eskalation

Bei der relativen Eskalation wird der nächste Eskalationszeitpunkt nicht bei der Eskalation bzw. der Eskalationsbearbeitung festgelegt, sondern bei jeder Bearbeitung des Jobs, d.h. der nächste Eskalationszeitpunkt wird wie folgt berechnet:

$$t_k = t_{letzter Bearbeitungszeitpunkt} + \Delta t_k$$

mit

$$\Delta t_k = \text{f(k, } q_5\text{:User, } q_6\text{:Primary-Group, } q_4\text{:Priorität)}$$

Charakteristisch für die relative Eskalation ist, daß die Eskalationszeitpunkte im Gegensatz zur absoluten Eskalation nicht eindeutig festgelegt sind. Der nächste Eskalationszeitpunkt wird nicht bei der Eskalationsbearbeitung bestimmt, sondern bei jeder Bearbeitung des Jobs neu berechnet. Der Eskalationsbearbeiter kann aber das Intervall Δt_k individuell setzen und somit den im System hinterlegten Standardwert überschreiben.

Absolute Eskalation versus relative Eskalation

Das **absolute Eskalationsverfahren** kommt dann zur Anwendung, wenn die Jobs eine **maximale Bearbeitungsdauer** nicht überschreiten dürfen und eine Eskalation zu erfolgen hat, wenn dies geschieht.
Das **relative Eskalationsverfahren** wird zur Sicherstellung der **kontinuierlichen Bearbeitung** der Jobs angewendet. Die Eskalation erfolgt dann, wenn die kontinuierliche Bearbeitung eines Jobs unterbrochen wird.

4.5.6 Modus “business hour”, “non business hour”

Die Modi “business hours” und “non-business hours” legen fest, welche Zeiten zur Berechnung der Zeitdifferenzen berücksichtigt werden. Wird ein Eskalationssystem im “non-business hours”-Modus betrieben, werden Zeit-Stunden zur Berechung der Differenzen zwischen den Eskalationszeitpunkten verwendet. Im “business hours”-Modus werden nur die Arbeitszeiten zur Berechnung der Differenzen zwischen den Eskalationszeitpunkten herangezogen.
Beispiel: Die Arbeitszeiten innerhalb einer Supportdienstleistungsorganisation sind von Mo. - Fr. von 08:00 - 12:00 Uhr und von 13:00 - 17:00 Uhr. Soll ein Job, der am Freitag um 16:00 Uhr erstellt wird, in 3 Stunden eskalieren, dann erfolgt die Eskalation im “non-business hours”-Modus am Freitag um 19:00 Uhr und im “business hours”-Modus erst am nächsten Montag um 10:00 Uhr.

4.5.7 Parametrisierung des Steuerungssystems

Die Parametrisierung des Verhaltens des Steuerungssystems erfolgt über folgende Funktionen, mittels derer die Werte, die im Automaten verwendet werden, dynamisch nach Lage der Zustände eines Jobs abgeändert werden. Hierdurch kann das Verhalten des Automaten für jeden Container individuell parametrisiert werden.

$\Delta t_{escalation}$ = f(q_3:Escl-Level, q_5:User, q_6:Primary-Group, q_4:Priorität, Container)

Δt_{accept} = f(q_3:Escl-Level, q_5:User, q_6:Primary-Group, q_4:Priorität, Container)

Mitteilung o_4*:Job nicht akzeptiert*
an Group = f(q_3:Escl-Level, q_6:Primary-Group, q_4:Priorität, Container)

Mitteilung o_6*:Eskalation bearbeiten*
an Group = f(q_3:Escl-Level, q_6:Primary-Group, q_4:Priorität, Container)

4.6 Aufbau eines Prototypen

Der **Nachweis der Anwendbarkeit des generischen Modells** bzw. des zustandsraumbasierten Steuerungssystems zum Informationsaustausch wird durch eine **prototypische Implementierung einer HelpDesk-Applikation** erbracht. Einsatzgebiet der HelpDesk-Applikation ist ein mittelständisches Unternehmen der Telekommunikationsbranche. Dieses betreibt als Supportdienstleistungsorganisation zwei HelpDesk-Systeme (vgl. Abbildung 1.1, Seite 4). HelpDesk-System 1 deckt die Niederlassungen im deutschsprachigen Raum ab. HelpDesk-System 2 deckt die weltweiten Niederlassungen ab. Der **Aufbau des generischen Modells für den Prototypen** wird anhand des HelpDesk-Systems 1 dargestellt, die Implementierung des Prototypen mittels einer Entwicklungs- und Betriebsumgebung (vgl. Abschnitt 2.1.2, Seite 20) anhand beider HelpDesk-Systeme.

4.6.1 Generisches Modell des Prototypen

Jedes HelpDesk-System wird im generischen Modell durch eine **Area** repräsentiert. Abbildung 4.21 zeigt die räumliche Verteilung der Niederlassungen des HelpDesk-Systems 1 und Abbildung 4.22 den Aufbau der Area.

Abbildung 4.21: Räumliche Verteilung der Niederlassungen des HelpDesk-Systems 1

Innerhalb der Area wird für jede Niederlassung eine **Primary-Group** erstellt und für jeden Mitarbeiter ein **User**. Gemäß der Niederlassungszugehörigkeit eines Mitarbeiters wird der korrespondierende User der jeweiligen Primary-Group zugeordnet.

Innerhalb der Area existiert eine Workfloweinheit und fünf Dateneinheiten. Die **Workfloweinheit** bzw. der **Job** setzt sich aus dem **Token** und dem Trouble-Ticket Container zusammen. Der **Trouble-Ticket Container** enthält alle Daten, die zur Beschreibung des Bearbeitungszustands einer Störung erforderlich sind.

Als **Dateneinheiten** wird ein **Trouble-Ticket Archiv** und folgende vier Listen angelegt:

Caller-List
: Adressliste aller Personen, die in der Vergangenheit einen Call an den HelpDesk gerichtet haben.

Contact-List
: Adressliste aller Personen, die in der Vergangenheit als Ansprechpartner bei der Bearbeitung von Calls genannt wurden.

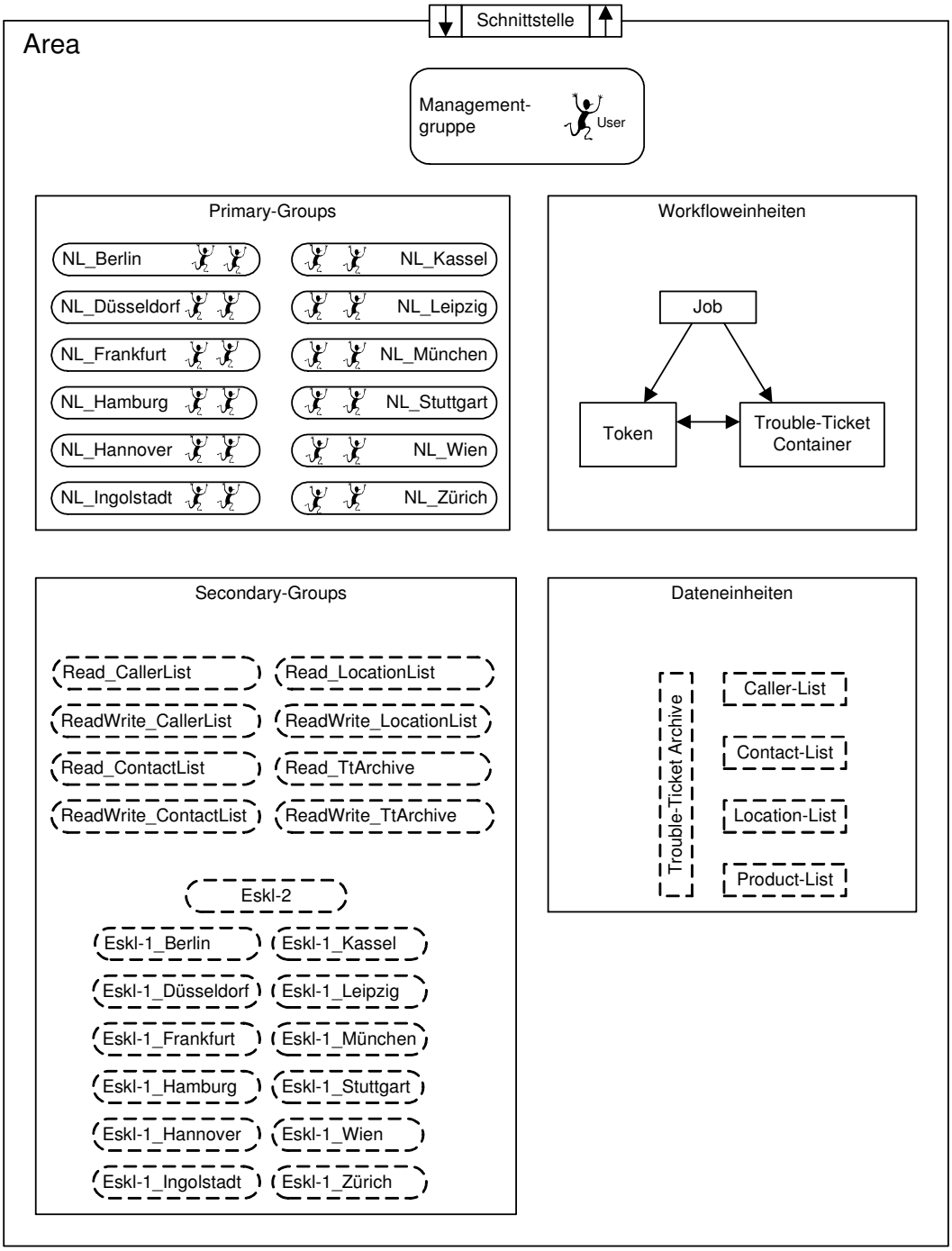


Abbildung 4.22: Aufbau der Area des HelpDesk-Systems 1

Location-List
Adressliste aller Lokationen, die Mitarbeiter des HelpDesk in der Vergangenheit angefahren haben.

Product-List
Liste aller Produkte, für die der HelpDesk Unterstützungsleistungen anbietet.

Die Listen dienen einerseits dazu, den Trouble-Ticket Container bei der Erstellung mit Daten zu befüllen und andererseits als Nachschlagewerk für die Mitarbeiter des HelpDesks. Zur **Steuerung der Zugriffskontrolle** werden pro Dateneinheit zwei **Secondary-Groups** angelegt. Die `Read_<Dateneinheit>` Gruppe gibt ihren Mitgliedern Leserechte und die `ReadWrite_<Dateneinheit>` Gruppe Lese- und Schreibrechte auf die jeweilige Dateneinheit.

Das **Eskalationssystem des Prototypen** ist von **Grundtype "absolut", "non business hours"** (vgl. Abschnitt 4.5.5, Seite 61) und verfügt über **zwei Eskalationsstufen**. Die Eskalationszeitpunkte hängen von der Priorität des Trouble-Tickets ab. Abbildung 4.23 zeigt die Funktionsweise des Eskalationssystems.

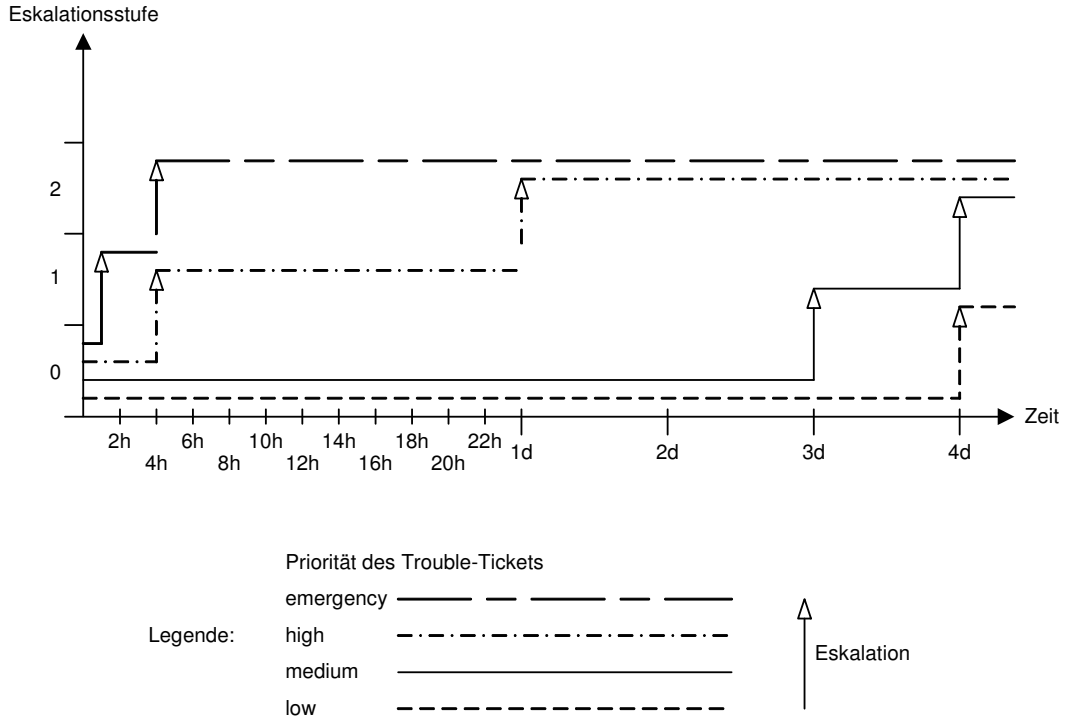


Abbildung 4.23: Funktionsweise des Eskalationssystems von HelpDesk-System 1

Zur **Steuerung der Berechtigung der Eskalationsbearbeitung** werden pro Eskalationsstufe "Eskalationsbearbeitergruppen" als Secondary-Groups angelegt, deren Mitglieder die Berechtigung zur Eskalationsbearbeitung auf der jeweiligen Stufe haben. Abbildung 4.24 zeigt den Zusammenhang zwischen den Eskalationsbearbeitergruppen und den Niederlassungen, auf der die Bearbeitung der Trouble-Tickets stattfindet.

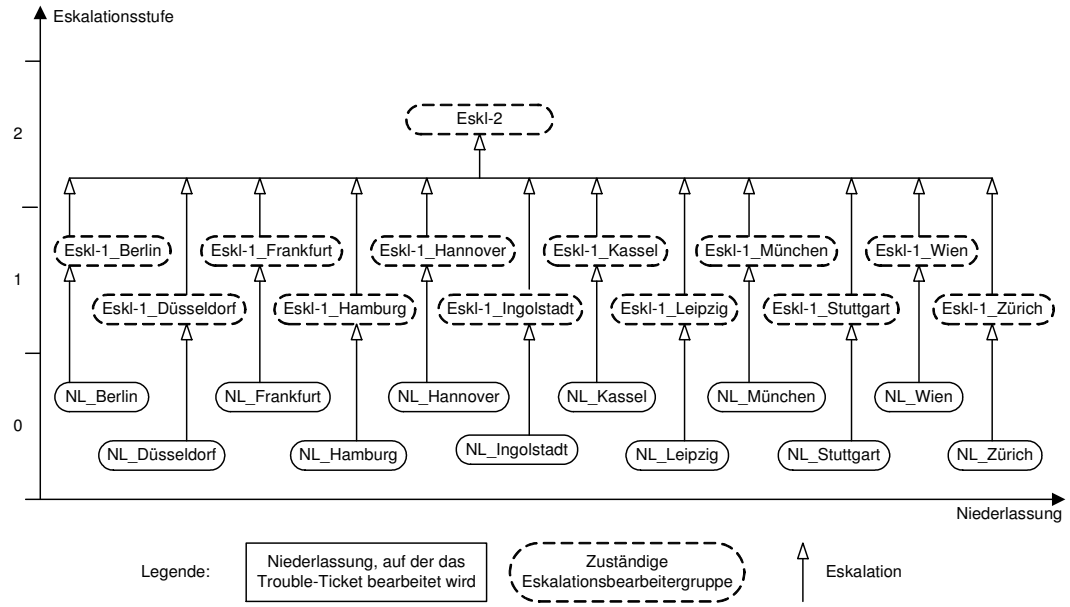


Abbildung 4.24: Zusammenhang zwischen den Eskalationsbearbeitergruppen und den Niederlassungen des HelpDesk-Systems 1

4.6.2 Implementierung des Prototypen

Zur prototypischen Implementierung der HelpDesk-Applikation wird eine **Entwicklungs- und Betriebsumgebung** eingesetzt und das **generische Modell** in das **Applikationsmodell** des verwendeten Softwareprodukts umgesetzt. Abbildung 4.25 zeigt den technischen Aufbau der prototypischen HelpDesk-Applikation.

Basis der Applikation sind **fünf Applikationsserver**, die über das Internet mittels **VPN-Verbindungen** ihren **Datenbestand synchronisieren**. Die Datenverteilung erfolgt bei der verwendeten Entwicklungs- und Betriebsumgebung über den Applikationsserver (siehe Abbildung 2.3, Seite 15, Option 1). Hierdurch ist es möglich, verschiedene Betriebssysteme und Datenbanken zum Aufbau der Applikationsserver zu verwenden.

Die Clients können sich mit allen Server verbinden, wobei sie den Server wählen, zu dem die größte Übertragungsbandbreite besteht. Hierdurch ist eine **Redundanz bei Serverausfällen** erreicht.

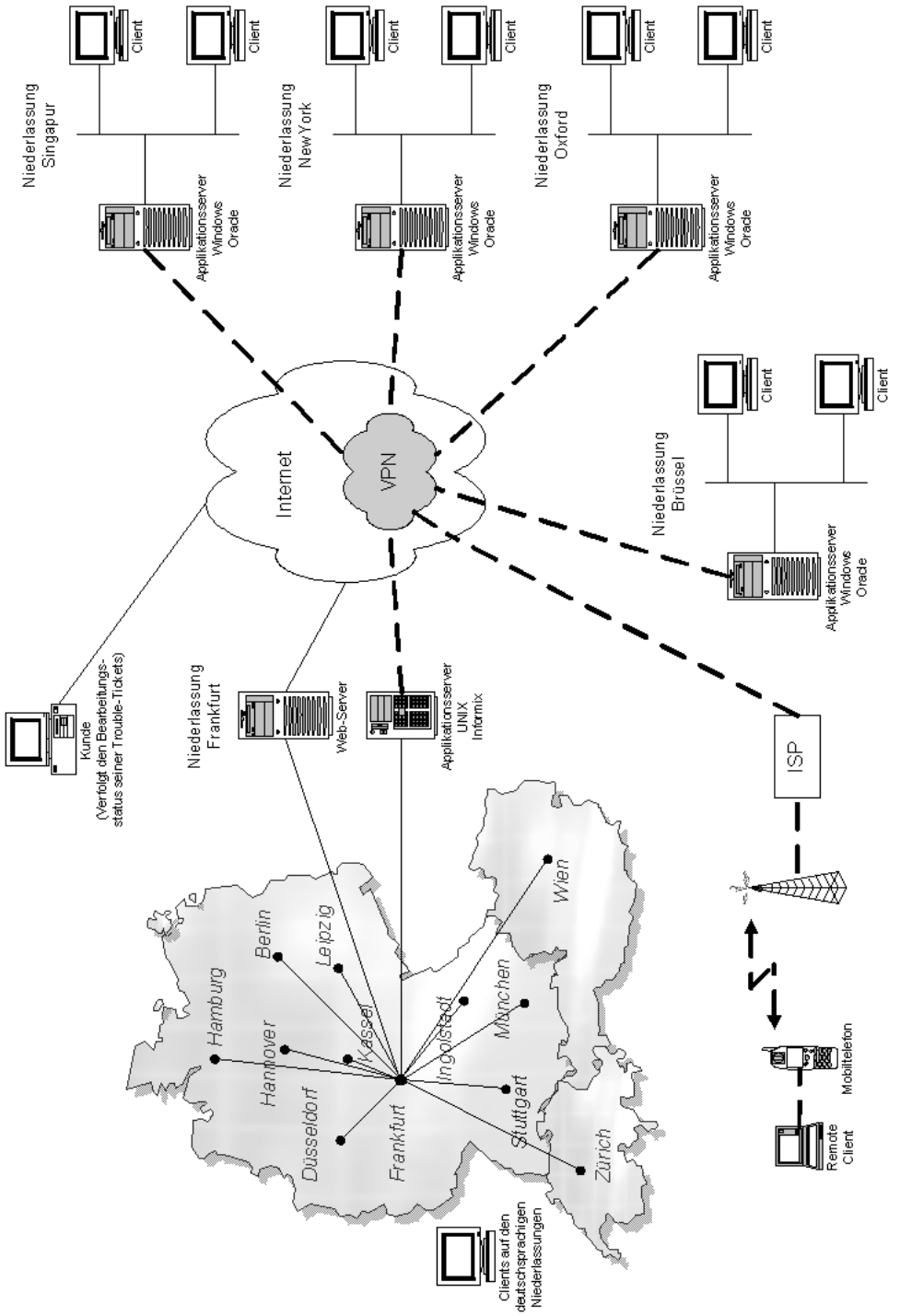


Abbildung 4.25: Technischer Aufbau der prototypischen HelpDesk-Applikation

Kunden können mittels eines Web-Servers den Bearbeitungsstatus ihrer Trouble-Tickets online mitverfolgen.

Die im Rahmen dieser Arbeit geschaffene prototypische HelpDesk-Applikation ist Rückgrad des Daten- und Informationsautauschs von ca. 230 Mitarbeitern in drei Kontinenten. Sie ist 24h/365d in Betrieb.

4.7 Fazit

Das generische Modell ermöglicht den **Aufbau einer HelpDesk-Applikation**, die **Softwareproduktunabhängig** ist. Es erlaubt ferner die unabhängige Entwicklung folgender **drei Teilbereiche** (siehe Abbildung 4.26).

- Abbildung der Struktur des HelpDesk-Systems.
- Aufbau eines Steuerungssystems zum Daten- und Informationsaustausch unabhängig von der Art des zu transportierenden Contents.
- Aufbau von Datenstrukturen und Automatismen zur Verwaltung bzw. teil- oder vollautomatischen Bearbeitung des Contents.

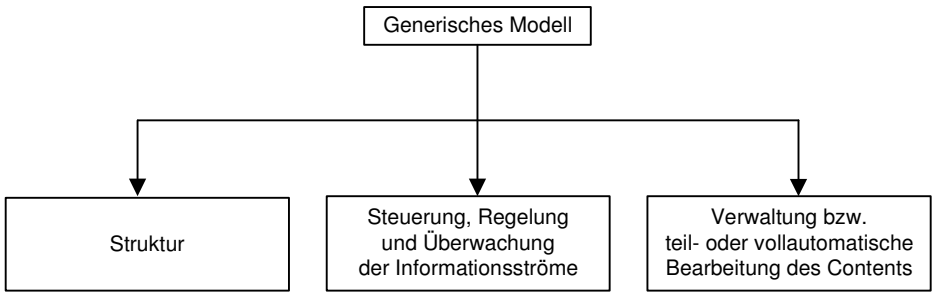


Abbildung 4.26: Teilbereiche des generischen Modells zur Entwicklung einer HelpDesk-Applikation

Das generische Modell bietet in den drei Teilbereichen folgende **Vorteile** gegenüber herkömmlichen HelpDesk-Applikationen:

Struktur

- Das **Area-Konzept** ermöglicht eine EDV-Unterstützung für mehrere HelpDesk-Systeme auf einer gemeinsamen IT-Infrastruktur. Hierbei sind die Datenbestände der HelpDesk-Systeme logisch getrennt.
- Unter Verwendung der **Primary-Groups** kann die dreigeteilte Struktur eines HelpDesk-Systems abgebildet werden (vgl. Abbildung 2.5, Seite 22). Darüberhinaus ist die Abbildung weiterer Strukturformen möglich (vgl. Abbildung 4.22, Seite 65).

- Das Zusammenspiel von **Secondary-Groups** und **Rechte-Konzept** ermöglicht eine einfache Verwaltung der Zugriffsrechte auf den Datenbestand des HelpDesk-Systems.

Steuerung, Regelung und Überwachung der Informationsströme

- Das **Token-Container-Konzept** ermöglicht den Aufbau eines Steuer-, Regelungs- und Überwachungssystems zur Kontrolle der Informationsströme. Dies ist unabhängig von der Art des transportierten Contents.
- Das Steuerungssystem des generischen Modells unterstützt die **Abwicklung von Aufgaben in einer vorgegebenen Zeitspanne**.
- Es gewährleistet die **Übertragung einer Nachricht** zwischen zwei Nutzern in einer **vorgegebenen Zeitspanne**.
- Durch entsprechende Parametrisierung bzw. Erweiterung des Zustandsautomaten können **Weiterleitungsentscheidungen**, **Priorisierungen** und das **Eskalationsmanagement teil- bzw. vollautomatisch** unterstützt werden.
- Die einfache Parametrisierung des Zustandsautomaten ermöglicht eine **schnelle Anpassung der HelpDesk-Applikation an unvorhergesehene und plötzlich auftretende Anforderungen**.

Content-Verwaltung

- Das **Daten- und Informationskonzept** ermöglicht die **Erstellung individueller Daten- und Workfloweinheiten** für den Content. Ferner die Erstellung von **Automatismen zur teil- oder vollautomatischen Bearbeitung des Contents**.

Die Verwendung eines **generischen Modells** zur Beschreibung der Funktionsweise der HelpDesk-Applikation schafft eine **Unabhängigkeit** hinsichtlich der verwendeten Entwicklungs- und Betriebsumgebung (vgl. Abbildung 4.1, Seite 39). Hierdurch wird das Auswechseln des Softwareprodukts einfach möglich.
Als **Nachweis für die Unabhängigkeit** ist die prototypische Implementierung der HelpDesk-Applikation für das HelpDesk-System 1 zuerst mit dem Produkt "Action-Request-System" (ARS) der Firma [Remedy] erfolgt. Danach ist das generische Modell des Prototypen erfolgreich auf das Produkt "Navision Attain" der Firma Microsoft umgesetzt worden. Abbildung 4.27 zeigt als Beispiel die Maske des Trouble-Tickets auf beiden Systemen.

Abschließend ist festzustellen, daß das generischen Modell den Anforderungen, die der praktische Betrieb an einen HelpDesk-Applikation stellt, gerecht wird (vgl. Abschnitt 2.2, Seite 21).

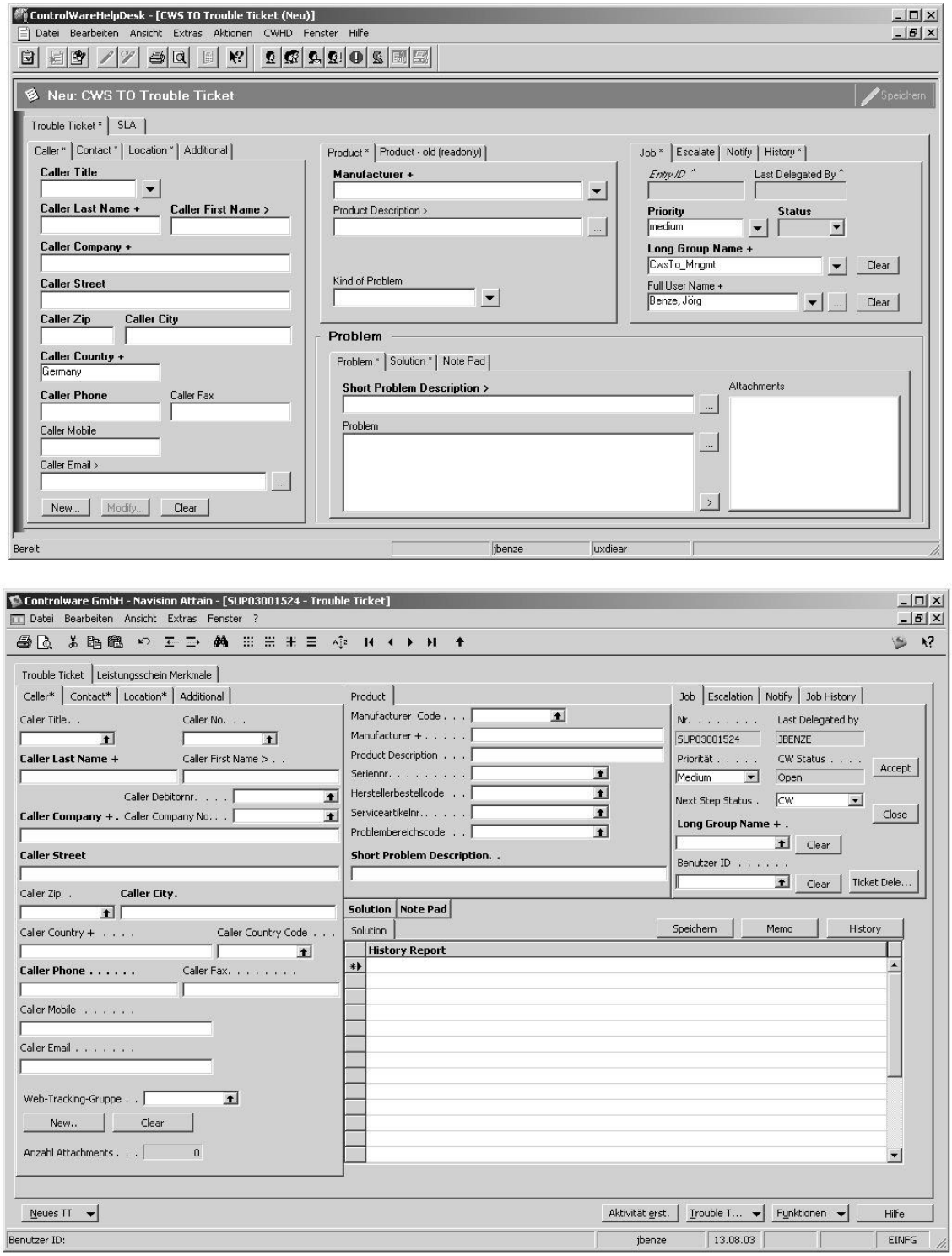


Abbildung 4.27: Maske des Trouble-Tickets auf verschiedenen Plattformen (oben: Action-Request-System / unten: Navision Attain)

Kapitel 5

Entwicklung einer verteilten Systemarchitektur zum Informationsaustausch

5.1 Ansatz zur Konzeptentwicklung

Der Ansatz für eine HelpDesk-Applikation auf Basis einer verteilten Systemarchitektur hat seinen Ursprung in der Beobachtung folgenden Sachverhalts. Eine Supportdienstleistungsorganisation verfügt in der Regel über mehrere Standorte, die mit Datenleitungen untereinander verbunden sind. Betrachtet man die Verfügbarkeit von Leitungsbandbreite in und zwischen den Standorten so ist folgendes festzustellen: In den Standorten verfügt man über eine hohe Leitungsbandbreite im LAN. Zwischen den Standorten ist die Leitungsbandbreite im WAN deutlich geringer (siehe Abbildung 5.1).
Analysiert man die Wege der Jobs zwischen den Nutzern der Applikation so ist festzustellen, daß sich bei mindestens 75% der Weiterleitungen Absender und Empfänger an demselben Standort befinden. Die Weiterleitungen von Jobs, bei denen sich der Absender und der Empfänger an verschiedenen Standorten befinden, macht einen weitaus geringeren Teil unter der Gesamtzahl der Weiterleitungen in einem Zeitraum aus (siehe Abbildung 5.2). Dieses Ergebnis ist zu erwarten, da die Jobs im Call-Center angelegt werden. Das Call-Center leitet die Jobs an die zuständige Niederlassung weiter. Bei der Bearbeitung in dem Standort werden die Jobs nach Zuständigkeit bzw. Schicht- und Vertretungsplan zwischen den Nutzer weitergeleitet.
Der oben festgestelle Sachverhalt hinsichtlich der Verfügbarkeit von Leitungsbandbreite paßt demnach zum Verkehrsaufkommen der Job-Weiterleitungen, d.h. dort wo ein hohes Datenverkehrsaufkommen durch Job-Weiterleitungen erwartet wird, ist auch eine hohe Leitungsbandbreite vorhanden und umgekehrt. Jedoch wird dieser Umstand von einer HelpDesk-Applikation mit einer Client-Server Architektur nicht genutzt (siehe Abbildung 5.3).

