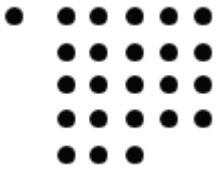

Fachhochschule Köln
Campus Gummersbach
University of Applied Sciences Cologne

Fakultät für Informatik und
Ingenieurwissenschaften

Diplomarbeit

zur Erlangung des Diplomgrades Diplom-Informatiker (FH)
in der Fachrichtung „Allgemeine Informatik“

Konzeption und Entwicklung eines Gatewaydämons für den Fax und Email Versand

Ein Zusatzmodul für die Kommunikationssoftware TOPCALL

In Zusammenarbeit mit der Firma
METRO Group Information Technology GmbH

Erstprüfer : Prof. Dr. rer. nat. Heide Faeskorn-Woyke
Zweitprüfer : Dipl. Kfm. Stephan Houben

vorgelegt am : 6. Mai 2004
von : Manuel Jung
Matr.-Nr. : 110 25 233
aus : Albert-Dimmers-Strasse 13
51469 Bergisch Gladbach

Inhaltsverzeichnis

Abbildungsverzeichnis

Aus Sicherheits- und Datenschutzgründen wurden interne Geschäfts- und systemkritische Daten der Metro Group in den Abbildungen durch einen schwarzen Balken unkenntlich gemacht. Die Verständlichkeit der Abbildungen wird durch diese Maßnahme jedoch nicht beeinträchtigt.

Listingverzeichnis

Analog zu den Abbildungen wurden in den Listings ebenso sensible Daten durch ein „X“ ersetzt.

Abkürzungsverzeichnis

API..... Application Programming Interface
BD..... Basisdienst
BD300..... Fax Basisdienst
BD313..... Fax-Dämon zur Bearbeitung von Kommunikationsaufträgen
BD314..... Fax-Dämon zur Bearbeitung von Rückmeldedateien
C..... Common Service
CRM..... Customer Relationship Management
DOM..... Document Output Management System
DTD..... Data Definition Type
EKSG..... Einkaufsachgebiet
ERP..... Enterprise Resource Planing
FIFO..... First In First Out
FTP..... File Transfer Protokoll
HTML..... Hypertext Markup Language
HTTP..... Hypertext Transfer Protokoll
JDBC..... Java Database Connectivity
JRE..... Java Runtime Engine
JVM..... Java Virtual Machine
MGI..... METRO Group Information Technology GmbH
NFS..... Network File System
PDF..... Portable Document Format
SDK..... Standard Develoment Kit
SQL..... Structured Query Language
TC/LINK-FI..... TOPCALL Link File Interface
TCP..... Transmission Control Protocol
TOM..... TOPCALL Open Message (Format)
URL..... Uniform Resource Locator
UTF8..... Zeichensatz - ländercodeunabhängig
XML..... Extensible Markup Language

Vorwort

Ziel dieser Diplomarbeit war die Neuentwicklung eines bereits bestehenden Softwareprodukts, welches in einem firmeneigenen Intranet als Backend-System[1] zum Einsatz kommt. Es werden die Konzeptplanung, die verwendeten Technologien und die Phasen der Programmierung bis hin zur Einbindung ins System erläutert. Zur schnellen Orientierung, folgt zuerst eine kurze Kapitelübersicht.

Kapitel 1 – Die Einführung

Gewährt einen kurzen Einblick in den größten Handelskonzern „die METRO“ und definiert das Ziel dieser Arbeit. Die hauseigene Fax-Applikation der Metro wird für das weitere Verständnis in seiner Bedienung und anschließend im technischen Ablauf näher erläutert.

Kapitel 2 – Verwendete Technologien

Gibt eine umfassende Einsicht in die verwendeten Technologien und deren Funktionsweisen im Hinblick auf den Fax-Basisdienst der MGI[2].

Kapitel 3 – Die Fax Kommunikationsanbindung

Beschreibt die Systemarchitektur des Fax-Basisdienstes und erläutert die Anbindung der Serversysteme zueinander – Insbesondere wird erläutert, warum einige gute Ideen für dieses Projekt leider wieder verworfen werden mussten.

Kapitel 4 – Der MGI Fax-Basisdienst

Schildert die Funktionsweise der neuen Java-Fax-Dämonen. Die Konfiguration und der Programmablauf wird an Strukturablaufplänen näher dargestellt. Dazu mussten für die Fax-Systemintegration zusätzliche Kopierdämonen entwickelt werden. Mit Hilfe eines Überwachungstools, – der „Admin GUI“, lässt sich der gesamte Prozess der Dämonen auch zur Laufzeit überwachen, und ggf. für eine Fehleridentifikation nutzen. Eine Integration der Module erfolgt später durch sog. Job-Scheduling-Systeme, deren Funktionsweise zuletzt beschrieben wird.

[1] besteht aus den Systemteilen, die zum Funktionieren eines Frontend-Systems benötigt werden.
[2] METRO Group Information Technology GmbH, IT-Servicegesellschaft der METRO

Danksagung

Die vorliegende Diplomarbeit entstand an der Fachhochschule Köln, Campus Gummersbach, im Fachbereich Informatik in Zusammenarbeit mit der Firma MGI - METRO Group Information Technology GmbH Düsseldorf, dem Informatik-Dienstleistungsunternehmen für die Metro Group.

Für die Unterstützung, Anregungen und konstruktive Kritik möchte ich mich an dieser Stelle bei folgenden Personen und Arbeitskollegen herzlich bedanken :

- Frau Heide Faeskorn-Woyke
- Frau Birgit Tang-Rave
- Herr Stephan Houben
- Herr Patric Scharoff
- Herr Peter Haasper
- Herr Olaf Boehm
- Herr Frank Bündgen
- Herr Lars Frik

Allen Mitarbeitern, die ich hier aus Platzgründen nicht auflisten kann, danke ich für die angenehme und kollegiale Arbeitsatmosphäre in der Abteilung für „Development Tool and Support“.

Bevor jedoch die eigentliche Ausarbeitung beginnt, folgt zunächst ein kurzer Überblick über das Unternehmen: Die „Metro Group“ – weit mehr als nur eine große Einkaufsmeile für Händler.

1. Kapitel – Einführung

1.1 Die Metro AG

1964 gilt als Geburtsjahr des Metro-Konzerns. Otto Beisheim eröffnete den ersten Metro Cash & Carry-Markt, einen Abholmarkt für Gewerbetreibende. Pioniere wie Beisheim und innovative Geschäftsmodelle – wie einst die Konsumvereine – prägten die Entwicklung des Handels zu dieser Zeit. Die Wurzeln der Konzernunternehmen, die heute zur Metro AG gehören, reichen aber teilweise bis ins 19. Jahrhundert zurück.

Im Jahre 1996 entstand durch Verschmelzung der selbstständigen traditionsreichen Handelsunternehmen dann die heutige Metro AG. Diese ging noch im gleichen Jahr an die Börse. Mit einer Marktkapitalisierung von 12,07 Milliarden Mark gehörte die Metro AG Ende 1996 zu den 20 größten deutschen börsennotierten Unternehmen. Seit November 2002 wurde die Metro AG organisatorisch in einzelne Querschnittsgesellschaften (Verantwortlichkeitsbereiche) unterteilt, die nun alle zusammen unter dem neuen Namen „Metro Group“ auftreten.

Die Metro ist inzwischen in 27 Ländern vertreten und gehört zu den TOP 5 der weltweit größten Handelskonzerne. 235.000 Mitarbeiter weltweit erzielen einen jährlichen Umsatz von ca. 51,5 Milliarden Euro.

Zu den Vertriebslinien in Deutschland zählen :

Abbildung 1 Vertriebslinien der Metro

Neben diesen Vertriebslinien, die für das operative Geschäft zuständig sind, gibt es in der Metro Group sog. Servicegesellschaften, die für die Vertriebslinien oder den Gesamtkonzern Aufgaben wie Einkauf, Logistik, Informatik, Werbung oder Abfall- und Umweltmanagement übernehmen.

Sie stellen die Schnittstellen der Metro Group Holding dar :

- **MGB** (Metro Group Buying GmbH)
  ist die zentrale Warenbeschaffungsorganisation im Konzern. Landesweit und international ist sie für die Beschaffung von Food- und Nonfood-Waren zuständig. Sie analysiert, bündelt und verhandelt über Beschaffungsvolumina mit Lieferanten. Die Metro ist weltweit der bedeutendste Abnehmer von SONY-Produkten. Jede vierte Socke in Deutschland wird letzten Endes über die Metro bestellt.

- **MGL** (Metro Group Logistics)
  Sie ist für die Organisation und Abwicklung der Warentransporte der Metro AG zuständig. Sie sorgt zusammen mit 3.500 Lieferanten dafür, dass alle Sendungen termingerecht zu den einzelnen Märkten bewegt werden. Das Handelsaufkommen ist inzwischen so groß geworden, dass die Metro jeden Tag mehr Pakete als die Deutsche Post AG transportiert.

- **MRE** (Metro Group Real Estate Management GmbH)
  Die MRE gehört mit rund 1,5 Millionen vermieteten Quadratmetern zu den größten Centermanagementgesellschaften in Deutschland. Sie entwickelt, bebaut, vermarktet und managt mit 500 Mitarbeitern mehr als 200 Standorte. Im Jahresschnitt eröffnet sie alle vier Tage irgendwo auf der Welt einen Markt.

- **MGP** (Metro Group Account Processing GmbH)
  Sie übernimmt zwischen der Industrie und den Vertriebslinien der Metro Group die komplette Rechnungsabwicklung und die papierlose Archivierung der Lieferantenrechnungen bis hin zu Service-Leistungen im Bereich der Zentralregulierung.

- **MGA** (Metro Group Advertising GmbH)
  beschäftigt sich in der Beschaffung, Abwicklung und Koordination von Werbedienstleitungen. Sie entwickelt und verwaltet eine konzernübergreifende Bilddatenbank mit allen lieferbaren Produkten und „harmonisiert“

Geschäftsprozesse, um nachhaltige Kosten- und Wettbewerbsvorteile zu erzielen.

- **Dinea** (Dinea Gastronomie GmbH)
  Die DINEA Gastronomie GmbH gehört zu den führenden Akteuren des deutschen Gastronomie-Marktes. Sie verwaltet bis jetzt in 250 Filialen Selbstbedienungsrestaurants in den Sektoren Shopping, Freizeit, Arbeit und Verkehr. Dineas-Restaurants befinden sich hauptsächlich in Warenhäusern und präsentieren sich als Marktplatz für gastronomisches Essen.

Zuletzt möchte ich die **MGI** (Metro Group Information Technology GmbH) erwähnen, bei der ich als erster Absolvent meine Diplomarbeit entwickeln konnte. Sie ist eine interne IT-Servicegesellschaft der Metro AG. Mit fast 1000 Mitarbeitern im In- und Ausland ist sie für die gesamte IT-Infrastruktur aller Vertriebslinien zuständig. Hierzu zählt die Entwicklung von Anwendungssystemen im nationalen und internationalen Umfeld, der Betrieb von Rechenzentren an mehreren Standorten sowie IT-Consulting zur Implementierung von IT-Systemen.

5.600 Benutzer in 900 Märkten und in der Zentrale von Metro Cash & Carry arbeiten mit dem von der MGI entwickelten Warenwirtschaftssystem. Am Standort Düsseldorf verwaltet sie dieses System mit insgesamt 75 Terrabyte Datenvolumen.

1.2 Ziel der Arbeit

Die Metro Group Information Technology GmbH in Düsseldorf stellt sicher, dass täglich mehrere Millionen Transaktionen der einzelnen Vertriebsmärkte in ihr Warenwirtschaftssystem eingespielt werden. Alle verkauften Artikelpositionen eines gesamten Tages werden mit sog. Kopierdämonen in große Oracle Datenbanken transferiert. Aus Performancegründen und Sicherheitsaspekten ist es aber immer wieder zwingend notwendig, „up to date“ zu sein. Zusätzlich kommt hinzu, dass dieses Jahr die Metro Group in Russland einen neuen Vertriebszweig gründen wird, der eine Herausforderung neben allen bis jetzt erschlossenen Ländern darstellt. Der russische Zeichensatz erzwingt ein Umstellen der vorhandenen Oracle Datenbanken von Version 8 auf die derzeit aktuellste Version 9i, die den UTF8 (Unicode) Standard erfüllt. Unicode gibt jedem Zeichen seine eigene Nummer — plattformunabhängig, programmunabhängig, sprachunabhängig. Der Einsatz von Unicode in Client-Server- oder vielschichtigen Anwendungen ermöglicht bedeutende Kosteneinsparungen im Vergleich mit herkömmlichen Zeichensätzen. Anwendungsprogramme und Internetseiten können für viele Systeme, Sprachen und Länder direkt eingesetzt werden, ohne sie jeweils speziell und aufwendig anpassen zu müssen. Text kann mit Unicode weltweit ohne Informationsverluste ausgetauscht werden.

Durch Umstellung dieser Oracle Datenbanken mussten auch die restlichen Oracle Produkte auf einen aktuelleren Versionsstand gebracht werden. Infolge dieser Umstellung sind bereits existierende interne Dienste nicht mehr aufwärtskompatibel zu den neuen Oracle-Software-Modulen. Hiervon ist auch die Schnittstelle für die Fax-Kommunikation betroffen, die in Oracle-Forms programmiert wurde. Ab Oracle Forms 9i wird kein Client/Server Schema[3] mehr unterstützt. Oracle Forms funktioniert ab der Version 9i zukünftig nur noch für Anwendungen im Web-Bereich. Aus diesem Grunde lassen sich die alten Fax-Dämonen-Prozesse nicht mehr vollständig auf eigenständige Rechner auslagern; die gesamte Businesslogik agiert nur noch auf dem Web-Server. Gerade diese muss aber bei systemkritischen Anwendungen kontinuierlich überwacht werden. Diese Aufgabe übernimmt in der Metro nicht etwa ein fleißiger Praktikant,

[3] s. Kapitel 2.1 “Oracle Forms”

sondern sogenannte automatisierte Job-Scheduling-Systeme[4] wie Majaestro. Es war nun zwingend notwendig, das alte Software-Produkt durch eine neue Lösung zu ersetzen.

Diese Diplomarbeit befasst sich mit der Ablösung der bewährten Fax-Dämonen BD313 und BD314 durch eine neue Implementierung in der plattformunabhängigen Programmiersprache Java. Die Auswahl dieser Sprache erfolgte nicht zufällig, sie macht es später möglich, die Fax-Dämonen bei Bedarf ohne großen Zeitaufwand auf unterschiedlichste Rechnerarchitekturen zu übertragen.

Diese Neuentwicklung war erst das zweitgrößere Projekt, das mit Java in der Metro MGI Abteilung realisiert worden ist. Durch die neu hinzugekommene Objektorientierung lassen sich wiederverwendbare Softwarekomponenten erstellen, die auch in anderen Softwareprojekten zum Einsatz kommen können. Das Fax-Datenbankmodell bewährte sich bereits in der Praxis, und sollte im Ursprung ohne größere Änderungen erhalten bleiben. Bereits implementierte SQL-Prozeduren[5], die in der Datenbank hinterlegt sind, sollten wiederverwendet werden.

Eine der wichtigsten Anforderungen, im Gegensatz zur Vorgängerversion, war die Realisierung der Mandantenfähigkeit[6] zum Betrieb von jeweils einer Fax-Dämonen-Version zu einem Metro-Markt. Praktisch ist es möglich, für alle Märkte insgesamt mehrere hundert Fax-Dämonen gleichzeitig laufen zu lassen. Insbesondere musste in diesen Zusammenhang darauf geachtet werden, dass eine reibungslose Kommunikation mit nur einer einzigen Server-Fax-Schnittstelle gewährleistet[7] wird. Die Zuverlässigkeit stand an oberster Stelle, evtl. auftretende Fehler sollten zum sofortigen Abbruch eines Dämonen führen und lückenlos in der entsprechenden Landessprache von einer zentralen Stelle (Datenbank) nachvollzogen werden.

Um auch mit moderneren Kommunikationswegen mitzuhalten, ist es nach Fertigstellung dieser Diplomarbeit optional möglich, außer Fax auch Emails zu versenden. Obwohl das

[4] s. Kapitel 4.7.3 “Job-Scheduling-Systeme”
[5] eine SQL-Prozedur bezeichnet ein allg. Verfahren um Datenbankinhalte zu lesen, abzuändern oder zu löschen.
[6] ability to clientele processing (Mantantenfähigkeit) = viele Datenbanken können auf einem Rechner betrieben werden, und innerhalb einer Datenbank zu einer Portalarchitektur organisiert werden
[7] als Server wird in der Metro der Kommunikationssoftware Topcall eingesetzt, TC/LINK-FI ist eine mögliche Schnittstelle zu diesem Fax-Server, die durch Dateien angesprochen werden kann.

Faxaufkommen in der Metro sich im Durchschnitt auf ca. 50 Faxe am Tage beschränkt, lassen sich durch Emails auf Dauer zusätzlich mehr Kosten einsparen als analog zu Faxen. Ein Fax benötigt immer eine Telefonleitung, die selbstverständlich nur gegen eine Gebühr der Telekom zur Verfügung gestellt wird und ausschließlich für den Faxgebrauch genutzt wird. Emails werden generell über ein firmeneigenes leistungsstarkes Backbone Netzwerk oder dem Internet versendet, welche in der Regel 24h im Betrieb sind, und zusätzlich auch für andere Zwecke genutzt werden können.

1.3 Basis- und Commondienste

In der Programmierwelt ist es heutzutage durch objektorientierte Programmierung möglich, einzelne Module[8], die universell auch in anderen Applikationen sinnvoll sind, wiederzuverwenden. In den Programmiersprachen Java und C++ lässt sich dieses Konzept sehr leicht anwenden, weil jedes Objekt durch eine Datei assoziiert wird, die sich später in neuen Projekten wieder in eine Paketstruktur einbinden lässt. Erweitert man dieses Konzept auch auf PL/SQL-Funktionen und –Prozeduren in einer Datenbank, lassen sich diese in anderen Anwendungen beliebig wiederverwenden. In der MGI bezeichnet man dieses Prinzip als Services. Unter diesem Begriff sind alle Funktionalitäten zusammengefasst, die vom Entwickler projektübergreifend genutzt werden können. Ursprünglich stellten die Common-Services die Basis für jede MGI-Applikation dar; die Basisdienste sind eine funktionale Erweiterung die applikationsübergreifend auf diese Dienste aufsetzt. Die MGI verwendet sowohl neben der normalen Kapselung für die Common-, als auch für die Basisdienste, zusätzlich noch ein Hierarchiekonzept für diese Softwarekomponenten. Abbildung 2 verdeutlicht, das alle Komponenten der „Common-Services“ auf einer Ausgangskomponente „Common Basics“ aufbauen. Diese unterstützt globalste Funktionalitäten wie z.B. Stringformatierungen (C97), Dateizugriffe (C99) oder Enviromentzugriffe (C98)...

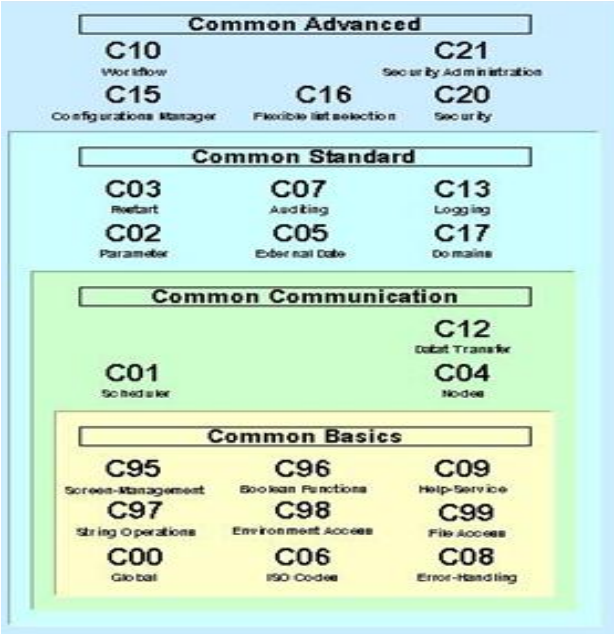

Abbildung 2 Hierarchiekonzept des Common-Service

[8] sind ein Bündel von Funktionen; das Aufgabengebiet kann zu einem Modul zusammengefasst werden.

Diese Funktionen dürfen vom Programmierer in keiner Weise doppelt und direkt in einem Softwaremodul hinterlegt werden, sondern müssen immer über die Hierarchieebene mittels eines „Stored Procedure Calls“[9] angewandt werden. Dieses Konzept der MGI ist sehr sinnvoll, falls Dienste in eine Datenbank neu installiert werden müssen. Updates der einzelnen Dienste können separat als Paket eingespielt werden und erzwingen nicht eine komplette Installation aller Dienste. Eine strenge Berechtigungshierarchie der Module erhöht zusätzlich die Sicherheit, und vermeidet unkontrollierte direkte Funktionsaufrufe wie das „Löschen einer Tabelle“.

Die heute zur Verfügung stehenden Basis- und Common-Dienste sind historisch bedingt aus zwei unterschiedlichen Projekten zusammengeführt worden. Im Rahmen der Internationalisierung wurde der Begriff „Shared-Services“ für die Basisdienste vergeben. Die Basisdienste umfassen die Bereitstellung von Basiskomponenten für applikationsübergreifenden Einsatz. Sie sind ein integriertes System zur Vereinfachung der Entwicklung von Anwendungssystemen und ermöglichen sowohl eine Kapselung von technischen Schnittstellen (Einbindung von Fremdprodukten) als auch Betriebssystemspezifika für Anwendungssysteme.

Die Kernfunktionalitäten lassen sich in folgender Tabelle zusammenfassen:

Kürzel	Funktion	Beschreibung
BD100	Date Routines	Ermittlung von Feiertagen, Anzahl von Arbeitstagen
BD200	GUI-Template	Vordefinierte Masken für GUI-Oberflächen
BD300	Adresses and Communication Interface	Modul zur Adresseneingabe, Versand von Listen aus Oracle-Anwendungen, Faxdienst
BD350	Wakeup Call	Anwenderinformation über wichtige Datenänderungen (z.B. neue Preise)
BD900	Messages	Anzeige von selbstdefinierten DB- oder Systemspezifischen Meldungen

Hier lässt sich auch der Basisdienst BD300 (Adress- and Communication Interface) wiederfinden, der z.T. durch neue Funktionalitäten der Java-Fax-Dämonen überarbeitet und ergänzt werden musste.

[9] bez. den Aufruf einer Methode o. Funktion, dessen Logik physikalisch in einer Datenbank hinterlegt ist.

Die Schnittstellen (ursprünglich „Common-Services“) sind ein Softwaresystem zum Datentransfer zwischen verschiedenen Datenbanken und Servern, und müssen in jedem Falle immer mit in eine Applikation eingebunden werden, weil sie absolute Grundfunktionen zur Verfügung stehen. Sie enthalten die Bereitstellung eines Layers zwischen technischem Datentransfer zu einer Anwendung, und bieten eine definierte Programmierschnittstelle für Anwendungslogik zur Daten-Aufbereitung. Spezielle Schnittstellenmodule wie z.B. Online-Hilfe für Forms-Anwendungen erweitern sich ständig. Aus diesem Grunde sind in nachfolgender Tabelle nur diejenigen auflistet, die auch relevant für die Java-Fax-Dämonen sind.

Kürzel	Funktion	Beschreibung
C02	Parameter	Verwaltung von Anwendungsparametern
C08	Error Handling	Standardisierung von Meldungen und mehrsprachiger Ausgabe
C16	Flexible List Selection	Aufruf und Verwaltung von Reports mit Parametern
C20	Security	Verwaltung Benutzerinformationen (Passwörter/Emailadressen)
C97	String Operationen	Verarbeitung von Zeichenketten
C98	Enviroment Access	Liest und schreibt Initialisierungsdaten
C99	File Access	Anlegen, Löschen, Lesen und Schreiben von Dateien

Die Herkunft der einzelnen Services ist auch an den Bezeichnungen der Softwaremodule zu erkennen. Common Services beginnen mit „C“ (z.B. C02) Basisdienste mit „BD“ (BD300). Entwickelte Applikationen der MGI sind meistens eine Mixtur aus Basisdiensten und Commondiensten.

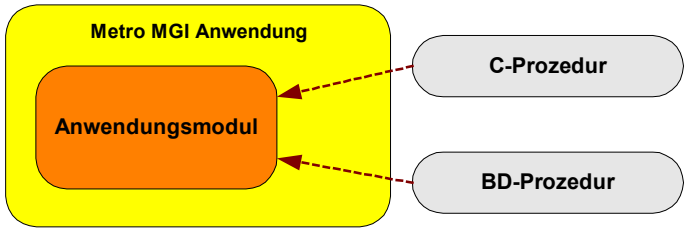

Abbildung 3 Aufbau einer MGI Anwendung

Es ist schwer, sich von dieser historisch gewachsenen Kapselung zu lösen. Die Services sind inzwischen so komplex geworden, dass eine komplette Migration aus wirtschaftlichen Gesichtspunkten unrealistisch erscheint. In der Entwicklungsphase der Java-Fax-Dämonen musste dieses Konzept durch eine entsprechende Paketstruktur in Java weitergeführt werden. Bereits existierende Bibliotheken, die im Internet zu beziehen sind, weisen aber von sich aus schon eine vorgegebene Struktur auf, die ohne tiefere Eingriffe in den Quellcode nicht problemlos in die Paketstruktur der Basis- bzw. Common Dienste integriert werden kann. Insbesondere hier mussten Kompromisslösungen ausgearbeitet werden.

Sehr viel Zeit hat die Entwicklung einer geeigneten Zugriffsschicht in Java zur weiteren Nutzung dieser Module in Anspruch genommen. Fertige Funktionalitäten, die bereits in Datenbankmodulen realisiert worden waren, sollten nicht noch zusätzlich in Standalone Javaklassen verwirklicht werden. Hierbei war es also wichtig zu wissen, welche Funktionen bereits in welchen Service-Dienst programmiert worden waren. Klarheit verschafften hier wöchentliche Meetings und Dokumentationen zu den Service-Diensten.

1.4 Fax & Co

Immer mehr Firmen setzen Faxserver ein, die den täglichen Faxverkehr beschleunigen, optimieren und dabei noch Telefongebühren sparen. Die Faxserver für PC-gestützten Faxversand und -empfang werden in Zukunft vermehrt durch diese intelligenteren Systeme erweitert oder abgelöst. Ein großer Vorteil von diesen Systemen: Man hat stets ein „freies Faxgerät“ zur Verfügung. Seit dem großen Internetboom von 1995 werden viele Geschäftsprozesse zum Teil nur noch durch Email über das Internet abgewickelt. Eine Email lässt sich mit Dateianhängen in Firmennetzwerken mit großer Bandbreite an das Internet blitzschnell und zuverlässig an die jeweilige Zielperson übermitteln. In Privathaushalten findet man immer öfter einen DSL-Anschluss, ein Faxgerät ist nicht mehr „state of the art“. Viele Faxgeräte sind somit für den normalen Endanwender unrentabel und vom Schreibtisch verdrängt worden. In der Metro AG werden Faxe aber noch durchaus für Bestellvorgänge von der MGB (Metro Group Buying GmbH) für den Einkauf verwendet.

1.4.1 Wo werden Faxe in der Metro eingesetzt?

Die MGI kümmert sich in erster Linie immer nur um die Basis den Funktionsumfang einer Applikation. Mit welchen Ausprägungen von Daten diese später „gefüttert“ werden, ist eigentlich uninteressant, lediglich die Datentypen müssen in der Entwicklungsphase bekannt sein. Der Inhalt aller Faxe besteht ausschließlich aus dem Warenwirtschaftsystem der MGI, der mittels eines Reportservers[10] in Form einer Datei generiert wird. Aufträge an diesen werden später mit der erwähnten Fax-Applikation (Faxe/Listen-Form[11]) bereitgestellt und anschließend über den Fax-Server Topcall[12] als Fax-Dokument versendet. Reportaufträge lassen sich durch Parametervariationen in den verschiedenartigsten Varianten generieren. Dadurch entstehen naturgemäß unterschiedlichste Faxausprägungen; zu viele, um sie hier alle als Beispiel aufzulisten.

Trotzdem soll an dieser Stelle kurz das Einsatzgebiet für Faxe in der Metro beleuchtet werden.

[10] vgl. Kapitel 2.2 „Oracle Reports“
[11] s. Kapitel 1.5 “Die Faxe/Listen-Form der MGI“
[12] s. Kapitel 3.1 „Die zentrale Kommunikationssoftware TOPCALL“

Typischerweise wird für jeden Lieferanten, der einen Metro Markt in der Welt beliefert, die bestellte Soll-Menge mit der tatsächlichen ausgelieferten Ist-Menge in den Metro Lagern mittels Bar-Code-Scannern verglichen. Unabhängig davon, ob nun Differenzen auftauchen, erhält der Zulieferer immer einen Auszug über die Waren, die auch wirklich im Zentrallager der Metro AG angekommen und registriert worden sind. Ein solcher Auszug stammt aus dem Warenwirtschaftsystem der Metro und wird durch Oracle Reports erzeugt und anschließend gefaxt. Den Hauptnutzen der Faxe/Listen-Form hat sicherlich die MBG. Für sie ist es wichtig, auch bei Lebensmittelwaren wirtschaftlich zu handeln. Sollten eingelagerte Produkte zu lange in den Verkaufregalen lagern, die kurz vor dem Verfallsdatum stehen, so wird auf diese Waren zusätzlich ein Preisnachlass gewährt. Die Verkäufer in den Metroverkaufshallen erhalten dann einen Auszug per Fax und deklarieren die Produkte mit niedrigeren Preisen. Dieses Procedere gilt natürlich auch bei speziellen Angeboten oder Sonderaktionen. Der häufigste Nutzen der Faxe/Listen-Form bezieht sich jedoch auf Bestellvorgänge oder Vorab Preisanfragen an einen Zulieferer. Abbildung 4 zeigt das Fax „Stück- und Einkaufspreisliste“ aus dem Warenwirtschaftssystem, welches an einen speziellen Lieferanten gesendet werden könnte.

STGRP und EK-Preisliste (mBL VL)

EKSG: 153 Gültig ab: 06.01.2004
Lieferantennummer: [redacted]
Lieferantenname: [redacted] Währungseinheit: EUR

Eingrenzung auf: Herstellerdaten DispKz < 2
Gruppierung: Sort./Grupp. X Variantenauflös. TSL

Art Nr	S/G Nr	Artikelbezeichnung	Var Nr	EK Geb	STGRP
1751		MD BUKO PAPRIKAROLLE 125G	1	8	[redacted]00
3405		MOZZARELLA GALBANI 1KG STANGE	1	8	[redacted]29
3503		BAUERNK.BURGUNDER WK 50% 2,6KG	1	1	[redacted]58
6897		BEAUZAC LANDBUTTERKAESE 50% 1,9KG	1	1	[redacted]40
7317		GORGONZOLA MIT MASCARPONE	1	1	[redacted]20
8827		MD MOMMARK SNK 45 % CA. 4 KG	1	1	[redacted]24
9888		BEL PAESE LAIBESNK 48 % CA. 2,5 KG	1	1	[redacted]67
9889		GORGONZOLA ERBORINATO 55 % CA.1,5KG	1	1	[redacted]55
15452		GORGONZOLA 50% 500G STK.	1	12	[redacted]43
16586		FRISCHKAESE CR.ERDB.500G 40% NICOL	1	6	[redacted]17
17723		CREMA BEL PAESE SNK 50% 24 X 25G	1	12	[redacted]32
26604		FRISCHKAESE CR.500G 20% NICOL BESNI	1	6	[redacted]49
26807		LE BRIN 50 % 150 G BARUF	1	6	[redacted]82
28307		MASCARPONE 500G DOPPELRAHM	1	8	[redacted]47
29926		CARRE REVEREND WK NATUR 52% 250G	1	6	[redacted]99
42480		DOLCELATTE SNK 63 % 1,5 KG	1	1	[redacted]55
89708		CARRE ROUSSOT RSK 50% 250G	1	6	[redacted]57
108333		RICOTTA NOSTRA FETA 50% 1,5KG	1	4	[redacted]17
120793		KR-KAESEZUB. 60% 1KG BARUFE	1	5	[redacted]56
120794		WEICHE SPUNDE 62,5G GESKA	1	40	[redacted]11
128093		MASCARPONE GALBANI 250G DOPPELR.	1	8	[redacted]25
[redacted]		[redacted]	1	12	[redacted]38

Abbildung 4 Fax für einen Lieferanten

Aufgrund der vorliegenden alten Einkaufspreise zu vorherigen Bestellungen der MGB bei Empfänger, lassen sich nun evtl. neue Konditionen durch größere Abnahmemengen aushandeln.