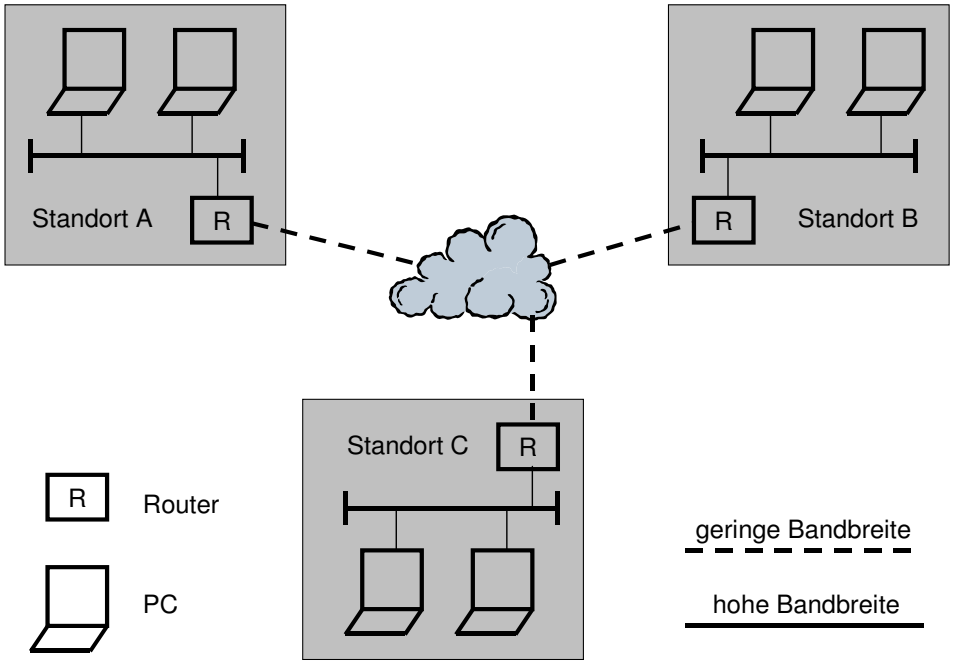


Abbildung 5.1: Verfügbarkeit von Leitungsbandbreite in und zwischen den Standorten einer Supportdienstleistungsorganisation

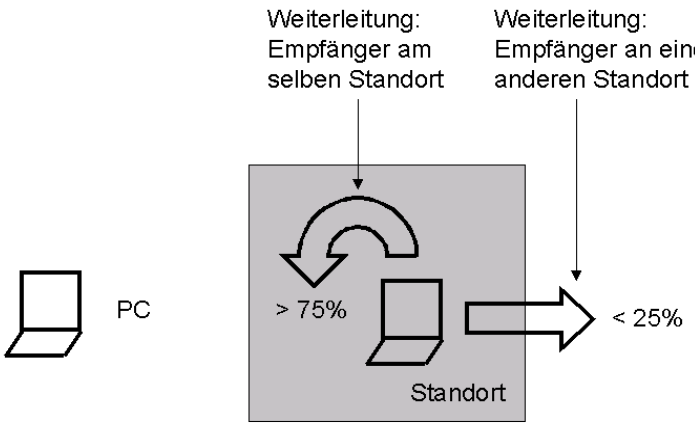


Abbildung 5.2: Verteilung der Anzahl der Weiterleitungen eines Jobs

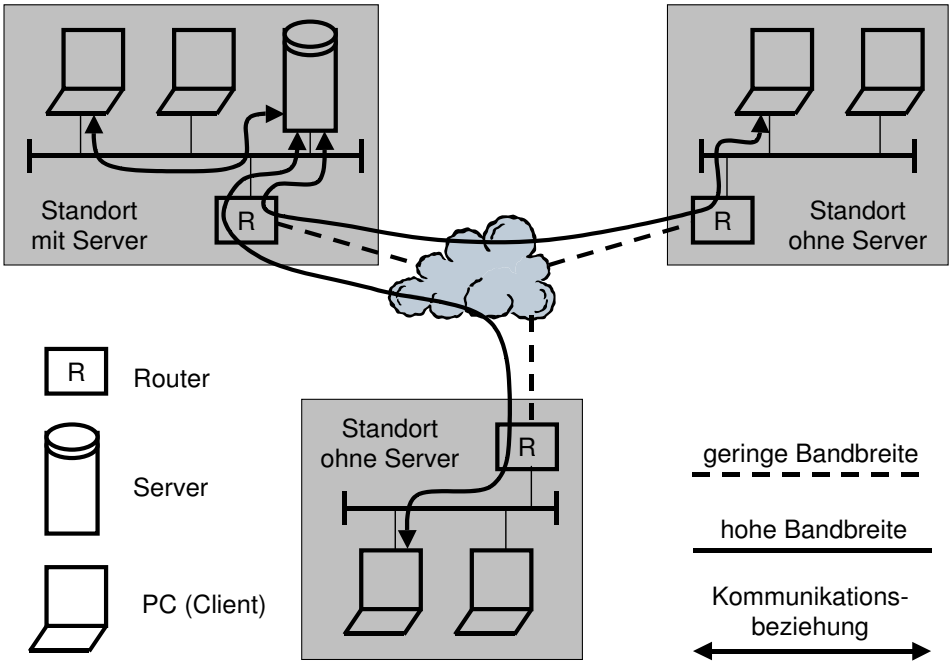


Abbildung 5.3: Datenverkehrsaufkommen bei einer HelpDesk-Applikation auf Basis einer Client-Server Architektur

Der Server befindet sich an einem Standort und durch die zentralistische Architektur kommunizieren die Clients anderer Standorte über die WAN-Leitung mit dem Server, was an diesen Standorten zu einer geringeren Leistungsfähigkeit der Applikation führen kann. Eine HelpDesk-Applikation mit einer **Client-Server Architektur** verfügt aus diesem Blickwinkel betrachtet über folgende **Nachteile**:

- Die lokale Weiterleitung eines Jobs zwischen zwei Nutzern an einem Standort, der nicht Aufstellungsort des Servers ist, erfordert eine Datenübertragung über die WAN-Leitung.

- Bei Ausfall der WAN-Verbindung eines Standorts ohne Server ist dieser nicht mehr arbeitsfähig, obwohl er über eine funktionierende Netzwerkinfrastruktur verfügt und die Mehrzahl der Jobs lokal bearbeitet werden.

- Mit zunehmender Client-Anzahl wird dem Server mehr und mehr Rechenleistung abverlangt und die Zunahme der Rechenleistung im Netz durch die Clients wird nicht genutzt. Dieser Effekt einer Client-Server Architektur begrenzt das Wachstum des Systems nach oben.

Durch die **Wahl einer anderen Architekturform** könnten die o.g. Nachteile einer Client-Server Architektur überwunden werden. Auch wäre man imstande das Gesamtsystem toleranter gegenüber Ausfällen von WAN-Leitungen und Einzelkomponenten zu gestalten. Eine **HelpDesk-Applikation auf Basis mobiler Agenten** stellt hier eine Alternative dar.

5.2 Mobile Agenten

Eine Möglichkeit zur **Definition** von mobilen Agenten ist folgende:

> Agenten sind Programme, die zwischen zwei Rechnern migrieren können. Sie entscheiden selbst über Zeitpunkt und Ziel. Der Zustand des Programms wird am Ausgangsrechner gespeichert, zum Zielrechner übertragen und der Programmablauf an der gleichen Stelle fortgesetzt.

Hinsichtlich der **Eigenschaften**, die ein Agent besitzen muß, wird ein starker und ein schwacher Agentenbegriff unterschieden. Der **schwache Agentenbegriff** definiert Merkmale wie z.B. *autonom*, *reaktiv*, *proaktiv*, *zielorientiert*. Sie sind als Grundvoraussetzungen von Agenten anzusehen. Der **starke Agentenbegriff** wird mehr von Ansichten aus dem Bereich der Künstlichen Intelligentz (KI) getragen. Als Beispiele sind Eigenschaften wie *Mobilität*, *Anpassungsfähigkeit*, *Kollaboration* zu nennen.

Besitzt ein Agent die **Eigenschaft der Mobilität**, wird er als **mobiler Agent** bezeichnet, sonst als **stationärer Agent**. Nur mobile Agenten können zwischen Rechner migrieren.

Bei der **Migration** wird zwischen der schwachen und der starken Migration unterschieden. Die **schwache Migration** erfordert die Festlegung eines speziellen Einstiegspunkts zur Ausführung des Agenten auf dem Zielrechner. Sie ist für Umgebungen geeignet, die keinen direkten Zugriff auf den Programmstack erlauben. Bei der **starken Migration** werden Programmstack, Programmzähler und Programmcode auf den Zielrechner übertragen, so daß zur Fortsetzung der Programmausführung keine Festlegung eines speziellen Einstiegspunkts notwendig ist.

Zur Ausführung eines Agenten auf einem Rechner ist ein Laufzeitsystem erforderlich. Dieses wird **Agentenhostsystem** genannt. Ein Agentenhostsystem kann Agenten keieren, ausführen, transferieren und beenden.

Agenten können miteinander interagieren. Als Beispiele für **Agenteninteraktion** sind die **Kooperation**, die **Koordination** und die **Kommunikation** zu nennen. Zur Kommunikation kommen Mechanismen wie *Remote Procedure Calling* (RPC), *Common Object Request Broker Architekture* (CORBA), *Distributed Component Object Model* (DCOM) und *Remote Method Invokation* (RMI) zum Einsatz.

5.3 Applikationskonzept auf Basis mobiler Agenten

Eine HelpDesk-Applikation auf Basis mobiler Agenten kann wie in Abbildung 5.4 gezeigt aufgebaut sein und besteht aus folgenden Komponenten:

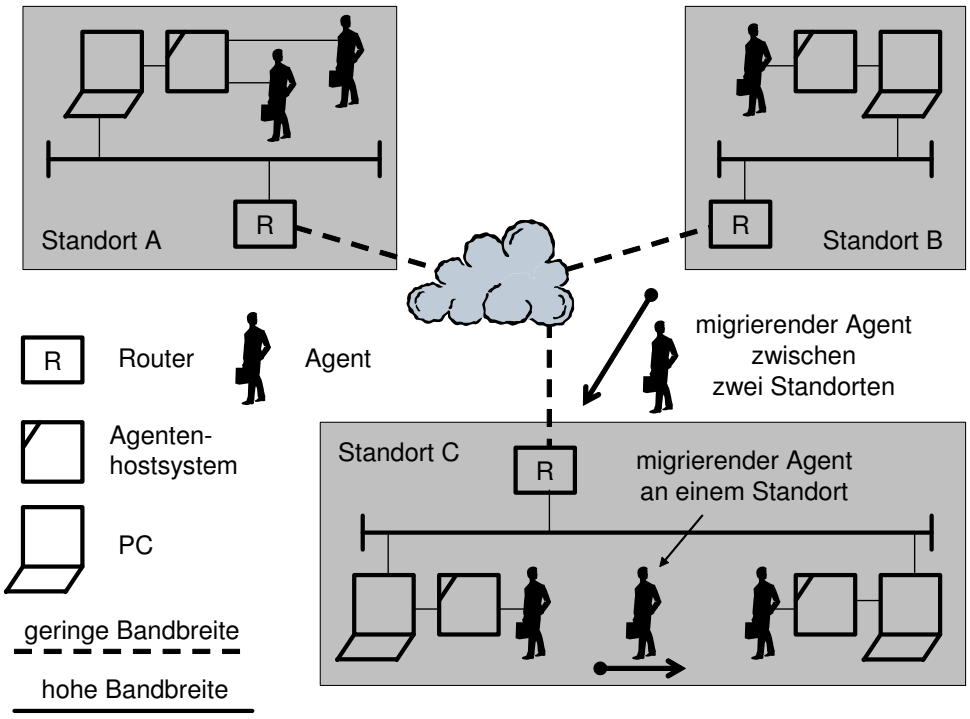


Abbildung 5.4: HelpDesk-Applikation auf Basis mobiler Agenten

Mobile Agenten Die Jobs werden als mobile Agenten repräsentiert, die auf dem Agentenhost des bearbeitenden Nutzers ausgeführt werden. Die Weiterleitung eines Jobs von einem Nutzer zu einem anderen bedingt die Migration des Agenten zu dem Agentenhostsystem des entsprechenden Nutzers. Voraussetzung zur Migration eines Agenten ist nicht immer eine Weiterleitung; ein Agent kann z.B. im Fall einer Eskalation sich selbständig zu einem Eskalationsbearbeiter bewegen bzw. im Falle der Nichtbearbeitung durch einen Nutzer sich einen anderen Bearbeiter suchen.

Die Verwendung von mobilen Agenten muß nicht auf das Bearbeiten durch eine Person beschränkt bleiben. Agenten können sich selbständig im Netz bewegen und ihren Job erledigen (z.B. Fehler in Netzwerken finden und Gegenmaßnahmen ergreifen).

Agentenhostsystem Ein Nutzer interagiert an dem Agentenhostsystem mit den Jobs, die sich in seiner Bearbeitung befinden. Er kann z.B. neue Jobs erstellen (d.h. mobile Agenten kreieren) und Jobs beenden (d.h. mobile Agenten beenden).

Eine solche HelpDesk-Applikation auf Basis mobiler Agenten verfügt nicht über die o.g. Nachteile einer Client-Server Architektur. Sie trägt den Anforderungen aus Verfügbarkeit von Leitungsbandbreite und den Erfahrungen zum Datenverkehrsaufkommen an und zwischen den Standorten bei Job-Weiterleitungen Rechnung. Ein Standort ist bei Ausfall seiner WAN-Anbindung weitgehend arbeitsfähig, da sich die mobilen Agenten, die die Jobs repräsentieren, auf den Agentenhostsystemen der Mitarbeiter des Standorts befinden.

5.4 Anforderungen und Bewertung

Aus den praktischen Erfahrungen des Betriebs eines Trouble-Ticket-Systems muß eine HelpDesk-Applikation mindestens **drei Anforderungen** erfüllen:

- Kein Job darf verloren gehen! Die Architektur und der Aufbau des Systems muß sicherstellen, daß keine Datenverluste im laufenden Betrieb auftreten. Eine vollständige Datenwiederherstellung nach Systemausfällen ist sicherzustellen.
- Die Datenbestände in dem System müssen für alle Anwender im laufenden Betrieb jederzeit erreich- und modifizierbar sein.
- Suchabfragen zur Filterung des Datenbestands im System nach vorgegebenen Kriterien müssen möglich sein und mit kurzen Antwortzeiten zu Resultaten führen.

Betrachtet man das Agentensystem vor dem Hintergrund dieser Anforderungen, so können nachfolgende Schwierigkeiten bei deren Erfüllung auftreten.

- Bei der Abschaltung bzw. beim Ausfall eines Agentenhostsystems stehen die Daten der dort befindlichen Agenten dem Gesamtsystem nicht mehr zur Verfügung. Wird durch das Agentenhostsystem nicht dem Verlust der Agenten und deren Daten bei Abschaltung der Plattform bzw. bei Systemabstürzen vorgebeugt, entstehen hierdurch unwiderrufliche Datenverluste. Die Erstellung eines vollständigen Backups, welches alle Daten des Gesamtsystems sichert, wird bei der verteilten Datenhaltung eines mobilen Agentensystems problematischer sein als bei einem Client-Server System.
- Die Erreichbarkeit der Datenbestände des Gesamtsystems im laufenden Betrieb kann bei einem System mobiler Agenten partiellen Einschränkungen unterliegen. Diese können ihre Ursache z.B. in Leitungsausfällen haben, die die Migrationsfähigkeit der Agenten einschränken, als auch in deaktivierten Agentenhostsystemen, die Agenten “eingefroren” haben.
- Suchabfragen auf große Teile des Datenbestands gestalten sich bei einem Client-Server System einfacher, da durch die zentralistische Architektur alle Daten an einem Ort durchsucht werden können. Hierdurch wird eine höhere Leistung bei den Antwortzeiten zu erwarten sein. Suchanfragen, die auf den dezentralen Datenbestand

eines Systems mobiler Agenten gerichtet sind, werden Probleme aufwerfen. Beispielhaft sind hier "eingefrorene" Agenten zu nennen, deren Datenbestand im Moment der Suchabfrage nicht zur Verfügung steht. Realisiert man die Suchanfrage mittels eines Suchagenten, der die Job-Agenten auf den Hostsystemen der Reihe nach besucht, wird sich dies negativ bei der Antwortzeit niederschlagen. Auch die in der Praxis beliebte Abfrage "zeige mir alle Jobs" würde in dem skizzierten Agentensystem eine Migration aller Agenten zu dem Agentenhostsystem des Abfrageursprungs nach sich ziehen. Die Anzahl der dort eintreffenden Agenten könnte das Agentenhostsystem überlasten. Alternativ könnte man die Agenten auffordern ihre Daten an das abfragende Agentenhostsystem zu senden.

- Das Weiterleiten eines Jobs (Agent) vom Nutzer A zum Nutzer B würde zeitaufwendig sein, da der Agent erst das Agentenhostsystem suchen müßte, an dem der Nutzer B arbeitet. Ein Auskunftsdienst, der das Agentenhostsystem von Nutzer B benennen könnte, wäre hier sinnvoll.

Fazit:
Es ist festzustellen, daß der Ansatz zum Aufbau einer HelpDesk-Applikation auf Basis von mobilen Agenten Vorteile bei der Ausnutzung der verfügbaren Bandbreite und der im Netz installierten Rechenleistung gegenüber einer Client-Server Architektur verspricht. Allerdings wird ein mobiles Agentensystem den Anforderungen an eine HelpDesk-Applikation ohne Nachbesserungen an der Architektur nicht gerecht.

5.5 Erweiterung des Applikationskonzepts

Eine Verbesserung der Leistungsfähigkeit von HelpDesk-Applikationen wäre zu erreichen, wenn man die Vorteile eines Systems mobiler Agenten mit denen einer Client-Server Architektur kombinieren könnte. Dies vermag u.U. die Nachteile beider Ansätze zu kompensieren. Die kurze Beschreibung der Idee eines solchen Architekturkonzepts ist Gegenstand dieses Abschnitts.

5.5.1 Komponenten

Bestandteile des Systems sind der **Workflow Objekt Agent (WOA)**, das **User Agent Interface (UAI)** und der **Area Domain Controller (ADC)**.

Workflow Objekt Agent (WOA)

Ein Workflow Object Agent (WOA) repräsentiert einen Vorgang im Rahmen eines Workflowprozesses, wie dies früher eine "Laufmappe" war. Es können **mehrere Typen von WOAs** im System existieren (z.B. Störung, Auftrag, Urlaubsmeldung, usw.) und sich im System bewegen. Ein WOA wird systemtechnisch durch einen mobilen Agenten realisiert, der auf einem im System vorhandenen Agentenhostsystem ausgeführt wird. Er kann zwischen diesen Agentenhostsystemen migrieren. Jeder WOA hat im System einen

eindeutigen Namen und gehört zu einer **Kategorie** (z.B. Störung, Auftrag, Urlaubsmeldung, usw.). Ein WOA enthält neben den Daten eines Vorgangs auch ein Verhalten, durch das er selbständig agiert und Entscheidungen trifft. Bei der Erstellung einer Urlaubsmeldung beispielsweise prüft der WOA, ob der gewünschte Zeitraum nicht mit einer Urlaubssperre kollidiert. Bei einem Trouble Ticket überwacht der WOA seine Eskalationszeiten selbständig bzw. sucht sich einen Mitarbeiter mit einem entsprechenden Skill-Profil. Bei einem Auftrag prüft der WOA den Bestand der zur Auftragsbearbeitung notwendigen Artikel im Lager, bevor er zu einem Mitarbeiter migriert. Der Vorteil bei der Verwendung von WOAs besteht in der **Individualität** jedes einzelnen WOAs, die es ermöglicht, jedem einzelnen ein eigenes Benutzerinterface zu geben oder ein spezielles Verhalten. Hierdurch wird es einfacher, neue Anforderungen zur IT-Unterstützung von Geschäftsprozessen mit dem System umzusetzen, da das Prozeßwissen des Systems sich in den WOAs befindet.

Die Verwendung von WOAs beschränkt sich aber nicht nur darauf, daß diese mit den Anwendern interagieren. WOAs können auch Teilaufgaben im Rahmen des Geschäftsprozesses selbständig erledigen. Beispielsweise ist hier die Durchführung einer Fehlerdiagnose bzw. Fehlereingrenzung bei Störungen im Netzwerk zu nennen. Hierzu ist es erforderlich, daß zukünftige **Netzwerkkomponenten Agentenhostsysteme bereitstellen**, zu denen der WOA migrieren kann, um eine Fehlerdiagnose durchzuführen.

User Agent Interface (UAI)

Das User Agent Interface (UAI) ist ein Agentenhostsystem, welches einerseits die Ausführung der Workflow Object Agenten (WOA) ermöglicht und andererseits ein Interface für die Nutzer des Systems darstellt, mittels dem die Anwender mit den Agenten bzw. den Diensten (vgl. Abschnitt 5.5.3, “Services”, Seite 83ff) kommunizieren können. Es hat einen eindeutigen Namen und sichert den Code der Agenten und deren Datenbestände auch bei Abschaltung des UAI und bei Systemabstürzen gegen Verlust. Das UAI dient auch zur Ausführung von Dienst-Agenten, die die im Abschnitt 5.5.3 “Services” (Seite 83ff) beschriebenen Dienste anbieten. Realisiert wird das UAI durch eine Softwarekomponente, die auf den gängigen Betriebssystemen (wie z.B. Windows, Linux, Solaris, HP-UX, usw.) installiert wird. Das visuelle Erscheinungsbild eines UAI könnte z.B. wie in Abbildung 5.5 gezeigt aussehen.

Auf der linken Seite des Fensters findet man eine (nach Kategorien sortierte) Auflistung aller auf dem UAI vorhandenen WOAs. Durch Markieren eines WOAs im Auswahlmenü der linken Seite erscheint dessen Benutzerinterface im rechten Teil des Fensters, wobei die Gestaltung des Benutzerinterfaces dem jeweiligen WOA obliegt. Unter dem Punkt “Services” erhält man eine Auflistung aller Dienstagenten (stationäre Agenten, vgl. Abschnitt 5.5.3, “Services”, Seite 83ff), die auf dem UAI ausgeführt werden, und Zugang zu deren Benutzerschnittstellen. Dies ist dann der Fall, wenn der UAI zu einem Area Domain Controller (ADC, siehe folgenden Abschnitt) gehört. Wird der UAI nicht auf einem ADC ausgeführt, erhält man unter diesem Punkt Zugang zu den zentralen Diensten, die der ADC für die Domain zur Verfügung stellt.

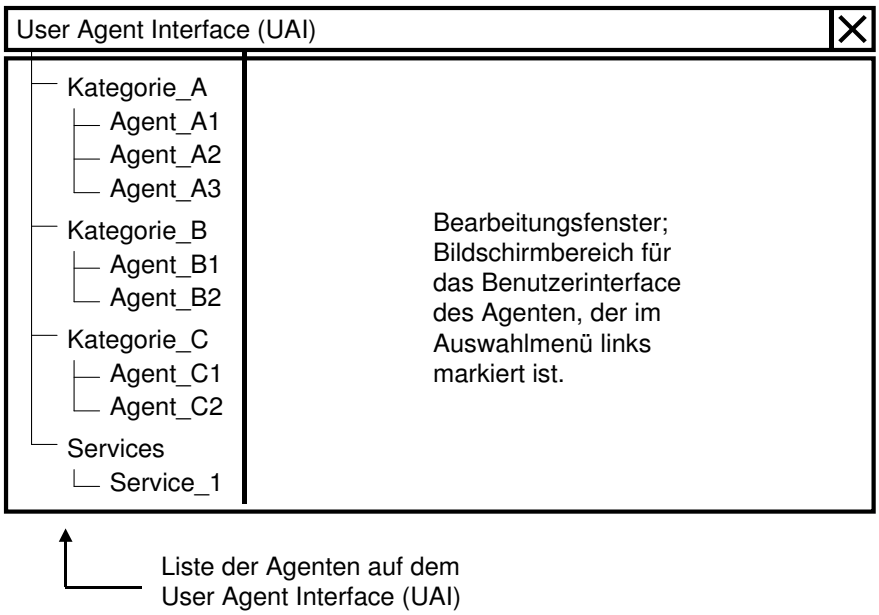


Abbildung 5.5: Visuelles Erscheinungsbild eines User Agent Interface (UAI)

Area Domain Controller (ADC)

Ein Area Domain Controller (ADC) dient den WOAs als Plattform für ein Agentenhostsystem in einer Domain (vgl. Abschnitt 5.5.2 "Strukturen", Seite 81ff), wenn diese Tätigkeiten ausführen, die keine Nutzer-Kommunikation erfordern. Auf einem ADC wird ein UAI als Agentenhostsystem ausgeführt. Auf dem UAI laufen mehrere Dienst-Agenten (vgl. Abschnitt 5.5.3 "Services", Seite 83ff), die von den WOAs in der Domain, für die der ADC zuständig ist, genutzt werden können. Die zum Aufbau des ADC verwendete Hardware ist auf ausfallsicheren Betrieb ausgelegt, d.h. Ausstattungsmerkmale eines ADCs sind ein Spiegelplattensystem, eine unterbrechungsfreie Stromversorgung, automatische Datensicherung usw. Bestandteil eines ADCs ist ferner ein Datenbankserver, auf dem die Dienst-Agenten ihre Datenbestände verwalten. Abbildung 5.6 zeigt den Aufbau eines ADCs und einer User-Station.

5.5.2 Strukturen

Weiterer Bestandteil eines Architekturkonzepts sind die Strukturen, in der die Komponenten des Systems eingebettet sind. Das Konzept beinhaltet zur Strukturierung **zwei Arten von Domains**, die **Local Area Domain (LAD)** und die **Wide Area Domain**

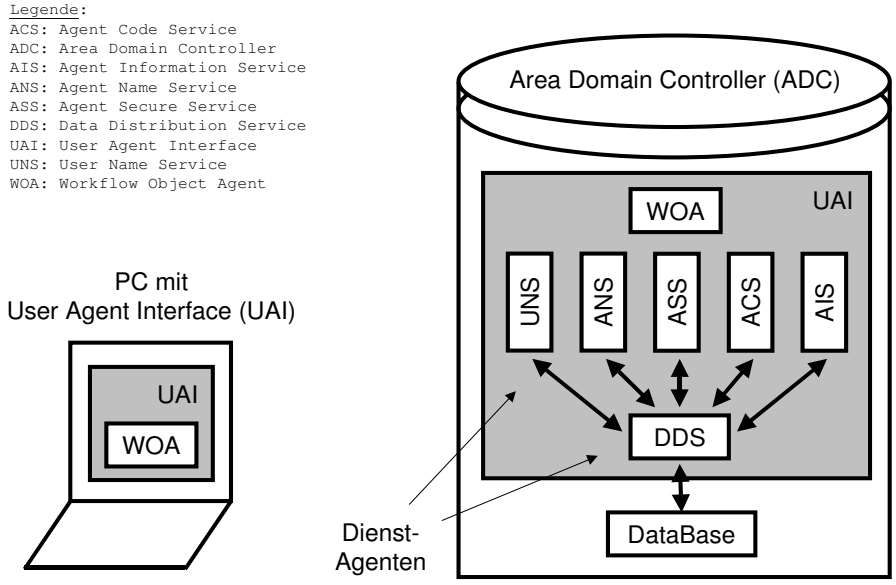


Abbildung 5.6: Aufbau eines Area Domain Controllers (ADC) mit Dienst-Agenten (vgl. Abschnitt 5.5.3 “Services”, Seite 83ff) und einer User-Station

(WAD). Abbildung 5.7 zeigt den Zusammenhang zwischen beiden auf.

Local Area Domain (LAD)

Wesentliches **Kennzeichen** der Local Area Domain (LAD) ist die **Verfügbarkeit einer hohen Netzwerkbandbreite** zwischen den zu ihr gehörenden Komponenten, weshalb ein Standort in der Regel eine LAD bildet. Jede LAD verfügt über einen ADC und einen eindeutigen Namen. LADs existieren in der Hierarchie gleichberechtigt nebeneinander und stehen eine Hierarchiestufe unter den Wide Area Domains (WADs).

Wide Area Domain (WAD)

Eine Wide Area Domain (WAD) besteht aus mehreren LADs und verfügt über einen ADC. **Charakteristisch** für die WAD ist die **geringe Netzwerkbandbreite** zwischen den LADs. Eine WAD wird in der Regel von den LADs eines Landes bzw. eines Kontinents gebildet. WADs existieren in der Hierarchie gleichberechtigt nebeneinander und stehen eine Hierarchiestufe über den Local Area Domains (LADs).

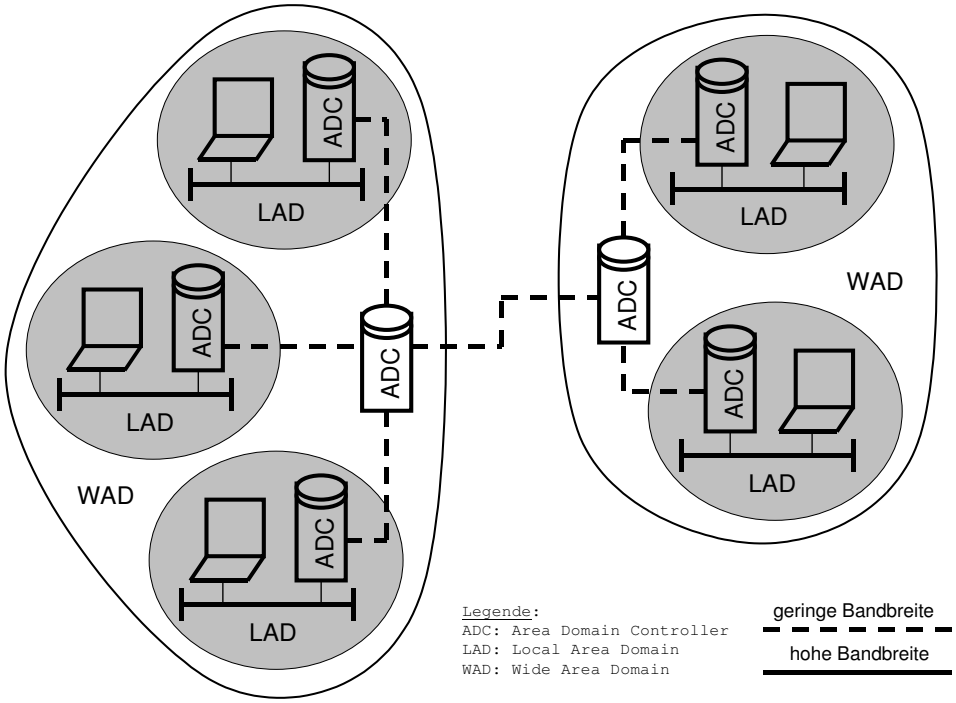


Abbildung 5.7: Strukturierung der Komponenten in eine Local Area Domain (LAD) und eine Wide Area Domain (WAD)

Verwendung der Domains
Die Untergliederung des Systems in Domains dient der **Kontrolle der Kommunikationsbeziehungen** zwischen den Domains. Innerhalb einer Local Area Domain (LAD) dürfen alle Einheiten in der Domain miteinander kommunizieren, da aufgrund der hohen Bandbreitenverfügbarkeit hieraus keine Einschränkungen zu erwarten sind. Kommunikationsbeziehungen über die Grenzen einer LAD hinaus dürfen von den Einheiten innerhalb der LAD nur über den DDS-Dienst (Data Distribution Service, vgl. nächsten Abschnitt "Services") der LAD, der zum Managen der Kommunikationsbeziehungen auf den schmalbandigen Leitungen zwischen den Domains dient, abgewickelt werden.
Mehrere LADs eines Landes oder eines Kontinents werden zu einer WAD zusammengefaßt, wobei der DDS-Dienst der WAD die Kommunikationsbeziehungen zu anderen WADs über schmalbandige Weitverkehrsverbindungen managt.

5.5.3 Services

Weitere Bestandteile des Architekturkonzepts sind Services, die von stationären Dienst-Agenten auf dem Area Domain Controller (ADC) für die jeweilige Domain angeboten werden.

Data Distribution Service (DDS)

Der Data Distribution Service (DDS) dient zur Verteilung und Synchronisation der Datenbestände zwischen den ADC der einzelnen Domains und zum Managen der Kommunikationsbeziehungen zwischen diesen. Hierzu befindet sich auf jedem ADC ein DDS-Agent (stationärer Agent), der mit anderen DDS-Agenten anderer ADC über die WAN-Strecken mit geringer Bandbreite kommuniziert. Zur Kommunikaton werden Technologien wie CORBA, DCOM, RMI oder RPC eingesetzt. Alle WOAs einer LAD und alle Dienst-Agenten müssen Kommunikationswünsche zu anderen Domains über den DDS-Dienst ihrer Domain abwickeln, sofern dieser in ihrer Domain verfügbar ist. Nur Kommunikationswünsche in derselben Domain dürfen direkt abgewickelt werden. Im Falle der Nichtverfügbarkeit des DDS-Dienstes in einer LAD dürfen Dienst-Agenten und WOAs dieser LAD direkt mit dem DDS einer anderen LAD über die WAN-Verbindung kommunizieren. Abbildung 5.8 zeigt den Aufbau des DDS.

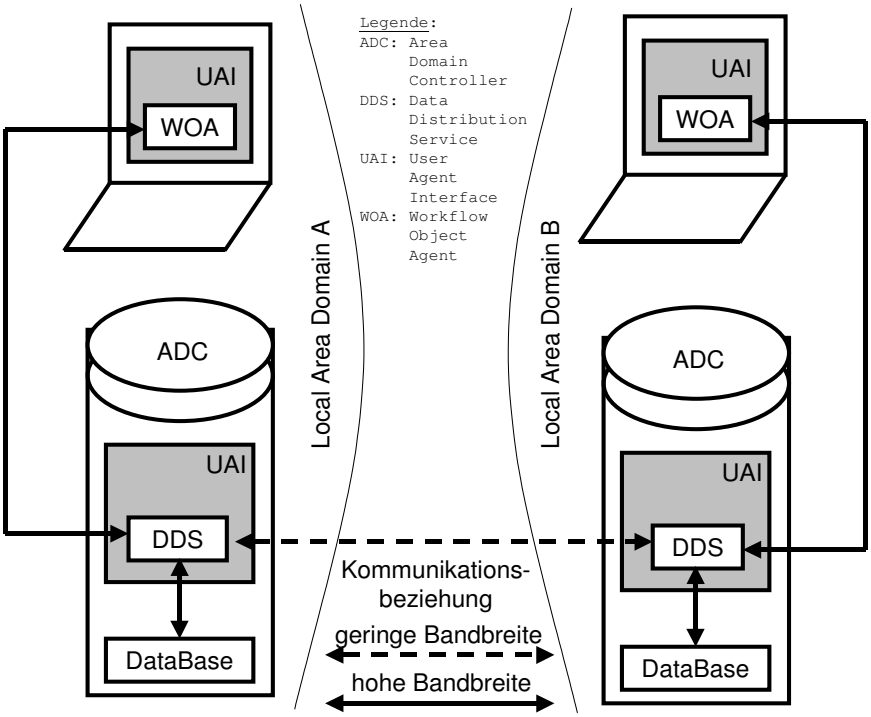


Abbildung 5.8: Aufbau des Data Distribution Service (DDS)

Der DDS in einer Domain dient dazu, alle Kommunikationswünsche, die die Domain verlassen und somit über Leitungen mit geringer Bandbreite geführt werden müssen, zentral zu erfassen und zu managen. Hierdurch ist es trotz der geringen Bandbreite zwischen den LADs möglich, zeitkritische Nachrichten innerhalb der Zeitvorgabe erfolgreich zu übertragen, da der DDS einer jeden Domain durch entsprechende Priorisierung dies bewirken kann. Der DDS hat auch die Aufgabe, Datenbestände in der Datenbank der Domain nach

einem vorgegebenen Regelwerk mit Datenbeständen in Datenbanken anderer Domains konsistent zu halten.

User Name Service (UNS)

Der User Name Service (UNS) dient zur Nutzerverwaltung und ist ein Auskunftsdienst zur Ermittelung des Namens des UAIs, an dem ein bestimmter Nutzer arbeitet. Der UNS gibt ferner Auskunft darüber, ob ein Nutzer derzeit im System angemeldet ist. Die primäre Nutzung des Dienstes erfolgt durch die UAIs bei einer Weiterleitung eines WOAs von Nutzer A nach Nutzer B. Hierzu muß der Name des UAIs von Nutzer B als Migrationsziel für den WOA ermittelt werden. Der UNS ordnet den Namen eines Nutzers dem Namen eines UAIs zu. Als Informationsgrundlage dienen die Meldungen der UAIs, die den Namen des Nutzers beim Einloggen an den UNS ihrer Domain senden. Der UNS-Datenbestand einer Domain wird über den DDS mit anderen Domains synchronisiert, so daß alle UNS in allen Domains über den gleichen Datenbestand verfügen. Abbildung 5.9 zeigt den Aufbau des UNS.

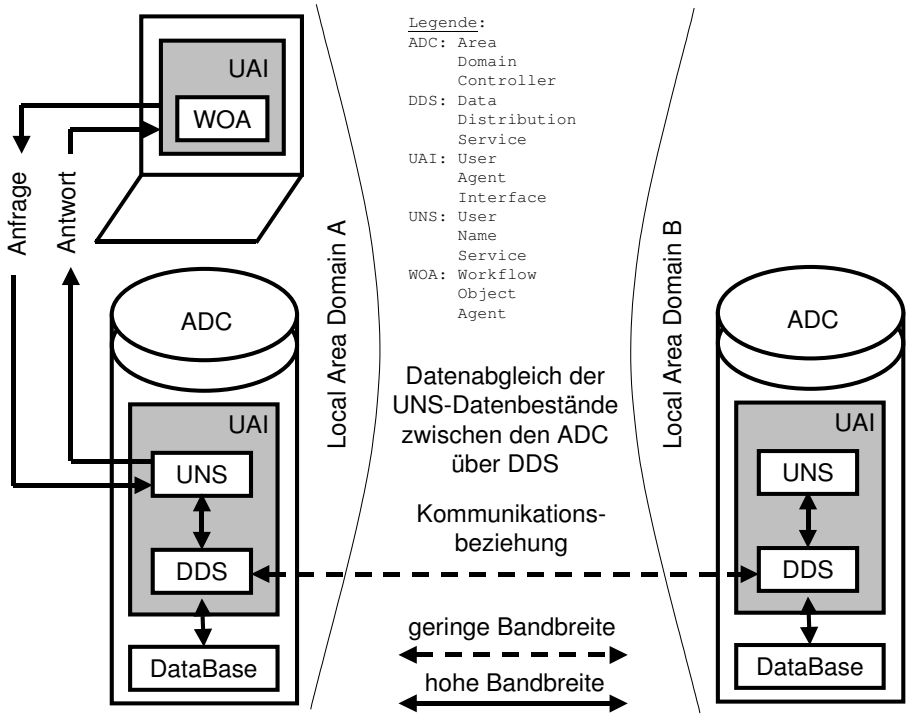


Abbildung 5.9: Aufbau des User Name Service (UNS)

Agent Name Service (ANS)

Der Agent Name Service (ANS) dient zur Zuordnung der WOAs zu den UAI. Mittels dem ANS kann man herausfinden, welche WOAs sich auf einem UAI aufhalten bzw. auf welchem UAI sich ein bestimmte WOA aufhält. Dieser Dienst wird z.B. von einem UAI in Anspruch genommen, wenn der Nutzer an dem UAI einen bestimmten WOA anfordert, um diesem über die Anforderung eine Mitteilung zu schicken. Als Informationsgrundlage dienen die Meldungen der UAIs, die diese bei der Migration eines WOAs zu einem anderen UAI an den ANS senden. Der ANS-Datenbestand einer Domain wird über den DDS mit anderen Domains synchronisiert, so daß alle ANS in allen Domains über den gleichen Datenbestand verfügen. Abbildung 5.10 zeigt den Aufbau des ANS.

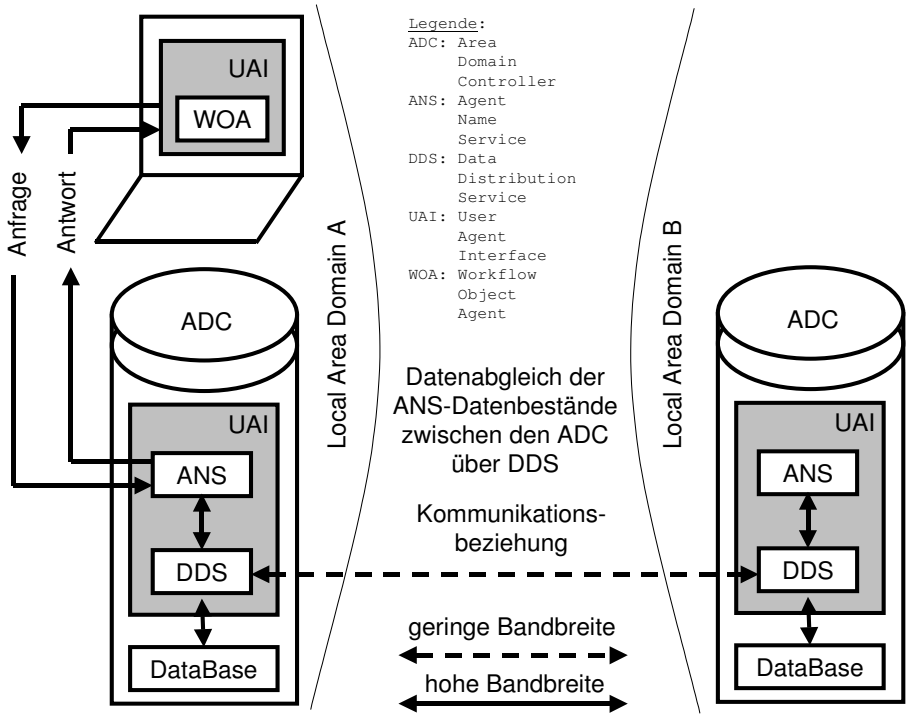


Abbildung 5.10: Aufbau des Agent Name Service (ANS)

Agent Secure Service (ASS)

Der Agent Secure Service (ASS) stellt die "Lebensversicherung" für die WOAs dar. Der ASS erstellt einen WOA anhand von Backup-Daten neu, wenn dieser durch außergewöhnliche Umstände (z.B. Systemabsturz eines UAI) zerstört wurde. Hierzu muß der WOA seinen Agentencode und nach jeder Änderung seinen aktuellen Datenbestand beim ASS seiner Aufenthaltsdomain hinterlegen. Der ASS prüft kontinuierlich die Existenz des WOAs und erzeugt bei negativem Ergebnis eine neue Kopie des WOA. Der ASS ist nur innerhalb

einer LAD verfügbar und kann nur von WOAs innerhalb der LAD genutzt werden, da hierzu höhere Bandbreiten notwendig sind, wie sie innerhalb einer LAD vorhanden sind. Abbildung 5.11 zeigt den Aufbau des ASS.

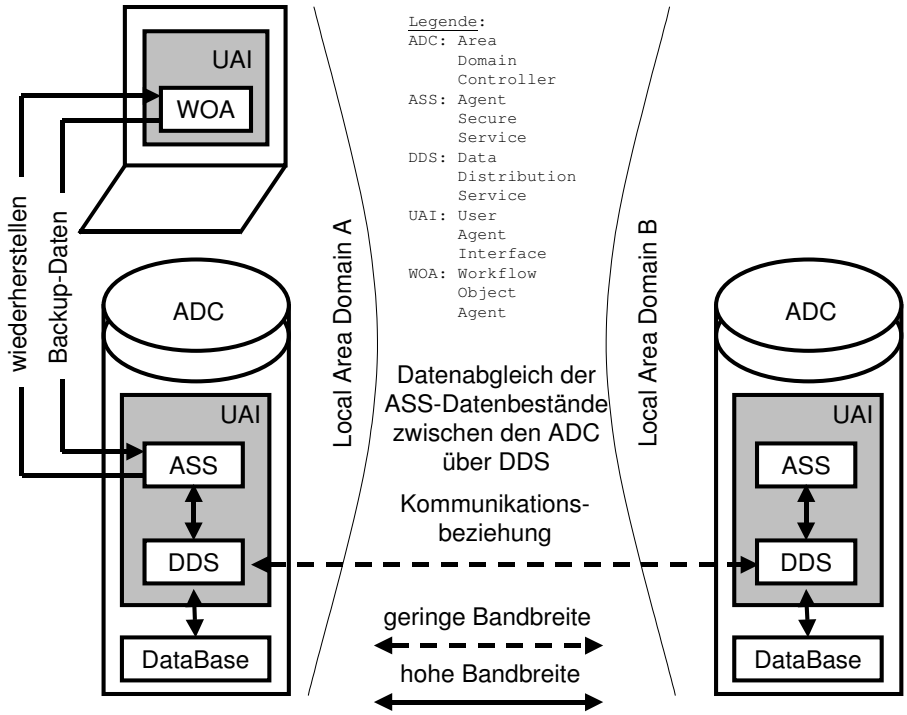


Abbildung 5.11: Aufbau des Agent Secure Service (ASS)

Wird die Verwendung des ASS für WOAs vorgeschrieben und werden die Datenbestände des ASS über den DDS mit anderen ADC synchronisiert, so ist es möglich, an jedem ADC einerseits ein Backup von allen Arbeitsvorgängen im System anzufertigen und andererseits Suchabfragen auf den gesamten Datenbestand der WOAs zu richten.

Hierdurch kann mittels des DDS bei der Migration eines WOAs über eine WAN-Verbindung (z.B. von LAD A zu LAD B) Bandbreite eingespart werden, da durch die Synchronisation der ASS-Datenbestände der ADC der LAD B bereits über alle Daten (Agentencode und Agentendaten) des WOAs verfügt. Insofern ist es nicht erforderlich den WOA komplett (d.h. Agentencode und Agentendaten) über die WAN-Leitung zu übertragen, sondern nur das Migrationskommando. Die Migration des WOAs geschieht dann wie folgt. Der ASS in LAD A eliminiert den WOA und der ASS in LAD B erzeugt eine neue Kopie des WOAs aufgrund der synchronisierten Backup-Daten.

Agent Code Service (ACS)

Der Agent Code Service (ACS) dient dazu, einem UAI den Agentencode für einen WOA-Typ zu liefern, wenn der Nutzer an dem UAI einen neuen WOA kreiert. Der Bestand der WOA-Typen des ACS wird vom Administrator verwaltet. Über den DDS kann man die Datenbestände aller ACSs sychronisieren, so daß systemweit alle WOA-Typen verfügbar sind. Abbildung 5.12 zeigt den Aufbau des ACS.

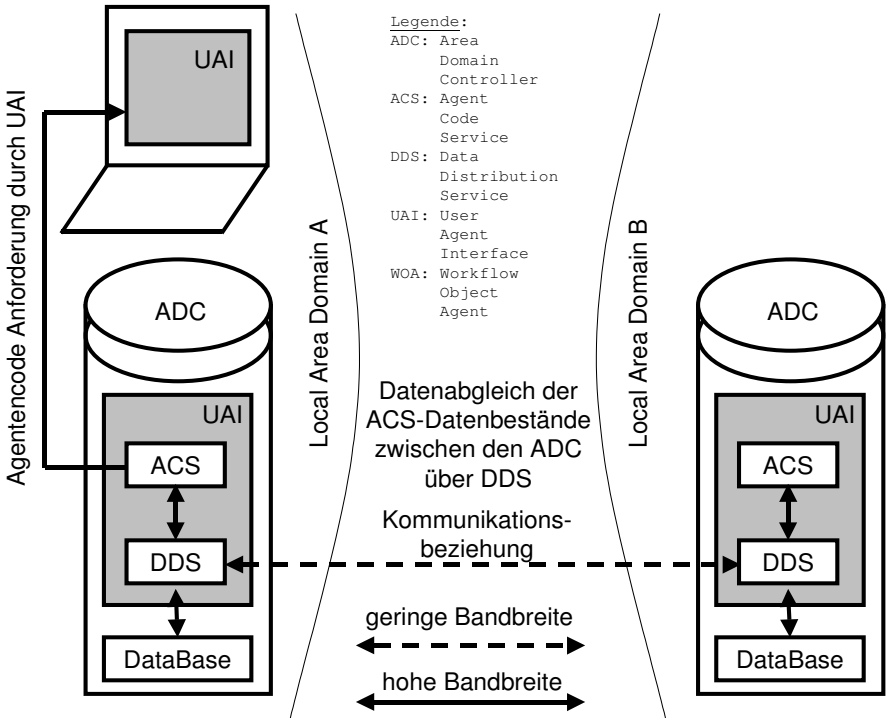


Abbildung 5.12: Aufbau des Agent Code Service (ACS)

Agent Information Service (AIS)

Der Agent Information Service (AIS) stellt für alle Einheiten im System eine Zugangsschnittstelle zu lokalen und globalen Datenbeständen bereit, mittels der die WOAs auf die Datenbestände zugreifen können. Der AIS nutzt den DDS zur Verteilung der Datenbestände. Er dient dazu, allgemein benötigte Datenbestände (wie z.B. Kunden- und Lieferantenadressen, Lagerbestände, Seriennummern, Artikelnummern und -bezeichnungen, internes Telefonbuch usw.) für die WOAs in den einzelnen LAD verfügbar zu machen. Der AIS sychronisiert mittels des DSS die einzelnen Teile seines Datenbestands nur mit den ADCs, auf denen die Teile auch benötigt werden, d.h. es wird nicht der komplette Datenbestand des AIS systemweit über alle ADC synchronisiert. Abbildung 5.13 zeigt den Aufbau des AIS.

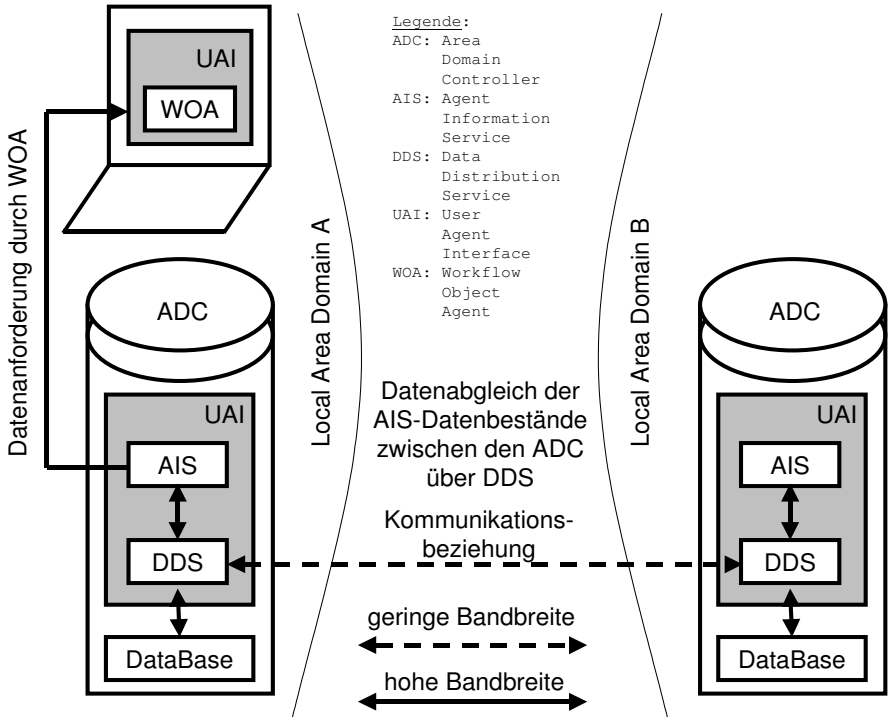


Abbildung 5.13: Aufbau des Agent Information Service (AIS)

5.6 Vergleich zur Client-Server Architektur

Vergleicht man das erweiterte Applikationskonzept auf Basis mobiler Agenten mit dem Client-Server Konzept, so sind folgende Unterschiede festzustellen.

Performance der Applikation aus Anwendersicht Anwender beurteilen die HelpDesk-Applikation subjektiv danach, wie gut sie damit ihre Arbeit verrichten können. Ein wesentlicher Punkt ist hier die Performance, mit der das System arbeitet bzw. wie schnell es auf Aktionen der Anwender reagiert. Bei einem System auf Basis einer Client-Server Architektur ist die Performance-Bremse an Standorten, die über keinen Server verfügen, die WAN-Verbindung zu dem Server-Standort. Die Systemarchitektur auf Basis mobiler Agenten läßt in diesem Punkt Vorteile erwarten, da der Agentencode und die Daten eines Vorgangs (WOA) direkt auf dem PC des Anwenders ausgeführt werden, was einen geringeren Bedarf an Netzkommunikation erfordert und die im System verfügbare Rechenleistung besser ausnutzt. Benötigt der Anwender (bzw. der WOA) Daten von zentralen Diensten (z.B. eine Kundenadresse von AIS), wird diese Anforderung schneller von einem ADC im lokalen LAN zu bedienen sein (Architektur auf Basis mobiler Agenten) als von einem zentralen Server über die WAN-Verbindung (Client-Server Architektur). Die Zeitdauer der Weiterleitung eines Jobs zu einem anderen Anwender (vor allem,

wenn dieser in einer anderen LAD ist) wird bei einem Agentensystem höher sein als bei einen Client-Server System, da neben den Daten u.U. auch der Agentencode des WOAs übertragen werden muß. Diese schlägt sich aber nicht auf die vom Anwender subjektiv empfundene Systemperformance nieder, da in der Zeitdauer der Weiterleitung der Anwender nicht auf eine Reaktion des Systems wartet.

Verfügbarkeit bzw. Ausfallsicherheit Ein Client-Server System hat zwei wesentliche Achillesfersen; den Server an sich und die Datenverbindungen der Clients zu dem Server, die im wesentlichen an die Verfügbarkeit der WAN-Verbindung zu den Standorten ohne Server gebunden ist. Bei Ausfall des Servers steht das komplette System still. Bei Ausfall einer WAN-Verbindung zu einem Standort, steht der komplette Betrieb des Standorts still. Die Systemarchitektur auf Basis mobiler Agenten verfügt nicht über Schwachpunkte, die einen vollständigen Ausfall bewirken können. Vielmehr sind hier Ausfälle von Teilfunktionen zu erwarten, d.h. der Ausfall einer WAN-Verbindung schränkt die Migrationsmöglichkeiten der Agenten ein bzw. beim Ausfall von zentralen Auskunftsdiensten stehen diese Services nicht mehr zur Verfügung.

Bandbreitenbedarf im WAN Der Bandbreitenbedarf im WAN wird bei dem vorgestellten Agentensystem geringer sein, als bei einem Client-Server System. Hierfür sind im wesentlichen zwei Gründe anzuführen. Zum einen ist bei der Bearbeitung eines Jobs keine Client-Server Kommunikation über die WAN-Verbindung nötig, da der Agent lokal auf dem PC des Anwenders bearbeitet wird. Zum anderen läßt sich der Bedarf an WAN-Bandbreite durch den DDS zentral managen, was weitere Möglichkeiten zur Reduktion des Bedarfs bietet.

Erstellung von Statistiken Bei der Erstellung von Statistiken ist die zentralistische Client-Server Architektur gegenüber einem System auf Basis mobiler Agenten klar im Vorteil. Auf dem Server sind alle Daten, die die Grundlage der Statistiken bilden, aktuell vorhanden und schnell erreichbar. Die oben vorgestellte verteilte Architektur würde dies nur mittels des ASS und der Verteilung der ASS-Daten über den DDS leisten. Vor Gewinnung der Rohdaten für eine Statistik müßte man sicherstellen, daß die Datenbestände des ASS auf allen ADCs über den DDS synchronisiert sind. Im laufenden Produktivbetrieb wird dieser Zustand nicht häufig anzutreffen sein, da die Sychronisation über die WAN-Strecken mit einem Zeitverzug verbunden ist, der sich in unsynchronisierten Datenbeständen zwischen den ADCs niederschlägt.

5.7 Fazit

Eine HelpDesk-Applikation auf Basis mobiler Agenten bietet gegenüber einem Client-Server System **Vorteile bei der Ausnutzung der im Netz verfügbaren Bandbreite**, wird aber den Anforderungen nicht gerecht. Um diese Anforderungen zu erfüllen, wurde das Applikationskonzept mobiler Agenten erweitert. Als **wesentlicher Unterschied** zwischen beiden Architekturen ist festzuhalten, daß bei einer **Client-Server Architektur** überwiegend **vollständige Systemausfälle** zu erwarten sind, entweder in Teilbereichen

des Systems (bei Ausfall von WAN-Strecken) oder komplett (beim Serverausfall). Im Gegensatz hierzu sind beim **Agentensystem** eher **Ausfälle von Teilfunktionalitäten** des Systems zu erwarten. Für **statistische Auswertungen** eignen sich Client-Server Systeme besser, da hier die Rohdaten aktuell und zentral vorliegen. Hinsichtlich der vom Anwender **subjektiv empfundenen Systemperformance** steht das erweiterte Agentensystem aufgrund der lokalen Datenverarbeitung besser da.
Die **Vorteile**, die man sich von einem Applikationskonzept auf Basis mobiler Agenten versprochen hat, sind **nur bedingt vorhanden**, da man zentralistische Dienste zum Systemmanagement benötigt, die eine Client-Server Struktur aufweisen. Der zu erwartende **Aufwand** hinsichtlich des Aufbaus und des Betriebs eines Systems auf Basis einer verteilten Architektur wird aufgrund der höheren Anzahl der Komponententypen, die ein solches System umfaßt, **größer** sein als bei einer Client-Server Architektur (bei dem es nur zwei Komponententypen gibt, den Server und den Client). Aus dem Blickwinkel von **Sicherheitsaspekten** betrachtet bietet ein Agentensystem potentiellen Angreifern mehr Angriffsfläche als eine Client-Server System, bei dem der Zugang zu den Datenbeständen zentral gesteuert und überwacht werden kann. Auch sind hier die Aktivitäten der Anwender leichter zu kontrollieren.

Kapitel 6

Fazit

Um einen störungsfreien Netzwerkbetrieb zu erreichen ist eine schnelle Kommunikation zwischen den Servicemitarbeitern und die Bereitstellung der von ihnen benötigten Daten notwendig. Zur **IT-technischen Unterstützung der Kommunikation** werden **HelpDesk-Applikationen** eingesetzt (vgl. Abbildung 2.4, Seite 19). Die Beschreibung der Struktur und der Funktionweise einer HelpDesk-Applikation erfolgt mit den herstellerproprietären Applikationsmodellen der verschiedenen Entwicklungs- und Betriebsumgebungen. Diese sind einerseits zueinander inkompatibel, andererseits erlauben sie nur eine Modellierung auf rechnernaher Ebene. Zur **Verbesserung der IT-Unterstützung** ist in dieser Arbeit ein **generisches Modell für HelpDesk-Applikationen** entwickelt worden (vgl. Abbildung 4.1, Seite 39), das als **Informationsverwaltungs-** und **Informationsaustauschsystem** die Kommunikation unter den Servicemitarbeiter unterstützt. Es ermöglicht die Modellierung einer HelpDesk-Applikation auf Geschäftslogikebene und erleichtert die Erstellung eines rechnernahen Applikationsmodells für die Implementierung. **Kernpunkt des generischen Modells** ist die Entwicklung eines **zustandsraumbasierten Steuerungssystems**, welches eine **neue Art der Steuerung, Regelung und Überwachung von Informationsströmen** in HelpDesk-Systemen darstellt (vgl. Abbildung 4.11, Seite 49). Der **Nachweis der Anwendbarkeit des generischen Modells** und des **zustandsraumbasierten Steuerungssystems** ist durch die **prototypische Implementierung einer HelpDesk-Applikation** erbracht worden (vgl. Abschnitt 4.6, Seite 63ff). Die in dieser Arbeit untersuchten HelpDesk-Applikationen basieren auf einer Client-Server Architektur. Ob eine **andere Architekturform** als Client-Server **Vorteile für eine HelpDesk-Applikation** aufweist, wurde **exemplarisch** anhand eines **Systems auf Basis mobiler Agenten** untersucht (vgl. Kapitel 5, Seite 73ff). Festzustellen ist, daß ein Architekturkonzept auf Basis mobiler Agenten **Vorteile bei der Ausnutzung der im Netzwerk verfügbaren Bandbreite** bietet. Damit das Agentensystem die Anforderungen erfüllen kann, die ein praktischer Einsatz fordert, ist die Einführung zentralistischer Dienste zum Systemmanagement in das Agentensystems notwendig. Diese weisen eine Client-Server Struktur auf, so daß die Vorteile eines Architekturkonzept für HelpDesk-Applikationen auf Basis mobiler Agenten nur eingeschränkt vorhanden sind.

Teil III

Technische Systeme zur automatischen Entstörung von Netzwerken

Kapitel 7

Automatisierungsansatz für das Netzwerkmanagement

7.1 Einleitung

Der Schwerpunkt des letzten Teils lag auf Konzepten zum Aufbau von HelpDesk-Applikationen, die die Mitarbeiter im HelpDesk beim Austausch von Informationen hinsichtlich der Störungsbearbeitung unterstützen, wobei die HelpDesk-Applikation steuernd, regelnd und überwachend in die Informationsströme eingreift. Die Qualität des Entstörungsmanagements hängt hierbei vom Wissen und der Erfahrung der Mitarbeiter ab. Um jedoch ein Fehlermanagement von gleichbleibender Qualität in komplexen Netzwerken zu betreiben, sollte das Entstörungsmanagement zukünftig unabhängig vom Wissensstand der Mitarbeiter sein. Dies kann mit einer Automatisierung des Fehlermanagements erreicht werden, welches Entstörungen ohne menschlichen Eingriff vornimmt. Vor diesem Hintergrund bildet die Betrachtung von Ansätzen zur Automatisierung des Netzwerkmanagements den Schwerpunkt dieses Kapitels.

7.1.1 Netzwerkfehler

Unter einem **Fehler** ist das Abweichen des Verhaltens eines Netzwerks von einer definierten Norm zu verstehen. Zur Beseitigung der Abnormität ist ein Eingriff in das Netzwerk erforderlich.

7.1.2 Kategorisierung von Netzwerkfehlern

Um die Vielzahl der Fehler in eine Ordnungsstruktur einzubetten, ist eine Kategorisierung notwendig. Nachfolgende **Merkmale zur Kategorisierung** sind durch Beobachtung der Praxis bzw. in [Steinder01] zu finden (siehe Abbildung 7.1):

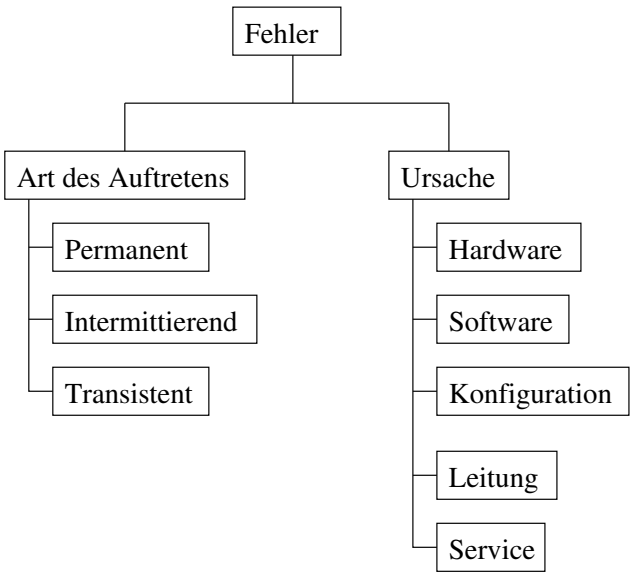


Abbildung 7.1: Kategorisierung von Netzwerkfehlern

Art des Auftretens

Permanente Fehler
Permanente Fehler existieren bis zu ihrer Beseitigung (und sind permanent durch Messung nachzuweisen).

Intermittierende Fehler
Intermittierende Fehler treten sporadisch und zusammenhanglos auf. Hierdurch ist die Fehlerursache schwerer zu ermitteln, da Messungen zur Fehlereingrenzung nicht permanent möglich sind. Fehler dieser Art sind vielfach von externen Einflüssen abhängig (z.B. kalte Lötstellen, thermische Einflüsse, Witterungsbedingungen, usw.).

Transistente Fehler
Transistente Fehler treten kurzzeitig auf und erfordern keinen Eingriff in das Netzwerk, da sie eigenständig in einen fehlerfreien Zustand übergehen. Als Beispiel ist hier der Spanning Tree Algorithmus (IEEE 802.1) zu nennen, der eine Zeit benötigt, ehe die Konvergenz des Netzwerks erreicht ist. In dieser Zeit können Schleifen (engl. Loops) und Verbindungsabbrüche auftreten, die als Fehler erkannt werden können. Diese verschwinden, wenn der Spanning Tree konvergiert und das Netzwerk schleifenfrei ist.

Fehlerursache

Hardwarefehler
Eine managebare aktive Netzwerkkomponente produziert einen Hardwarefehler,

wenn der Fehler nur durch Austausch der Komponente bzw. eines Moduls der Komponente behoben werden kann. Wesentliches Kennzeichen eines Hardwarefehlers ist, daß er vom Servicepersonal ausschließlich durch einen mechanischen Eingriff am Standort des Geräts behoben werden kann. Eine Entstörung ist durch den Austausch einer Softwarekomponente bzw. durch eine Änderung der Konfiguration des Geräts nicht möglich. Dies gilt auch für redundante Systeme. Hier können die Auswirkungen eines Hardwareausfalls durch Umschaltung auf die Redundanz vermieden werden. Die Wiederherstellung des ursprünglichen Zustands erfordert den Austausch der ausgefallenen Komponente.

Leitungsfehler
Ein Leitungsfehler ist der Ausfall einer logischen Datenverbindung zwischen zwei managebaren Netzwerkkomponenten. Bei dieser Sichtweise auf eine Leitung werden auch Modems, Patch-Panel usw. zur Leitung gezählt. Ein Leitungsfehler liegt auch dann vor, wenn die Qualität der Datenübertragung für den benötigten Zweck nicht ausreichend ist.

Softwarefehler
Ein Softwarefehler liegt dann vor, wenn die Betriebssoftware eines Netzwerkgeräts eine Fehlfunktion aufweist, die durch den Austausch der Softwarekomponente oder ein "Reset" beseitigt werden kann. Eine Fehlerbeseitigung durch eine Änderung der Parametrisierung des Geräts ist nicht möglich. Ein wesentliches Merkmal des Softwarefehlers ist die Auswechselbarkeit der betroffenen Softwarekomponente ohne den Austausch einer Hardwarekomponente.

Konfigurationsfehler
Ein Konfigurationsfehler ist ein Fehler in der Parametrisierung der Betriebsdaten einer managebaren Netzwerkkomponente. Fehler dieser Art können in der Regel durch Austausch der Konfigurationsdaten per Remotezugriff behoben werden.

Servicefehler
Ein Servicefehler liegt dann vor, wenn ein im Netzwerk angebotener Dienst gestört ist. Beispiele hierfür sind der DHCP-Dienst, Mail-Server, File-Server, Print-Server usw. Ein Servicefehler hat seine Ursache vielfach in einer defekten Hardware, fehlerhafter Software oder in einem Konfigurationsfehler einer zur Serviceerbringung notwendigen Komponente. Allerdings werden Hardware-, Software-, und Konfigurationsfehler, die direkt an einer Komponente zur Serviceerbringung (z.B. Server) auftreten, als Servicefehler klassifiziert.

7.1.3 Häufigkeit des Auftretens von Fehlern

Interessant ist die Frage, wie sich beim Netzwerkbetrieb die **Verteilung der Fehler nach o.g. Kategorisierung** darstellt. Hierzu wurden die Trouble-Tickets eines Network Operations Centers (NOC) ausgewertet. Für die Fehlerursache ergibt sich eine Verteilung nach Abbildung 7.2. Hierbei ist anzumerken, daß zur Häufigkeit von Servicefehlern an

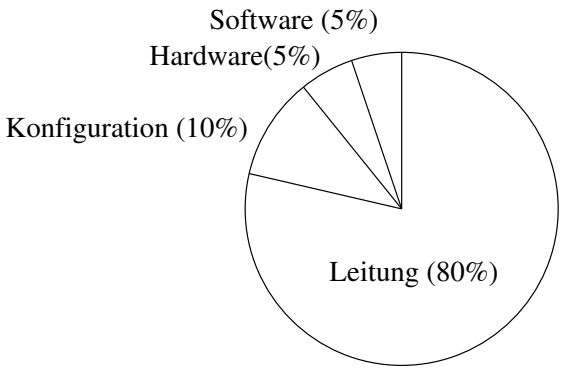


Abbildung 7.2: Verteilung der Fehlerursachen beim Netzwerkmanagement eines Network Operations Centers (NOC)

dieser Stelle keine Aussage möglich ist, da Servicefehler im o.g. NOC nicht bearbeitet werden.