1.4.2 Warum gerade ein Fax-Gateway-Dämon?

Viele „Home-Anwender“ können heutzutage mittels der Druckerschnittstelle in Microsoft Windows ihre Ausgaben über ein handelsübliches Modem[13] direkt auf ein Faxgerät senden bzw. drucken. Nachteil: „Jeder Rechner muss über ein Modem verfügen“, in großen Firmennetzwerken wie der Metro ist dies ein immenser Kostenfaktor, der nicht zu vertreten wäre. Nicht nur die einmalige Anschaffung der Hardware, sondern die Administration und Wartung dieser ist der eigentliche Kostenfaktor. Aus Sicherheitsgründen sollte auch niemals ein direkter Zugang über ein Modem zu einem Rechner möglich sein, sondern stets durch eine Firewall[14] geschützt sein.

Viel besser ist in diesem Falle nur ein Modem; ein Fax-Server[15], der Aufträge aller Benutzer über ein Netzwerk in einer Warteschlange kontinuierlich abarbeitet und versendet. In der Metro findet das TOPCALL-Fax-System Verwendung. Es stellt analog zu Windows sog. Schnittstellen zur Verfügung, mit dem das System angesteuert werden kann. Einige dieser Schnittstellen sind bereits vom Hersteller TOPCALL[16] für Businessanwendungen wie Lotus Notes, Exchange u.v.w. vorkonfiguriert und können ohne großen Aufwand direkt angebunden werden. Die Metro MGI nutzt jedoch Aufgrund einer hauseigenen Fax-Applikation eine Universalschnittstelle[17] zum Fax-Server, die durch sog. Dämonen bei Bedarf automatisch versorgt wird.

Dämonen sind generell besondere Programme, die niemals vom Anwender aufgerufen werden. Sie sind sehr häufig auf Serversystemen zu finden und werden meist schon

[13] Ein Modem dient dazu, digitale Daten in analoge Signale umzuwandeln und umgekehrt. Diese können über ein Telefonnetz zu anderen Systemen übertragen werden, z.B. einem Faxgerät.
[14] Firewalls sind Sicherheitssysteme innerhalb eines Netzwerkes die nur eine bestimmte Art von Protokollen und Anfragen zwischen Rechner zulassen.
[15] Die Metro verwendet Topcall als Fax-Server vgl. Kapitel 3.1 „Die zentrale Kommunikationssoftware TOPCALL“
[16] s. Homepage des Unternehmens <http://www.topcall.de> (12.04.2004)
[17] vgl. Kapitel 3.1.2 „Die TC/LINK-FI Schnittstelle“

beim Booten[18] mitgestartet. Als Dämon[19] bezeichnet man ein Programm, das auf einem Server eine bestimmte Aufgabe erfüllt, ohne dass dabei irgendeine Eingabe erforderlich wäre. Solche Programme werden deshalb auch als nicht-interaktive Programme bezeichnet. Sie laufen sozusagen im Hintergrund. Zu unbestimmten Zeitpunkten "erwachen" sie dann, und schauen nach, ob Arbeit für sie vorliegt und führen diese aus.

Die Eigenentwicklung der Dämonen für den Basisdienst BD300 bezeichnet die MGI als Fax-Dämonen. Insgesamt gibt es pro Installationseinheit (Store[20]) zwei Stück, die beide auf einem Server laufen müssen:

- **BD313 (Bearbeitung von Kommunikationsaufträgen)**, er spürt neue generierte Fax-Aufträge von Benutzern auf, und versorgt automatisch die TC/LINK-FI Schnittstelle des TOPCALL-Servers mit entsprechenden Transaktionsdateien
- **BD314 (Bearbeitung von Rückmeldungen)**, kümmert sich um dem Rückmeldestatus eines Faxes und speichert dieses zu einem Kommunikationsauftrag.

Besonders wichtig bei der Neuentwicklung der Fax-Dämonen in der Programmiersprache Java, war nicht in erster Linie die Performance, sondern die Aspekte der Zuverlässigkeit und Sicherheit. Insbesondere in Java kommt man schnell in die Versuchung, als Programmierer zu vergessen, wie viele Objekte bei einem kontinuierlich laufenden Prozess jedes Mal neu erstellt werden können. Ein Programmablauf kann dadurch schnell in einem „out of memory“ enden, selbst wenn der Server eigentlich reichlich Speicher zur Verfügung stellt. Die genaue Fehlersuche ist in diesem Falle besonders schwierig.

Als letzter Punkt bleibt noch zu klären, warum die ganze Sache eigentlich als Gateway bezeichnet wird.

[18] bezeichnet das Laden von Konfigurationseinstellungen beim Start eines Betriebssystems
[19] wört. übersetzt “Disk And Execution MONitor”
[20] bez. einen Markt in der Metro, z.B. Saturn, Praktiker, MediaMarkt

1.4.3 Der Sinn von Gateways

Gateways werden oft eingesetzt, um eine Schnittstelle zwischen zwei Kommunikationssystemen zu schaffen. Vielfach sind Sie in großen Netzwerkstrukturen von Firmen anzutreffen. Gateways erlauben es Netzwerken, die auf völlig unterschiedlichen Protokollen basieren, sich miteinander zu verbinden. Häufig werden sie in diesem Zusammenhang auch als „Router“ bezeichnet. Falls dieser in einem Netzwerk betrieben wird, kann z.B. auch ein Server als Gateway eingesetzt werden, sodass der gesamte Internet-Verkehr über ihn abgewickelt wird.

Analog zum Internet bezeichnen die Fax-Dämonen BD313 und BD314 nun eine Vorstufe zum eigentlichen Fax-Gateway, den TOPCALL-Fax-Server. Diese Zwischenstufe basiert zwar auf dem gleichen TCP/IP-Protokoll[21] wie es auch im vorgeschalteten Firmennetzwerk vorzufinden ist, der komplette Informationsfluss zu TOPCALL wird aber immer in Fax-Steuer-Informationen transformiert und für keinerlei andere Daten genutzt, daher spricht man an dieser Stelle von einem Gateway.

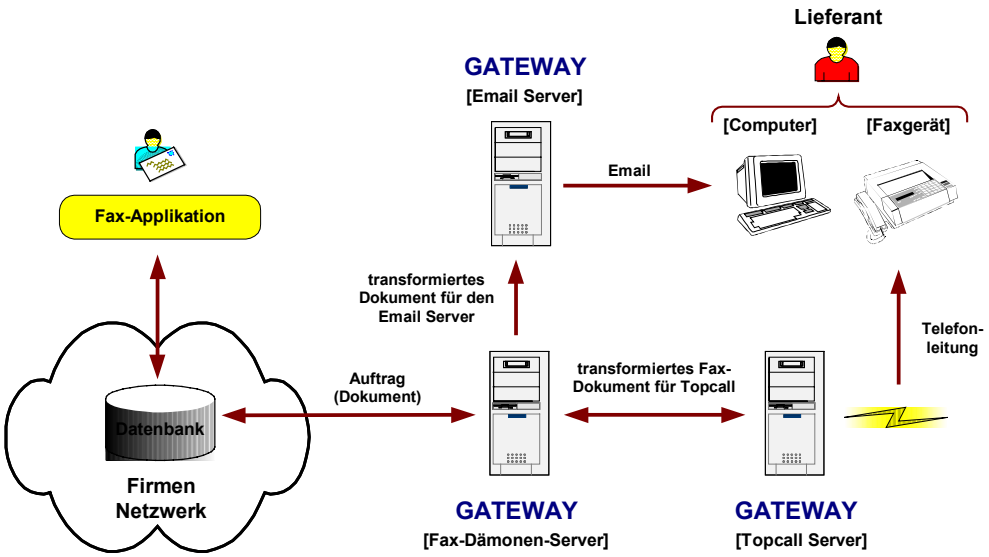

Abbildung 5 Kommunikationsweg eines Faxes/Email zum Empfänger

Abbildung 5 zeigt den kompletten Kommunikationsweg zwischen einem Faxbeauftragten und einem Endanwender. Über eine hauseigene MGI-Fax-

[21] eins der am häufigsten verwendeten Protokolle in Netzwerken, mit denen sich Information zwischen Rechnersystemen austauschen lassen. Näheres s. <http://de.wikipedia.org/wiki/TCP/IP> (13.12.2003)

Applikation[22] lassen sich Fax-Aufträge generieren und über ein Firmennetzwerk in eine Datenbank ablegen. Ein Fax-Dämonen-Server sucht nun ständig nach neuen Aufträgen in einer Fax-Datenbank. Sollte er fündig werden so transformiert er diese formgerecht an den eigentlichen Fax-Server TOPCALL, der diese schließlich über eine Telefonleitung an ein Fax-Gerät des Endanwenders (Lieferant) sendet. Der Informationsaustausch zwischen der Fax-Applikation und dem TOPCALL-Server erfolgt dank der Fax-Dämonen sogar bidirektional[23], d.h. es können gleichzeitig Faxe versendet und Rückmeldungen verarbeitet werden.

Insgesamt lässt sich erkennen, dass somit zwei Gateways für das Senden von Faxen verantwortlich:

- **Fax-Dämonen-Server [Gateway]**, Verbindungsglied zwischen der Fax-Applikation für den Benutzer und dem
- **TOPCALL-Fax-Server [Gateway]**, der die Verbindung über eine Telefonleitung zu einem Faxgerät herstellt.

Historisch bedingt sprechen die Mitarbeiter der Metro immer noch von einem Fax-Gateway-Dämon, obwohl dieser analog dank Java nun zusätzlich Aufträge auch als Email versenden kann. In diesem Falle fungiert der Fax-Dämonen-Server natürlich immer noch als Gateway, nur dass dieser anstatt Faxe Email-Informationen an einen Emailserver weiterleitet.

Für das weitere Verständnis der Fax-Dämonen ist es jedoch zuerst notwenig, den Aufbau und die Funktionsweise der hauseigenen Fax-Applikation „Die Faxe/Listen-From“ näher zu erläutern.

[22] vgl. Kapitel 1.7 „Die Faxe/Listen-Form der MGI“
[23] gleichzeitiger Kommunikationfluss in beide Richtungen

1.5 Die Faxe/Listen-Form der MGI

Die Faxe/Listen Form entstand im Jahre 1999 aufgrund einer Projektanforderung „Integration von Preispflegesystemen“ innerhalb der Metro. Damals musste eine Möglichkeit geschaffen werden, Listen und Formulare aus Oracle-Anwendungen heraus zu faxen. Man entschied sich, diese Funktionalität in einem Basisdienst zu realisieren, weil ähnliche Anforderungen auch aus anderen Teilprojekten gestellt wurden.

1.5.1 Ein Fax geht auf Reisen

Ein Aufruf der Applikation erfolgt mittels eines handelsüblichen Internet-Browsers über das Intranet der Metro AG. Nach erfolgreicher Autorisierung an eine Oracle-Datenbank erscheint eine Steuerungszentrale für die gesamte Artikelverwaltung der MGB.

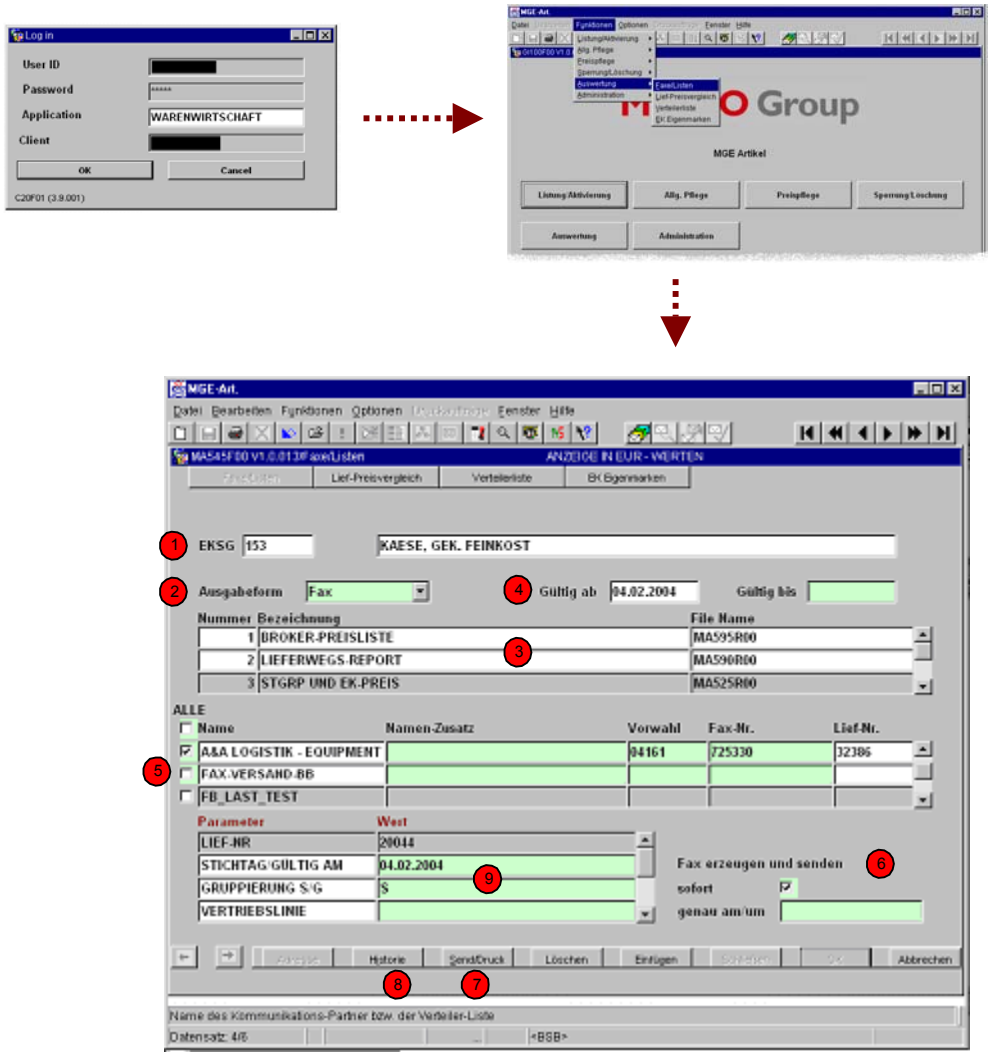

Abbildung 6 Faxe/Listen-Form

In dieser Oracle-Forms-Anwendung gelangt man nach Auswahl der Faxanwendung in die Faxe/Listen Form. Von hier aus lassen sich nun beliebige Reports an bestimmte Empfänger (meist Lieferanten) erstellen und mittels der Fax-Dämonen versenden. Bevor mit einer näheren Auswahl begonnen werden kann, ist ein entsprechendes Einkaufssachgebiet (EKSG) für die Reportsgenerierung auszuwählen. [s. Punkt 1 in Abb. 6] – Die Einkaufsachgebietnummer 153 entspricht in diesem Beispiel „Käse und gekochte Feinkost“. Für eine Versendung von Berichten muss nun die Ausgabeform festgelegt werden [2]. Derzeit ist es nur möglich, zwischen einem Fax oder verschiedenen Dateiformaten (z.B. MS-Excel) auszuwählen. Zukünftig wird es durch die Neuentwicklung der Fax-Dämonen von Oracle-Forms in Java auch möglich sein, Emails zu versenden. Als Nächstes lässt sich durch eine Selektion vordefinierter Parameterwerte [3] für einen Report die Art eines Berichts festlegen. Optional ist es auch, möglich neue Report-Arten zu definieren. Zwischen folgenden kann der Anwender derzeit unterscheiden: Broker Preisliste, Lieferwegsreport, Stückgrundpreis und Einkaufspreis.