Aus der Untersuchung folgt, daß mehr als drei Viertel der vom NOC zu bearbeitenden Störungen Leitungsstörungen sind. Aus diesem Grund sollten **Ansatzpunkte zur Automatisierung des Netzwerkmanagements** primär zur Bearbeitung von Leitungsstörungen gesucht werden, da eine Automatisierung in diesem Bereich den größten Nutzen für die Arbeitsentlastung der Operator bringt.

7.1.4 Komplexität des Entstörungsmanagements

Nach dem Auftreten und der Diagnose eines Fehlers werden entsprechende Maßnahmen zum Troubleshooting ergriffen. Das **Komplexitätniveau** der Troubleshootingmaßnahmen ist für die einzelnen Fehlerkategorien unterschiedlich (siehe Abbildung 7.3).

Leitungsfehler
: Die Leitungen im WAN-Bereich werden vielfach nicht vom Supportdienstleister selbst betrieben, sondern von Carriern angemietet. Insofern beschränkt sich das Entstörungsmanagement bei einem Leitungsausfall auf die Information des jeweiligen Carriers und das Aktivieren einer Backupleitung für die Dauer des Ausfalls. Die defekte Leitung wird ständig geprüft und bei Wiederherstellung der Verbindung die Backupleitung abgeschaltet. Die Tätigkeit des Entstörungsmanagements ist für einen Leitungsfehler somit von geringer Komplexität.

Softwarefehler
: Ein Softwarefehler in einem Netzwerkgerät kann meistens nicht vom Supportdienstleister behoben werden. In diesem Fall wird der Gerätehersteller von der Fehlfunktion informiert und es wird von diesem ein Patch oder eine andere Version zur Verfügung gestellt. Die Komplexität eines Softwarefehlers hat zwei Ursachen. Zum einen ist ein

Testaufbau erforderlich, um den Fehler zu ermitteln und zu beschreiben. Andererseits muß der Patch oder das Update des Geräteherstellers auf eine Korrektur des Fehlers hin überprüft werden. Der Austausch der Software in dem Gerät kann vielfach remote erfolgen. Da zum Austausch der Produktivbetrieb unterbrochen werden muß, erfordert eine solche Aktion bei wichtigen Netzwerkkomponenten einen entsprechenden Organisationsaufwand.

Konfigurationsfehler
Konfigurationsfehler treten vielfach bei Erweiterungen bzw. bei größeren Änderungen im Netzwerk auf. Sie resultieren aus einem schlechten Netzdesign. Die Fehlerbearbeitung kann ein zeitraubender Prozeß sein, welcher nicht selten in einem logischen Neudesign des Netzwerks endet.

Servicefehler
Beim Servicefehler sind einerseits eine geringe Anzahl von Netzwerkkomponenten betroffen, weshalb Troubleshootingmaßnahmen nur auf einem begrenzten Gebiet notwendig sind, andererseits laufen in den Netzwerkkomponenten komplexe Software-Anwendungen, was letztendlich eine hohe Komplexität der Entstörung dieser Fehler nach sich zieht.

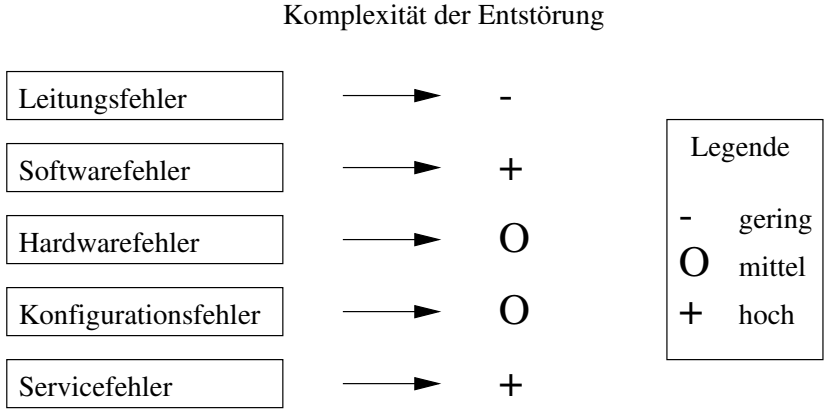


Abbildung 7.3: Komplexität der Entstörung für die verschiedenen Fehlerarten

7.1.5 Ansatzpunkte und Grenzen bei der Automatisierung des Netzwerkfehlermanagements

Bei dem Vorhaben das Netzwerkfehlermanagement zu automatisieren stellt sich zuerst die Frage, wo die **Grenzen der Automatisierung** liegen. Der wesentliche Charakter des Netzwerkmanagements liegt darin, remote Parameter aus Netzwerkgeräten abzufragen bzw. Meldungen von dort zu empfangen. Änderungen in dem Verhalten der Netzwerkkomponenten können mittels Remotekommandos an diese bewirkt werden (siehe Abbildung 7.4).

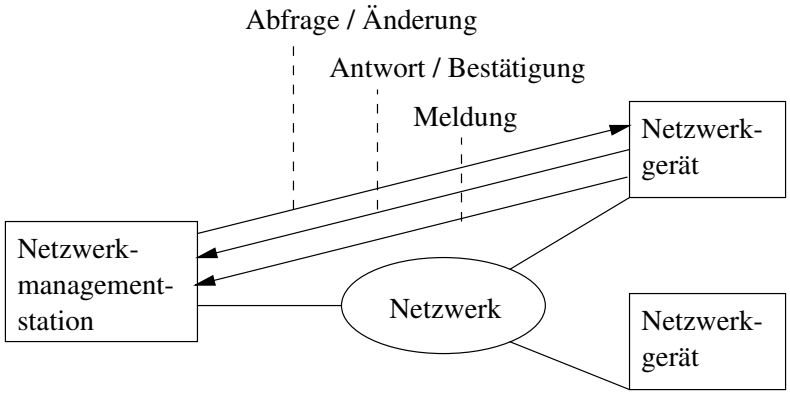


Abbildung 7.4: Grundsätzliche Funktionsweise des Netzwerkmanagements

Insofern ist prinzipiell alles der Automatisierung zugänglich was auf dem grundsätzlichen Funktionsprinzip (siehe Abbildung 7.4) basiert. Speziell für die **Möglichkeit des automatischen Fehlermanagements** ergeben sich folgende Aspekte:

Hardwarefehler
: Hardwarefehler erfordern zur Beseitigung den Austausch der betroffenen Hardwarekomponente. Die Beseitigung der Auswirkung eines Hardwarefehlers durch ein automatisches Netzwerkmanagement beschränkt sich auf die Delegation der Aufgaben, die die defekte Komponente nicht mehr erfüllen kann, auf andere Netzwerkkomponenten. Ferner nach Austausch der Hardware den ursprünglichen Zustand wiederherzustellen.

Softwarefehler
: Bei einem Softwarefehler wäre der erste Schritt zur Automatisierung der Austausch des betroffenen Softwaremoduls durch eine andere Version oder ein Patch, sofern verfügbar. Schwieriger wird es sein, ausgehend von einer Fehlerdiagnose vollautomatisch den Fehler in der defekten Software zu beseitigen bzw. ein Patch zu erstellen.

Konfigurationsfehler
: Wie bei Softwarefehlern wäre bei Konfigurationsfehlern der erste Schritt zur Automatisierung der Austausch des Konfigurationsfiles auf der betroffenen Netzwerkkomponente. Da sich die Konfiguration einer Komponente aus dem Netzwerkdesign ableitet, ist die vollautomatische Erstellung eines Konfigurationsfiles im Sinne des Netzwerkdesigns eine anspruchsvolle Aufgabe.

Leitungsfehler
: Für Leitungen, die gegen Ausfall zu sichern sind, existieren Ersatzleitungen, die im Störungsfall aktiviert werden. Eine Automatisierung der Bekämpfung von Leitungsfehlern besteht darin, Leitungsausfälle automatisch zu erkennen und die entsprechende Backupleitung zu schalten. Bei aktiviertem Backup wird der Status der gestörten

Leitung ständig geprüft. Nach erfolgreicher Entstörung wird auf die ursprüngliche Leitung zurückgeschaltet und die Backupleitung deaktiviert. Diese Tätigkeit gilt bei den Operatorn als “Monkey-Job”, den ihnen eine Automatisierung abnehmen könnte.

Servicefehler
Wie beim Leitungsfehler wird beim Servicefehler der erste Schritt zur Automatisierung die Erkennung von Fehlern sein. Dies wird dann genutzt, um auf entsprechende Redundanzsysteme umzuschalten. Die Beseitigung der Störungsursache an dem ausgefallenen System ist oftmals ein komplexer Prozeß, der schwer zu automatisieren sein wird.

7.1.6 Fazit

Abschließend ist festzustellen, daß eine **Automatisierung des Netzwerkmanagements** primär darauf ausgerichtet sein wird, **Ausfälle zu erkennen** und durch **Umschaltung auf Redundanzen** die Auswirkung eines Ausfalls für den Nutzer des Netzwerks gering zu halten. Ansätze zur Automatisierung bieten sich in folgenden Punkten:

1. Automatische Erkennung von Standard-Fehlersituationen im Netzwerk.

2. Basierend auf der Fehleridentifikation die Durchführung der zugehörigen Standard-Entstörungsstrategie.

Eine Automatisierung der Entstörung von Standard-Situationen entlastet die Operator von Routine-Aufgaben. Unter Berücksichtigung der Häufigkeit des Auftretens und der Komplexität der Entstörung bietet sich der **Bereich der Leitungsstörungen** als **Ausgangspunkt für eine Automatisierung** an.

7.2 Architekturmodelle zum Aufbau eines automatischen Fehlermanagements

7.2.1 Umbrella-Management-Architektur

Die Netzwerkmanagementsysteme in der Vergangenheit waren darauf ausgelegt, den Operator mit so vielen Informationen wie möglich zu versorgen. Ferner einen weitgehenden Remotezugriff auf die Elemente des Netzwerk zu ermöglichen. Die Erfahrung zeigte jedoch, daß für das Fehlermanagement Informationen aus dem Netz für die Lösungsfindung wichtig sind, jedoch der Informationsgehalt vieler Events (ein gemeldetes Ereignis aus dem Netzwerk) sehr gering ist. Ziel moderner Netzwerkmanagementsysteme ist es deshalb, die Events auf eine minimale Anzahl mit maximalem Informationsgehalt zu reduzieren. Hierzu verwenden sie die **Umbrella-Management-Architektur**, die alle in [Krause03] untersuchten Netzwerkmanagement-Applikationen als gemeinsames Merkmal aufweisen (siehe

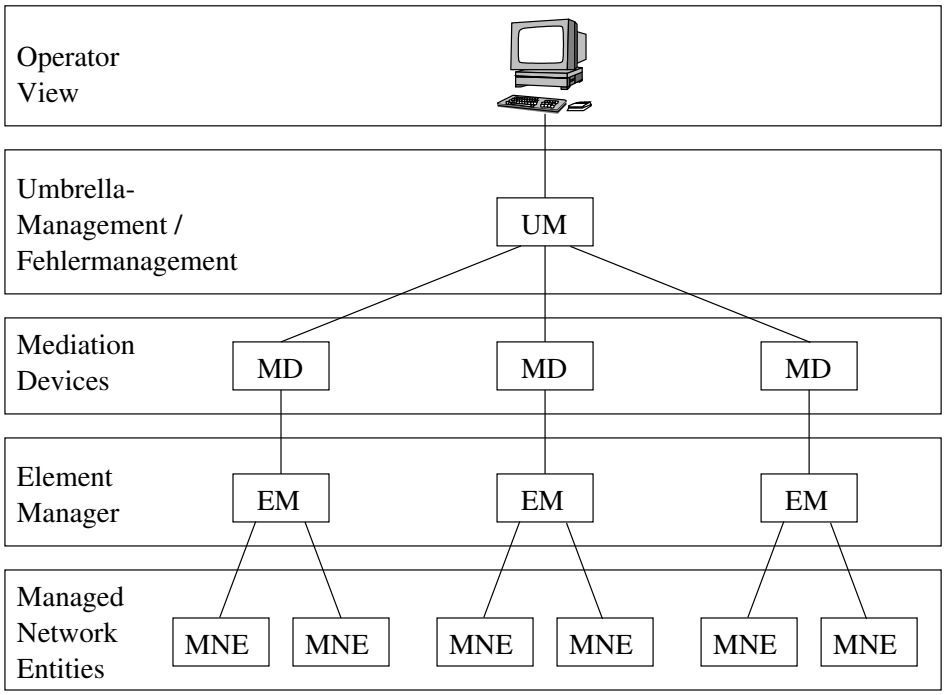


Abbildung 7.5: Umbrella-Management-Architektur nach [Krause03]

Abbildung 7.5). Sie ermöglicht eine einheitliche Sicht auf die verschiedenen Datenquellen und eine Minimierung der vom Operator zu bearbeitenden Event-Anzahl.
Ziel zukünftiger Systeme ist es, Tätigkeiten der Operator teil- oder vollzuautomatisieren. Die Umbrella-Management-Architektur besteht aus **fünf Ebenen**:

(1) Managed-Network-Entities (MNE)
Die Managed-Network-Entities (MNE) sind die zu überwachenden Netzwerkgeräte, deren Events vom Netzwerkmanagement ausgewertet werden.

(2) Element-Manager
Element-Manager sind Softwarekomponenten zur Konfiguration und Überwachung von Netzwerkgeräten. Diese werden von dem Netzwerkgerätehersteller angeboten und bieten proprietäre Konfigurations- und Überwachungsmöglichkeiten.

(3) Mediation-Devices
Mediation-Devices sind Softwarekomponenten, die die Einbindung eines Element-Managers in ein Umbrella-Management ermöglichen. Mittels eines entsprechenden Mediation-Devices läßt sich jede Komponente in das Umbrella-Management einbinden.

(4) Umbrella-Management
Die Umbrella-Management-Komponente ist für die Konsolidierung der verschiedenen

Events zuständig. Ziel der Komponente ist eine Vorverarbeitung und Aufbereitung der eingehenden Events, um den Operator nur mit den essentiellen Informationen über Netzwerkfehler zu versorgen.

(5) Operator-View
Bei der Operator-View handelt es sich um eine Benutzeroberfläche, mittels der der Operator mit dem Umbrella-Management kommuniziert.

Die Umbrella-Management-Struktur ist darauf ausgelegt, dem Operator eine umfassende Sichtweise auf die für ihn relevanten Daten des Netzwerks zu ermöglichen. Eine auf dieses Ziel ausgelegte Architektur eignet sich nur bedingt zur Automatisierung des Troubleshootings.

7.2.2 Eigenes Architekturmodell zum Aufbau eines automatischen Fehlermanagements

Der Ablauf des Fehlermanagements erfolgt in den vier Schritten **Events sammeln**, **Events filtern**, **Fehlerdiagnose** und **Fehlerbeseitigung** (siehe Abbildung 7.6).

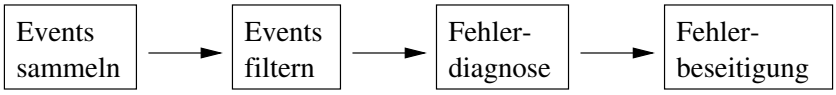


Abbildung 7.6: Ablauf des Fehlermanagements

Um den heute überwiegend manuell betriebenen Vorgang zu automatisieren, wird in dieser Arbeit folgendes Architekturmodell vorgestellt (siehe Abbildung 7.7). Wesentlicher Kern des Architekturmodells sind **vier Engines**, deren Aufgaben folgende sind:

Event Collection Engine (ECE)
Aufgabe der ECE ist das Sammeln von Daten aus allen verfügbaren Quellen, die zur Beschreibung des Netzwerkzustands herangezogen werden können. Zur Beschreibung der Events in einem einheitlichen Format, dem **Event Description Format (EDF)**, werden die gesammelten Daten von der ECE in das EDF konvertiert und zentral an einer Stelle gespeichert.

Event Filter Engine (EFE)
Aufgabe der EFE ist die Grundfilterung der von der ECE gesammelten Daten, um weitere Informationen zu den Daten hinzuzufügen und die Menge der Events zu reduzieren. Die EFE nutzt das EDF zur Beschreibung der Events.

Fault Detection Engine (FDE)
Die Fault Detection Engine liefert auf Basis der gefilterten Events und durch aktive Tests im Netzwerk (die die Konfiguration des Netzwerks nicht ändern dürfen) eine Beschreibung der Fehlerursache (engl. Root Cause). Diese wird aus der Menge der

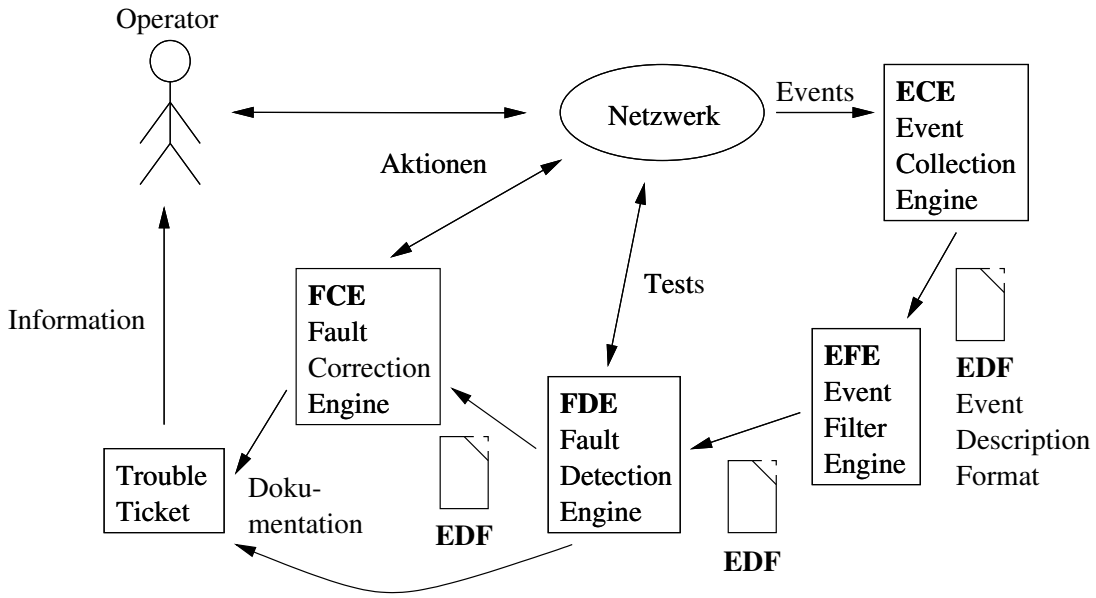


Abbildung 7.7: Eigenes Architekturmodell zum Aufbau eines automatischen Fehlermanagements

Events ermittelt (wenn er darin enthalten ist) oder von der FDE erzeugt. Hierzu wird ebenfalls das EDF verwendet.

Fault Correction Engine (FCE)
Aufgabe der FCE ist es, basierend auf dem Root-Cause Event (Fehlerbeschreibung) durch Aktionen im Netz den Fehler zu beseitigen bzw. zu überbrücken, damit der Nutzer des Netzwerks durch den Fehler nicht beeinträchtigt wird.

Die FDE und die FCE dokumentieren ihre Aktionen in einem Trouble-Ticket. Hierüber können sich die Operator einerseits über den aktuellen Stand der automatischen Störungsbearbeitung informieren. Andererseits können die Operator so die Bearbeitung einer Störung nahtlos übernehmen, wenn die automatische Störungsbearbeitung keine Entstörung durchführen konnte.
Die nachfolgenden Abschnitte beschreiben für jede Engine heute bekannte Technologien, mittels derer sie aufgebaut werden können. ECEs und EFEs sind im Netzwerkmanagement weit verbreitet. FDEs können heute einzelne Netzwerkfehler aus einer vorher festgelegten Menge von Fehlerfällen mit akzeptabler Zuverlässigkeit erkennen. FCEs stecken noch in den Kinderschuhen.

7.2.3 Technologien zum Aufbau einer Event Collection Engine (ECE)

Das Sammeln von Events kann aus verschiedenen Quellen und auf verschiedene Arten erfolgen [Micromuse02/1]. Als Quelle für Events dienen u.a.:

- Netzwerkmanagementprotokolle (z.B. SNMP, Polling & Traps)
- Logfiles
- Datenbanken
- Windows Ereignisanzeige
- UNIX Syslog-Dienst
- usw.

Aufgabe der ECE ist das Sammeln von Informationen aus verschiedenen Quellen. Ferner das Konvertieren der Informationen in ein einheitliches Datenformat, den EDF, und das Ablegen der Events in eine Datenbank. Abbildung 7.8 zeigt den Aufbau der ECE.

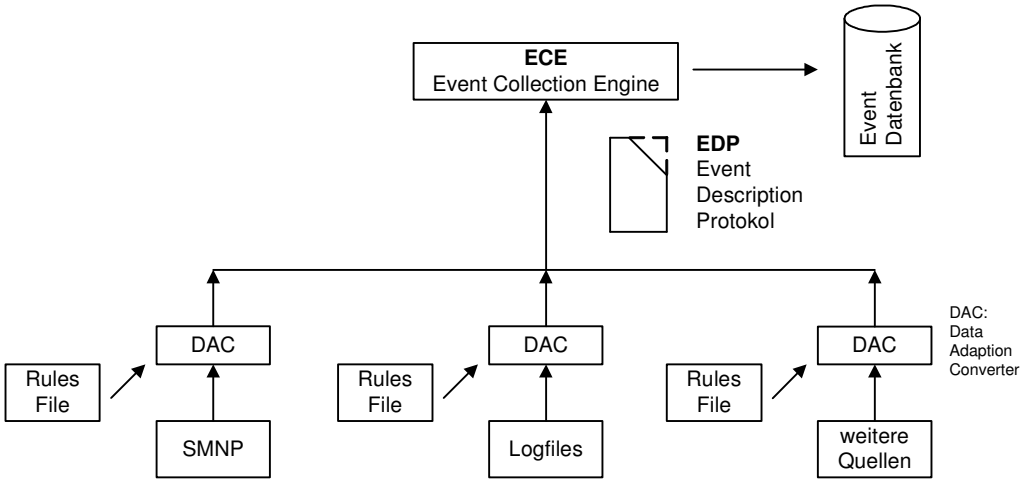


Abbildung 7.8: Aufbau der Event Collection Engine (ECE)

Zur Konvertierung der Datenformate der Events aus den unterschiedlichen Quellen werden für jede Quellenart Adapter benötigt, sogenannte **Data Adaption Converter** (DAC), die das Event einer Quelle in das EDF übertragen. Mittels eines entsprechenden DACs kann jede Datenquelle als Eventquelle verwendet werden. Jeder DAC verfügt über ein Regelwerk (Rules File), mittels dem festgelegt wird, für welche Ereignisse in der Datenquelle ein Event zu generieren ist.

7.2.4 Technologien zum Aufbau einer Event Filter Engine (EFE)

Zur Reduzierung der Anzahl der Events und um den Informationsgehalt der Events zu erhöhen, erfolgt in der EFE eine **Grundfilterung der Events** auf **vier Arten** [Micromuse02/1]:

Deduplizierung

Deduplizierung bedeutet, daß eine Anzahl gleicher Events zu einem Event zusammengefaßt werden. Abbildung 7.9 verdeutlicht dies.

Test_Node	17.04.03	15:48:27	Link up on Port
Test_Node	17.04.03	15:49:40	Link up on Port
...	...	...	...
Test_Node	17.04.03	16:12:32	Link up on Port
Test_Node	17.04.03	16:13:48	Link up on Port

Node	Count	First Occurrence	Last Occurrence	Description
Test_Node	42	17.04.03 15:48:27	17.04.03 16:13:48	Link up on Port

Abbildung 7.9: Deduplizierung in der Event-Darstellung

Klassifikation

Eine Klassifikation ordnet die Events nach einem Zuordnungsschema verschiedenen Klassen von Ereignisarten zu (siehe Abbildung 7.10).

Suppression / Priorisierung

Suppression bzw. Priorisierung von Events bedeutet, daß Events nach ihrer Wichtigkeit sortiert werden (Voraussetzung hierzu ist eine vorherige Klassifikation der Events nach Prioritätsstufen). Events unterhalb einer gewissen Prioritätsstufe werden aussortiert bzw. nicht weitergeleitet (siehe Abbildung 7.11).

Korrelation

Korrelation von Events bedeutet, daß bestimmte Events andere Events in einem bestimmten Zeitintervall neutralisieren können. So kann z.B. das Event **Link down** von einem Port durch das Event **Link up** neutralisiert werden (siehe Abbildung 7.12).

7.2.5 Technologien zum Aufbau einer Fault Detection Engine (FDE)

Ziel der Fault Detection Engine (FDE) ist es, aus einer Menge von vorgefilterten Events und durch aktive Tests im Netzwerk eine Fehlermeldung zu erstellen. Für diese **Root-Cause Analyse (RCA)** kommen verschiedene Verfahren zur Anwendung, die in diesem Abschnitt näher untersucht werden. Die RCA dient zur **Fehlerlokalisierung**, nicht zur

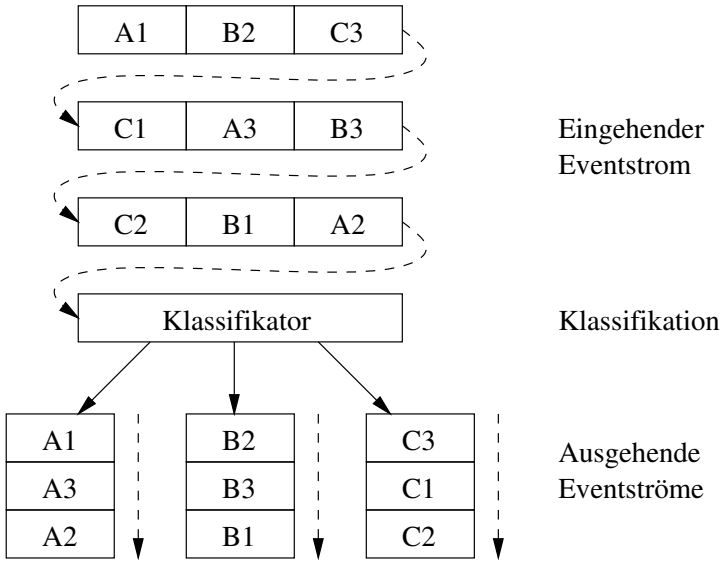


Abbildung 7.10: Klassifikation von Events

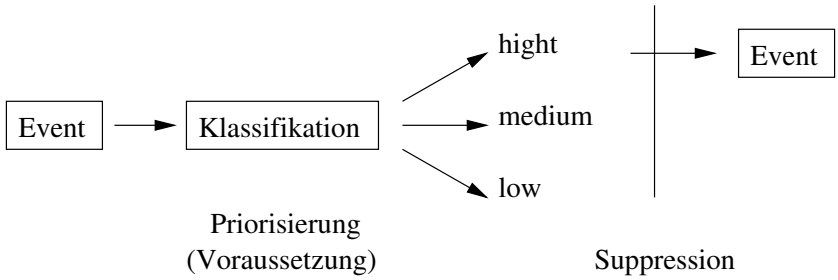


Abbildung 7.11: Suppression bzw. Priorisierung von Events

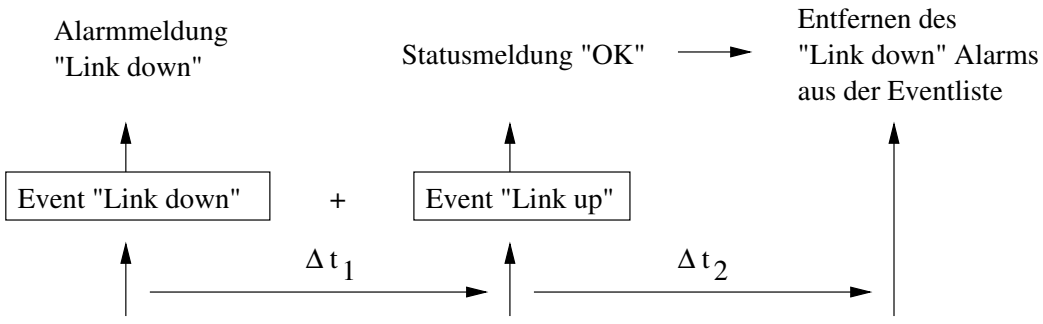


Abbildung 7.12: Korrelation von Events

Fehlerbehebung. Ihr Vorteil liegt darin, daß der Operator direkt an der Fehlerursache mit dem Troubleshooting beginnen kann, ohne vorher eine große Anzahl von Events bearbeiten zu müssen. Vor der Betrachtung der verschiedenen RCA-Verfahren werden zunächst einige im folgenden verwendete Begriffe definiert:

Symptom
: Ein Symptom ist ein Event, welches das Netzwerk-Managementsystem (NMS) über eine Fehlfunktion im Netzwerk informiert.

Fehler
: Ein *Fehler* ist die Ursache für ein oder mehrere *Symptome*, die beim Auftreten des Fehlers erzeugt werden. Das Fehler-Event (sofern es existiert) ist sowohl Fehler als auch Symptom. Existiert das Fehler-Event (auch **RCA-Event** genannt) nicht (z.B. wenn dies durch einen unbestätigten UDP-Transport verlorengegangen ist (Trap)), muß es aufgrund der Symptomlage vom RCA-Verfahren erzeugt werden.

Die **Verfügbarkeit von Topologie-Informationen**, die vielfach von automatischen Topologieerkennungsverfahren erstellt werden, ist **Grundvoraussetzung für die Mehrheit der RCA-Verfahren**. Zur Abbildung der Topologie-Informationen wird vielfach ein objektorientiertes Modell des Netzwerks verwendet. Ein Erkennungsmechanismus instanziiert die verschiedenen Geräteklassen und erstellt hierdurch ein Modell des Netzwerks, welches die Funktionalität und den Aufbau des realen Netzwerks widerspiegelt.

Rule Based Reasoning (RBR)

Ziel des Rule Based Reasoning (RBR) ist es, das Fachwissen des Operators in Regeln zu hinterlegen und diese zur automatischen Korrelation der Events zu verwenden. Hierbei gibt es **zwei Varianten**:

Top-Down: Es wird von einer bestimmten Fehlerhypothese ausgegangen und durch Filterung der Events versucht, diese zu beweisen.

Botton-Up: Basierend auf initialen Informationen wird durch Filterung versucht, in iterativen Schritten eine Diagnose zu erstellen.

Die Funktionsweise des RBR ist in Abbildung 7.13 dargestellt. Der RBR-Agent versucht, die innerhalb der letzten Vergangenheit aufgetretenen Events über einen **Reasoning Algorithmus** zu korrelieren, wobei dieser ein hinterlegtes Regelwerk verwendet. Das Regelwerk besteht aus einer Vielzahl von Regeln, die je nach Variante des Reasoning Algorithmuses in unterschiedlicher Reihenfolge abgearbeitet werden. Die **Problematik des RBR ist, daß die Topologieinformationen in den Regeln stecken** und die komplette Netzwerktopologie über die statischen Regeln formuliert wird. Hieraus resultieren **Nachteile** bei der **Skalier- und Anpaßbarkeit** von RBR-Systemen, da bei Änderungen in der Netzwerktopologie alle Regeln der betroffenen Komponenten angepaßt werden müssen. Der Erstellungs- und Wartungsaufwand für ein Regelwerk, welches ein modernes

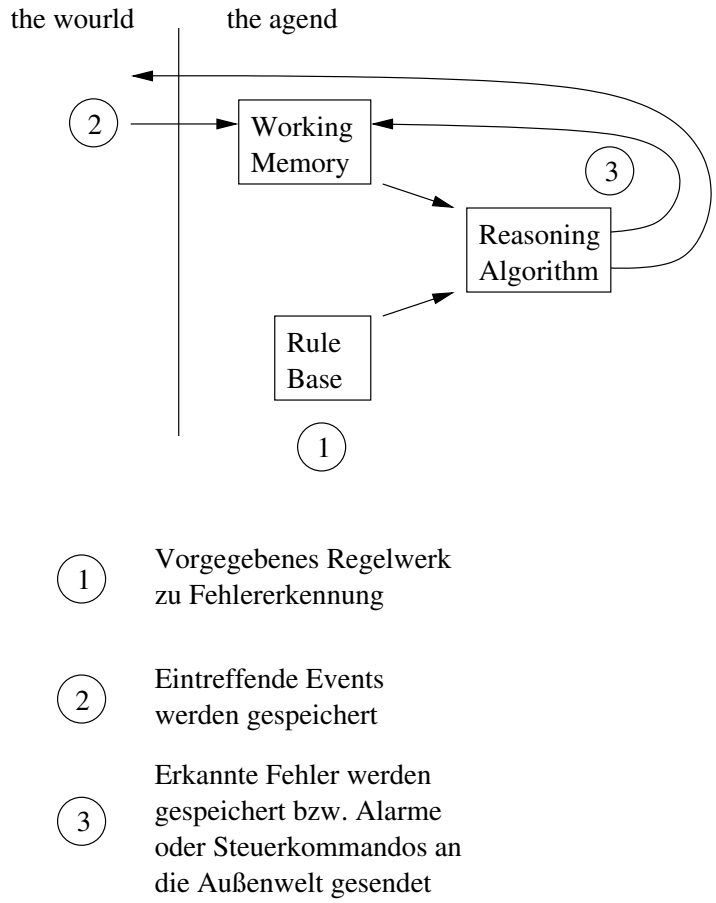


Abbildung 7.13: Funktionsweise des Rule Based Reasonings (RBR) nach [Lewis00]

Netzwerk benötigen würde, wäre enorm. Ein weiteres Problem besteht in dem Verlust eines Events. In diesem Fall wäre es für den Reasoning Algorithmus nicht möglich, die richtige Korrelation durchzuführen. Hieraus folgt, daß der Einsatz des RBR-Verfahrens nur sinnvoll für kleine Netzwerke ist, deren Topologie sich nicht oder nur selten ändert und deren Funktionalität einfach zu erfassen ist. Sind diese drei Bedingungen nicht erfüllt, ist der Einsatz des RBR Verfahrens nicht anzuraten.

Model Based Reasoning (MBR)

Das Model Based Reasoning (MBR) wird auch als Topologie Based Reasoning bezeichnet, da hier das komplette Netzwerk in Aufbau und Funktion per Software abgebildet wird. MBR ist eine Erweiterung der RBR. Die Topologie-Information wird nicht statisch in den Regeln hinterlegt, sondern die Regeln greifen auf aktuelle Topologie-Informationen aus einer externen Datenquelle zu (siehe Abbildung 7.14). Die Schwierigkeit bei diesem Verfahren ist die aktuelle und akkurate Erkennung der Topologie-Information.

Die Güte eines modellgestützten Tools hängt von der **Correlation Engine** und von der **Qualität der Topologieerkennung** ab [Plessner02].

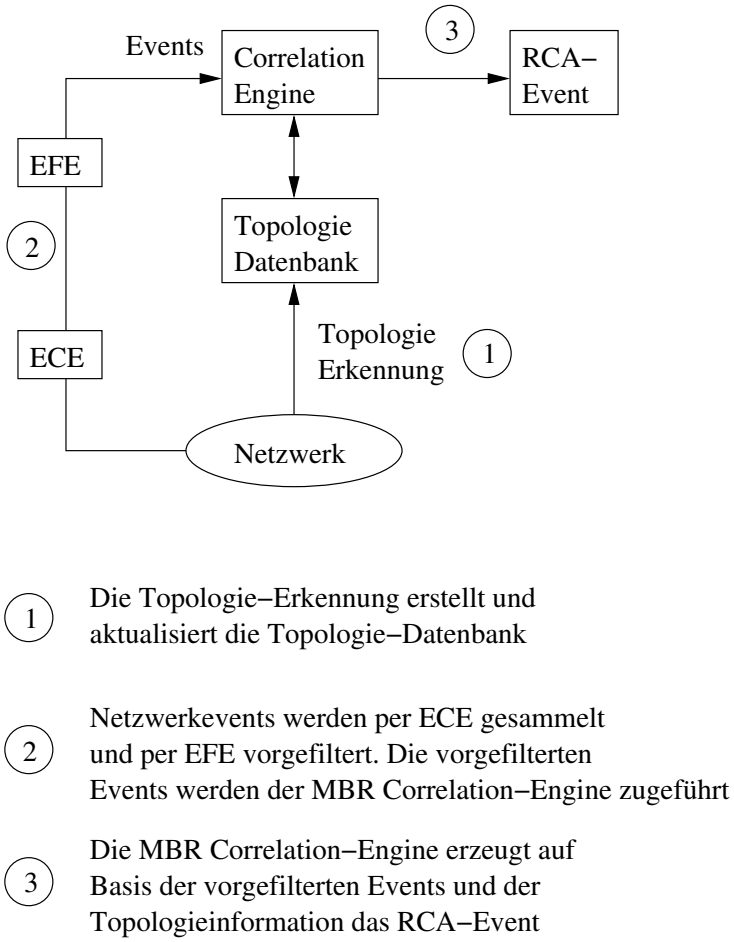


Abbildung 7.14: Funktionsweise des Model Based Reasonings (MBR)

Case Based Reasoning (CaBR)

Beim Case Based Reasoning (CaBR) wird das Wissen über bestimmte Fälle in der Datenbank gespeichert. Für ein Symptom oder eine Menge von Symptomen wird ein Case eröffnet, der mit bekannten Fällen aus der Datenbank verglichen wird. Abbildung 7.15 zeigt die Funktion des CaBR-Systems.

Eine Eigenschaft des CaBR-Systems ist seine Lernfähigkeit bzw. die Möglichkeit auf ein spezielles Netzwerk trainiert zu werden. Ein großes Problem beim Einsatz des CaBR-Systems ist die Zuordnung der Symptome zu einem bestimmten Fall aus der Datenbank. Denn zu einer eindeutigen Fehlererkennung ist es erforderlich, daß unvollständig und zeit-

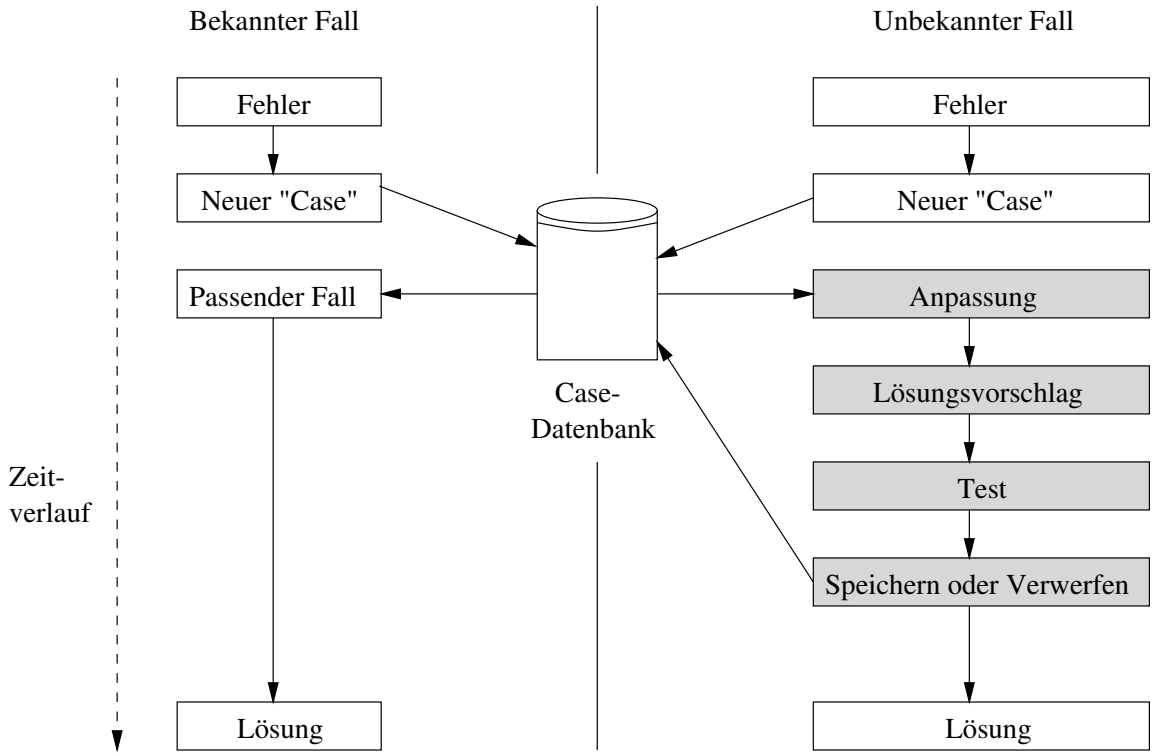


Abbildung 7.15: Funktionsweise des Case Based Reasonings (CaBR)

lich versetzt eintreffende Symptome eindeutig vom CaBR-System einem Fehlerfall zugeordnet werden müssen, was beim CaBR-System (zumindest im Zusammenhang mit Netzwerkmanagement) heute noch ein Problem bereitet [Krause03].

Code Based Reasoning (CoBR)

Das Code Based Reasoning (CoBR) ist eine abgeschwächte Form des CaBR. Es wird durch folgende Eigenschaften gekennzeichnet [Smarts01]:

1. Ein CoBR-System verfügt über eine Case Datenbank (hier **Fehlerdatenbank** genannt), die eine begrenzte Menge von Fällen enthält. Hieraus resultiert, daß das CoBR-System nur auf die **Erkennung einer begrenzten Menge von Fehlerfällen** ausgelegt ist.

2. Das CoBR-System wertet nicht alle Events aus, sondern nur die, die zur Erkennung der in der Fehlerdatenbank hinterlegten Menge von Fehlerfällen notwendig sind.

Abbildung 7.16 zeigt den Aufbau der Fehlerdatenbank. Für jeden Fehlerfall F_1 bis F_m, der vom CoBR-System erkannt werden soll, gibt es eine Spalte in der Matrix.

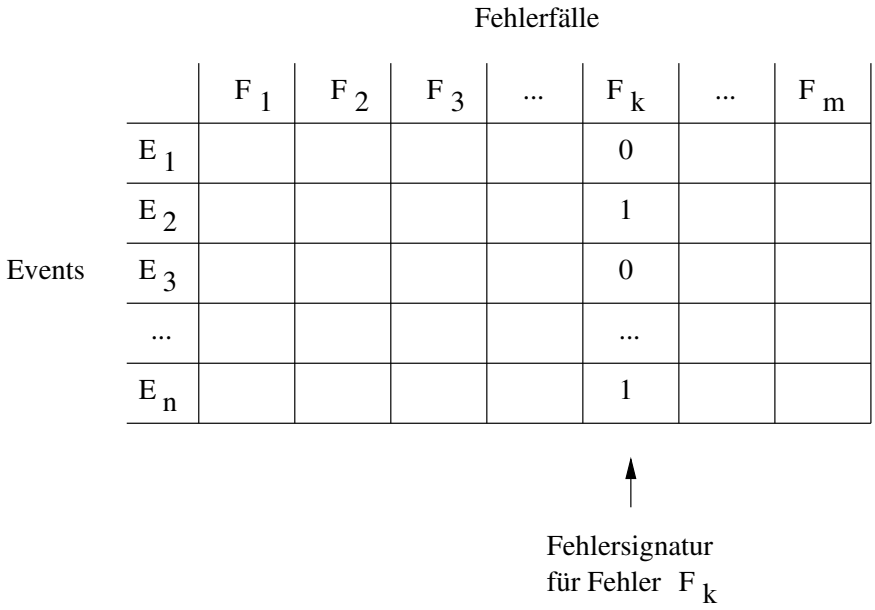


Abbildung 7.16: Aufbau der Fehlerdatenbank beim Code Based Reasoning (CoBR) - System

Für jedes Event E_1 bis E_n, welches vom CoBR-System ausgewertet wird, gibt es eine Zeile in der Matrix. Jede Spalte unterhalb eines Fehlers F_k steht für die Fehlersignatur des Fehlers F_k. Hierin bedeutet eine "0", daß das zugehörige Event nicht auftritt und eine "1", daß das zugehörige Event auftritt. Jeder Fehler F_k verfügt über einen eindeutigen Eventvektor $\vec{E} = (E_1, E_2, E_3, ..., E_n)^T$. Treten in einer zeitlich begrenzten Umgebung ein oder mehrere Events im Netzwerk auf, so wird aus diesen eine Eventsignatur in Form eines Eventvektors erstellt, der mit den Fehlersignaturen in der Fehlerdatenbank verglichen wird. Hierbei wird die Menge von Fehlerfällen aus der Fehlerdatenbank ermittelt, deren Fehlersignatur zur Eventsignatur paßt bzw. am ähnlichsten dazu ist. Die gefundenen Fehlerfälle werden nach der Größe der Ähnlichkeit zur Eventsignatur der Reihe nach sortiert.
Abbildung 7.17 zeigt ein Beispiel zur Veranschaulichung im zweidimensionalen Raum. Hierin wird der "Abstand" der Fehlersignaturen zu der Eventsignatur bestimmt. Für das Beispiel in Abbildung 7.17 ergibt sich als Reihenfolge F_2, F_1, F_3.
Gemäß der Reihenfolge ihrer Ähnlichkeit zur Eventsignatur werden die in der Fehlerdatenbank gefundenen Einträge nun dem Operator präsentiert. Dieser kann den Vorschlag akzeptieren oder sich den nächsten anzeigen lassen.

Systeme, die auf Basis des CoBR-Verfahrens arbeiten, führen in der Initialisierungsphase eine automatische Topologieerkennung durch. Hierdurch wird ein objektorientiertes Modell des Netzwerks erstellt. Auf Basis dieses Modells werden nun die Fehlerfälle und die zugehörigen Fehlersignaturen berechnet, mit denen die Fehlerdatenbank aufgebaut wird. Diese Fehlerdatenbank wird als **Codebook** bezeichnet.

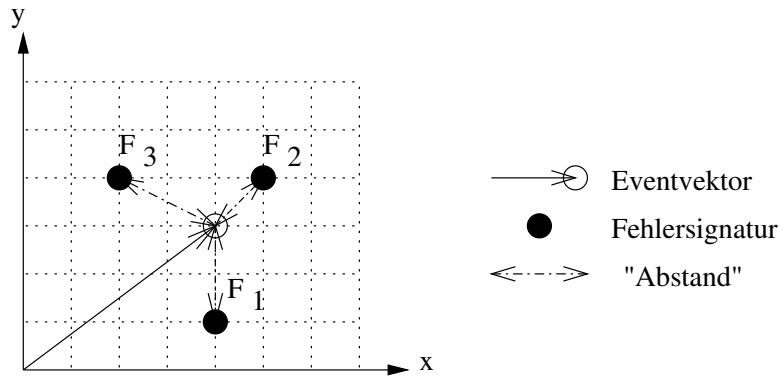


Abbildung 7.17: Funktionsweise des Code Based Reasonings (Veranschaulichung) - Festlegen der Fehlerreihenfolge

Der Vorteil des CoBR-Verfahrens für die RCA liegt in der hohen Erkennungsgeschwindigkeit. Dies hat seine Gründe einerseits in der Auswertung einer begrenzten Menge von Events und andererseits in dem Erkennen nur einer begrenzten Anzahl von Fehlerfällen. Neue Fehlerfälle können durch Hinzufügen der entsprechenden Fehlersignatur zum Codebook erkannt werden. Aufwendig ist die Erstellung des Codebooks, welches aber nur in der Initialisierungsphase erforderlich ist (oder nach großen Umbauten im Netzwerk).

Downstream Event Suppression (DES)

DES ist nur in stern- oder baumförmigen Netzwerktopologien einsetzbar. Typisch für diese Topologieformen ist, daß Netzknoten (engl. Nodes) hinter einer Fehlerstelle nicht erreichbar sind. Aufgabe von DES ist die Unterdrückung von Events, die einen Bezug zu einem Netzwerkknoten hinter einer Fehlerstelle haben (siehe Abbildung 7.18). Da DES nur Events unterdrückt und keine Fehler aus Events folgert, ist es **kein RCA-Verfahren** im strengen Sinn.

Bewertung der verschiedenen Ansätze

Bei der Bewertung der verschiedenen RCA-Verfahren sind die wichtigsten **Vergleichskriterien** folgende:

- Wie geht das Verfahren mit Topologieänderungen um? Dies ist ein wichtiger Aspekt, da Netzwerke sich im Laufe der Zeit ändern.
- Ist das Verfahren skalierbar? Es ist zu klären, wie sich der Rechenaufwand für das Verfahren zur Netzwerkgröße verhält.
- Wie geht das Verfahren mit verloren gegangenen oder verspäteten Events um? Da gegenwärtig das Netzwerkmanagement zu großen Teilen auf SMNPv1 basiert (wel-

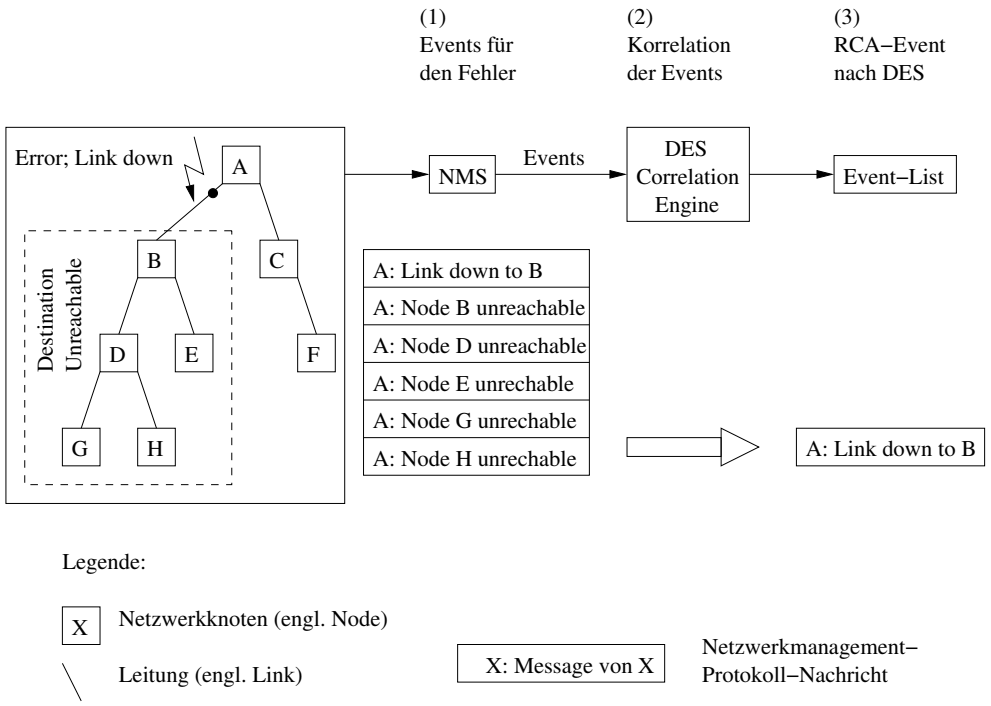


Abbildung 7.18: Funktionsweise von Downstream Event Suppression (DES)

ches inzwischen bei der IETF den Status **historic** hat), ist dies eine reale Situation, die ein RCA-Verfahren nicht beeinflussen darf.

- Werden unbekannte Events an den Operator weitergegeben oder verworfen? In ständig wachsenden Netzwerken wird es niemals möglich sein, alle möglichen Fehler im Vorfeld zu definieren. Aus diesem Grund müssen unbekannte Events an den Operator weitergeleitet werden können. Nur dieser kann anhand seines Wissens und seiner Erfahrung diese Events bearbeiten.

Tabelle 7.1 zeigt die Ergebnisse auf o.g. Fragen für die betrachteten RCA-Verfahren.

	RBR	MBR	CaBR	CoBR
Topologie-Änderungen	—	x	x	x
Skalierbarkeit	—	x	x	x
verlorene und verspätete Events	—	x	durch Tests	durch Tests
Unbekannte Events	x	x	x	—

Tabelle 7.1: Vergleich der RCA-Verfahren

RBR
Aufgrund der in Tabelle 7.1 dargestellten Schwachpunkte ist das Verfahren als Basis für eine RCA nicht die erste Wahl. RBR kann zusätzlich zu einem anderen RCA-Verfahren verwendet werden, um dem Nutzer die Möglichkeit zu bieten, eigene Regeln zu definieren.

MBR
Dieses Verfahren ist der am besten geeignete Ansatz für eine RCA.

CaBR
Dieses Verfahren ist noch in der Entwicklung und wird im industriellen Bereich derzeit nicht verwendet, bietet für die Zukunft aber vielversprechende Ansätze.

CoBR
CoBR ist ein funktionierendes RCA-Verfahren. Es wertet nicht alle Events aus, sondern nur die Events, die in den Fehlersignaturen auftreten. Alle anderen Events werden verworfen und nicht an den Operator weitergegeben. Diese Schwäche des Verfahrens ist gleichzeitig der Grund für seine Effizienz.

Fazit

Es ist festzustellen, daß für ein reines Netzwerkmanagement (Layer 1 - 3 des OSI-Referenzmodells) die Kombination von MBR zur RCA und RBR zur Anpassung an spezielle Bedürfnisse ideal ist. Sind darüberhinaus spezielle Dienste oder Funktionen des Netzwerks zu überwachen, eignet sich für diesen Bereich der RCA das CoBR-Verfahren besser.

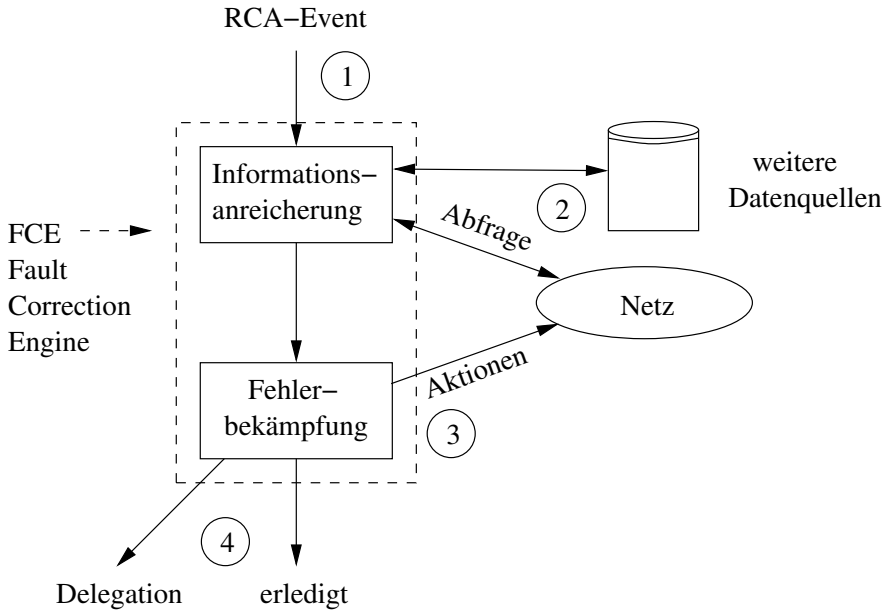
7.2.6 Technologien zum Aufbau einer Fault Correction Engine (FCE)

Ziel der Fault Correction Engine (FCE) ist die automatische Fehlerbeseitung des mittels dem RCA-Event gemeldeten Fehlers. Die FCE ist **zweistufig** aufgebaut (siehe Abbildung 7.19).
Die **erste Stufe** dient zur **Verbesserung der Informationslage** hinsichtlich des RCA-Events. Die **zweite Stufe** übernimmt die **Fehlerbekämpfung**. Im Erfolgsfall werden die durchgeführten Maßnahmen dokumentiert (und dem Operator visualisiert). War die Fehlerbekämpfung nicht möglich (vgl. Abschnitt 7.1.5, “Grenzen der Automatisierung”, Seite 101), so delegiert die FCE die Störung mittels eines Trouble-Ticket Systems an einen entsprechenden Bearbeiter. Die hierbei über die Störung gesammelten Informationen werden dabei übergeben.

Verfahren zur automatischen Informationsanreicherung

Zur Informationsanreicherung in **Stufe 1 der FDE** stehen zwei Verfahren zur Verfügung:

Key-Schlüssel-Verfahren
Beim Key-Schlüssel-Verfahren [Micromuse02/2] erhält der RCA-Event Informationen, die in externen Datenbanken als Key-Schlüssel verwendet werden können.



1 Eingang des RCA−Events

2 Automatische Verbesserung der Informationslage hinsichtlich der Fehlermeldung (z.B. Auflistung der betroffenen Benutzer, Verfügbarkeit von Redundanzsystemen und Ersatzteilen zur Fehlerbekämpfung)

3 Automatische Planung und Durchführung der Fehlerbekämpfung auf Basis der verbesserten Informationslage

4 Automatische Dokumentation der durchgeführten Maßnahmen bzw. Delegation der Störung mittels eines Trouble−Ticket Systems zur weiteren Bearbeitung

Abbildung 7.19: Zweistufiger Aufbau der Fault Correction Engine (FCE)

Hierüber können weitere Informationen aus einer externen Datenbank zum RCA-Event hinzugefügt werden. Dieser Vorgang kann iterativ betrieben werden, d.h. die Quelldatenbank enthält weitere Key-Schlüssel für eine weitere Suche in den externen Datenbanken (oder im Netz). Der Nachteil des Key-Schlüssel-Verfahrens liegt in der manuellen Erstellung der Abfrageskripte. Jedes RCA-Event, welches mit Informationen anzureichern ist, wird in einer Liste eingetragen und beim Auftreten des Events wird das korrespondierende Abfrageskript gestartet (siehe Abbildung 7.20).

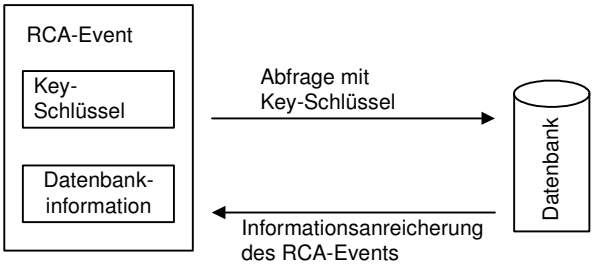


Abbildung 7.20: Funktionsweise der Informationsanreicherung eines RCA-Events mittels dem Key-Schlüssel-Verfahren

Eigenes modellbasiertes Informationsanreicherungsverfahren

In dieser Arbeit wird ein modellbasiertes Informationsanreicherungsverfahren vorgeschlagen, um die o.g. Nachteile des Key-Schlüssel-Verfahrens zu überwinden. Voraussetzung ist hier die **Existenz eines objektorientierten Netzwerkmodells**. Für das RCA-Event wird ein Fehlerobjekt bzw. RCA-Objekt erzeugt, welches das RCA-Event im Netzwerkmodell repräsentiert. Dieses RCA-Objekt wird dann mit den Objekten des Netzwerkmodells verbunden, die in irgendeinem Zusammenhang zu dem RCA-Objekt stehen. Die Art des Zusammenhangs wird durch ein weiteres Objekt beschrieben. Abbildung 7.21 verdeutlicht den Ansatz.

Der **Vorteil** des **modellbasierten Informationsanreicherungsverfahrens** gegenüber dem **Key-Schlüssel-Verfahren** liegt in einem **größeren Informationsgehalt**, mittels dem u.U. die Erstellung von Skripten zur Informationsanreicherung automatisiert werden kann. Eine detaillierte Darstellung dieses Vorschlags und dessen Nutzen für die automatische Fehlerbeseitigung wird in Abschnitt 7.3 ausgeführt.

Bewertung beider Ansätze Der Einsatz des Key-Schlüssel-Verfahrens ist dann anzuraten, wenn für wenige RCA-Events einfache Informationsanreicherungen benötigt werden. Beispielsweise für visuelle Zwecke oder für einfache Skripte, die in Stufe 2 der FCE einfache Gegenmaßnahmen auf einen RCA-Event unternehmen. Der Einsatz des vorgeschlagenen modellbasierten Informationsanreicherungsverfahrens wird dann interessant, wenn eine Informationsanreicherung für viele unterschiedliche RCA-Events erfolgen soll. Da das Netzwerkmodell Angaben zum Aufbau und der Funktionsweise des Netzwerks enthält,

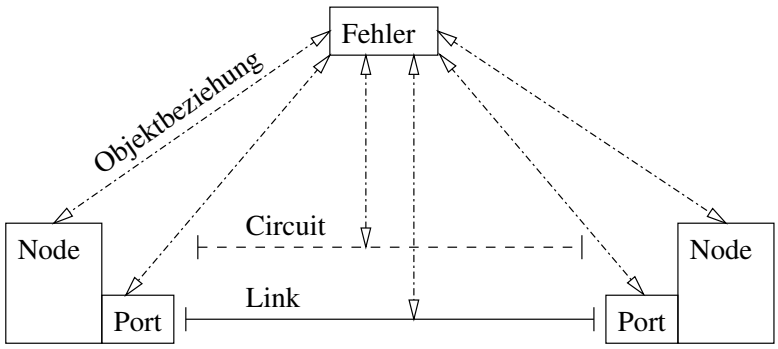


Abbildung 7.21: Grundlegende Vorgehensweise zur Dokumentation eines RCA-Events als Fehlerobjekt, welches Voraussetzung für ein modellbasiertes Informationsanreicherungsverfahren ist

kann über das modellbasierte Informationsanreicherungsverfahren die Informationsanreicherung automatisch erfolgen, ohne hierfür ein Skript erstellen zu müssen. Dies gilt jedenfalls für die Informationen, die das Netzwerkmodell enthält. Für darüberhinausgehende Informationen bleibt als Anreicherungsmöglichkeit nur das Key-Schlüssel-Verfahren.

Verfahren zur automatischen Fehlerbekämpfung

Zur automatischen Fehlerbekämpfung in **Stufe 2 der FCE** steht derzeit ein Verfahren zur Verfügung, das **Virtual-Operator-Modell (VOM)** [Micromuse02/2]. VOM ermöglicht Standard-Vorgehensweisen beim Troubleshooting nach dem Entscheidungsbaumprinzip zu hinterlegen. Treten RCA-Events auf, für die im VOM eine Standard-Vorgehensweise hinterlegt ist, dann wird diese automatisch abgearbeitet. Die Standard-Vorgehensweisen sind vom Operator zu erstellen. Sie basieren auf seinem Wissen und seiner Erfahrung über das Netzwerk und den RCA-Event. Der Erfolg oder Mißerfolg der Standard-Vorgehensweise wird dem Operator angezeigt.
Vor diesem Hintergrund kann man beim VOM eigentlich nicht von einer echten automatischen Fehlerbekämpfung sprechen (z.B. plant VOM keine Entstörungsstrategien und ist nicht robust gegen Änderungen im Netzwerkaufbau). VOM wird primär dazu verwendet, auf Fehlermeldungen zu reagieren (z.B. eine SMS an einen Mitarbeiter zu versenden oder ein Trouble-Ticket zu eröffnen).

Fazit

Ausgehend von einem RCA-Event leisten Verfahren zum Aufbau einer FCE für eine **vorher bestimmte Menge** von RCA-Events eine Informationsanreicherung und eine automatische Reaktion auf das Event. Für die der FCE **unbekannten Events** ist **keine Bearbeitung** möglich. Die Verfahren zur Informationsanreicherung und zur automatischen Reaktion auf ein Event basieren auf den Entscheidungsbaumprinzip. Die Erstellung der Entscheidungsbäume für beide Stufen der FCE erfolgt händisch durch einen Operator.

Aus diesem Grund kommen heute FCE nur zur Bearbeitung von immer wiederkehrenden Standard-Vorgehensweisen zum Einsatz, weshalb hier **nicht von einer echten automatischen Fehlerkorrektur gesprochen** werden kann.

7.3 Eigene Konzepte zum Aufbau einer modellbasierten Fault Detection Engine (FDE) und Fault Correction Engine (FCE)

Ein **Netzwerkmodell** ist die **Grundlage** für viele Verfahren im Bereich des **automatischen Netzwerkmanagements**. Zur Beschreibung eines Fehlers in einem objektorientierten Netzwerkmodell ist zuerst der Aufbau eines solchen Modells erforderlich. Das hier gezeigte Modell, welches als Basis für die Fehlerbeschreibung dient, wird aus den Prinzipien der relationalen Modelle entwickelt, auf denen die heutigen Softwareprodukte zur Netzwerkdokumentation basieren. Die Erstellung eines objektorientierten Netzwerkmodells erfolgt nur soweit, wie dies zur Erläuterung der Fehlerbeschreibung in einem objektorientierten Netzwerkmodell erforderlich ist.

7.3.1 Objektorientiertes Netzwerkmodell

Der Aufbau eines objektorientierten Netzwerkmodells erfordert die Erstellung einer Klassenhierarchie. In dieser werden die Klassen (bzw. die Bausteine) definiert, aus denen das Netzwerkmodell aufgebaut wird. Hierüber werden die Beziehungen der Klassen untereinander festgelegt. Abbildung 7.22 zeigt das Metamodell und die Klassen, die folgende sind:

Root
: Root ist die Basisklasse, von der sich alle anderen Klassen ableiten. Root ist eine abstrakte Klasse, d.h. es werden von Root keine Instanzen erstellt.

Hardware
: Hardware ist eine abgeleitete Klasse von Root, die Basisklasse für alle Objekte ist, die zur Beschreibung von Hardwareobjekten dienen. Hardware ist eine abstrakte Klasse, d.h. es werden von Hardware keine Instanzen erstellt.

Connection
: Connection ist eine abgeleitete Klasse von Root und Basisklasse für alle Objekte, die zur Beschreibung von Verbindungen (physikalische und logische) im Netzwerkmodell verwendet werden. Connection ist eine abstrakte Klasse, d.h. es werden von Connection keine Instanzen erstellt.