Zuletzt kann noch ein Gültigkeitsdatum für den Reportauftrag mitgegeben werden [4]. Dies ist ein zwingend notwendiges Feature für sehr große Reports, die u.U. einige Stunden für die Generierung auf dem Reportserver bräuchten, obwohl kleinere, die von anderen Benutzern benötigt werden, evtl. sogar eine höhere Priorität haben. Diese warten in einer Warteschleife, die nach dem FIFO-Prinzip[24] arbeitet. Der Startzeitpunkt wird festgelegt durch das Attribut „Gültigkeitsdatum ab“. Ressourcenintensive Reports sollten somit auf das Wochenende verlegt werden. Optional lässt sich noch ein „Gültig bis“-Datum vergeben. Dies kann nützlich sein, wenn ein Report infolge von Wartungsarbeiten an Datenbanken nicht mehr generierbar wäre. Alle Benutzerdefinierten Parameterwerte, die an einen Report Server geschickt werden, lassen sich bei auftretenden Problemen in der Faxe/Listen-From unter den Spalten Parameter/Wert [9] nochmals überprüfen. Aus einem persönlichen Adressbuch [5] setzt man nun einen oder mehrere Empfänger für diesen Fax-Vorgang mittels einer Checkbox fest (optional lassen sich auch Verteilerlisten[25] erstellen). „Senden/Druck“ [7] veranlasst die Faxe/Listen-Form, eine neue Auftragsnummer zu erstellen und Laufnummern für jeden Report in der Fax-Datenbank zu hinterlegen. Anschließend werden alle Faxe über

[24] First In First Out
[25] Verteilerlisten können mehrere Empfänger beinhalten

eine Kommunikationssoftware Namens Topcall[26] versendet. Die Schnittstelle zwischen der Back Office Applikation (Faxe/Listen-Form) und dem Fax-Server wird durch zwei von der Metro entwickelte Fax-Dämonen BD313 (Bearbeitung Kommunikationsvorgänge) und BD314 (Bearbeitung von Rückmeldungen) hergestellt. Deren Funktionsweise ist im nächsten Kapitel „Faxgenerierung“ näher beschrieben.

Zu jedem Zeitpunkt hat der Anwender zusätzlich die Möglichkeit, sich über eine Fax-Historie-Form [8] über den Status seiner Aufträge zu informieren (vgl. Abbildung 7).

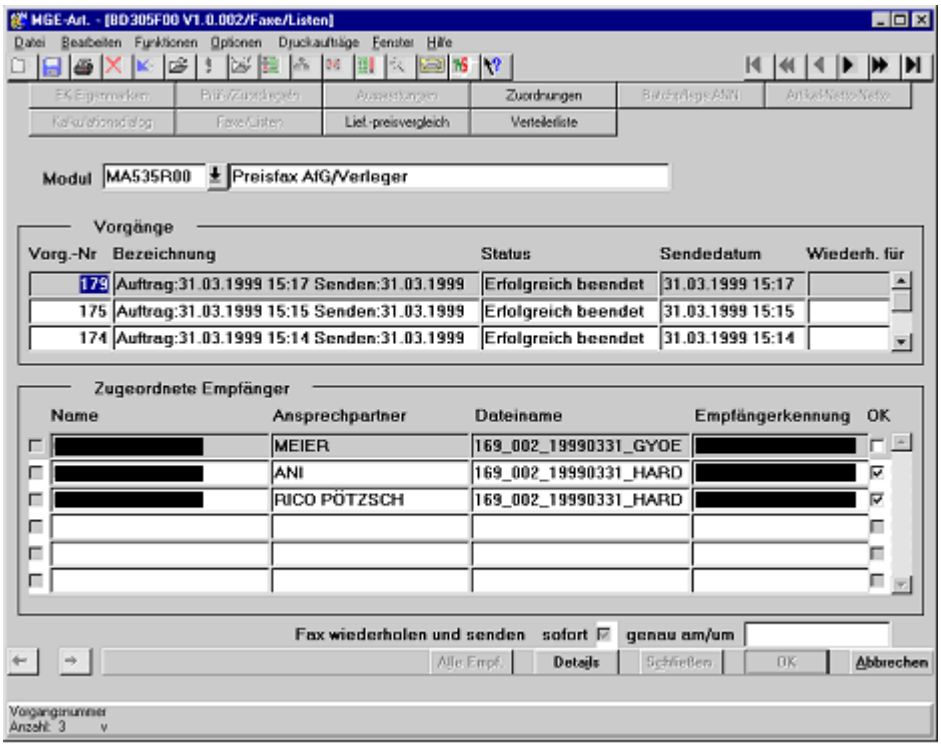

Abbildung 7 Faxe/Listen-Form Historie

Nach Auswahl eines Reports, z.B. MA535R00, werden alle zu diesem Report erstellten Kommunikationsvorgänge, absteigend nach Vorgangsnummern, sortiert dargestellt. Angezeigt werden die Vorgangsnummer, die Bezeichnung des Vorganges, der Bearbeitungsstatus des Gesamtvorganges und das Soll-Absendedatum für das Fax. Im unteren Block werden alle dem Kommunikationsvorgang zugeordneten Empfänger nach Name sortiert aufgeführt. Neben den Namen des Adressaten werden der Name des Ansprechpartners für evtl. Rückfragen, sowie die Empfängerkennung (Faxnummer)

[26] s. Kapitel 4.7.4 „DOM/File.Net Archivierung“

angezeigt. Optional können Faxe auch in einem DOM-System[27] archiviert werden, um auch zu einen späteren Zeitpunkt noch Zugriff auf alle versandten Faxe zu erhalten. Sofern eine Archivierung des für den Empfänger erstellten Reports in DOM vorgesehen ist, wird der Name der Datei dargestellt. Eine Checkbox „OK“ kennzeichnet, ob das Fax erfolgreich verschickt worden ist. Sollte dies einmal nicht geschehen, so können empfängerspezifische Informationen zu den einzelnen Empfängern mit „Details“ eingesehen werden (s. Abb. 8).

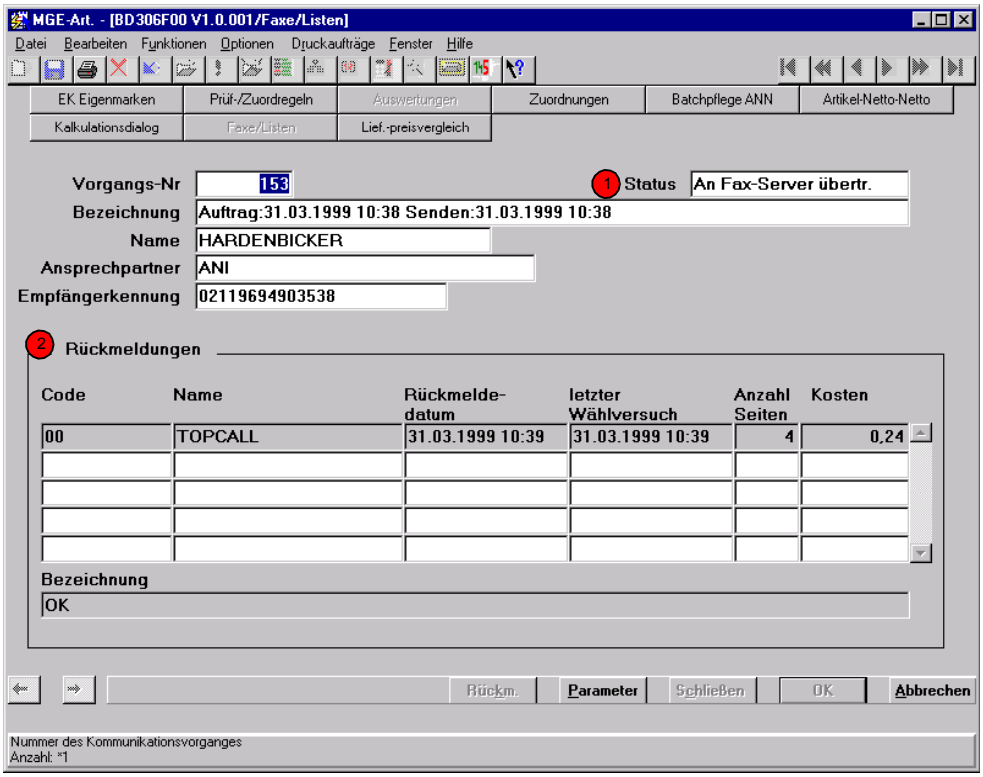

Abbildung 8 Faxe/Listen-Form Details

Ein Statusfeld [1] kennzeichnet dabei den aktuellen Status für den ausgewählten Empfänger:

- undefinierter Zustand
- Auftrag angelegt – durch die Faxe/Listen-Form
- Auftrag erkannt - wird vom Fax-Dämon BD313 beim Aufgreifen eines Datensatzes für ein Dokument gesetzt, wenn dieser einen Vorgang erkannt hat, der abgearbeitet werden soll. Dieser Vorgang wird dann exklusiv von diesem Dämon bearbeitet. (Im Fehlerfalle lässt sich somit schnell

[27] Document Output Management, s. Kapitel 4.6

nachvollziehen, welcher Auftrag zuletzt von dem Dämonen bearbeitet worden ist).

- Auftrag angelegt und Report gestartet - wird von der Faxe/Listen Form gesetzt, direkt bevor der erste Report gestartet wird. Wenn sich aus einem Vorgang der mehrfache Aufruf von Reports ergibt, wird der Status gesetzt, wenn der erste Report gestartet wird.
- Steuerdatei fertig - wird vom Dämon BD313 gesetzt, nachdem die letzte Steuerdatei für den Fax-Server zu einem Vorgang erstellt worden ist
- An Fax-Server übertragen - wird vom Dämon BD313 gesetzt, wenn die Steuerdatei für ein Dokument inkl. des generierten Reports an den Fax-Server übertragen wurde
- Erfolgreich beendet - wird vom Dämon BD314 gesetzt, wenn für alle Empfänger eines Auftrages positive Rückmeldungen eingetroffen sind.
- Report Fehler - wird vom Dämon BD314 gesetzt, wenn zu einem Dokument ein Report nicht erzeugt werden konnte.
- TOPCALL Fehler - wird vom Dämon BD314 gesetzt, wenn der TOPCALL-Server ein Fax nicht erfolgreich versenden konnte

Im unteren Block „Rückmeldungen“ der in Abb. 8 dargestellten Detailansicht wird die original Statusinformation vom Fax-Server angezeigt. Der Name entspricht dem aktuell verwendeten TOPCALL-Server, und das Feld „letzter Wählversuch“ gibt Aufschluss darüber, wann zuletzt versucht wurde, einen Empfänger zu erreichen (standardmäßig probiert der Fax-Server dies 5 mal). Zu allen Fax-Aufträgen protokolliert der Fax-Server die Anzahl der versendeten Seiten, die dazu anfallenden Kosten und einen eigenen TOPCALL-Code, der dem Anwender Auskunft über Erfolg bzw. Misserfolg der Sendung gibt. Im Fehlerfalle, z.B. „Telefonleitung besetzt“, liefert dieser detaillierte Informationen zur Ursache.

1.5.2 Faxgenerierung

Wie zuvor bereits erwähnt, wird eine Schnittstelle zwischen der Faxe/Listen-Form, dem Fax-Server und dem Reportserver durch zwei Fax-Dämonen BD313 und BD314 hergestellt. Folgendes Schema zeigt einen groben Ablaufplan zur Faxgenerierung. An dieser Stelle wird bewusst auf nähere Implementierungsdetails des Datenbankschemas

verzichtet. Nur die neu zu erstellenden Fax-Dämonen in Java werden im Kapitel 4 „Der MGI Fax-Basisdienst BD300“ näher erläutert - alleine das Tabellenmodell für die Reportserstellung und die dazugehörigen Basisdienstkomponenten im Einzelnen zu erklären, würden den Umfang der Arbeit sprengen.

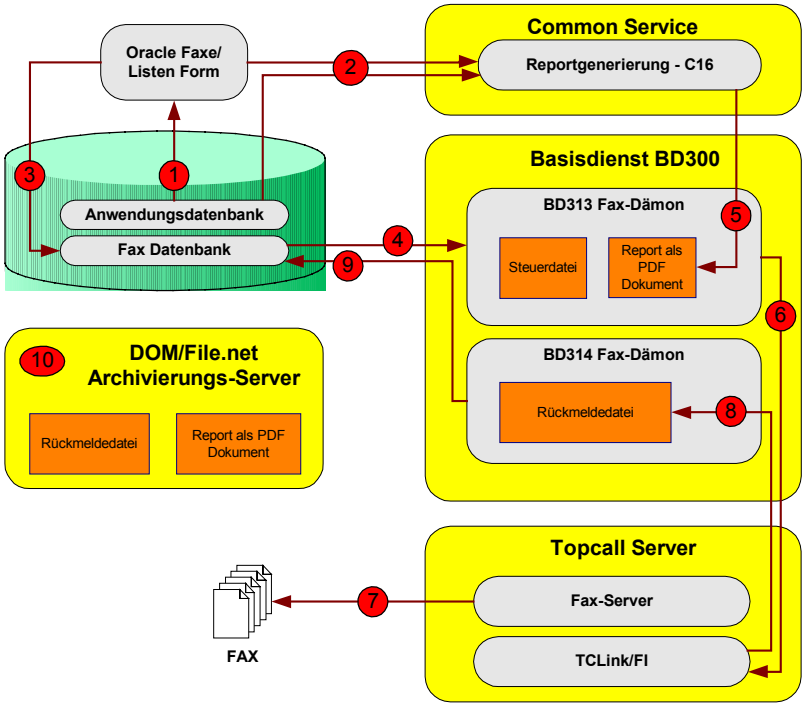

Abbildung 9 Ablaufschema eines Faxes

Die komplette Generierung des Inhaltes eines Faxes, läuft derzeit unter einem einzigen Windows 2000 System, lediglich die Datenbank wird zusätzlich auf einem AIX-Rechner bereitgestellt. An der Erstellung, Generierung und anschließender Versendung von Faxen sind mehrere Fax-Dienste beteiligt. Als erstes ist es erforderlich, über einen Browser die Faxe/Listen Form zu starten, welche sich (wie in Abb. 9 zu sehen) über die Anwendungsdatenbank [1] alle notwendigen Parameter für einen gewünschten Report holt. Sind alle Empfänger für ein Fax oder in einer Verteilerliste vom Anwender bekannt gegeben, so kann mittels des „Senden/Druck“ Knopfes der Common Service Dienst C16 [2] angestoßen[28] werden, der den Report- Auftrag an einen Oracle Report Server weiterleitet. Zuletzt wird der neu angelegte Auftrag für ein Fax in einer Fax-Datenbank hinterlegt [3]. Insgesamt zwei Fax-Dämonen (BD313/BD314), die in Oracle Forms im

[28] anstoßen bez. hier ausführen einer Funktion

Client/Server Prinzip[29] operieren, sind nun für die weitere Abarbeitung von Faxen zuständig.

Dämon BD313 ist vor BD314 immer als Erstes am Zug und überprüft ständig die Fax-Datenbank, ob Faxvorgänge mit dem Status „versenden“ gekennzeichnet sind [4]. Findet dieser nun einen Faxauftrag, der noch versendet werden muss, so wird eine Steuerdatei für den Fax-Server TOPCALL generiert und die dazugehörige Reportdatei vom Report Server abgeholt. Beide Dateien werden nun auf ein Verzeichnis der Link/FI[30] Schnittstelle verschoben [6]. Zuletzt wird ein Fax, bestehend aus dem Report und einer Steuerdatei, vom Fax-Server generiert und als Fax versendet [7]. Dämon BD313 ist mit seiner Aufgabe fertig und legt sich solange in den Standbymodus[31], bis er neue Fax-Aufträge in der Fax-Datenbank entdeckt.

Mögliche Statusausprägungen zu Fax-Vorgängen (z.B. Empfänger nicht erreichbar, Leitung belegt) werden nun pro Empfänger in eine Rückmeldedatei geschrieben, die von der Link/FI Schnittstelle generiert wird. Dämon BD314 entscheidet nun, ob ein Fax erfolgreich versendet wurde, indem er ständig nach neuen Rückmeldedateien auf dieser Schnittstelle sucht [8]. Eine Rückmeldedateien-Zuordnung zu einem Faxvorgang in der Fax-Datenbank kann nun mittels einer Laufnummer und einer Telefonkennung vorgenommen werden. Nach dem Auslesen dieser Datei wird der entsprechende Fax-Server-Status in der Fax-Datenbank hinterlegt [9]. BD314 legt sich ebenfalls in den Standbymodus bis er wieder neue Rückmeldedateien auffindet.

Um später eine Kontrolle über alle versandten Faxe zu erhalten, unabhängig von der Faxdatenbank, wird der Report inkl. der Rückmeldedatei endgültig auf ein Archivierungssystem wie Dom bzw. File.net gespeichert.

[29] s. Kapitel 2.1 „Oracle Forms“
[30] TC/Link-FI definiert eine Datei-Schnittstelle zum TOPCALL-Fax-Server, vgl. dazu Kapitel 3.1.2 „TC/Link-FI Schnittstelle
[31] bei Dämonen bez. man den Standbymodus als einen Ruhezustand; die Prozessorauslastung wird auf ein Minimum beschränkt.

2. Kapitel – Technologien

2.1 Oracle Forms

Oracle Forms ist ein deklaratives Maskenentwicklungstool unter Verwendung von eventgesteuerten Triggern[32], die in PL/SQL formuliert werden. Die Anwendungsentwicklung mit Oracle Forms erfolgt mit der Entwicklungsumgebung Oracle Forms Developer. Diese verwendet eine leistungsstarke deklarative Funktionalität, um Anwendungen aus Datenbankdefinitionen zu erstellen, welche eine enge Integration mit Oracle Datenbank nutzen. Mit Forms lassen sich im Vergleich zu anderen Programmiersprachen mit geringem Aufwand schnell anspruchsvolle Formulare und Geschäftslogiken entwickeln. Die Produktivität des Entwicklers wird hierbei durch eine umfassende GUI-Entwicklungsumgebung mit drag & drop und zahlreichen vordefinierten Wizzards unterstützt, so dass mit wenig oder ohne selbstgeschriebenem Code Anwendungen erstellt oder angepasst werden können. Neben Vererbungsmechanismen, die auch in anderen Programmiersprachen zu finden sind, bietet Forms eine Anwendungspartionierung, die es ermöglicht, einzelne PL/SQL-Programmblöcke auf einem Datenbankserver, dem Anwendungsserver oder in der Client-Anwendung abzulegen und von Fall zu Fall zu unterscheiden, welche Lösung am vorteilhaftesten ist. Forms Applikationen können von wenigen bis auf mehrere zehntausende Benutzern skaliert werden, ohne die Anwendung selbst zu ändern.

Es gab also viele Gründe, die im Jahre 1994 die Metro MGI dazu bewegten, Oracle Forms einzusetzen. Inzwischen ist es aber sehr schwer geworden, von dieser Lösung wegzukommen und nach Alternativen zu suchen. Gerüchten im Internet zufolge ist es ferner Zukunft bereits abzusehen, dass der Support seitens Oracle eingestellt wird und diese Technologie komplett durch Java abgelöst werden soll. Die Gesamtheit aller Forms-Anwendungen der Metro AG wird auf einen sehr hohen Entwicklerwert geschätzt. Die Migration auf andere Konzepte kann somit nur step by step erfolgen. Ein erster Ansatz ist hierbei die Realisierung der Fax-Dienste in Java. Oracle Forms Anwendungen können auf der Client/Server Ebene (nur bis Version 6i), als auch im

[32] Trigger reagieren auf sog. Events (Ereignisse). Trigger können auf unterschiedlichste Art und Wiese im Programm implementiert und ausgelöst werden, z.B. kann die Bewebung eines Mauszeigers auf einen bestimmten Fixpunkt auf dem Bildschirm ein neues Fenster öffnen.

Internet oder dem Intranet einer Firma aufgerufen werden. Abbildung 10 veranschaulicht das Client/Server Modell.

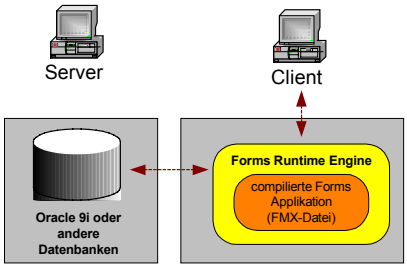

Abbildung 10 Client/Server Prinzip bei Oracle Forms

Der Anwender am Client startet eine kompilierte Form, die als FMX-Datei über die Oracle Forms Runtime-Engine ausgeführt wird. Der gesamte Prozess läuft auf seinem eigenen Rechner. Dieses Prinzip wird in der Informatik auch als Client/Server bezeichnet. Komplexe Forms sind sehr ressourcenhungrig, daher sollte auf dem Client eine leistungsfähige Hardware zur Verfügung stehen. Die Kosten für eine Anschaffung der Rechner erweist sich als ein großer Nachteil des Client/Server-Prinzips. Durch eine Software der MGI auf dem Client, werden Sourcen und Executables, die für den Start einer Oracle Forms Applikation notwendig sind, im Netzwerk und zusätzlich lokal auf dem Rechner verteilt. Auch Updates und Patches werden auf diese Weise installiert. Dieser Prozess erfolgt aber nicht automatisch, sondern muss z.T. immer noch von Hand vom Anwender durchgeführt werden. Das ist sicherlich nicht im Sinne eines Systemadministrators, der auf einer zentralen Umgebung die jeweils aktuellste Software für einen Anwender zur Verfügung stellen möchte, und nicht auf die Installationsbereitschaft eines Anwenders angewiesen sein will. Aus diesem Grunde gibt es für Forms zusätzlich eine Web-Server-Architektur, das Web Prinzip (vgl. Abb. 11).

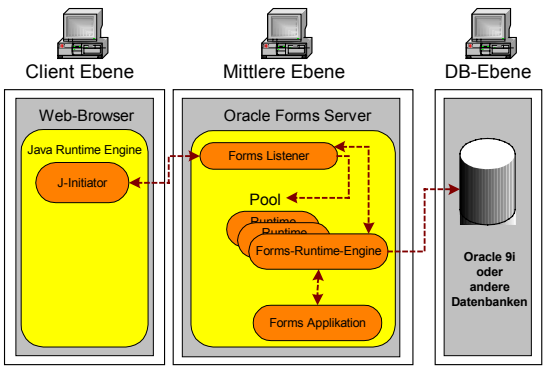

Abbildung 11 Web Prinzip bei Oracle Forms

Oracle Forms 6i wird in der Metro zukünftig durch die Version 9i abgelöst. Diese unterstützt nur noch das Web Prinzip. Überlicherweise verteilen sich Oracle Forms Anwendungen auf die folgenden drei Ebenen auf:

- Die **Client Ebene** enthält einen Web Browser, in dem die Anwendung angezeigt und verwendet wird
- Die **Mittlere Ebene** enthält den Anwendungs-Server, auf dem sich die Anwendungslogik und die Server-Software befinden
- **Datenbankebene** (optional) – Sie besteht aus einem Datenbankserver, auf dem Geschäfts- und Unternehmensdaten gespeichert und abgerufen werden können

Der Oracle Forms Server ist hierbei das zentrale Element für die Bereitstellung komplexer und transaktionsorientierter Forms-Anwendungen über das Internet oder das firmeneigene Intranet. Das Web Prinzip bei Oracle Forms legt drei Hauptkomponenten zugrunde: Client, Forms Listener und einer Forms Runtime Engine.

Ein Client zeigt die Benutzeroberfläche einer Form an und verwaltet die Interaktionen zwischen Benutzer und Forms Server. Die Bedienelemente der Benutzerschnittstelle (z.B. Maskenfelder, Checkboxen etc.) werden von einer Client-Komponente, einem Java Applet verwaltet. Durch die Verwendung von Applets können Forms-Oberflächen in ihrer Detailfülle besser angezeigt werden, als durch die eingeschränkten Layouteigenschaften einer HTML-Seite[33]. Aktuell wird hierfür nicht die native Browser JRE für die Ausführung eines Applets benutzt, sondern ein von Oracle gepackter JInitiator, der auf der jeweiligen Sun-Version basiert, eingesetzt. Er ermöglicht das Ausführen von javabasierten Anwendungen, die unter dem Aspekt einer maximalen Geräteunabhängigkeit entwickelt wurden und auf nahezu jedem Computer laufen. Über eine Internetadresse lässt sich nun der Oracle Forms Listener kontaktieren.

Ein Oracle Forms Listener startet eine Forms Server-Anwendungssitzung und stellt eine Verbindung zwischen dem Client und der Forms Runtime Engine her. Der Forms Listener verwaltet auch einen Pool mit verbindungsbereiten Laufzeit-Engines, durch den die Herstellung einer Verbindung mit dem Client so schnell wie möglich erfolgen kann.

[33] Hyper Text Markup Language, Sprache in der Internetseite geschrieben werden

Die Forms Runtime Engine verwaltet die Anwendungslogik und -verarbeitung. Ferner verwaltet sie im Auftrag vom Client eine Verbindung mit einer Datenbank. Der von der Forms Runtime Engine ausgeführte Code entspricht dem Code (From-, Menü- und Bibliotheksmodule), der für die Ausführung in einer Client/Server Umgebung eingesetzt wird. Um Formsanwendungen im Internet bereitzustellen sind keine Änderungen am Anwendungscode erforderlich.

Der Trend geht eindeutig immer mehr zu Webanwendungen hin. Viele Applikationen wurden in der Metro bereits für den Einsatz im Web entwickelt, doch leider gibt es immer noch vereinzelte Oracle Froms Programme wie z.B. den Fax-Basisdienst, die auf der Client/Server-Architektur basieren. Dies sind meist Systemprozesse, die nicht durch einen Web-Server an verschiedene Anwender verteilt werden müssen. Für Job-Scheduling-Systeme ist es erst durch den Wegfall des Runtime-Engine-Pools möglich, diese Prozesse zu überwachen.

Zuletzt sei noch angemerkt

Oracle verschweigt bis jetzt, dass Anwendungen, die bereits in Oracle Forms 6i entwickelt wurden, sich nicht ohne größeren Zeitaufwand problemlos nach Forms 9i kompilieren lassen. Die Unterschiede[34] zu den jeweiligen Oracle Forms Versionsständen sind zum Teil so groß, dass viele Consulting-Firmen Dienstleistungpakete für eine Komplettmigration anbieten.

Neukompilierungen nach 9i möchte die Metro MGI aber stets vermeiden („never touch a running system“). Oracle wird durch die neue Version 9i zukünftig nur noch Support für die Web Struktur anbieten, auf lange Sicht wird sich die Metro MGI also auf eine Umstellung vorbereiten müssen.

[34] Unterschiede <http://otn.oracle.com/products/forms/pdf/forms_upgrade_reference.pdf> (23.11.2003)

2.2 Oracle Reports

Ein Report ist ein Bericht, der Daten aus Tabellen liest, sie auswertet und in einem definierten Layout auf dem Bildschirm, Drucker oder in eine Datei ausgibt. Mit Oracle Reports Services lassen sich Reports anlegen, vorhandene Reports in einem internen oder externen Unternehmensnetzwerk ausführen oder im Internet publizieren. Die Faxe/Listen-Form wird eingesetzt, um auf diese Weise Faxe zu erzeugen. Dazu verschickt sie Anforderungen an den Reports Server. Diese werden nun weiter an die Runtime Engine des Oracle Report Services geroutet, die einen Report generiert. Als Resultat erhält man ein Dokument, das in verschiedenen Ausgabeformaten ausgegeben werden kann (HTML, Text, MS-Excel oder PDF). Am häufigsten werden Reports in der Metro als PDF-Form verwendet, da es aufgrund der Unabhängigkeit vom Betriebssystem und Textverarbeitungsprogrammen als das beste Publikationsformat gilt.

In Abbildung 12 ist nun zu sehen, wie ein Report durch die Faxe/Listen-Form generiert wird.

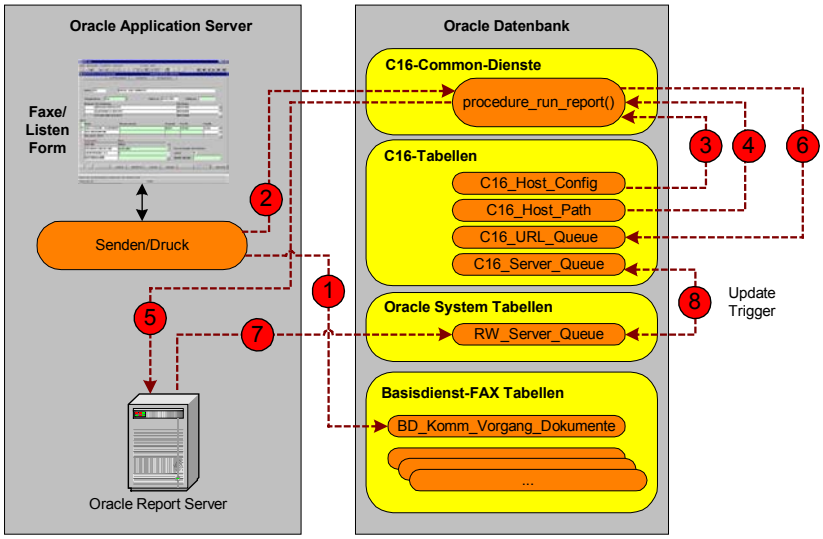

Abbildung 12 Ablaufplan Reportsgenerierung[35]

Nachdem der Anwender sich mit über einen Oracle Web Cache[36] mit einem Oracle Applikationsserver verbunden hat, erscheint die Faxe/Listen-Form in seinem Browserfenster. Der Anwender merkt nicht auf welchem Rechner die Fax-Anwendung tatsächlich läuft. Sind alle erforderlichen Eingaben in den Masken durchgeführt, so entscheidet er sich sicherlich zum Senden des (der) Report(s) mittels des

[35] zeigt einen AIX-Server aus einer verteilten Umgebung (bereits neue Fax-Systemlösung)
[36] s. Kapitel 1.8 „Der Oracle Web Cache“

„Senden/Druck“ Buttons. Der Fax-Vorgang wird nun mit dem Status[37] 3 (Auftrag erkannt, Report gestartet) in den Basisdienst-Fax Tabellen hinterlegt (vgl. Abbildung 12) [1], die später von den Fax-Dämonen ständig gepollt und abgearbeitet werden. Ab hier beginnt nun eigentlich schon die Reportinitialisierung, obwohl der Oracle Reportsserver bisher noch von nichts weiß. Von der Form aus wird über die Common-Service-Dienste die Prozedur `run_report()` im C16 Package („Flexible Listenselektion“) angestoßen [2]. Dieses Package befindet sich in der Datenbank und ist ausschließlich für die Reportsgenerierung zuständig. Damit die „Flexible Listenselektion“ einen Reportauftrag auf dem Reportserver starten kann, ist es zuerst erforderlich, festzustellen, auf welchem Host und im welchen Verzeichnis ein Report abgelegt werden soll. Hierfür muss zuerst der zugewiesene Host der durch einen installierten Oracle Web Cache variieren kann, durch die Faxe/Listen-Form festgestellt[38] werden; anschließend bekommt die Flexible Listenselektion durch die Common Service C16 Tabelle `C16_Host_Config` [3] den Reportservernamen auf dieser Maschine. Durch die Tabelle `C16_Host_Config_Path` [4] erhält die Faxe/Listen-Form einen absoluten Verzeichnispfad, indem später auch die Reportdateien (PDF) wiederzufinden sind. Nun sind alle Informationen für eine Reportgenerierung vorhanden. In Abbildung 13 ist nun zu sehen, dass eine Steuerdatei, die alle notwendigen Parameter der Faxe/Listen-Form enthält (wie z.B. Einkaufssachgebiet, Erstellungsdatum, Reportmodul[39] ...), mit einer *.par Dateiendung erzeugt wird. Im nächsten Schritt wird diese durch eine generierte Shell Datei[40] (*.sh) an einen Reportauftrag gekoppelt und anschließend von der Flexible Listen Selektion aus gestartet.