Object2Object
: Object2Object ist eine abgeleitete Klasse von Root und Basisklasse für alle Klassen die beschreiben, welche Metaobjekte miteinander verbunden werden können. Beispielsweise kann ein LWL-Kabel mit einem SC-Stecker nicht mit einem Port verbunden werden, der über ein RJ45-Connector verfügt. Object2Object ist Basis für

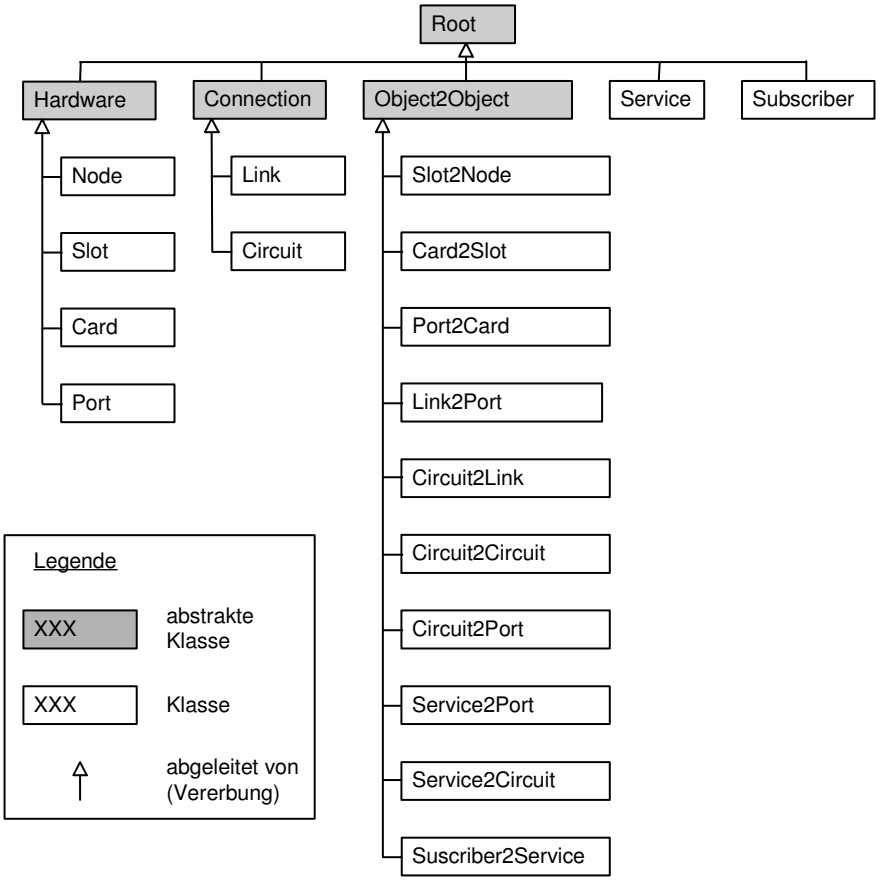


Abbildung 7.22: Metamodell des objektorientierten Netzwerkmodells (Klassenhierarchie)

alle Klassen, die zur Beschreibung o.g. Sachverhalte dienen. Object2Object ist eine abstrakte Klasse, d.h. es werden von Object2Object keine Instanzen erstellt.

Subscriber

Subscriber ist eine abgeleitete Klasse von Root und beschreibt die Nutzer der Services im Netzwerk.

Service

Ein Service ist eine abgeleitete Klasse von Root und beschreibt einen Dienst, den das Netzwerk für einen Nutzer (Subscriber) erbringt.

Hardware-Klassen

Node

Eine Node ist eine Klasse zur Beschreibung einer MNE (Managed Network Entitiy). Die Node hat einen eindeutigen Namen, mit der sie im Netzwerk identifiziert

wird. Beispielsweise modelliert man ein Chassis einer Netzwerkkomponente (ohne Kartenbestückung) als Node.

Slot
Ein Slot ist ein Einschub in einer Node, der einen freien Steckplatz für eine Karte darstellt. Eine Node kann über Slots verschiedener Art verfügen.

Card
Eine Card beschreibt ein Modul, welches in einen Slot eingeschoben wird. Eine Node verfügt über viele Arten von Cards (z.B. Netzwerkmanagement-Karte, Concentrator-Karte, Power-Supply-Karte, usw.). Die Cards werden in die entsprechenden Slots der Node eingeschoben.

Port
Ein Port beschreibt einen Anschluß auf einer Card.

Connection-Klassen

Link
Ein Link beschreibt eine physikalische Verbindung zwischen zwei Ports, die eine physikalische Signalübertragung ermöglicht (z.B. Kupfer-Leitung, LWL-Leiter, Richtfunk usw.).

Circuit
Ein Circuit beschreibt eine logische Verbindung, die auf einer anderen logischen Verbindung (Circuit) oder auf einer physikalischen Verbindung (Link) basiert (z.B. STM-4 Verbindung zwischen zwei SDH-Switchen, die über ein LWL-Kabel verbunden sind).

Object2Object-Klassen

Slot2Node
Beschreibt, welche Slots an welchen Nodes vorhanden sind.

Card2Slot
Beschreibt, welche Cards in welche Slots eingebaut werden dürfen.

Port2Slot
Beschreibt, welche Ports auf welchen Cards vorhanden sind.

Link2Port
Beschreibt, welche Links mit welchen Ports verbunden werden können.

Circuit2Link
Beschreibt, welche Circuits auf welchen Links basieren können.

Circuit2Circuit
Beschreibt, welche Circuits auf welchen Circuits basieren dürfen.

Circuit2Port
Beschreibt, welche Circuits auf welchen Ports basieren dürfen.

Service2Circuit
Beschreibt, welche Services auf welchen Circuits basieren dürfen.

Service2Port
Beschreibt, welche Services auf welchen Ports basieren dürfen.

Subscriber2Service
Beschreibt, welcher Subscriber welchen Service benutzen darf.

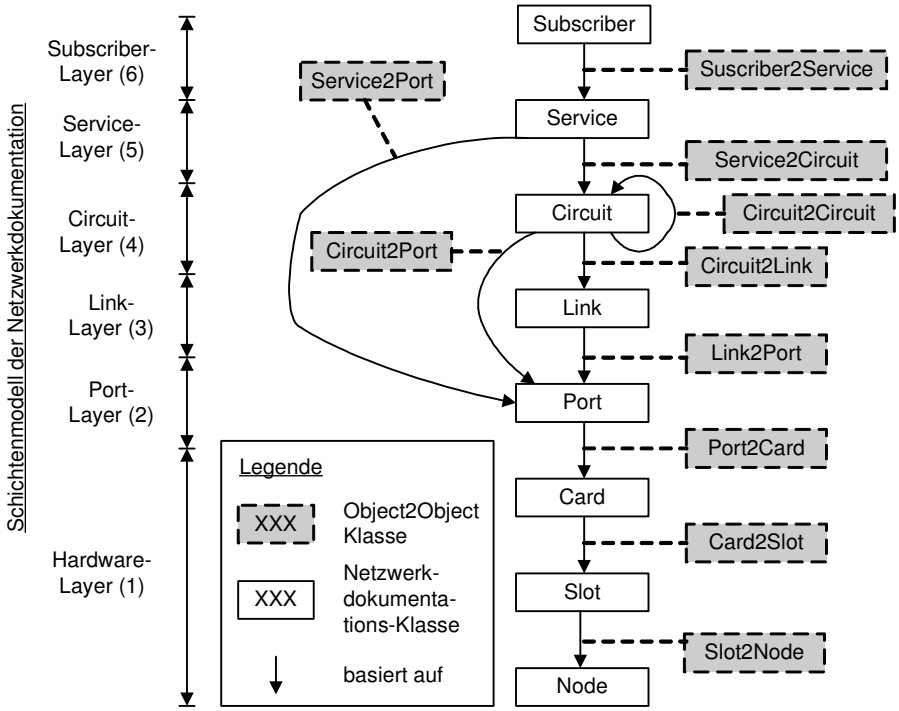


Abbildung 7.23: Metamodell des logischen Aufbaus der Netzwerkdokumentation

Mit der definierten Klassenhierarchie und den Klassen wird der Aufbau des Netzwerkmodells realisiert. Grundprinzip beim Aufbau des Netzwerkmodells ist, daß alle Bestandteile der Netzwerkdokumentation direkt oder indirekt auf der Klasse **Node** basieren. Dies ist erforderlich, da z.B. ein Slot nur existieren kann, wenn eine Node vorhanden ist, in der er sich befindet. Auch kann eine Karte nur dann im Netzwerk verwendet werden, wenn sie in den Slot einer Node eingebaut ist. Hieraus ergibt sich eine logische Hierarchieordnung

der Klassen (siehe Abbildung 7.23). Zur Modellierung der **x basiert auf y** Beziehung werden die Klassen der **Object2Object**-Familie verwendet. Beispielsweise wird durch die Klasse **Card2Slot** festgelegt, welche Karte in welchen Slot eingebaut werden kann. Mittels **Card2Slot** wird die mechanische und logische Kompatibilität der Karte zum Slot modellierungstechnisch sichergestellt.

7.3.2 Schichtenmodell der Netzwerkdokumentation

Aus dem Metamodell des logischen Aufbaus des objektorientierten Netzwerkmodells wird das **Schichtenmodell der Netzwerkdokumentation** abgeleitet (siehe Abbildungen 7.23 und 7.24). Hierbei basiert eine Schicht n auf der darunterliegenden Schicht $n-1$. Als Kriterium für die hierarchische Ordnung der Schichten wird der Abstand einer Schicht von der Highspeed-Backplane einer Netzwerkkomponente herangezogen. Die Schichten des Netzwerkdokumentationsmodells haben im einzelnen folgende Aufgaben:

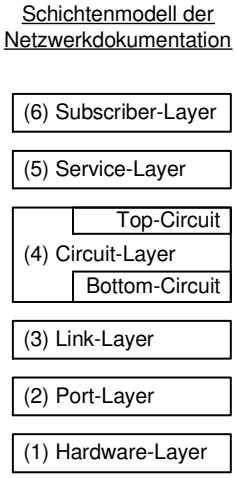


Abbildung 7.24: Schichtenmodell der Netzwerkdokumentation

(1) Hardware-Layer
Aufgabe des Hardware-Layers ist die Beschreibung des physikalischen Aufbaus der Hardware-Komponenten, die im Netzwerk verbaut sind. Hierzu gehört der Aufbau aller Managed Network Entities (MNE) bzw. Nodes. Eine Beschreibung der in den Nodes vorhandenen Slots und deren Belegung. Desweiteren eine Auflistung der in den Slots vorhandenen Cards.

(2) Port-Layer
Aufgabe des Port-Layers ist die Beschreibung der physikalischen und logischen Anschlüsse der modular aufgebauten Hardwarekomponenten. Ein physiklischer Port steht hier für eine Verbindungsmöglichkeit, an die ein Mechanismus zur physikalischen Signalübertragung (Link) an die zum Port gehörende Node angeschlossen

werden kann (z.B. Kupfer- oder LWL-Kabel). Ein logischer Port stellt eine Verbindungsmöglichkeit für eine logische Verbindung (Circuit) dar.

(3) Link-Layer
Aufgabe des Link-Layers ist die Beschreibung der physikalischen Verbindungen zur Signalübertragung zwischen den Ports.

(4) Circuit-Layer
Aufgabe des Circuit-Layers ist die Beschreibung der logischen Verbindungen, die auf den Links betrieben werden und die Verschachtelung der logischen Verbindungen ineinander. Durch die Verschachtelung der Circuits entsteht eine Circuit-Hierarchie. Der unterste Circuit (d.h. der Circuit, der dem Link-Layer am nächsten ist) wird **Bottom-Circuit** genannt. Der oberste Circuit wird **Top-Circuit** genannt. Hierbei dient der Bottom-Circuit zur Beschreibung des Basis-Protokolls, welches zur Datenübertragung auf dem Link verwendet wird. Der Top-Circuit dient zur Beschreibung einer logischen Verbindung, die von einem Service des Service-Layers verwendet wird.

(5) Service-Layer
Der Service-Layer dient zur Beschreibung eines Dienstes im Netz. Aufgabe des Service-Layers ist es einerseits, die technischen Daten eines Dienstes im Netz zu beschreiben, andererseits enthält der Service-Layer Informationen, mittels derer die zu einem Service gehörenden betriebswirtschaftlichen Daten referenziert werden können (z.B. werden die betriebswirtschaftlichen Daten im Störungsfall zur Priorisierung des Trouble-Tickets für diesen Service herangezogen).

(6) Subscriber-Layer
Aufgabe des Subscriber-Layers ist es, den Nutzer eines oder mehrerer Services zu beschreiben. Im Störungsfall ermöglicht es der Subscriber-Layer, die Kontaktinformationen der von der Störung betroffenen Nutzer zu ermitteln.

Das *Schichtenmodell der Netzwerkdokumentation* ermöglicht die Dokumentation einer Netzwerkinfrastruktur, wie sie für die Administration und zur Automatisierung des Netzwerkmanagements erforderlich ist. Es umfaßt den physikaltischen Aufbau der Netzwerkkomponenten und der Verbindungen, die logischen Kommunikationsbeziehungen, die Dienste und die Nutzer der Dienste.

7.3.3 Aufbau der Netzwerkdokumentation

Zur Veranschaulichung des Aufbaus eines objektorientierten Netzwerkmodells werden nachfolgende **Beispiele** betrachtet.

Modellierung einer Managed Network Entity (MNE)

Zur Veranschaulichung der Modellierung einer MNE wird das exemplarische Beispiel in Abbildung 7.25 verwendet. Die dort gezeigte MNE besteht aus einem Chassis mit vier

Slots. In zwei der vier Slots sind Karten eingebaut. Die eine Karte verfügt über drei Ports, die andere Karte über zwei Ports. Eine Modellierung der MNE durch Instanzierung o.g. Klassen zeigt Abbildung 7.26.

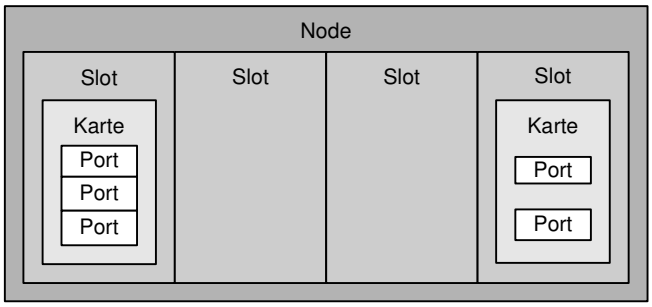


Abbildung 7.25: Beispiel für den Aufbau einer Managed Network Entitiy (MNE) mit 4 Slots und zwei Karten (mit 2 bzw. 3 Ports)

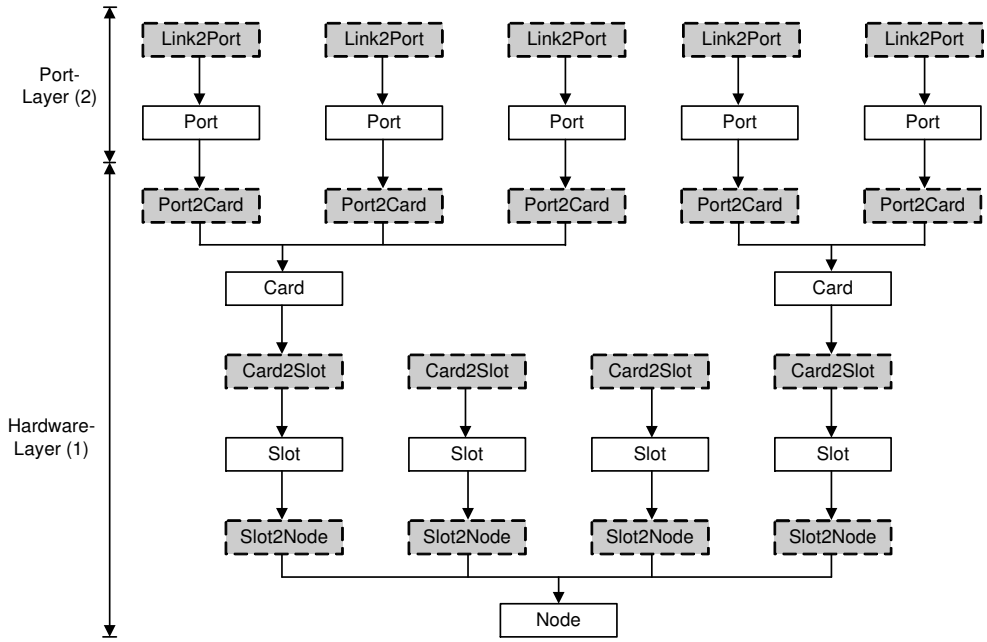


Abbildung 7.26: Beispiel für die Modellierung einer Managed Network Entitiy (MNE) aus Abbildung 7.25

Modellierung einer Verbindung zwischen zwei MNE

Abbildung 7.27 zeigt ein Beispiel zur Modellierung von physikalischen und logischen Verbindungen zwischen den Ports (A- und Z-Port) zweier MNE. Das **Link**-Objekt, welches

den A- und Z-Port miteinander verbindet, wird mit jeweils einem **Link2Port**-Objekt mit dem A- und Z-Port Objekt verbunden. Auf dem **Link**-Objekt wird im Sinne des Schichtenmodells mit einem **Circuit2Link**-Objekt das erste **Circuit**-Objekt aufgesetzt bzw. mit diesem verbunden. Das erste **Circuit**-Objekt auf einem **Link**-Objekt beschreibt die Basistechnologie der Signalübertragung auf dem Link (z.B. Ethernet, ATM, SDH, ISDN usw.). Auf dem untersten **Circuit**-Objekt können nun mit **Circuit2Circuit**-Objekten weitere **Circuit**-Objekte aufgesetzt werden, die logische Verbindungen beschreiben, die von der verwendeten Basistechnologie eingesetzt werden (z.B. IP-Verbindungen, verschlüsselte VPN-Channels usw.).

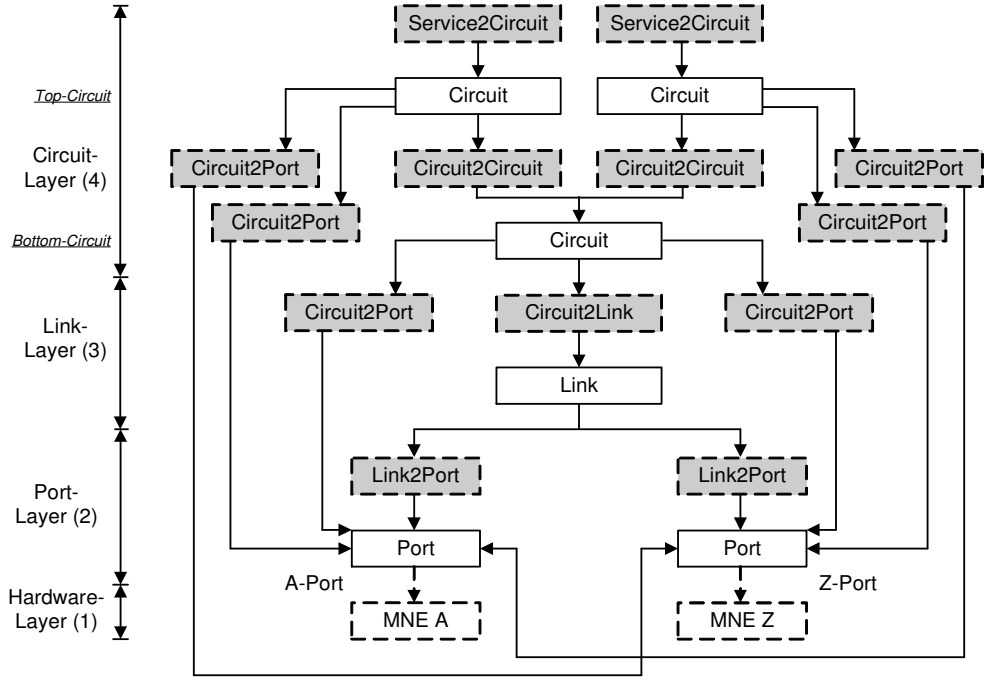


Abbildung 7.27: Beispiel zur Modellierung von physikalischen und logischen Verbindungen zwischen den Ports zweier MNE

Circuit Hierarchie

Die Circuits im Circuit-Layer des Schichtenmodells sind in einer Hierarchie angeordnet, die sich aus den Protokollschichten der zu dokumentierenden logischen Netzwerkverbindungen ergibt. Man betrachte hierzu das **Beispiel einer Backbone-Verbindung** zwischen zwei Backbone Switchen eines Carrier-Netzwerks (siehe Abbildung 7.28). Der A-Port und der Z-Port der jeweiligen Switche sind durch ein LWL-Kabel verbunden, was durch ein **Link**-Objekt dokumentiert wird. Die logische Verbindung beider Switche ist eine STM-1 Verbindung (155 Mbit/sec). Hierzu wird ein Circuit-Objekt **STM-1-Circuit** mit dem **Link**-Objekt verbunden bzw. in Sinne des Schichtenmodells auf diesem aufgesetzt. Dieser

ist der Bottom-Circuit der Circuit-Hierarchie.

Eine STM-1 Verbindung kann 3 VC3-Container (34 Mbit/sec) enthalten. Aus diesem Grund werden 3 **VC3-Circuits** auf den **STM-1-Circuit** aufgesetzt. Ein VC3-Container kann 21 VC12-Container (2Mbit/sec) enthalten. Daher werden auf jeden **VC3-Circuit** 21 **VC12-Circuits** (2 Mbit/sec) aufgesetzt. In dieser Konfiguration stehen zwischen den Backbone-Switchen 63 Kanäle mit einer Bandbreite von je 2 Mbit/sec pro Kanal zur Verfügung. Einer dieser **VC12-Circuits** wird dazu verwendet, für einen Kunden eine Telefonanlagenkoppelung (30 B-Kanäle) zu realisieren. Hierzu wird auf einem **VC12-Circuit** der **ISDN-S2M-Circuit** aufgesetzt. Dieser ist der Top-Circuit für die Telefonanlagenkoppelung.

Ein anderer **VC12-Circuit** wird für eine IP-Verbindung zwischen zwei Backbone-Switchen (Transfernetz) verwendet. Hierzu wird der **Backbone-IP-Circuit** auf einem **VC12-Circuit** aufgesetzt. Innerhalb des **Backbone-IP-Circuits** werden drei verschlüsselte VPN-Verbindungen für drei verschiedene Kunden übertragen. Dazu werden 3 **VPN-Circuits** auf den **Backbone-IP-Circuit** aufgesetzt. Die **VPN-Circuits** sind die Top-Circuits für die VPN-Verbindung.

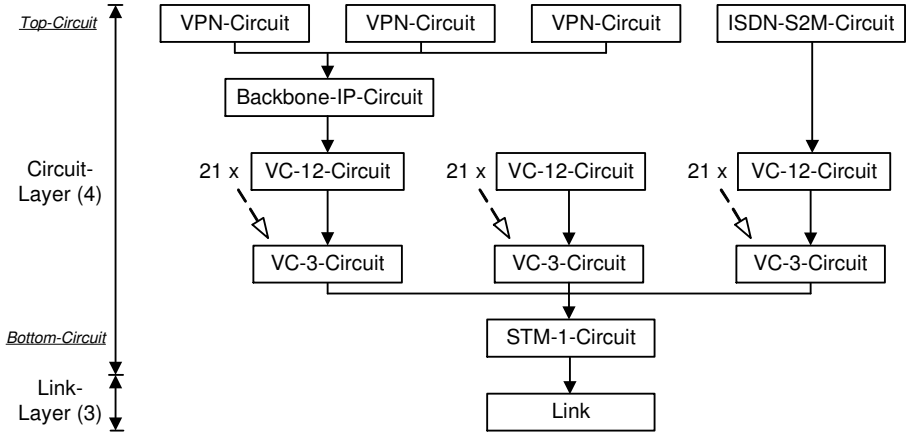


Abbildung 7.28: Circuit-Hierarchie im Circuit-Level (vereinfachte Darstellung) am Beispiel eine Backbone-Verbindung

Modellierung von Services und Subscribern

Ein **Beispiel für die Modellierung von Services und Subscribern** für die logischen Verbindungen im Netzwerk zeigt Abbildung 7.29. Der oberste Circuit der Circuit-Hierarchie beschreibt eine logische Datenverbindung durch das Netzwerk. Mittels eines **Service2Circuit**-Objekts wird der oberste Circuit mit einem **Service**-Objekt verbunden. Das **Service**-Objekt seinerseits wird mit einem **Subscriber2Service**-Objekt mit einem **Subscriber**-Objekt verbunden.

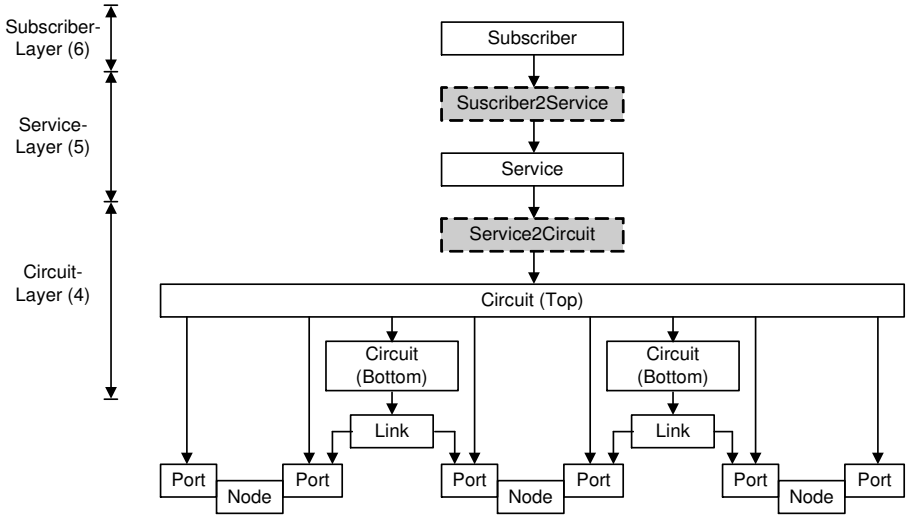


Abbildung 7.29: Beispiel zur Modellierung von Services und Subscribern für die Verbindungen im Netzwerk (vereinfachte Darstellung)

7.3.4 Erweiterung des Netzwerkmodells zur Fehlerbeschreibung

Zur Darstellung eines Events im objektorientierten Netzwerkmodell und zur Planung und Durchführung von automatischen Test- bzw. Troubleshootingmaßnahmen im Netzwerk ist eine Erweiterung des Metamodells (vgl. Abbildung 7.22, Seite 122) erforderlich. Die Erweiterung besteht aus folgenden Metaobjekten (siehe Abbildung 7.30).

Abstrakte Klassen

Die abstrakten Klassen des Metamodells werden um die zwei Klassen **Test** und **Correct** erweitert.

Test
: Test ist eine abgeleitete Klasse von Root und dient zur Beschreibung von Testprozeduren, die die Funktionalität von Teilbereichen des Netzwerks online im Netz nachprüfen. Test ist eine abstrakte Klasse, d.h. es werden von Test keine Instanzen erstellt.

Correct
: Correct ist eine abgeleitet Klasse von Root und dient zur Beschreibung von Korrekturprozeduren, die beim Auftreten eines Fehlers an einer Komponente des Netzwerks versuchen, diesen zu beheben. Correct ist eine abstrakte Klasse, d.h. es werden von Correct keine Instanzen erstellt.

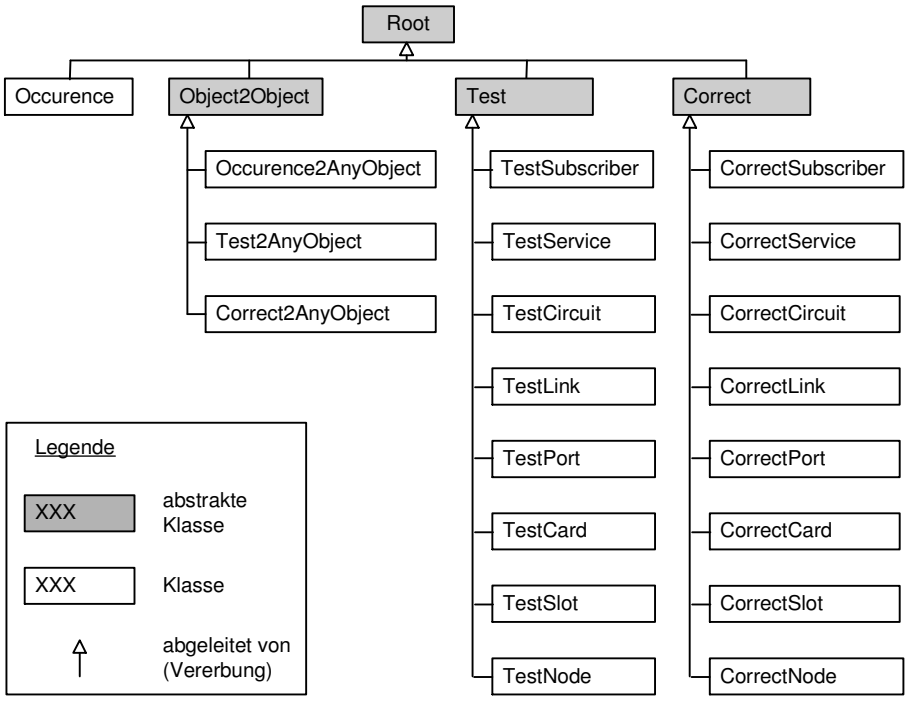


Abbildung 7.30: Erweiterung des Metamodells (vgl. Abbildung 7.22, Seite 122) zur Fehlerbeschreibung

Object2Object-Klassen

Die im Metamodell bereits bestehenden Object2Object-Klassen werden um nachfolgende erweitert.

Occurence2AnyObject
: Beschreibt, welche Occurence-Objekte mit welchen anderen Objekten verbunden werden können.

Test2AnyObject
: Beschreibt, welche Test-Objekte mit welchen anderen Objekten verbunden werden können.

Correct2AnyObject
: Beschreibt, welche Correct-Objekte mit welchen anderen Objekten verbunden werden können.

Test-Klassen

TestNode
: Testet die zugehörige Node auf Fehlfunktionen.

TestSlot
Testet den zugehörigen Slot auf Fehlfunktionen.

TestCard
Testet die zugehörige Card auf Fehlfunktionen.

TestPort
Testet den zugehörigen Port auf Fehlfunktionen.

TestLink
Testet den zugehörigen Link auf Fehlfunktionen.

TestCircuit
Testet den zugehörigen Circuit auf Fehlfunktionen.

TestService
Testet den zugehörigen Service auf Fehlfunktionen.

TestSubscriber
Testet den zugehörigen Subscriber auf Fehlfunktionen.

Correct-Klassen

CorrectNode
Dient zum Troubleshooting von festgestellten Fehlern auf der zugehörigen Node.

CorrectSlot
Dient zum Troubleshooting von festgestellten Fehlern auf dem zugehörigen Slot.

CorrectCard
Dient zum Troubleshooting von festgestellten Fehlern auf der zugehörigen Card.

CorrectPort
Dient zum Troubleshooting von festgestellten Fehlern auf dem zugehörigen Port.

CorrectLink
Dient zum Troubleshooting von festgestellten Fehlern auf dem zugehörigen Link.

CorrectCircuit
Dient zum Troubleshooting von festgestellten Fehlern auf dem zugehörigen Circuit.

CorrectService
Dient zum Troubleshooting von festgestellten Fehlern auf dem zugehörigen Service.

CorrectSubscriber
Dient zum Troubleshooting von festgestellten Fehlern auf dem vom Subscriber verwendeten Services.

Occurence

Occurence ist eine abgeleitete Klasse von Root und Metaobjekt zur Beschreibung von Netzwerkevents im objektorientierten Netzwerkmodell. Hierzu wird für jedes Event, welches in die Netzwerkdokumentation aufgenommen werden soll, eine Instanz von **Occurence** gebildet, die über eine Instanz von **Occurence2AnyObject** mit dem Objekt in der Netzwerkdokumentation verbunden wird, welches den Grund für das Event darstellt (siehe Abbildung 7.31).

Abbildung 7.31: Verbindung einer Occurence mit dem betroffenen Objekt (z.B. einem Port), für welches ein Event in die Netzwerkdokumentation aufgenommen wird

Verwendung der Erweiterungen für das objektorientierte Netzwerkmodell

Die Verwendung der Erweiterungen zum Aufbau eines objektorientierten Netzwerkmodells ist in Abbildung 7.32 verdeutlicht, die den Aufbau von zwei MNE mit einer Verbindung zwischen beiden zeigt.
Der wesentliche **Unterschied des erweiterten Netzwerkmodells zu den bisherigen Modellen** besteht darin, daß **jede Instanz eines Metaobjekts** über ein **Test**- und **Correct**-Objekt verfügt. Mittels dem **Test**-Objekt ist man in der Lage, die Funktionsweise des jeweiligen Dokumentationsobjekts durch Onlineabfragen im Netz nachzuprüfen und gegebenenfalls bei Fehlfunktionen eine Fehlerbeschreibung zu erstellen. Dies ist dann die Grundlage für automatische Troubleshootingmaßnahmen durch das **Correct**-Objekt des jeweiligen Dokumentationsobjekts. **Test**- und **Correct**-Objekte für die Netzwerkdokumentation und für das Troubleshooting könnten **vom Hersteller der jeweiligen MNE mitgeliefert** werden, so wie heute der Hersteller den Private-MIB-Tree zur Einbindung der MNE in das Netzwerkmanagementsystem mitliefert.