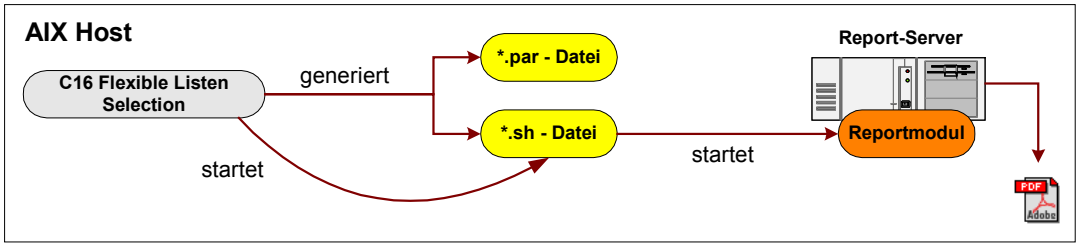

Abbildung 13 Reportgenerierung durch die Flexible-List-Selection

[37] Statusausprägungen wurden bereits im Kapitel 1.5.1 „Ein Fax geht auf Reisen“ beschrieben.
[38] ausführliche Erklärung der Funktionsweise im Kapitel 2.3 „Der Oracle Web Cache“.
[39] Ein Reportmodul generiert zu Projektdaten einen benutzerspezifischen Report. Reportmodule werden mit Hilfe des Report-Builder von Oracle erstellt.
[40] Mit einem Shell Script lassen sich mehrere Shell-Kommandoeingaben auf einer Konsole in eine Datei verlagern, die später nur noch mit dem entsprechenden Dateinamen aufgerufen werden muss.

Durch den Start dieser Shell Datei wird der Report Server Abb. 12 [5] aufgerufen, dem nun ein Reportmodul mit entsprechenden Reportparametern von der Faxe/Listen-From durch eine *.par-Datei übergeben wird. Die Flexible/Listen-Selection ist mit ihrer Aufgabe fertig und fügt in die `C16_URL_Queue` Abb. 12 [6] Tabelle den Namen erzeugten Reportdatei vom Reportserver ein und den dazugehörigen Downloadpfad, der später für die Fax-Dämonen relevant sein werden. Hier sei noch angemerkt, dass auf jeder AIX Maschine ein Oracle Report Server läuft und ein Report (sofern der Inhalt gleich ist) nicht jedes Mal für Lieferanten neu erzeugt wird. Der in Abbildung 13 dargelegte Prozess ist auf jedem AIX-Server zu finden, der von einem Oracle Web Cache zugewiesen wird.

Die Fertigstellung eines Reports wird nun durch eine Oracle Systemtabelle `RW_Server_Queue` gekennzeichnet Abb 12 [7]. Alle Reportaufträge werden dort mit einem Status versehen; im Fehlerfalle einer Reportsgenerierung lässt sich dort die Ursache feststellen. Weil ein direkter Zugriff auf diese Systemtabellen nicht ratsam ist, wird sie mittels eines Update-Triggers[41] Abb. 12 [8] ständig in eine C16 Tabelle `C16_Server_Queue` mit einem ähnlichen Aufbau geklont. Die Fax-Dämonen können so später anhand dieser Tabelle erkennen, ob bereits ein Report zu einem Fax erstellt worden ist, und abgeholt werden kann.

Im Normalfall sind alle Reports ohne größere Probleme erzeugt worden, doch der komplette Ausfall eines Reportsservers wurde erst durch ausgiebige Tests der Fax-Dämonen hervorgerufen. Bis zu diesem Zeitpunkt hatte sich noch niemand Gedanken darüber gemacht, auch die veralteten Reportdateien einmal zu löschen. Der Reportserver verweigerte seinen Dienst, mangels Speicherplatz auf den Festplatten. Man einigte sich darauf, diese Verzeichnisse durch Job-Scheduling-System von Zeit zu Zeit nach dem Dateierstellungsdatum zu säubern.

[41] Trigger sind Event-Handler in Datenbanken, die auf definierte Ereignisse reagieren und entsprechende Aktionen ausführen.

2.3 Der Oracle Web Cache

Der Oracle Web Cache ist ein Caching Service für Webserver. Er wird in der Metro eingesetzt und verbessert die Performance und Verfügbarkeit von Web Seiten, die auf einem Oracle9iAS[42] laufen und mit Datenbanken kommunizieren. Webmaster führen oft Statistiken auf Websites auf, um zu wissen, welche Seiten am häufigsten aufgerufen werden. Werden diese Seiten auf einem virtuellen Speicher gecached[43], so vermeidet der Oracle Web Cache, diese URL zum wiederholten Male vom Web Server aufzurufen. Dadurch werden Seiten schneller angeliefert und die Belastung der Http-Web-Server spürbar reduziert.

Ein Oracle Web Cache wird immer vor einen Web Server positioniert. Greift nun ein Browser eines Anwenders auf Web Sites zu, so sendet er seine Anforderungen via Http-Request an den Oracle Web Cache. Gilt der angeforderte Inhalt als veraltet oder modifiziert[44], so holt sich der Oracle Web Cache den neuen vom Web Server. Vergleichbare Produkte, die ausschließlich diese Funktionalität zur Verfügung stellen, gibt es auch kostenlos im Internet, wie z.B. den Squid-Proxy[45] für Linux bzw. Unix.

Neben der Aufgabe des Cachings bietet der Oracle Web Cache zusätzlich einige Vorzüge, die bei hoher Netzwerklast äußerst sinnvoll sind.

Zu diesen Eigenschaften zählen u.a. :

- Absicherung der Performance - Durch Statistiken lassen sich Cache-Konsistenz Performancefragen verwalten. Statistiken weisen den Inhalten Prioritäten zu und ermitteln, welche Dokumente in der vorhandenen Version übermitteln werden und welche neu abzuholen sind.

- Überlastungsschutz - Es ist möglich, die Anzahl der Requests an einen Oracle Web Cache einzuschränken.

[42] Application Server stellen eine Infrastruktur für den Ablauf von Business-Logik-Komponenten zur Verfügung.
[43] zwischengespeichert
[44] Ein typisches Merkmal hierfür ist das Änderungsdatum der angeforderten Seite
[45] siehe <www.squid.org> (03.04.2004)

- Backend Fail-over - Scheitert ein Server, so wird die Belastung automatisch auf die verbleibenden Web Server verteilt. Steht er wieder zur Verfügung, so nimmt der Oracle Web Cache diesen wieder in seinen Server-Pool auf.

- Sicherheit - Bietet Passwortauthentifizierung, Sperren von Ports und ein Timeout für inaktive Sessions.

Als Problemfall zur Faxe/Listen-Form erwies sich das Feature des automatischen Routings und des Serverbalancing, der hauptsächliche Grund für den Einsatz des Web Caches in der Metro. Hierdurch lassen sich Anforderungen auf mehrere Web Server verteilen, die noch genügend Ressourcen haben, um eine Anfrage zu verarbeiten. Die Faxe/Listen-Form läuft dadurch, wie in Abbildung 14 dargestellt, nicht immer zwangsläufig auf dem selben Rechner.

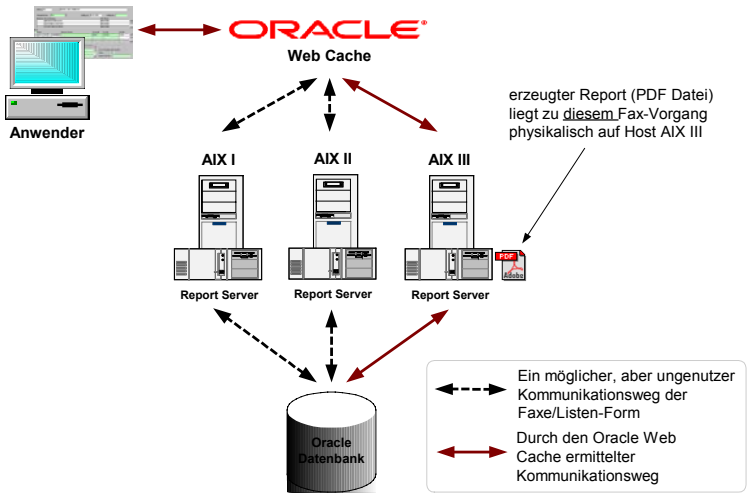

Abbildung 14 Funktionsprinzip des Oracle Web Caches

Möchte ein Benutzer ein Fax abschicken, so landet er auf einem der verfügbaren AIX-Web-Server der Metro (z.B. AIX III). In einem Browser lassen sich nun Faxe über die Faxe/Listen-Form erstellen, anschließend werden Faxe in Form eines generierten Reports als PDF-Datei physikalisch auf dem vom Web Cache zugewiesenen Host gespeichert. Diese lassen sich später über die Java-Fax-Dämonen von einem installierten Webverzeichnis[46] wieder abholen. Aufgrund dieser Tatsache ist es für die Fax-Dämonen nicht erkennbar, welcher Report auf welchem Rechner liegt. Die Lösung dieses

[46] Dateiordner der sich per Http-Protokoll ansprechen lässt

Problems liegt in der Tabelle `C16_URL_QUEUE`[47] des Common Service Dienstes. Diese enthält eine Spalte „URL“ mit dem kompletten Downloadpfad zu einem Report. Abbildung 15 zeigt, wie diese durch den C16-Dienst von der Faxe/Liste-Form nach Erstellung eines Vorganges und nach dem Start eines Reports gefüllt wird.

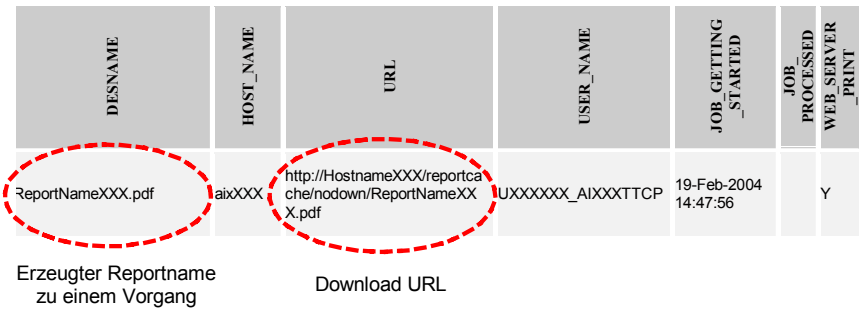

Abbildung 15 Lösung des Web Cache Problems

Zuvor gab es dieses Feld ebenfalls, allerdings war es damals ohne den Oracle Web Cache nicht erforderlich, eine komplette URL inkl. Hostname einzutragen, weil die gesamte Fax-Applikation inkl. der Fax-Dämonen nur auf einem Windows NT Rechner lief. Eine Modifikation der Faxe/Listen-Form war also nötig. Obwohl Oracle Forms eine Fülle von Bibliotheken zur Programmierung bereitstellt, gibt es derzeit keine Möglichkeit, über Forms den Namen[48] oder die IP-Adresse[49] eines Servers herauszubekommen. Java lässt sich aber glücklicherweise in diese Sprache einbetten. (vgl. Listing 1)

```
1  public String GetHostname() throws Exception, IOException {
2      String hostName;
3       try { hostName = InetAddress.getLocalHost().getHostName().toString();
4           } catch(Exception ex) {throw ex;}
5      return hostName;
6  }
```

Listing 1 Hostname eines Rechners mit Java ermitteln

Die Funktion `InetAddress.getLocalHost().getHostName()` ermittelt den Namen des Rechners auf dem die Faxe/Listen-Form läuft.

[47] s. Tabellenanhang
[48] um sich IP-Adressen leichter zu merken, können auch Namen sog. „Host-Names“ vergeben werden
[49] eindeutige Adresse, zum auffinden eines Rechners in einem Netzwerk

2.4 Logging Mechanismen

Während der Entwicklungsphase eines Programms ist es oft zur Gewohnheit geworden, auftretende Fehler mit Hilfe eines Debuggers zu eliminieren. Dies ist generell eine zuverlässige und schnelle Methode. Offen bleibt jedoch die Frage, ob der Programmierer auch alle Eventualitäten während des alltägliches Produktivbetriebes auf einem Server berücksichtigt hat. Ein Setzen von Breakpoints oder Watches ist nach der Kompilierung eines Programms nicht mehr möglich.

Die einfachste Form eines Loggings ist die Verwendung eines „Print-Befehls“, wie z.B. in der Programmiersprache Java `System.out.println()`, der einen einfachen Text auf der Console ausgibt (s. Listing 2).

```
1  import java.io.* ;
2
3  public class Test {
4    public static void main (String[] args) {
5         // Start kritischer Programmcode
6           System.out.println(„Bin im kritischen Programmcode...“);
7       // Ende kritischer Programmcode
8     };
9  };
```

Listing 2 Einfachste Form eines Loggings in Java

Häufig wird er verwendet, um bestimmte Warnungen zur Lokalisierung von Fehlern auszugeben. Das Konzept des Loggings geht jedoch weit über diese Funktion hinaus. Es wird eine einfache Möglichkeit geboten, sowohl Fehler während der Entwicklungszeit einzugrenzen als auch Programmabläufe zu überprüfen. Durch Logging Routinen kann später zur Ausführungszeit das Verhalten der Anwendung zur Laufzeit überwacht werden. Durch Loggingsysteme wird also nicht nur dem Programmierer die Fehlererkennung und -beseitigung erleichtert. Es wird die Überwachung einer Applikation erheblich vereinfacht, indem zum Beispiel die Ressourcenverwaltung, Systemabstürze oder auftretende Fehler aufgezeichnet und später ausgewertet werden können. Für den Fax-Dämon ist eine solche Überwachung unbedingt notwendig, weil diese auf Servern als Hintergrundprozesse laufen und somit z.T. auch keine Ausgabekonsole für auftretende Fehler besitzen.

2.4.1 Logging Bibliotheken

Loggingsysteme basieren auf dem Grundgedanken, Meldungen einzelner Softwarekomponenten unterschiedlich festzuhalten und zu verarbeiten, so dass nur eine Nachricht gespeichert oder ausgegeben wird, wenn dies vom Benutzer erwünscht ist. Prinzipiell bestehen alle Logger aus einem Loggerobjekt, welches jeder Softwarekomponente zugeordnet ist und den Loggingmechanismus übernimmt, einem Überwacher, der festlegt, auf welchen Ausgabekanal welcher Typ von Ereignis geloggt werden soll und einem Formatter, der das Layout einer Nachricht festlegt. Nach kurzer Recherchearbeit im Internet nach entsprechenden Bibliotheken, so stößt man schnell auf das sehr beliebte und bekannte Logging-Framework „Log4j“[50] von Apache. Unterstützt werden derzeit folgende Sprachen: C++, Java, DotNet, Perl, PHP und PL/SQL.

Alternativ wird seit der Version 1.4 des Java Development Kits von Sun ein eigenes Logging Interface zur Entwicklungsumgebung mitgeliefert. Leider entfiel diese Lösung, weil es zur Zeit der Planungsphase dieser Arbeit nur die ältere 1.3 JRE[51] Version für die AIX-Server Welt gab. Log4j ist ein Open-Source-Projekt von Jakarta und eine ständig weiterentwickelte Alternative zum Java Logging API von SUN. Zur Überwachung der Basisdienst Dämonen BD313 / BD314 wurde eine C98-Klasse `C98Logging.java` programmiert, die den Zugriff auf Log4j kapselt. Diese Kapselung war notwendig, weil alle Nachrichten ein einheitlich festgelegtes Layout zugewiesen bekommen, welches zentral in einer Datenbank hinterlegt ist. (vgl. Abb. 16)

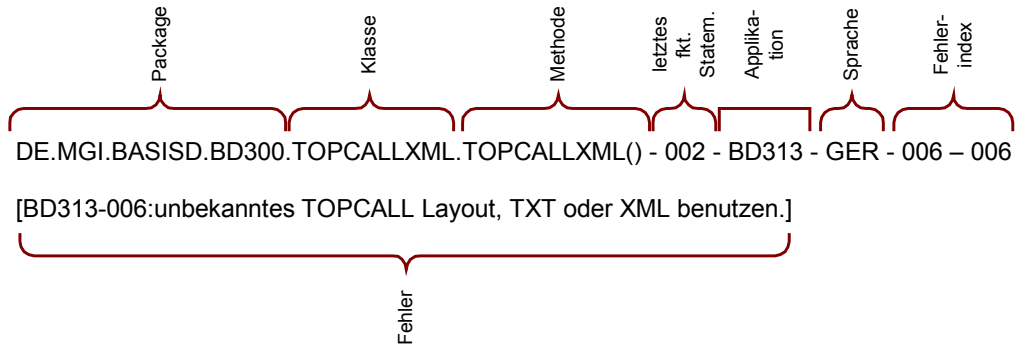

Abbildung 16 Typisches Layout einer MGI Fehlermeldung

[50] s. <http://logging.apache.org/log4j> (03.04.2004)
[51] Java Runtime Engine

Modifikationen dieses Layouts bewirken, dass auch in anderen Programmkonstrukten der Metro, die derzeit noch kein Log4j benutzen, wie zum Beispiel in PL/SQL Packages, ein Layout mitverändert wird. Somit haben alle Log-Nachrichten ein einheitliches Format. Als zusätzliches Feature, das Log4j bislang noch nicht anbietet, sollte eine Mehrsprachigkeit bei bekannten Nachrichten gewährleistet sein. Die verwendete Sprache des Log4j wird beim Start der jeweiligen Applikation in einem Konstruktor[52] gesetzt. Hier ein Programmausschnitt aus der Initialisierungsmethode für eine Landessprache im Fax-Dämons BD314 (Listing 3):

```
1   BD314_EXCEPTION_LANGUAGE = c02.c02P01_Get_Par_C(MODULE, APPL + "Java_EXCEPTION_LANGUAGE");
2   c98Logging.set_exception_language(BD314_EXCEPTION_LANGUAGE);
```

Listing 3 Mehrsprachenfähigkeit eines Fax-Dämonen setzen

Zu jeder Applikation ist eine feste Sprache in der Common Service Datenbanktabelle *C08_Messages* definiert. Die Funktion *c02.c02P01_Get_Par_C()* holt sich die verwendete Sprache mittels des Primary Key der sich aus der Applikation *APPL* → *„BD314"* und der Sprache *Java_EXCEPTION_LANGUAGE* → *„ENG"* (für Englisch) ergibt und überliefert diese der Log4j Klasse.

Log4j bietet unter anderem zusätzlich die Möglichkeit der dynamischen Konfiguration von mehreren Log Levels, durch die eine Dringlichkeit einer Meldung festlegt werden kann. Es besitzt 5 vorgegebene Log Levels inkl. die Level ALL und OFF, die jeweils durch ein Hierarchie von Prioritäten ausgegeben werden:

- FATAL (für die höchste Priorität)
- ERROR
- WARN
- INFO
- DEBUG (für die niedrigste Priorität)

Im gesamten Projekt wurden ausschließlich die Level ERROR und DEBUG verwendet. Damit diese und alle anderen Levels nutzbar werden, muss jede programmierte

[52] In der objektorientierten Programmierung ist ein Konstruktor eine spezielle Methode einer Klasse. Sie wird aufgerufen, wenn eine Instanz dieser Klasse (ein Objekt) erstellt wird.

Javaklasse in der *C08_MESSAGES* Tabelle der Datenbank registriert werden. Alle derzeit laufenden Applikationen der MGI greifen im Fehlerfalle auf diese Tabelle zu und geben eine benutzerdefinierte bzw. standardisierte Fehlermeldung in der gegenwärtigen Landessprache aus. Die Einspielung erfolgt bei der Metro MGI mit sog. Patches, die mittels SQL-Dateien eingespielt werden. Listing 4 zeigt eine solche Datei, die Log-Nachrichten für den Fax-Dämon BD314 in der Sprache ENG (Englisch) registriert.

```
1   INSERT INTO c08_messages
2        VALUES ('BD314JAVAFAT', 'ENG', '1', 'Java-Fatal occured: V1V.');
3   INSERT INTO c08_messages
4        VALUES ('BD314JAVAERR', 'ENG', '2', 'Java-Error occured: V1V.');
5   INSERT INTO c08_messages
6        VALUES ('BD314JAVAWRN', 'ENG', '3', 'Java-Warning occured: V1V.');
7   INSERT INTO c08_messages
8        VALUES ('BD314JAVADBG', 'ENG', '4', 'Java-Debug occured: V1V.');
9   INSERT INTO c08_messages
10       VALUES ('BD314JAVAIFO', 'ENG', '5', 'Java-Info occured: V1V.');
11  INSERT INTO c08_messages
12       VALUES ('BD314JAVAERR', 'ENG', '6', 'User data not tended, invalid email format.');
```

Listing 4 Typisches Installationsskript für die Mehrsprachenfähigkeit

Die Variable V1V ist ein freier Parameter und wird durch eine Common Service PL/SQL Prozedur mit der entsprechenden Fehlermeldung gefüllt und anschließend in einen der Ausgabekanäle des Log4j's geschrieben. Die letzte Zeile enthält einen benutzerparametrisierten Fehler, der später durch die Laufnummer „6" im Javaprogramm identifiziert wird. Die Hinterlegung eines generellen Layouts für alle Log-Nachrichten und die Zusatzfunktionalität der Mehrsprachigkeit setzt eine ständige Datenbankverbindung voraus, auch wenn die Businesslogik gar nicht mit der Datenbank verbunden ist. So trat schnell die Frage auf, wie Fehler eigentlich im Log4j-Framework abgefangen und verarbeitet werden sollten. Ein Abriss der Datenbankverbindung würde das Log4j-Framework veranlassen, in einer Endlosschleife hängen zu bleiben (sog. Deadlock). Die Funktionen, die eine entsprechende Log-Nachricht zu einem Fehler aus der Datenbank holen,

- *c08DBExp.get_undefinedMsg()* holt den Text für unbekannte Log-Nachrichten aus C98_MESSAGES-Tabelle

- `c08DBExp.get_definedMsg()` holt den Text für benutzerdefinierte Log-Nachrichten aus C98_MESSAGES-Tabelle

würden selbstständig wieder einen Fehler auslösen, und sich so selbst rekursiv wieder aufrufen. Für diese systembedingten Spezialfälle einigten wir uns wieder auf „Back to Basics“ und geben einen Text auf der Console aus, anschließend beendet sich der Fax-Dämon mit dem Fehlercode `System.exit(1)`. Dieser wird von einem Majesto-Scheduling-System aufgefangen, welches eine Alarmmeldung an den Operator abschickt. Neben den üblichen Logging-Ausgabekanälen (Datei und Console) des log4js, ist zusätzlich eine Common Service Tabelle `C98_INFO_MESSAGES` in der MGI-Datenbank angelegt worden. Durch die Erweiterung des log4j mittels eines Appenders[53] wird diese neben den anderen Ausgabekanälen zusätzlich immer mit gefüllt. Der Vorteil dieser Lösung ist es, dass es nicht notwendig ist, direkt Kontakt mit einen zuständigen System-Operator aufzunehmen, der den Host nach Fehlerdateien absucht. Mittels des Loggings in einer Datenbank ist es nun möglich, Fehler von einer zentraler Stelle aus zu finden, ohne auf Mitarbeiter von andere Abteilungen abgewiesen zu sein. Folgende Tabelle zeigt nun einen typischen Auszug der `C98_INFO_MESSAGES` Tabelle.

IP_ ADRESS	HOST_ NAME	DAEMON	APPL_ID	MODULE_ID	CRASHS_AT	STMTLINE	TIMESTAMP DATE	MESSAGE_ TYPE	MESSAGE
XXX.XXX.XXX.XXX	aixXXX	BD313	BD	BD313	DE.MGI.BASISD.BD313.MAIN()	000	2004-02-17 04:39:17,062	ERR	Eine JAVA-Error Meldung ist aufgetreten: DE.MGI.BASISD.BD300.TOPCALLXML.TOPCALLXML() - 002 - BD313 - GER - 006 - 006[BD313-006:unbekanntes TOPCALL Layout, TXT oder XML benutzen.]
XXX.XXX.XXX.XXX	aixXXX	BD313	BD	BD313	DE.MGI.BASISD.BD313.BD331()	015	2004-02-17 04:39:13,109	ERR	Eine JAVA-Error Meldung ist aufgetreten: DE.MGI.BASISD.BD300.TOPCALLXML.TOPCALLXML() - 002 - BD313 - GER - 006 - 006[BD313-006:unbekanntes TOPCALL Layout, TXT oder XML benutzen.]

Als zusätzliches Feature der neuen Appenderklasse ist es nun sogar möglich, nicht nur die Fehlerklasse und die Methode ausfindig zu machen, die einen Fehler verursacht hat,

[53] Appender sind Erweiterungen des log4js Frameworks. Sie lassen sich je nach Konfiguration ein- oder ausschalten und werden bei Log-Nachrichten aufgerufen, um entsprechend darauf zu reagieren.

sondern viel wichtiger ist es auch zu wissen, auf welchem Rechner dies geschehen ist. Eindeutig identifizieren lässt sich dies durch die Spalten `IP` und `HOSTNAME` in der Tabelle.

Ein großer Nachteil der Hinterlegung einer Mehrsprachigkeit und eines allgemeinen Layouts in einer Datenbank ist es, dass diese auch jederzeit zur Verfügung stehen müssen. Der Abriss einer Datenbankverbindung würde die Dämonen zwar ordnungsgemäß mit einen Fehlerreturn Code beenden, Fehlermeldungen lassen sich dadurch aber nicht mehr generieren. Für diesen Sonderfall sollte aber nicht noch zusätzlich ein Standardlayout im log4j mit hinterlegt werden. Ein Ausfall der kompletten Datenbank hätte zur Folge, dass andere Datenbankanwendungen der MGI (z.B. Forms-Anwendungen) ebenfalls nicht mehr funktionieren würden. Der Fehler würde somit zwangsläufig schnell bemerkt werden.

2.4.2 Exception Chaining

Eine Ausnahme[54] oder Ausnahmesituation bezeichnet die Möglichkeit, in einem Programmabschnitt einen unvorhergesehenen Zustand zu entdecken und angemessen zu behandeln. Im Fehlerfalle der neu zu programmierenden Fax-Dämonen ist eine lückenlose Dokumentation zur genaueren Fehlersuche unbedingt notwendig. Exception Chaining ist eine Erweiterung von Standardfehlermeldungen. Unter Exception Chaining versteht man die Verkettung von Ausnahmeereignissen. Sie macht bei mehreren aufeinander folgenden Fehlerbedingungen nicht nur die zuletzt, sondern auch alle zuvor aufgetretenen für den Programmierer sichtbar.

Die Kapselung ist einer der wichtigsten Vorteile der objektorientierten Programmierung. Sie erlaubt es einem Objekt, einem anderen Objekt Anweisungen zu geben, ohne dass es wissen muss, wie diese Anweisungen ausgeführt werden. Leider bricht diese Detailfreiheit, die eine gute Kapselung möglich macht, in sich zusammen, wenn Fehler auftreten. Ab SDK 1.4 wird Exception Chaining[55] standardmäßig unterstützt und kann durch eine Erweiterung der Basis-Exceptionklasse im SDK[56] 1.4 genutzt werden. Ein verursachter Fehler (z.B. LowLevelException, Listing 5, Zeile 4) wird durch einen Try & Catch-Block abgefangen, und anschließend als neue Exception geworfen. Der

[54] engl. Exception
[55] vgl. Java 1.4 API <http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Throwable.html> (15.04.2004)
[56] "Software Development Kit" - Entwicklungs-Werkzeugkasten für Java Programme

Unterschied zum SDK 1.3 liegt dabei in der `initCause()` Methode (Zeile 5), sie hängt eine LowLevelException an eine neue HighLevelException. Eine vollständige Ausgabe aller Fehler erhält man mit `Exception.printStackTrace()`, der Ursprung wird durch einen caused by [Zeile 9 in Listing 11] gekennzeichnet.

```
1   public void do_error() {
2   try {
3        // do something critical and throw an LowLevelException("Datei nicht gefunden")
4   } catch (LowLevelException ex) {
5      throw new HighLevelException().initCause(ex); }}
6
7   Der dazu gefüllte Printstack hat folgenden Inhalt:
8   java.lang.Exception: Konnte Datei nicht lesen
9     at mypackage1.Testclass.do_error(Testclass.java:32)
10    at mypackage1.Testclass.main(Testclass.java:46)
11  Caused by: java.io.FileNotFoundException: Datei nicht gefunden
12    at mypackage1.Testclass.do_next(Testclass.java:30)
```

Listing 5 Vollwertiges Exception Chaining in Java SDK 1.4

Leider ist auf allen Web-Servern der Metro noch die veraltete Version des SDKs 1.3 installiert, später sollen unter dieser auch die Java-Fax-Dämonen laufen. Folgende drei einfache Beispiele zeigen, wie es trotzdem möglich ist, mit dieser Situation auch im SDK 1.3 umzugehen. Um das Prinzip zu erläutern, wird in allen Beispielen eine einfache Scheduler Klasse verwendet, die mit Downloadaufgaben aus dem Internet (sog. Tasks) gefüllt werden kann.

In nachstehendem Listing 6 wird eine IOException durch die `Scheduler.getTotalDuration()` Funktion in der Scheduler Klasse ausgelöst [Zeile 6]. Der Ursprung des Fehlers stammt aber aus `Downloadtask.getDuration()`, und wird durch einen throw[57] [Zeile 27] an die obersten Schichten im aufrufenden Stapel weitergeleitet [Zeile 3]. Diese werden nun von `Scheduler.getTotalDuration()` aufgefangen. Leider kann diese Funktion nur allgemeinere Exceptions werfen. Der eigentliche Typ des Fehlers „IOException“ geht in diesem Falle verloren und kann später nicht mehr vom Programmierer aufgefangen werden.

[57] throw wirft in Java benutzerdefinierte oder systembedingte Fehler die später aufgefangen werden können

```
1   public class Scheduler {
2          private List todoList = new LinkedList();
3          public long getTotalDuration() throws Exception {
4                           long total = 0L;
5                           for (Iterator i=todoList.iterator(); i.hasNext();  ) {
6                                       total += ((Task)(i.next())).getDuration();}
7                           return total;}
8          public void main(String[] args) {
9                          Scheduler s = new Scheduler();
10                         s.todoList.add(new DownloadTask("http://google.com"));
11                         try { System.out.println(s.getTotalDuration());
12                         } catch (Exception doh) {
13                                      doh.printStackTrace();}}
14  }
15   class DownloadTask {
16          private String url;
17          public DownloadTask(String u) {
18                         url = u; }
19          public long getDuration() throws IOException {
20                         // do something critical and throw an IOException  }}
```

Listing 6 Scheduler-Beispiel 1 für Exception Chaining

Eine etwas verbesserte Version zeigt das nächste Listing. Hier wird die IOException von *Scheduler.getTotalDuration()* in eine allgemeinere benutzerdefinierte TaskException Klasse umgewandelt [def. in Zeile 22]. Dem Programmierer ermöglicht dies eine saubere Behandlung der Ausnahme im Scheduler. Schade nur, dass die IOException, mit der alles begann, nun verloren ist. Sie wurde von dem Objekt Task verbraucht und durch eine TaskException ersetzt. Außerdem ist es zwecks Übersichtlichkeit nicht sinnvoll, für jede Methode einer Klasse eine extra Exception-Klasse zu definieren. Somit bleibt zusätzlich noch die Frage offen, welche Methode den Fehler eigentlich geworfen hat. Eine spätere Suche nach dem eigentlichen Grund des Fehlers in den oberen Klassen würde sehr schwierig werden.

```
1    public class Scheduler {
2       private List todoList = new LinkedList();
3       public long getTotalDuration() throws TaskException {
4           long total = 0L;
5           for (Iterator i=todoList.iterator(); i.hasNext(); ) {
6               total += ((Task)(i.next())).getDuration();
7           }
8           return total; }
9       public void main(String[] args) {
10          Scheduler s = new Scheduler();
11          s.todoList.add(new DownloadTask("http://google.com"));
```

```
12          try {
13              System.out.println(s.getTotalDuration());
14          } catch (TaskException doh) {
15              System.out.println("Unable to perform task");
16              doh.printStackTrace();}
17      }
18  }
19  class TaskException extends Exception {
20      public TaskException(String msg, Throwable cause) {
21          super(msg, cause);}
22  }
23  class DownloadTask {
24      private String url;
25      public DownloadTask(String u) {
26          url = u;}
27  public long getDuration() throws TaskException {
28          try {
29              // do something critical and throw an IOException
30          } catch (IOException doh) {
31              throw new TaskException("Couldn't download", doh);}}
32  }
```