7.3.5 Beschreibung von Fehlern im Netzwerkmodell

Die Beschreibung von Fehlern im objektorientierten Netzwerkmodell erfolgt durch Einfügen von **Occurence**-Objekten für vorgefilterte Events in das Modell. Abbildung 7.33 zeigt hierzu ein vereinfachtes Netzwerkmodell mit 5 MNEs (**Node-1**, **Node-2**, **Node-3**, **Node-4**, **Node-5**) und 2 Diensten (**Service-1**, **Service-2**), welches 3 Events aufweist, die durch **Occurence**-Objekte (**Occurence-1**, **Occurence-2**, **Occurence-3**,) repräsentiert werden.
Die Abbildung der vorgefilterten Events in das objektorientierte Netzwerkmodell ermöglicht eine **automatische Klassifikation der Events** in Kategorien, die den Ebenen des **Schichtenmodells der Netzwerkdokumentation** entsprechen. Es ergeben sich hieraus folgende **Klassifikationskriterien**:

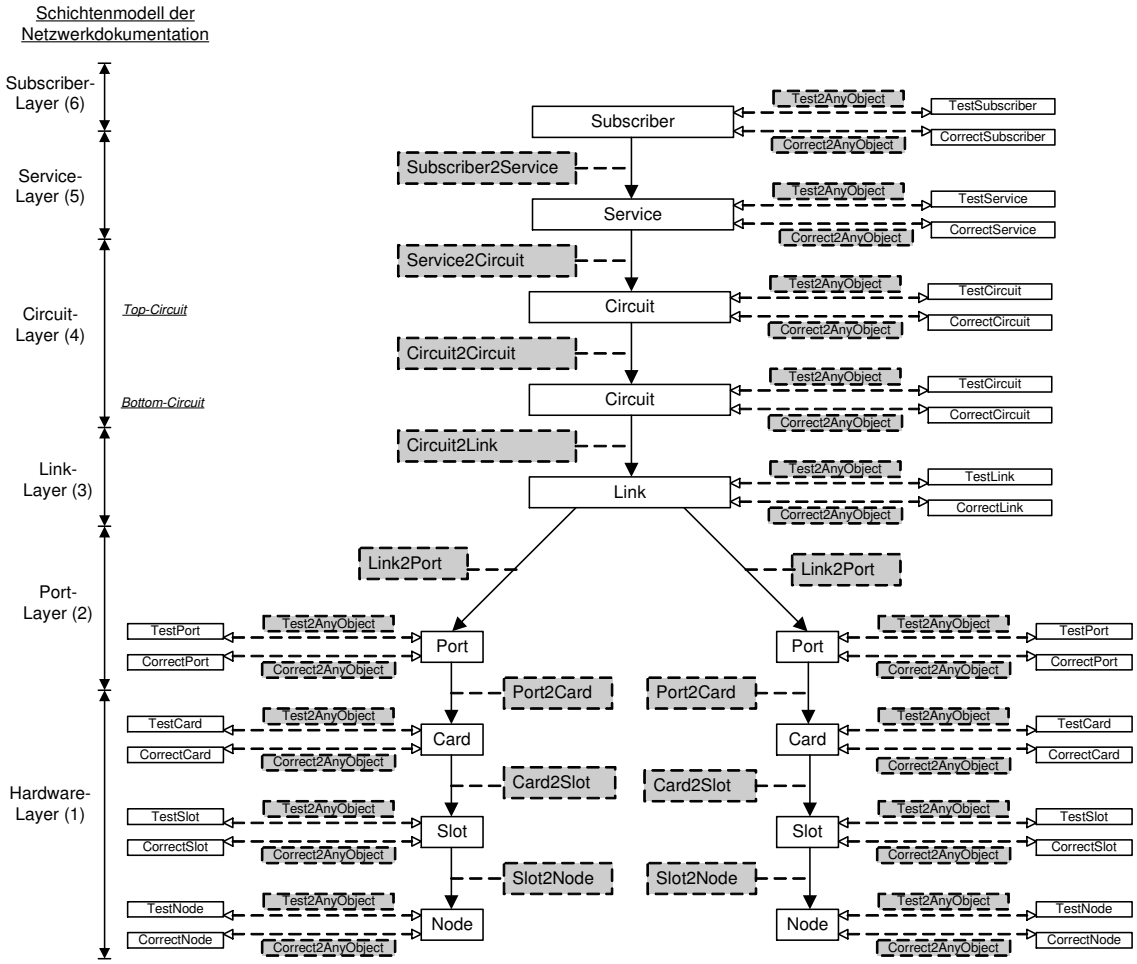


Abbildung 7.32: Aufbau des objektorientierten Netzwerkmodells für zwei MNE und einer Verbindung zwischen beiden unter Nutzung der Erweiterungen des Metamodells

- Hardware-Event
- Port-Event
- Link-Event
- Circuit-Event
- Service-Event
- Subscriber-Event

Hierbei ist zu bemerken, daß ein “Subscriber-Event” von einem technischen Netzwerküberwachungssystem nicht generiert wird. Allerdings könnte der Call eines Nutzers im Call-

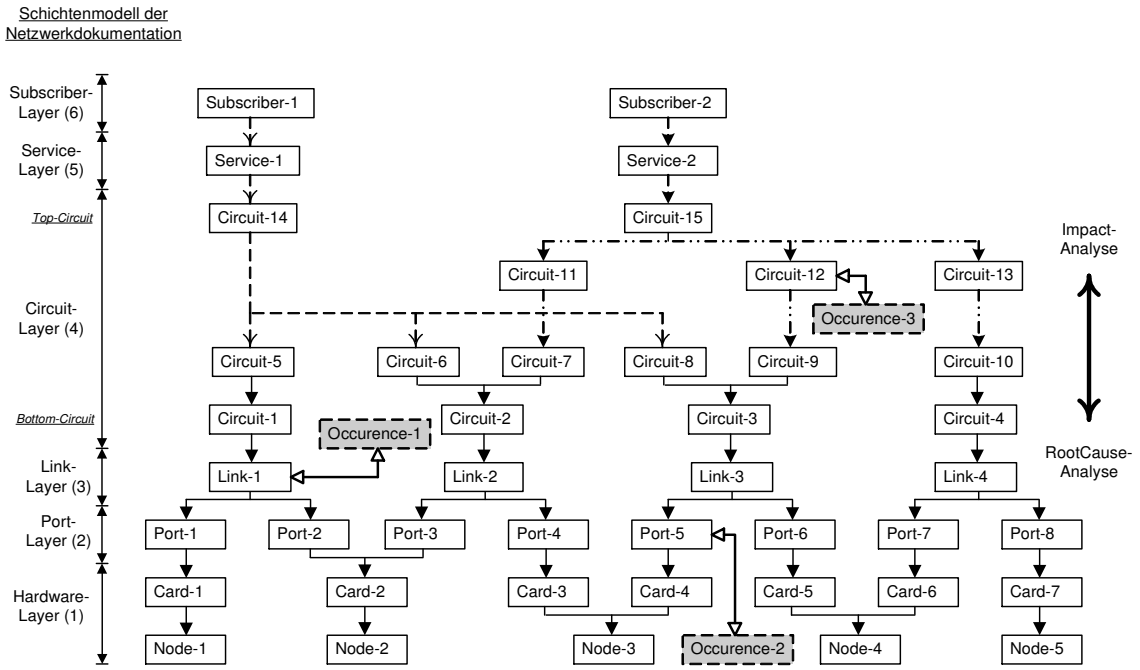


Abbildung 7.33: Beschreibung von Fehlern im objektorientierten Netzwerkmodell (vereinfachte Darstellung ohne Slots)

Center als “Subscriber-Event” in das objektorientierte Netzwerkmodell aufgenommen werden.

Setzt man die Eventklassifikation nach dem **Schichtenmodell der Netzwerkdokumentation** ins Verhältnis zur anfangs genannten Ursachenklassifikation (vgl. Abschnitt 7.1.2, Seite 97ff), so erlaubt die Klassifikation eines Events in dem einen Klassifikationsschema Rückschlüsse auf die Klassifikation des Events in dem anderen Klassifikationsschema. Abbildung 7.34 zeigt die Relation zwischen beiden Klassifikationsschemata.

Hieraus folgt, daß bis zu einem gewissen Grad die Ursachenklassifikation eines Events aus der Klassifikation nach dem **Schichtenmodell der Netzwerkdokumentation** bestimmt werden kann.

Durch die Verlinkung der **Occurence**-Objekte mit den Objekten des Netzwerkmodells, die in die Hierarchieordnung des Schichtenmodells der Netzwerkdokumentation eingebunden sind, läßt sich hieraus eine **Hierarchieordnung der Occurence-Objekte** ableiten. Hierdurch wird eine **Hierarchieordnung der Events** möglich, die die **Abhängigkeit der Fehler untereinander** beschreibt. Abbildung 7.35 zeigt die Fehlerhierarchieordnung für das Beispiel in der Abbildung 7.33. Die Hierarchieordnung der **Occurence**-Objekte kann für zwei weitere Analysen, wie nachfolgend beschrieben, verwendet werden.

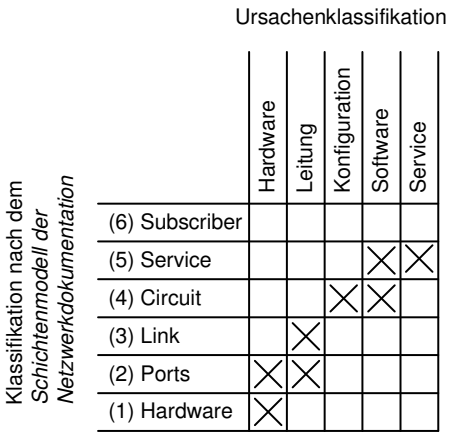


Abbildung 7.34: Relation zwischen der Fehlerklassifikation nach dem *Schichtenmodell der Netzwerkdokumentation* und der Ursachenklassifikation (vgl. Abbildung 7.1, Seite 98)

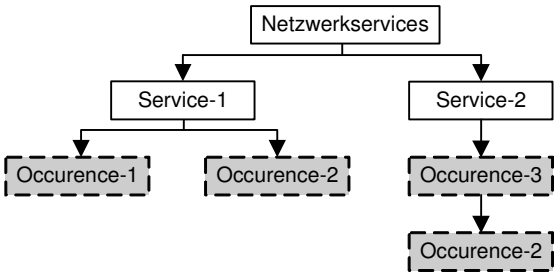


Abbildung 7.35: Hierarchieordnung der Servicefehler im Netzwerk

Impact-Analyse

Die Impact-Analyse dient zur Bestimmung der Auswirkung, die ein Fehler für die Nutzer des Netzwerks hat. Die Nutzer verspüren immer dann eine Auswirkung eines Fehlers, wenn Services gestört sind. Da jeder Service auf einer eindeutigen hierarchischen Struktur von Netzwerkdokumentationsobjekten aufgebaut ist, sind von einem Fehler im Netzwerk alle die Services betroffen, die das vom Fehler betroffene Objekt in ihrem Baum haben. Abbildung 7.36 zeigt die Baumstruktur der Netzwerkobjekte, auf die sich der **Service-1** stützt. Abbildung 7.37 zeigt das für den **Service-2**. Hieraus ist ersichtlich, daß **Occurence-1** die Services **Service-1** und **Service-2** betrifft, **Occurence-2** nur **Service-2** und **Occurence-3** nur **Service-2**.

Root-Cause Analyse

Die Root-Cause Analyse dient zur Ermittlung der Fehlerursachen für einen gestörten Service, wobei die **Hierarchieordnung der Servicefehler** einen weiteren **Ansatzpunkt für ein RCA-Verfahren** liefert. Um bei der Störung eines Services die Ursache festzustel-

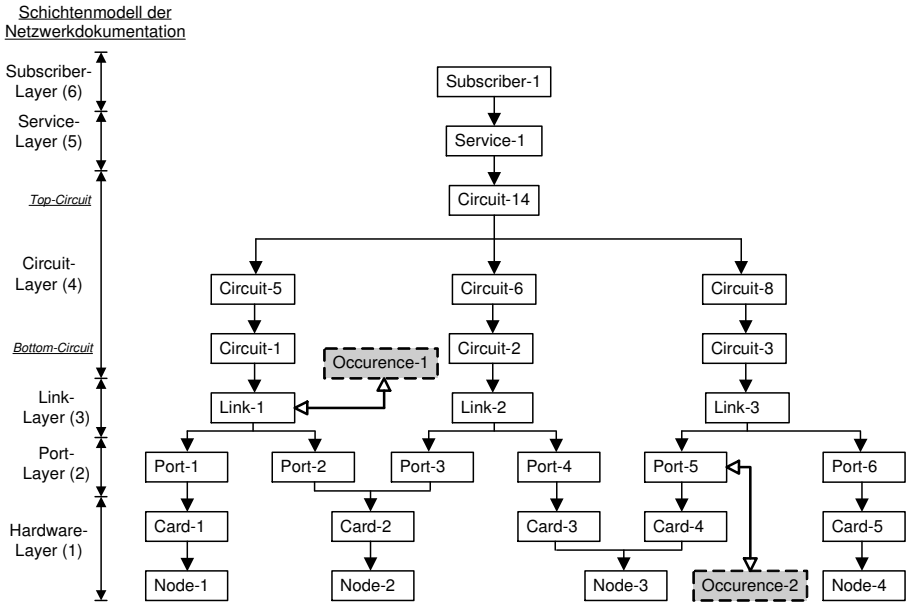


Abbildung 7.36: Objekthierarchie der Netzwerkdokumentationsobjekte, auf denen der *Service-1* basiert (vereinfachte Darstellung ohne Slots)

len, erstellt man aus dem Netzwerkmodell heraus eine Fehlerhierarchie für den gestörten Service. Abbildung 7.35 zeigt dies für die Services **Service-1** und **Service-2**.
Root-Cause für die Fehler eines Services sind die Blattknoten der Fehlerhierarchie für diesen Service. Für das Beispiel oben ist Root-Cause für den **Service-1** **Occurence-1** und **Occurence-2**, für den **Service-2** **Occurence-2**.
Hierbei ist zu beachten, daß die Störung von **Service-1** ihre Ursache in einer "Parallelschaltung" von Fehlern hat, d.h. zur Entstörung von **Service-1** müssen beide Fehler (**Occurence-1** und **Occurence-2**) beseitigt werden. Die Störung von **Service-2** hat ihre Ursache in einer "Reihenschaltung" von Fehlern, d.h. zur Entstörung von **Service-2** wird zunächst **Occurence-2** beseitigt. Da **Occurence-3** mit hoher Warscheinlichkeit seine Ursache in **Occurence-2** hat, wird **Occurence-3** u.U. mit der Behebung von **Occurence-2** verschwinden.

7.3.6 Eigenes Konzept zum Aufbau einer Fault Detection Engine (FDE)

Die Erweiterungen des Netzwerkmodells und die Aufnahme der vorgefilterten Events in das Netzwerkmodell ermöglichen den Aufbau einer zweistufigen FDE (siehe Abbildung 7.38). Hierbei dient die erste Stufe zur Evaluierung von Fehler-Verdachtsfällen anhand der vorgefilterten Events. Aufgabe der zweiten Stufe ist es, die Verdachtsfälle durch Onlineabfragen im Netz zu überprüfen. Die Funktionsweise der FDE sieht im Detail wie folgt aus:
In der **ersten Stufe** werden die von der Event Filter Engine (EFE, vgl. Abschnitt 7.2.4,

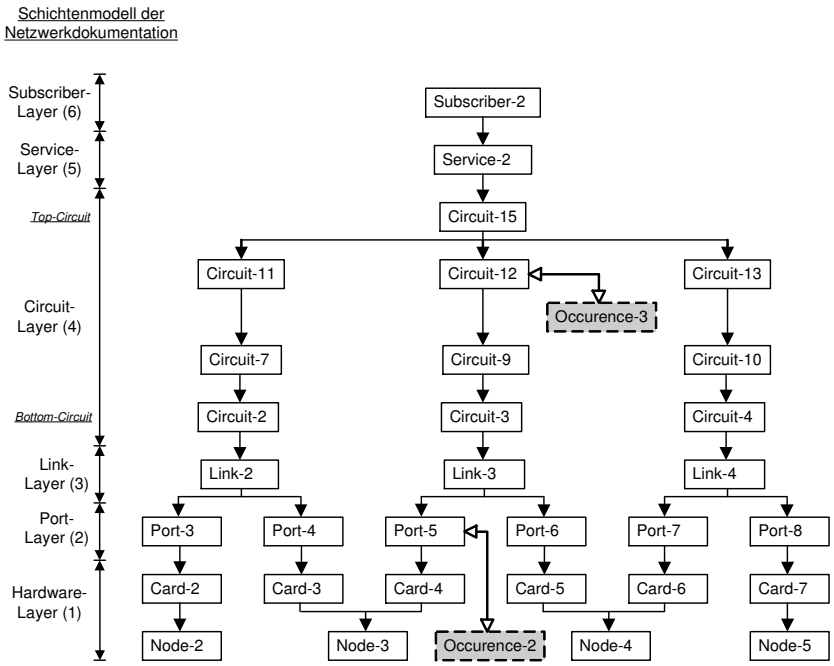


Abbildung 7.37: Objekthierarchie der Netzwerkdokumentationsobjekte, auf denen der *Service-2* basiert (vereinfachte Darstellung ohne Slots)

Seite 107ff) gelieferten vorgefilterten Events in das Netzwerkmodell abgebildet. Auf Basis des Netzwerkmodells wird eine Root-Cause Analyse und Impact-Analyse nach oben beschriebenem Verfahren durchgeführt. Die Root-Cause Analyse liefert eine Liste der fehlerhaften Netzwerkkomponenten und die Abhängigkeit der Events untereinander. Ferner eine Klassifikation der Events auf Basis der Ebenen des **Schichtenmodells der Netzwerkdokumentation** (vgl. Abbildung 7.24, Seite 125). Die Impact-Analyse liefert eine Liste der gestörten Services. Beide Listen werden zu einer Liste korreliert. Diese wird als **Verdachtsliste** für potentielle Netzwerkfehler und für potentielle Auswirkungen der Fehler auf die Services des Netzwerks an die zweite Stufe der FDE übergeben.

In der **zweiten Stufe** erfolgt ausgehend von dieser Verdachtsliste und auf Basis des Netzwerkmodells die Erstellung einer Prüfroutine (unter Nutzung der **Test**-Objekte), die durch Online-Abfragen im Netzwerk die Verdachtsfälle überprüft.

Die Prüfroutine wird wie folgt aus der Fehlerhierarchie der gestörten Services gewonnen. Jedem Dokumentationsobjekt sind durch die Erweiterungen des Netzwerkmodells (vgl. Abschnitt 7.3.4, Seite 130ff bzw. Abbildungen 7.31, 7.32) drei Objekte zugeordnet, die für das automatische Netzwerkmanagement benötigt werden (siehe Abbildung 7.39).

Hierdurch läßt sich aus der Fehlerhierarchie der Services die Menge der **Test**-Objekte und deren hierarchischer Zusammenhang ermitteln, die zum Test der Verdachtsfälle benötigt werden (siehe Abbildung 7.40). Zur Gewinnung der jeweiligen **Test**-Objekte werden die

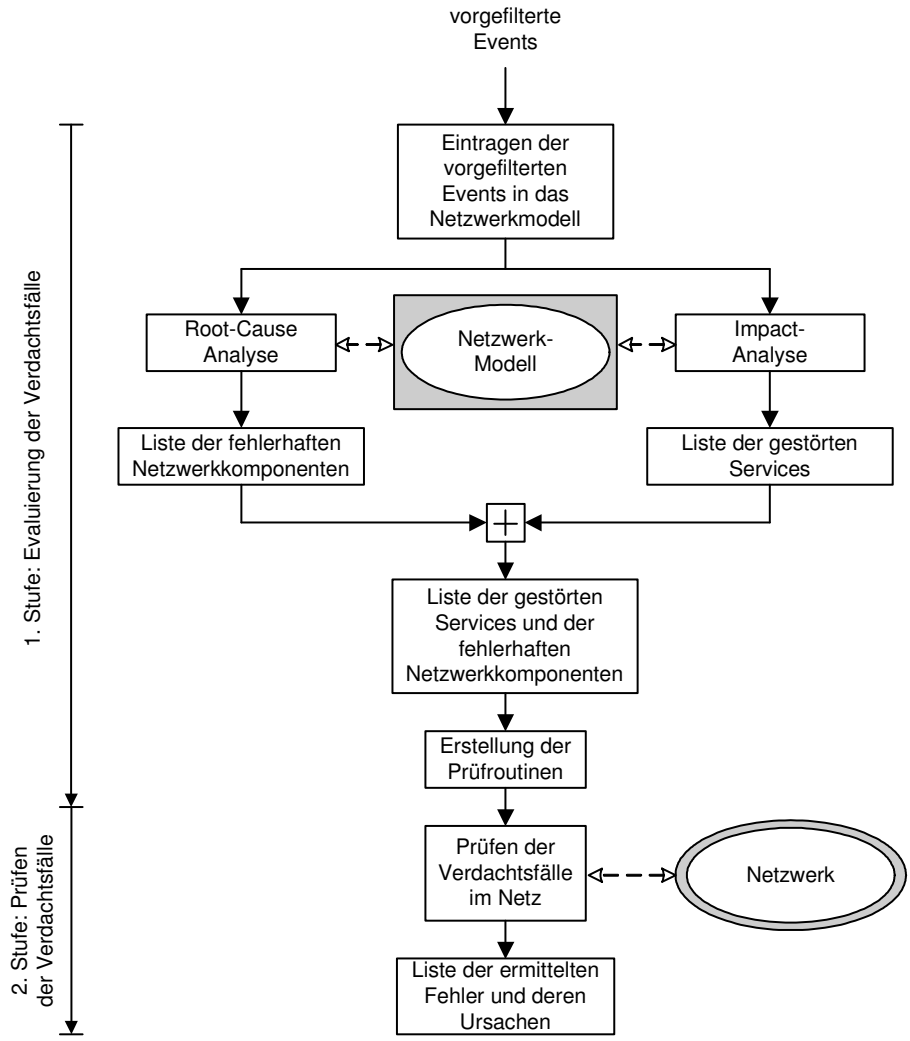


Abbildung 7.38: Eigenes Konzept zum Aufbau einer zweistufigen Fault Detection Engine (FDE)

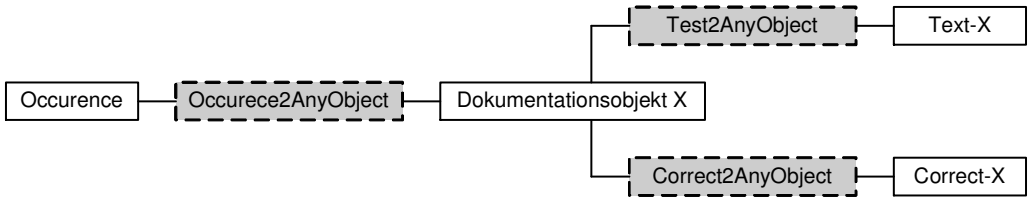


Abbildung 7.39: Einem Dokumentationsobjekt zugeordnete Objekte, die zum automatischen Netzwerkmanagement benötigt werden

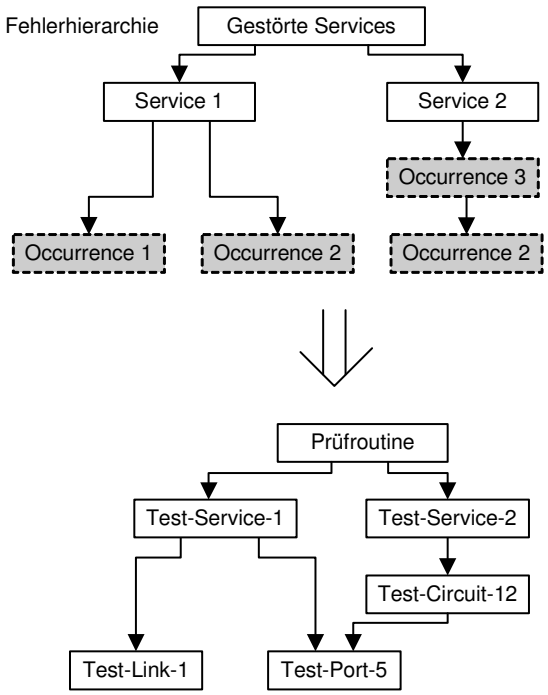


Abbildung 7.40: Beispielhafte Erstellung der Prüfroutine aus der Hierarchieordnung der Servicefehler

Netzwerkdokumentationsobjekte verwendet, die mit einem **Occurence**-Objekt markiert sind.

Die Prüfroutine wird aus der Menge der **Test**-Objekte unter der Berücksichtigung der hierarischen Zusammenhänge erstellt. Als Ergebnis der Überprüfung wird eine Liste der ermittelten Fehler und deren Ursachen (im *Event Description Format* (EDF), vgl. Abbildung 7.7, Seite 106) erstellt. Die Liste enthält Angaben, mit denen später die **Correct**-Objekte parametrisiert werden, aus denen sich die Korrekturroutine in der FCE zusammensetzt.

Der **Vorteil des** hier **vorgestellten zweistufigen Aufbaus der FDE** gegenüber dem heute verwendeten einstufigen Aufbau liegt in folgenden Punkten:

- Die Fehleranalyse der ersten Stufe basiert auf der Auswertung der vorgefilterten Events. Da im praktischen Netzwerkbetrieb nicht sichergestellt werden kann, daß alle Events übertragen werden (z.B. verlorengegangene UDP-Traps), kann die Root-Cause-Analyse und Impact-Analyse auf Basis der Eventauswertung fehlerhaft sein.

  Betrachtet man die Ergebnisse der RCA- und Impact-Analyse auf Basis der Eventauswertung als "Verdachtsfälle", die in einer zweiten Stufe durch Abfragen im Netz überprüft werden, so läßt sich mit einem zweistufigen Verfahren die **Genauigkeit der Fehleranalyse** und die Beschreibung der Fehlerauswirkung auf die Services im Netzwerk **steigern**.

- Durch die Einbindung der Events in das Netzwerkmodell ist die Erstellung der Prüfroutinen automatisch möglich. Diese Vorgehensweise ist robust gegen Änderungen der Topologie und der Konfiguration des Netzwerks (sofern die Modifikationen im Netzwerkmodell dokumentiert werden). Dies ermöglicht die **Bearbeitung beliebiger eintreffender Events**. Die **Beschränkung auf eine begrenzte und vorher bestimmte Menge von Events**, der die gegenwärtigen Systeme unterliegen, **entfällt**.
- Die Netzlast, die aus den Online-Abfragen zur Fehlereingrenzung im Netzwerk resultiert, beschränkt sich auf das notwendige Maß, da nur Abfragen aufgrund von Verdachtsfällen gestartet werden.

7.3.7 Eigenes Konzept zum Aufbau einer Fault Correction Engine (FCE)

Mit der Erweiterung des Netzwerkmodells und dem Ergebnis der Fault Detection Engine (FDE) kann der Aufbau einer Fault Correction Engine (FCE) nach Abbildung 7.41 erfolgen. Wesentliches Merkmal der Architektur der FCE ist der **dreistufige Aufbau**. Hierbei dient die erste Stufe zur Planung der Troubleshootingmaßnahme. Die zweite Stufe führt die geplante Troubleshootingmaßnahme durch. Die dritte Stufe dient der Erfolgskontrolle und Dokumentation der durchgeführten Maßnahme bzw. der Delegation der Störung an ein Informationsverwaltungs- und Informationsaustauschsystem (vgl. Teil II), wenn eine automatische Entstörung nicht erfolgreich war. Die Funktionsweise der FCE ist folgende.

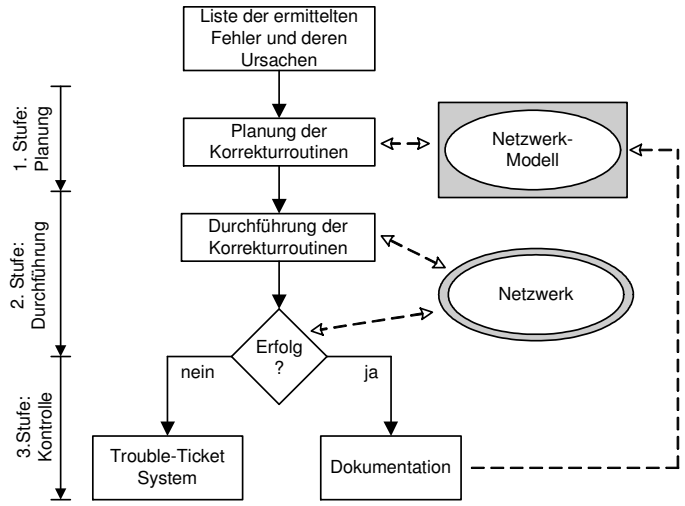


Abbildung 7.41: Eigenes Konzept zum Aufbau der Fault Correction Engine (FCE)

In der **ersten Stufe (Planung)** wird ausgehend von der Fehlerhierarchie und der Prüfroutine eine Korrekturroutine erstellt. Diese setzt sich aus den **Correct**-Objekten der mit **Occurence**-Objekten markierten Komponenten des Netzwerks zusammen. Abbildung 7.42

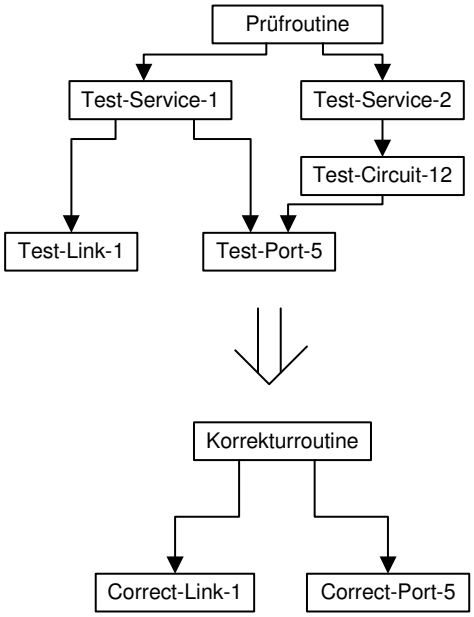


Abbildung 7.42: Beispielhafte Erstellung der Korrekturroutine aus der Prüfroutine

zeigt die Erstellung der Korrekturroutine aus der Prüfroutine für das Beispiel in Abbildung 7.33, Seite 135 (vgl. auch Abbildung 7.40, Seite 140)
Der Aufbau der Korrekturroutine ergibt sich aus folgender beispielhafter Annahme. Von der Prüfroutine wurde festgestellt, daß zur Entstörung von **Service-1** die Komponenten **Link-1** (**Occurence-1**) und **Port-5** (**Occurence-2**) entstört werden müssen. Zur Entstörung von **Service-2** ist ebenfalls die Entstörung der Komponente **Port-5** (**Occurence-2**) erforderlich (vgl. Abschnitt "Root-Cause Analyse", Seite 136). Zur Entstörung von **Service-2** ist die Entstörung von **Circuit-12** (**Occurence-3**) nicht erforderlich, da in diesem Beispiel folgender Sachverhalt angenommen wird. Die Prüfroutine hat mittels dem **Test-Circuit-12**-Objekt festgestellt, daß der Fehler von **Circuit-12** (**Occurence-3**) eine Folge des Fehlers von **Port-5** (**Occurence-2**) ist, weshalb eine Entstörung von **Circuit-12** (**Occurence-3**) nicht erforderlich ist.

In der **zweiten Stufe (Durchführung)** werden die geplanten Troubleshootingmaßnahmen durchgeführt und das Ergebnis protokolliert.

In der **dritten Stufe (Kontrolle)** erfolgt eine Erfolgsprüfung der vorgenommenen Troubleshootingmaßnahmen. Im Erfolgsfall werden die vorgenommenen Korrekturmaßnahmen dokumentiert und gegebenenfalls die hieraus resultierenden Veränderungen in das Netzwerkmodell eingepflegt. Ist eine automatische Entstörung nicht möglich (z.B. Hardwareaustausch), so wird der Fehlerfall inklusive aller bis dahin vom System gesammelten Daten an ein Informationsverwaltungs- und Informationsaustauschsystem (vgl. Teil II) zur weiteren Bearbeitung übergeben.

Der **Vorteil** des vorgestellten Aufbaus einer FCE im Vergleich zu den gegenwärtig exi-

stierenden Lösungen liegt darin, daß **Troubleshootingmaßnahmen für vorher nicht bestimmte Events bzw. Fehlersituationen geplant und durchgeführt werden können**, da auf Basis der **Correct**-Objekte und deren Parametrisierung individuelle Korrekturroutinen erstellbar sind. Heutige Systeme leisten dies nur für eine vorher bestimmte Menge von Fehlersituationen, wobei die Troubleshootingmaßnahme für jede Fehlersituation vom Operator manuell erstellt werden muß.
Das vorgeschlagene Verfahren ist **robust gegenüber Änderungen in der Topologie** bzw. der Konfiguration des Netzwerks, da das Verfahren zur Planung der Troubleshootingmaßnahmen entkoppelt ist von den Topologieinformationen des Netzwerks bzw. diese aus dem Netzwerkmodell bezieht.

7.4 Fazit

Betrachtet man den Stand der Netzwerkmanagementsysteme so ist festzustellen, daß gegenwärtig kein System ein automatisches Netzwerkmanagement zu leisten vermag. Allenfalls findet man Lösungen, die immer wiederkehrende Routineaufgaben von geringer Komplexität bearbeiten können. Diese Lösungen sind optionale Module von Umbrella-Management-Systemen, spielen dort aber eine untergeordnete Rolle.
Der Hauptgrund für diese Situation liegt in dem Architekturkonzept der Umbrella-Management-Systeme (vgl. Abbildung 7.5, Seite 104), welche auf eine passive Überwachung des Netzwerks ausgelegt sind. Durch eine Informationsvorverarbeitung der Netzwerkbetriebsdaten wird der Operator mit den für ihn relevanten Informationen versorgt. Ferner wird ein schneller administrativer Zugang zu den Netzwerkkomponenten ermöglicht.
Um eine **Automatisierung des Netzwerkmanagements** zu erreichen, ist in der Arbeit ein **neues Architekturkonzept für ein aktives Netzwerkmanagementsystem** entstanden, welches nicht auf passive Überwachung, sondern auf aktives Management ausgelegt ist (vgl. Abbildung 7.7, Seite 106).
Grundvoraussetzung für eine **Automatisierung des Netzwerkmanagements** ist die **Netzwerkdokumentation**, da diese Informationen über den Aufbau und die Struktur des Netzwerks enthält, die zur Analyse der Fehler und zur Planung der Troubleshootingmaßnahmen benötigt werden.
Aus diesem Grund ist eine **neue Art der Netzwerkdokumentation** entwickelt worden, mit der eine automatische Analyse von Netzwerkfehlern und die automatische Planung von Troubleshootingmaßnahmen möglich wird (vgl. Abschnitte 7.3.1 bis 7.3.5).
Für das o.g. Architekturkonzept eines aktiven Netzwerkmanagementsystems sind **neue Konzepte zum Aufbau von Fehlererkennungs- und Fehlerkorrekturkomponenten** entwickelt worden, die auf Basis der neuen Art der Netzwerkdokumentation **automatische Fehlererkennungs- und Troubleshootingmaßnahmen** vornehmen (vgl. Abschnitte 7.3.6 und 7.3.7). Dies ermöglicht die automatische Bearbeitung der Fehlerfälle, so daß **die Beschränkung der automatischen Bearbeitung auf eine vorbestimmte und begrenzte Menge von Fehlerfällen**, der heutige Systeme unterliegen, **entfällt**.

Teil IV

Fazit und Ausblick

Kapitel 8

Fazit und Ausblick

Die **IT-technische Unterstützung von HelpDesk-Systemen** untergliedert sich in **zwei Teilbereiche**. Der eine Bereich ist die **IT technische Unterstützung der manuellen Entstörung von Netzwerken**. Der anderen Bereich ist die **automatisierte Überwachung und Entstörung von Netzwerken**.

Zur **IT-technischen Unterstützung der manuellen Entstörung** werden **Trouble-Ticket Systeme** eingesetzt, die eine schnelle Kommunikation zwischen den Servicemitarbeitern ermöglichen. Die Beschreibung der Funktionsweise und der Struktur eines Trouble-Ticket Systems erfolgt mit **herstellerproprietären Applikationsmodellen**. Diese sind einerseits zueinander inkompatibel. Andererseits ermöglichen sie nur Modellierungen auf rechnernaher Ebene, die Anforderungen an ein Trouble-Ticket System werden aber auf Geschäftslogikebene gestellt.

In dieser Arbeit wird das Konzept der Trouble-Ticket Systeme zu **Datenhaltungs- und Informationsaustauschsystemen** weiterentwickelt, die schnell und flexibel an Supportdienstleistungseinheiten anpaßbar sind. Hierzu ist ein **generisches Modell einer HelpDesk-Applikation** entwickelt worden, welches die Planung und Modellierung eines Datenhaltungs- und Informationsaustauschsystems ermöglicht. **Kernpunkt** des generischen Modells ist ein **zustandsraumbasiertes Steuerungssystem**, welches eine **neue Art der Steuerung, Regelung und Überwachung von Informationsströmen** in HelpDesk-Systemen darstellt. Der **Nachweis der Anwendbarkeit** des generischen Modells und des zustandsraumbasierten Steuerungssystems ist durch eine **prototypische Implementierung** erbracht worden.

Da alle in der Arbeit untersuchten HelpDesk-Applikationen auf einer Client-Server Architektur basieren, ist untersucht worden, ob andere Architekturformen Vorteile beim Aufbau einer HelpDesk-Applikation bieten. Exemplarisch dafür ist ein Architekturkonzept für eine HelpDesk-Applikation auf Basis mobiler Agenten entworfen und untersucht worden. Ein solches Architekturkonzept bietet **Vorteile bei der Ausnutzung der Netzwerkressourcen**. Zur Erfüllung der Anforderungen, die der praktische Einsatz an eine

HelpDesk-Applikation stellt, ist in dem Architekturkonzept der Aufbau von Systemmanagementdiensten notwendig, die eine Client-Server Struktur aufweisen, weshalb der Vorteil durch eine andere Architekturform nur bedingt vorhanden ist.

Der **praktische Nutzen des generischen Modells** liegt darin, daß es eine Beschreibung des Aufbaus und der Funktionsweise einer HelpDesk-Applikation ermöglicht, die unabhängig von der verwendeten Entwicklungs- und Betriebsumgebung ist. Desweiteren erfordert die Portierung einer HelpDesk-Applikation von einer Entwicklungs- und Betriebsumgebung zu einer anderen nicht mehr den Neuentwurf der HelpDesk-Applikation, wie das bei Verwendung der Applikationmodelle erforderlich ist.

Entwicklungsmöglichkeiten sind in einer weiteren Ausarbeitung des Konzepts einer HelpDesk-Applikation auf Basis mobiler Agenten zu sehen, da hierdurch HelpDesk-Applikationen ermöglicht werden, die ein Potential von Adaptions- und Automatisierungsmöglichkeiten aufweisen, welches weit über dem heutiger Client-Server basierter Systeme liegt.

Zur **automatischen Überwachung des Netzwerks und zur automatisierten Entstörung von Netzwerkfehlern** werden **Umbrella-Management-Systeme** eingesetzt. Diese sind auf eine **passive Überwachung des Netzwerks** ausgelegt. Sie zeigen dem Servicemitarbeiter durch eine Informationsvorverarbeitung nur die für ihn wichtigen Informationen an und ermöglichen einen schnellen Zugang zu den Netzwerkkomponenten. Durch optionale Module können wiederkehrende Routineaufgaben mit geringer Komplexität automatisiert bearbeitet werden. **Ein automatisches Netzwerkmanagement vermögen die Umbrella-Management-Systeme** aufgrund ihrer Architektur **nicht zu leisten**.

Grundvoraussetzung für die Automatisierung des Netzwerkmanagements ist die **Netzwerkdokumentation**. Zur Unterteilung des Netzwerks in funktionale Ebenen, wie sie für eine Netzwerkdokumentation benötigt wird, ist in dieser Arbeit ein **Schichtenmodell der Netzwerkdokumentation** entwickelt worden. Ausgehend davon ist eine **neue Art der Netzwerkdokumentation** entstanden, mit der aktuelle Fehlerfälle aus dem Netzwerk in das Modell abbildbar sind. Dadurch ergeben sind **neue Ansätze zum Aufbau modellbasierter Impact- und Root-Cause Analysen**.

Für den **Aufbau eines automatischen Netzwerkmanagements** ist in dieser Arbeit ein **Architekturkonzept für ein aktives Netzwerkmanagement** geschaffen worden.

Es sind **Architekturkonzepte zum Aufbau von Komponenten für die automatische Fehlererkennung und Fehlerkorrektur** entwickelt worden, die zu folgendem Vorteil führen. Die Beschränkung heutiger Systeme auf eine begrenzte Menge vorher festgelegter Fehlerfälle, die automatisch bearbeitet werden können, entfällt.

Der **praktische Nutzen** der neuen Art der Netzwerkdokumentation liegt in der Verknüpfung der Informationen über den Aufbau des Netzwerks, seiner Dienste und dessen

Nutzer einerseits, und der Informationen über den aktuellen Betriebszustand andererseits. Hierdurch können Aussagen getroffen werden, wie sich technische Netzwerkfehler auf die Dienste und die Nutzer auswirken bzw. ist feststellbar, aufgrund welcher technischen Störungen ein Dienst nicht funktioniert.

Weitere **Entwicklungsmöglichkeiten** für den Aufbau eines automatischen Netzwerkmanagements sind in der Entwicklung des Aufbaus und der Funktionsweise von **Komponenten zur Fehlererkennung und zur Fehlerkorrektur** zu sehen. Der Schwerpunkt sollte hierbei auf **robuste Verfahren und Strategien** liegen, die in heterogenen Telekommunikationsnetzwerken anwendbar sind.

Langfristiges Ziel der **Weiterentwicklung der IT-Unterstützung von HelpDesk-Systemen** muß die **Integration** von Datenhaltungs- und Informationsaustauschsystemen, automatischen Netzwerkmanagementsystemen und Netzwerkdokumentationssystemen zu einem **Operation-Support-System** sein. Dieses betreibt alle operativen Netzwerkprozesse automatisch.

Literaturverzeichnis

[aCs01/1] aCs (Distributor BlueOcean Software) - Datenbankinformationen
http://www.acscompany.de/f3_pro_ee.htm
Seitenabruf: 07.03.2001

[aCs01/2] aCs (Distributor BlueOcean Software) - Standinformationen CeBit 2001
Standbesuch am 22.03.2001

[ARS-00] "Action Request System" Produkt-Dokumentation (v4.0.x)
Remedy Cooperation, 2000
Mountain View, Californien (USA)

[Avensoft01/1] Perfect Tracker (Avensoft) - Produktinformationen,
http://www.avensoft.com
Seitenabruf: 06.03.2001

[Balzert96] Balzert, Helmut
Lehrbuch der Softwaretechnik: Software-Entwicklung
Spektrum Akademischer Verlag GmbH 1996
Heidelberg, Berlin, Oxford

[Balzert98] Balzert, Helmut
Lehrbuch der Softwaretechnik:
Software-Management,
Software Qualitätssicherung,
Unternehmensmodellierung
Spektrum Akademischer Verlag GmbH 1998
Heidelberg, Berlin

[Balzert01] Balzert, Heide
UML kompakt
Spektrum Akademischer Verlag GmbH 2001
Heidelberg, Berlin

[Benze99] Benze, Jörg
Einsatz von HelpDesk-Systemen zur Optimierung des

System-Supports bei Dienstleistungsunternehmen
der Kommunikationsbranche
5. Workshop Multimediale Informations- und Kommunikationssysteme (MIK)
innerhalb der
5. Fachkonferenz Smalltalk und Java in Industrie und Ausbildung (STJA)

[Benze00] Benze, Jörg
Unterlagen zum 1. Netzwerkmanagement-Workshop innerhalb der Vorlesung Netzwerkmanagementsysteme des Fachgebiets Telematik.
10. Januar 2000, Technische Universität Ilmenau

[Benze00/2] Benze, Jörg
Umdruck zum Vortrag:
HelpDesk-Systeme: Neue Ansätze zur rechnerunterstützten Optimierung des Supportmanagements
4. Remedy User Group Deutschland (http://www.rug-germany.de)
30. - 31. Mai 2000 / Hamburg

[Benze00/3] Benze, Jörg
HelpDesk-Systeme
- Aufbau, Betrieb, Optimierung -
45. Internationales Wissenschaftliches Kolloquium (IWK)
Technische Universität Ilmenau

[Benze00/4] Benze, Jörg
HelpDesk-Systeme: Struktur und Aufgabenmanagement
6. Workshop Multimediale Informations- und Kommunikationssysteme (MIK) innerhalb der Net.ObjectDays 2000.

[Benze00/5] Benze, Jörg
Umdruck zum Vortrag:
Vorstellung des Controlware Support-Tools:
Softwarestudie: Datenaustausch über die AR-System API
5. Remedy User Group Deutschland (http://www.rug-germany.de)
28. - 29. November 2000
Berlin

[Benze01] Benze, Jörg
Unterlagen zum 2. Netzwerkmanagement-Workshop innerhalb der Vorlesung Netzwerkmanagementsysteme des Fachgebiets Telematik.
18. Januar 2001, Technische Universität Ilmenau

[Benze01/2] Benze, Jörg
Umdruck zum Vortrag:
Geschäftsprozessmodellierung mit UML und Realisierung mit dem AR-System
6. Remedy User Group Deutschland (http://www.rug-germany.de)
18. - 19. Juni 2001
Dresden

[Benze01/3] Benze, Jörg
Analyse und Design von Geschäftsprozessen mit der Unified Modeling Language (UML)
7. Workshop Multimediale Informations- und Kommunikationssysteme (MIK) innerhalb der Net.ObjectDays 2001.

[Benze02] Benze, Jörg
Unterlagen zum 3. Netzwerkmanagement-Workshop innerhalb der Vorlesung Netzwerkmanagementsysteme des Fachgebiets Telematik.
18. Januar 2002, Technische Universität Ilmenau

[Benze02/2] Benze, Jörg
Umdruck zum Vortrag:
Analyse, Design und Optimierung von Geschäftsprozessen in Supportorganisationen
7. Remedy User Group Deutschland (http://www.rug-germany.de)
27. - 28. Mai 2002
München

[Benze02/3] Benze, Jörg
Ansätze zum automatischen Fehlermanagement in Netzwerken
8. Workshop Multimediale Informations- und Kommunikationssysteme (MIK) innerhalb der Net.ObjectDays 2002.

[Benze03] Benze, Jörg
Unterlagen zum 4. Netzwerkmanagement-Workshop innerhalb der Vorlesung Netzwerkmanagementsysteme des Fachgebiets Telematik.
15. Januar 2003, Technische Universität Ilmenau

[Benze03/2] Benze, Jörg
Beschreibung von Fehlern in heterogenen Telekommunikationsnetzwerken und Ansätze zur Automatisierung des Troubleshootings
9. Workshop Multimediale Informations- und Kommunikationssysteme (MIK) innerhalb der Net.ObjectDays 2003.

[Benze03/3] Benze, Jörg
Unterlagen zum 5. Netzwerkmanagement-Workshop innerhalb der Vorlesung Netzwerkmanagementsysteme des Fachgebiets Telematik.
24. November 2003, Technische Universität Ilmenau

[Benze04] Benze, Jörg
Unterlagen zum 6. Netzwerkmanagement-Workshop innerhalb der Vorlesung Netzwerkmanagementsysteme des Fachgebiets Telematik.
29. November 2004, Technische Universität Ilmenau

[Blanchard98] Blanchard, Kenneth * Carlos, John * Randolph, Alan
Management durch Empowerment
Rowohlt Taschenbuchverlag GmbH 1999
Hamburg

[Blanchard00] Blanchard, Kenneth * Johnson, Spencher
Der Minuten Manager
Rowohlt Verlag GmbH 2000, 25. Auflage
Hamburg

[BlueOcean01/1] Track-It (BlueOcean Software) - Produktblatt

[BlueOcean01/2] Track-It (BlueOcean Software) - Produktinformationen
http://www.blueocean.com/product.asp
Seitenabruf: 06.03.2001

[BlueOcean01/3] Track-It (BlueOcean Software) - Systemanforderungen
http://www.blueocean.com/trackit.asp
Seitenabruf: 07.03.2001

[BlueOcean01/4] Track-It (BlueOcean Software) - Enterprise Edition
http://www.blueocean.com/enterprise.html
Seitenabruf: 07.03.2001

[Booch99] Booch, Grandy * Rumbaugh, Jim * Jacobson, Ivar
Das UML-Benutzerhandbuch
Addison-Wesley 1999, 2. Auflage
Bonn

[Bruton97] Bruton, Noel
How to Manage the I.T. HelpDesk
Butterworth-Heinemann 1997
Oxford

[Class98] C. Class
Exkurs: Endliche Automaten
Übung zur Vorlesung "Lokale Netze" im Wintersemster 1998/1999
http://www.tik.ee.ethz.ch/tik/education/lecture/LAN/automaten.pdf
Seitenabruf: 28.11.2001

[ComConsult01/1] CCM4 (ComConsult) - Produktinformationen
http://www.comconsult.de/ccm/uebersicht.html
Seitenabruf: 06.03.2001

[ComConsult01/2] CCM4 (ComConsult) - Systemanforderungen
http://www.comconsult.de/ars/index.html
Seitenabruf: 06.03.2001

[ConSol01/1] CallManager Help (ConSol) - Produktinformationen
http://www.consol.de/de/callmanager/main.hei
Seitenabruf: 09.03.2001

[ConSol01/2] CallManager Help (ConSol) - Systemanforderungen
http://www.consol.de/de/callmanager/reqs.hei
Seitenabruf: 09.03.2001

[ConSol01/3] CallManager Help (ConSol) - Integrationsmöglichkeiten
http://www.consol.de/de/callmanager/integration.hei
Seitenabruf: 09.03.2001

[ConSol01/4] CallManager Help (ConSol) - Standinformationen CeBIT 2001
Standbesuch am 22.03.2001

[Ct8/2000] c't Magazin für Computer und Technik
Ausgabe 08/2000
Artikel: "Dasein oder Nicht-Dasein"
- Analyse der Ausfallzeiten von Web-Servern - / Seite 174ff
Verlag Heinz Heise GmbH & CoKG
Hannover

[Czegel98] Czegel, Barbara
Running an effective HelpDesk, 2. Auflage
John Wiley & Sons, Inc. 1998
New York, Chichester, Weinheim, Brisbane, Singapore, Toronto

[Czegel99] Czegel, Barbara
HelpDesk Practitioner's Handbook
John Wiley & Sons, Inc. 1999
New York, Chichester, Weinheim, Brisbane, Singapore, Toronto

[Czegel01] Czegel, Barbara
Technical Support on the Web
John Wiley & Sons, Inc. 2001
New York, Chichester, Weinheim, Brisbane, Singapore, Toronto

[Davenport99] Davenport, Thomas H.
Wenn Ihr Unternehmen wüßte, was es alles weiß ...:
das Praxishandbuch zum Wissensmanagement.
Verlag Moderne Industrie 1998 / 2. Auflage 1999
Landsberg/Lech

[DeMarco97] De Marco, Tom
Warum ist Software so teuer?: ... und andere Rätsel des Informationszeitalters
Carl Hanser Verlag 1998
München, Wien

[DeMarco98] DeMarco, Tom
Der Termin: Ein Roman über Projektmanagement
Carl Hanser Verlag 1998
München, Wien

[DeMarco99] De Marco, Tom * Listener, Timothy R.
Wien wartet auf Dich!: Der Faktor Mensch im DV-Management
Carl Hanser Verlag 1999 / 2. aktualisierte und erw. Aufl.
München, Wien

[DeMarco01] De Marco
Spielräume
Carl Hanser Verlag 2001
München, Wien

[Drzyzga00] Drzyzga, Uwe
Personalgespräche richtig führen -
Ein Kommunikationsleitfaden
Deutscher Taschenbuchverlag GmbH & Co. KG 2000
München

[Föllinger94] Föllinger, Otto
Regelungstechnik
Einführung in die Methoden und ihre Anwendungen
Hüthig-Verlag 1994/ 8. überarbeitete Auflage
Heidelberg

[Füser99] Füser, Karsten
Modernes Management -

Lean Management, Business Reengineering, Benchmarking und viele andere Methoden
Deutscher Taschenbuchverlag GmbH & Co. KG 2000
2. überarbeitete Auflage
München

[Grässle00] Grässle, Patrick * Baumann, Henriette * Baumann, Philippe
UML projektorientiert -
Geschäftsprozessmodellierung, IT-System-Spezifikation und Systeminteration mit der UML
Galileo Press GmbH 2000
Bonn

[GNats01/1] GNats - Programmbeschreibung
http://sources.redhat.com/gnats
Seitenabruf: 15.03.2001

[HelpSTAR01/01] HelpSTAR 2000 (HelpStar) - Produktinformationen
http://www.helpstar.com/products/workflowhs2000.asp
Seitenabruf: 05.03.2001

[HelpSTAR01/02] HelpSTAR 2000 (HelpStar) - Webschnittstelle
http://www.helpstar.com/products/wi2000.asp
Seitenabruf: 05.03.2001

[HelpSTAR01/03] HelpSTAR 2000 (HelpStar) - Systemanforderungen
http://www.helpstar.com/products/techspecs.asp
Seitenabruf: 06.03.2001

[HelpSTAR01/04] euro-tools (Distributor HelpSTAR 2000) - Standinformationen CeBIT 2001
Standbesuch am 22.03.2001

[Horn95] Horn, Christian * Kerner, Immo O. (Hrsg.) Lehr- und Übungsbuch Informatik: Band 1: Grundlagen und Überblick
Fachbuchverlag Leipzig GmbH 1995
Leipzig

[Horn97] Horn, Christian * Kerner, Immo O. (Hrsg.) Lehr- und Übungsbuch Informatik: Band 3: Praktische Informatik
Fachbuchverlag Leipzig im Carl Hanser Verlag 1997
München, Wien

[Horn98] Horn, Christian * Kerner, Immo O. (Hrsg.) Lehr- und Übungsbuch Informatik: Band 4: Technische Informatik und Systemgestaltung
Fachbuchverlag Leipzig im Carl Hanser Verlag 1998
München, Wien

[Horn00] Horn, Christian * Kerner, Immo O. (Hrsg.) Lehr- und Übungsbuch Informatik: Band 2: Theorie der Informatik
Fachbuchverlag Leipzig im Carl Hanser Verlag 2000
München, Wien

[Horvath00] Horváth & Partner
Das Controllingkonzept -
Der Weg zu einem wirkungsvollen Controllingsystem
Deutscher Taschenbuchverlag GmbH & Co. KG 2000
4. überarbeitete und erweiterte Auflage
München

[Intraware01/1] OCTOHelp+ (Intraware) - Produktinformationen
http://www.intraware.de
Seitenabruf: 08.03.2001

[Intraware01/2] OCTOHelp+ (Intraware) - Standinformationen CeBIT 2001
Standbesuch am 22.03.2001

[iX3/2002] iX Magazin für professionelle Informationstechnik
Ausgabe 03/2002
Artikel: "Ausgeliefert"
- Überwachung von Servicevereinbarungen - / Seite 106ff
Verlag Heinz Heise GmbH & CoKG
Hannover

[Jablonski97] Jablonski, Stefan * Böhm, Markus * Schulze, Wolfgang (Hrsg.)
Workflow-Management: Entwicklung von Anwendungen und Systemen; Facetten einer neuen Technologie
dpunkt-Verlag / 1. Auflage 1997
Heidelberg

[Johnson01] Johnson, Spencer
Die Mäusestrategie für Manager -
- Veränderungen erfolgreich begegnen
Heinrich Hugendubel Verlag 1998 / 5. Auflage
München

[KemmaSoftware01/1] BridgeTrak (Kemma Software) - Produktinformationen,
http://www.kemma.com/fact.htm
Seitenabruf: 06.03.2001

[KemmaSoftware01/2] BridgeTrak (Kemma Software) - Modul BridgeWeb
http://www.kemma.com/bridgeweb.htm
Seitenabruf: 06.03.2001

[KemmaSoftware01/3] BridgeTrak (Kemma Software) - Modul Architektur,
http://www.kemma.com/bridgetrakable.htm
Seitenabruf: 06.03.2001

[Krause01] Krause, David
Geschäftprozessanalyse mit UML
- Verfahren zur Geschäftsprozeßmodellierung -
Praktikumsbericht zum 2. Praxissemester,
Kooperativer Studiengang Informatik (KoSI),
FH Darmstadt (Fachbereich Informatik) 2001

[Krause03] Krause, David
Ansätze zur Automatisierung des Netzwerkmanagements in heterogenen Telekommunikationsnetzwerken (Diplomarbeit)
Kooperativer Studiengang Informatik (KoSI),
FH Darmstadt (Fachbereich Informatik) 2003

[Kruchten99] Kruchten, Philippe
Der Rational Unified Process
- Eine Einführung - Addison-Wesley 1999
München

[Lewis00] Lewis, Lundy
Event Correlation in SPECTRUM and Other Commercial Products
http://www.aprisma.com/literature/white-papers/wp0551.pdf
Seitenabruf: 10.07.2003

[Lobscheid98] Lobscheid, Hans Gerd
Mitarbeiter einvernehmlich führen,
Verlag C. H. Beck / 2. überarbeitete und ergänzte Auflage
München 1998

[Macfarlane03] Macfarlane, Ivor * Rudd, Colin
IT Service Management
Ein Begleitband zur IT Infrastructure Library
IT Service Management Forum
Webbs Court
8 Holmes Road
Earley
Reading RG6 7BH
Großbritannien

[MagicSolutions01/1] Magic (Magic Solutions) - Produktinformationen
http://www.magicsolutions.com/products/help/default.asp
Seitenabruf: 09.03.2001

[MagicSolutions01/2] Magic (Magic Solutions) - Standinformationen CeBit 2001
Standbesuch am 22.03.2001

[Materna00] Unterlagen zu AR-Benutzer- und Administratorschulung
Materna GmbH, 2000
Dortmund

[Materna01] Materna Helpdesk (Materna) - Produktinformationen
http://www.materna.de/helpdesk.html
Seitenabruf: 07.03.2001

[Maxpert01/1] ServiceDesk (Maxpert) - Produktinformationen
http://www.maxpert.de
Seitenabruf: 06.03.2001

[Maxpert01/2] - Standinformationen CeBit 2001
Standbesuch am 22.03.2001

[Meffert00] Meffert, Heribert
Marketing -
Grundlagen marktorientierter Unternehmensführung
Konzepte - Instrumente - Praxisbeispiele
Gabler Verlag 2000 / 9. Auflage
Wiesbaden

[Micromuse02/1] Micromuse Inc., San Francisco
Micromuse Schulungsunterlagen 2002
Netcool User Schulung
Netcool Administrator Schulung
http://www.micromuse.com

[Micromuse02/2] Micromuse Inc., San Francisco
Netcool / Virtual Operator
http://www.micromuse.com/downloads/pdf_lit/Netcool_VO.pdf
http://www.micromuse.com

[Oestereich98] Oestereich, Bernd
Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language
Oldenburg Verlag / 4. aktualisierte Auflage 1998
1. korrigierter Nachdruck 1999
München, Wien

[Oestereich01] Oestereich, Bernd
Teilnehmerunterlagen zum Kurs:
Objektorientierte Analyse und Design mit der UML

Version/Exemplar: 27.08.2000/2
08. - 11. Januar 2001, Hamburg

[Oestereich01/2] Oestereich, Bernd
UML Kurzreferenz für die Praxis: kurz, bündig, ballastfrei
Oldenburg Verlag 2001
München, Wien

[Peregrine01/1] ServiceCenter (Peregrine) - Produktinformationen
http://www.peregrine.de
Seitenabruf: 06.03.2001

[Peregrine01/2] ServiceCenter (Peregrine) - Standinformationen CeBIT 2001
Standbesuch am 22.03.2001

[Phaidros01/1] Phaidros - Programmbeschreibung
http://www.phaidros.de
Seitenabruf: 12.04.2001

[Phaidros01/2] Phaidros - Systemanforderungen
http://www.phaidros.de/german/phaidros/index.htm
Seitenabruf: 12.04.2001

[Plessner02] Plessner, Dr. Klaus
Root-Cause Analyse / Spezialisten für die Fehlersuche
NetworkWorld 04/2002
http://www.networkworld.de/index.cfm?
pageid=194&id=77781&type=detail
Seitenabruf: 16.07.2003

[Ploenzke01] Haible, Jürgen * Kiesel, Michael * Renninger, Wolfgang
Wunsch und Wirklichkeit -
Kundenorientierung in deutschsprachigen Unternehmen
Studie zum Thema
Customer Relationship Management
2. Management-Enquete
CSC PLOENZKE AG, Fachhochschule Amberg-Weiden

[Quintus00/1] CustomerQ (Quintus) - Produktinformationen
http://www.quintus.com
Seitenabruf: 06.03.2001

[Quintus00/2] CustomerQ (Quintus) - Systemanforderungen
http://www.quintus.com/pdfs/custq.pdf
Seitenabruf: 07.03.2001

[Quintus00/3] CustomerQ (Quintus) - Systembeschreibung
http://www.quintus.com/prodServ/prod/econtact56.cfm
Seitenabruf: 07.03.2001

[Remedy] Remedy Corporation
Mountain View
Californien
Die Firma Remedy Corporation ist von der Firma Peregrine Mitte 2001 aufgekauft worden; die Produktline "Action Request System" von Remedy wird von Peregine weiterentwickelt und weiterhin vertrieben. Anfang 2003 ist Peregrines Remedy-Sparte von BMC Software übernommen worden.

[Remedy01/1] Remedy Help Desk (Remedy) - Produktinformationen
http://www.remedy.com/solutions/ebis/itsm/datasheets/help_desk.htm
Seitenabruf: 06.03.2001

[Remedy01/2] Remedy Help Desk (Remedy) - Systemanforderungen
http://www.remedy.com/shared/datasheets/pdfs/ars_german.pdf
Seitenabruf: 06.03.2001

[Remedy01/3] Remedy Help Desk (Remedy) - Frequently Asked Questions (FAQ)
http://www.remedy.com/solutions/core/arsystem_faq45.htm
Seitenabruf: 09.03.2001

[Reschke02] Krüger, Gerhard * Reschke, Dietrich
Lehr- und Übungsbuch Telematik
Fachbuchverlag Leipzig im Carl Hanser Verlag 2002
München, Wien

[Röhrig00] Röhrig, Christian * Sefke, Alex
Corporate Servicemanagement: Organisation und Technik interner Dienstleistungen
Addison-Wesley Verlag / 1. Auflage 2000
München, Boston, San Francisco, Harlow (Englang), Don Mills (Ontario), Sydney, Mexico City, Madrid, Amsterdam

[Sattler98] Sattler, Ralf
Unternehmerisch denken lernen -
Das Denken in Strategie, Liquidität, Erfolg und Risiko
Deutscher Taschenbuchverlag GmbH & Co. KG 1998
München

[Schelle99] Schelle, Heinz
Projekte zum Erfolg führen -
Projektmanagement systematisch und kompakt

Deutscher Taschenbuchverlag GmbH & Co. KG 1999
2. überarbeitete und erweiterte Auflage
München

[SDS01/1] SDS Helpdesk (ScottDataSystems)
- Produktinformationen
http://www.scottdatasystems.com/product.html
Seitenabruf: 06.03.2001

[Smarts01] Smarts
System Management ARTS, Inc., New York
Automating Root Cause Analysis
Codebook Correlation Technology vs. Rules Based Analysis
http://www.smarts.com/resources/CCT_WhitePaper_1001.pdf
Seitenabruf: 16.07.2003

[Soffront01/1] TrackWeb Helpdesk (Soffront Software) - Produktinformationen
http://www.trackhelpdesk.com
Seitenabruf: 06.03.2001

[Soffront01/2] TrackWeb TRACKKB (Soffront Software) - Knowledge Base
http://www.trackkb.com/02_Knowledge_Base_Features_Benefits.htm
Seitenabruf: 06.03.2001

[Soffront01/3] TrackWeb Helpdesk (Soffront Software) - Produktbeschreibung
http://www.trackhelpdesk.com/03_Help_Desk_Software.html
Seitenabruf: 06.03.2001

[Soffront01/4] TrackWeb Helpdesk (Soffront Software) - Internetkomponente
http://www.trackhelpdesk.com/04_Internet_Help_Desk.html
Seitenabruf: 06.03.2001

[Softlab01/1] STARS (Softlab) - Produktinformationen
http://www.softlab.de/solutions/frm_care00.asp
Seitenabruf: 06.03.2001

[Softlab01/2] STARS (Softlab) - Systemanforderungen
http://www.softlab.de/solutions/dsp_care03.asp
Seitenabruf: 06.03.2001

[Steinder01] Steinder, Malgorzata * Sethi, Adarshpal S.
The present and future of event correlation:
A need for end-to-end service fault localization
http://www.cis.udel.edu/∼steinder/PAPERS/sci2001.pdf
Seitenabruf: 16.07.2003

[Stelzer03] Stelzer, Dirk
Informations- versus Wissensmanagement
Versuch einer Abgrenzung
http://www.wirtschaft.tu-ilmenau.de/deutsch/institute/
wi/wi3/infothek/documents/Stelzer_IMvsWM_2003-07-10.pdf
Seitenabruf: 05.08.2003

[Talisma01/1] Talisma (Talisma) - Produktinformationen,
http://www.talisma.com
Seitenabruf: 23.03.2001

[Teledialogteam] Teledialogteam
http://www.teledialogteam.de/tds.html
Seitenabruf: 03.12.1999

[Umbehauen92] Umbehauen, Heinz
Regelungstechnik I
Lineare Kontinuierliche Regelsysteme
Vieweg Verlag 1992 / 7. überarbeitete und erweiterte Auflage
Braunschweig, Wiesbaden

[Umbehauen89] Umbehauen, Heinz
Regelungstechnik II
Zustandsregelung, digitale und nichtlineare Regelsysteme
Vieweg Verlag 1989 / 5. durchgesehene Auflage
Braunschweig, Wiesbaden

[Umbehauen93] Umbehauen, Heinz
Regelungstechnik III
Identifikation, Adaption, Optimierung
Vieweg Verlag 1993 / 4. durchgesehene Auflage
Braunschweig, Wiesbaden

[Wald99] Wald, Egbert
HelpDesk-Management
MITP-Verlag 1999
Bonn

[WebWonderland01/1] Wonderdesk (Web Wonderland) - Produktinformationen
http://www.wonderdesk.com
Seitenabruf: 07.03.2001

[WebWonderland01/2] Wonderdesk (Web Wonderland) - Features
http://www.wonderdesk.com/features.html
Seitenabruf: 06.03.2001

[WebWonderland01/3] Wonderdesk (Web Wonderland) - Anpassungsmöglichkeiten
http://www.wonderdesk.com/customizing.html
Seitenabruf: 06.03.2001

[Winpeak01/1] Helpdesk Manager (Winpeak Software) - Produktinformationen
http://www.winpeak-software.com/produkte.htm
Seitenabruf: 06.03.2001

[Winpeak01/2] Helpdesk Manager (Winpeak Software) Agent
http://www.winpeak-software.com/AgentDesktop/index_hm.htm
Seitenabruf: 09.03.2001

[Wöhe00] Wöhe, Günter
Einführung in die Allgemeine Betriebswirtschaftslehre
Verlag Franz Vahlen GmbH 2000 / 20. neubearb. Auflage
München