Listing 7 Scheduler-Beispiel 2 für Exception Chaining

Noch besser ist es, wenn man eine eigene Exception-Unterklasse erstellt, die es ermöglicht, eine geworfene Exception zwischenzuspeichern und selbst wieder eine auszulösen. Über die `getWrappedThrowable()` Methode der Klasse ließe sich somit die ursprüngliche Exception wiederholen. Dieses Prinzip bezeichnet man als eine Nested[58]-Exception. Von diesen Optionen ist die dritte sicherlich die beste Wahl. Leider ließ sich diese Lösung bei den Java-Fax-Dämonen nicht anwenden, weil man durch die Verwendung von fremden APIs anderer Hersteller daran gehindert wird, seine eigene Nested-Exception überall einzusetzen.

```
1   public class NestedException extends Exception
2      private Throwable wrappedThrowable;
3         public NestedException(String s) {
4              super(s); }
5         public NestedException() {
6              super();}
7         public NestedException(String s, Throwable t) {
8              super(s);
9              setWrappedThrowable(t);}
10     public NestedException(Throwable t) {
11             super();
```

[58] Einnisten bzw. Einschachtelung, im SDK 1.4 ist diese Funktionalität bereits in die allgemeinere Exceptionklasse eingebunden worden.

```
12              setWrappedThrowable(t);}
13      public Throwable getWrappedThrowable() {
14              return wrappedThrowable;}
15      public void setWrappedThrowable(Throwable t) {
16              wrappedThrowable = t;}
```

Listing 8 Scheduler-Beispiel 3 (Nested Exception Klasse)

Trotzdem wurde auf ein Exception Chaining zwecks Logging nicht verzichtet. Das nächste Listing 9 zeigt nun einen Auszug aus einer der vielen JDBC-Wrapperklassen, die den Zugriff auf die Datenbank kapseln.

```
1    public bd332j00() throws Exception {
2            final String METHODNAME = "bd332j00()";
3            int stmtLine = 0;
4            this.registerClass(System.getProperty("Daemon"), "BD332", "BD");
5            try { // do something critical and throw an Exception }
6            catch(Exception ex) {
7                c98Logging.error(APPL, MODULE, PACKAGENAME, METHODNAME, stmtLine, ex.getMessage());
8                throw ex;}}
```

Listing 9 Auszug einer JDBC-Wrapper-Klasse

In erster Linie war es wichtig, dass ein auftretender Fehler zum sofortigen Programmabbruch der Dämonen führen sollte. Spezielle Fehlerbehandlungslogiken konnten durch die oben genannten Gründe im SDK 1.3 leider nicht realisiert werden. Das Logging-Framework „log4j“ macht es aber dennoch möglich, eine Art Exception Chaining zu realisieren. Mit der vorzeitigen Ausnutzung des Log4j’s [Zeile 7], ist es möglich, einen Fehler in die konfigurieren Ausgabekanäle zu loggen, bevor die Methode durch das Werfen einer Exception mit *`throw`* [Zeile 8] verlassen wird. Wendet man dieses Procedere in allen Klassen und Methoden kontinuierlich an, so hat man in allen Ausgabekanälen des Log4j’s eine Hierarchieordnung von aufgetretenen Fehlern (vgl. Seite 48 – Tabelle *`C98_INFO_MESSAGES`*).

Über die Übergabeparameter des Log4j's [Zeile 7]

- Dämon – Bezeichnet die Anwendung unter der alle Javaklassen laufen z.B. „Faxdienst", (wird über Systemproperties global allen Klassen zur Verfügung gestellt)
- Module – Modul der Services BD-Basisdienst oder C-Commondienst
- Appl - Klassenname
- Packagename – Packagestruktur unter der die Klasse in Java zu finden ist
- Methodenname – Funktion bzw. Prozedur, in der ein Fehler aufgetreten ist
- StmtLine – Letztes funktionierendes Statement im Programmabschnitt

lässt sich später die Fehlerursache leicht lokalisieren.

Durch ein Weiterreichen einer aufgetretenen Exception an den eigentlichen Aufrufer, der selbst wieder eine Exception auslöst, ist es erst möglich gewesen, diesen in den obersten Klassen zu verarbeiten und einen Dämonstatus in die Fehlertabelle `C98_INFO_MESSAGES` einzutragen. Obwohl dies sicherlich kein vollwertiges Exception-Chaining ist, war es somit möglich, die auslösende Ursache zu einem Fehler nicht zu verlieren und an die Oberklassen weiter zu reichen.

Die Oberklassen (Hauptprogramm der Fax-Dämonen) verarbeiten diesen Fehler, indem sie einen Rückgabecode an das Betriebssystem zurückliefern, der später von einen Job-Scheduling-System[59] ausgewertet werden kann.

[59] überwachen Systemprozesse sog. „Jobs" und schlagen bei einem entsprechenden Rückgabecode Alarm – vgl. Kapitel 4.8 "Job-Scheduling-Systeme"

2.5 Datenbankanbindung

2.5.1 Java Database Connection

JDBC ist eine standardisierte Schnittstelle, um aus Java Programmen den Zugriff auf beliebige relationale Datenbanken zu ermöglichen. Als Alternative gibt es noch ODBC[60] von Microsoft mit gleicher Funktionalität, da es aber in der Programmiersprache C++ realisiert wurde, ist es nicht plattformunabhängig und findet deshalb keinen Einsatz in der Metro. Oft wird JDBC fälschlicherweise mit "Java Database Connectivity" gleichgesetzt, tatsächlich ist JDBC aber ein Warenzeichen und kein Akronym. JDBC bietet ein generisches SQL-Datenbank-Interface, das von einem speziellen Datenbanksystem unabhängig ist. Der Zugriff erfolgt transparent. Ein Programmierer muss sich nicht mehr mit unterschiedlichen Implementierungen einer Anfrageoperation auseinandersetzen, sondern kann sich ganz auf die Applikationsentwicklung konzentrieren.

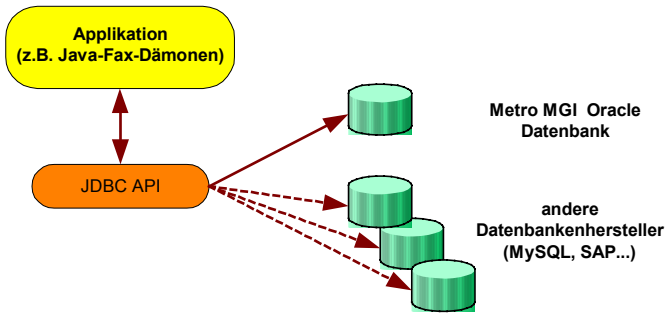

Abbildung 17 JDBC Anbindung

Dabei werden über einen Treibermanager Datenbanktreiber dynamisch geladen, die einheitliche API[61]-Aufrufe in Befehle übersetzen, die das Datenbanksystem verstehen und ausführen kann. Wird nach der Abfrage der Datenbank ein ResultSet[62] zurückgegeben, wird sie durch den Treiber in eine für Java verständliche Form transformiert und für die Fax-Dämonen zur Verfügung gestellt. Alle Datenbankaufrufe im Fax-Projekt wurden durch sog. Wrapper gekapselt, um sie auch in anderen Klassen wiederzuverwenden. Listing 10 zeigt ein Beispiel einer typischen Wrapperklasse. Jeder Methodenaufruf in diesen Klassen öffnet zu Beginn einer Datenbankoperation mittels

[60] Open Database Connectivity
[61] Application Programming Interface - Schnittstelle für die Anwendungsprogrammierung
[62] Ergebnismenge einer Datenbankanfrage

Connection-Pooling eine Datenbankverbindung [Zeile 7], und schließt diese nach vollendeter Arbeit wieder [Zeile 14]. Ein Programmierer muss sich somit nicht zusätzlich um eine Datenbankverbindung kümmern. Dieses Prinzip funktioniert ohne Probleme, solange man keine „ResultSets“ als Rückgabewerte verwendet. Kurz vor der Projektfertigstellung kam die Idee, anstatt der handelsüblichen ResultSetKlasse *java.jdbc.resultSet* des SDK die spezielle *Oracle.jdbc.OracleResultSet* [Zeile 1] des Oracle JDBCs zu verwenden.

```
1   public OracleResultSet bd331j00_get_status_of_Doc(int pi_vorgang_nr,
2                                                     Connection con)  throws Exception {
3     OracleCallableStatement ocstmt;
4     String sqlQuery;
5     OracleResultSet rs;
6       try {
7         // bei java.jdbc.resultSet (hier öffnen einer Datenbankverbindung)
8         sqlQuery = "{call ? := bd331k00.bd331p16_get_status_of_doc(
9                     '" + pi_vorgang_nr + "')}";
10        ocstmt = (OracleCallableStatement)con.prepareCall(sqlQuery);
11        ocstmt.registerOutParameter(1, OracleTypes.CURSOR);
12        ocstmt.execute();
13        rs = (OracleResultSet)ocstmt.getObject(1);
14        ocstmt.close();
15        // bei java.jdbc.resultSet (hier schliessen der Datenbankverbindung)
16        return rs;}
17  …
```

Listing 10 Beispiel OracleResultSet einer JDBC Wrapperklasse

Diese unterstützt neben den normalen Rückgabewerten auch spezielle Oracle Datenbankentypen. Anschließende Tests ergaben, dass das *Oracle.jdbc.OracleResultSet* beim Auslesen einer Ergebnismenge eine ständige Verbindung voraussetzt, *java.jdbc.ResultSet* speichert diese im Gegenzug in einem Puffer. Bei großen Ergebnismengen (z.B. Bilder) sollte man deshalb auf *java.jdbc.ResultSet* verzichten, die schnell die Speicher-Ressourcen eines Rechners belegen kann. Aus Sicherheits- und Performancegründen wurden alle JDBC-Klassen des Projekts, die einen *Oracle.jdbc.OracleResultSet* als Rückgabewert beinhalten, mit einer eigenen Connection versehen, die dann als Übergabewert [Zeile 2] mitgegeben werden muss. Die Businesslogik wird somit wieder zum Verwalter der Connection, was anfangs vermieden werden sollte.

2.5.2 Connection Pooling

Unter Connection Pooling versteht man generell die Wiederverwendung physischer Verbindungen. Sinnvoll ist dieser Einsatz auch bei den Java-Fax-Dämonen. Nach der erfolgreichen Bearbeitung von Faxvorgängen schließen alle Dämonen ihre Datenbankverbindungen und verweilen einige Zeit[63] im Standbybetrieb. Nehmen diese anschließend ihre Arbeit wieder auf, so muss eine neue Verbindung an die Datenbank angefordert werden. Der Aufbau einer Datenbankverbindung ist aber eine relativ ressourcenintensive Operation. Deshalb ist es besser, eine Datenbankverbindung im Server nicht sofort zu schließen, wenn ein Client seine Arbeit beendet hat. Eine geöffnete Datenbankverbindung kann gespeichert und für nachfolgende Client-Zugriffe wiederverwendet werden. Connection Pooling in JDBC 2.0 soll dem Entwickler genau diese Funktionalität zur Verfügung stellen. Beim Connection Pooling mit dem Oracle-Treiber kann man zwischen zwei Stufen unterscheiden:

- **Pooled Connection** - Sie repräsentiert eine wiederverwendbare Datenbank-verbindung. Sie wird beim Zurücklegen in den Pool nicht physisch geschlossen, sondern lediglich freigegeben und bleibt somit für nachfolgende Client-Zugriffe erhalten.

- **Connection Pool** - In einem Connection Pool wird eine Menge von Pooled Connections verfügbar gemacht, die bei Bedarf aus dem Pool angefordert und nach der Verwendung wieder in den Pool zurückgestellt werden. Alle Verbindungen im Connection Pool sind dabei mit derselben Datenbank verbunden.

Eine einzelne Pooled Connection implementiert somit den generellen Mechanismus für die Wiederverwendung von Datenbankverbindungen. Der Connection Pool macht von diesem Mechanismus Gebrauch, indem er mehrere dieser Verbindungen in einem Cache bereitstellt. Connection Pooling wurde anfänglich durch simple Klassen von mir selbst implementiert. Grund hierfür war die Erfordernis der Mehrsprachenfähigkeit der Applikation. Die Sprache wird bei jedem Auschecken einer Verbindung aus dem Pool explizit in einer neuen Datenbank-Session gesetzt.

[63] je nach Konfiguration 0 - ... Minuten

Leider wurde durch diese Lösung jede JDBC Verbindung nach einer undefinierten Zeit geschlossen, was zum sofortigen Abbruch der Fax-Dämonen führte. Recherchen ergaben, das es sich hierbei um einen Bug[64] im JDBC Treiber handelt, der ohne Dekompilierung und Abänderung des Source-Codes der JDBC-Bibliothek nicht behebbar war - Ursache hierfür war ein Closing-Event, das nicht an andere Klassen durchgereicht wurde – eine physikalisch geschlossene Verbindung wurde fälschlicherweise als „offen“ deklariert. Fertig implementiertes Connection Pooling in der JDBC Bibliothek zeigte dieses Verhalten jedoch nicht. Aus diesem Grunde wurde die alte Lösung verworfen und eine Wrapperklasse über den Pool programmiert.

2.5.3 Die SQLJ Technologie

Die Anforderungen der Fax-Dämonen setzte voraus, dass alle Zugriffe auf eine Datenbank gekapselt werden müssen. Strategisch gesehen ist eine solche Kapselung generell immer sinnvoll, solange sie später ohne große Mühe in einem Projekt vollständig nachvollzogen werden kann. Eine Kapselung wird meist mit Wrapperklassen realisiert. Sie stellen im Falle der Fax-Dämonen ein Signaturmapping[65] der SQL-Methoden und -Prozeduren in einer Datenbank her, und erleichtern es dem Programmierer, sich nicht noch zusätzlich mit der Datenbankverbindung zu beschäftigen. Leider verlagert sich dadurch ein Teil der Businesslogik vollständig in eine Datenbank. SQL-Anweisungen sollten daher am besten in extra XML-Dateien abgelegt werden, welche anschließend nur noch zur Datenbank übertragen werden. Durch die historische Entwicklung von Programmen in der MGI, (insb. durch Forms) war diese Lösung aber nicht erwünscht.

Um diese Wrapperklassen zu erstellen fiel der Fokus zuerst auf SQLJ. SQLJ ist eine eingebettete Datenbanksprache, die sehr eng in die eigene Host-Sprache eines Programmes (Java) integriert ist. Diese enge Verbundenheit mit der Host-Sprache zeigt sich besonders dadurch, dass Java-Klassen als Datentypen der Tabellenspalten verwendet werden können. Ein großer Vorteil von SQLJ ist es, dass es sich um eine standardisierte Sprache handelt. Im Unterschied zu SQL, wurden bei SQLJ zuerst die Grundlagen der Sprache festgelegt, die erst dann durch die einzelnen

[64] Oracle JDBC-Bug No. 2377695
[65] Eine Signatur bezeichnet die Übergabeparameter einer Funktion bzw. Prozedur

Datenbankhersteller implementiert wurde. SQLJ besteht aus einer Menge von Klauseln, die die Sprache Java erweitern und immer mit dem Präfix `#sqlj` [Listing 11 – Zeile 1 und 3] beginnen:

```
1  #sql { CREATE TABLE Personen (Name    char(4) not null, Vorname char(20) not null) };
2  System.out.println("Tabelle erstellt");
3  #sql { DROP TABLE Personen };
4  System.out.println("Tabelle gelöscht");
```

Listing 11 Embedded SQLJ in einer Javaklasse

Anwendungen, die in Java und SQLJ geschrieben sind, sind portierbar und können auch mit Datenbanken anderer Hersteller kommunizieren. SQLJ kann derzeit leider nur statisch verwendet werden, es ist aber durchaus möglich, durch die Kombination von SQLJ und JDBC dynamisches SQL zu simulieren.

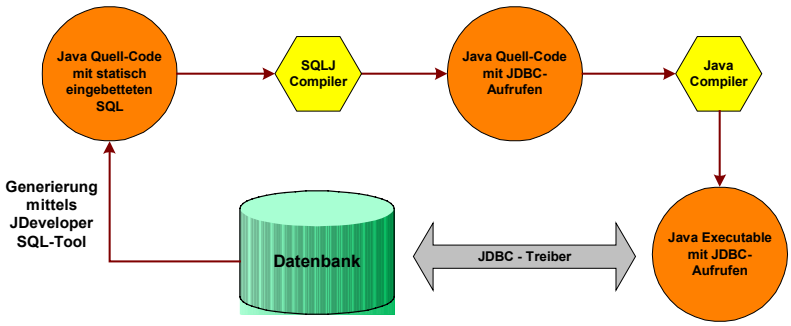

Abbildung 18 Übersetzung und Ablauf eines SQLJ-Programmes

Um SQLJ zu nutzen, ist es erforderlich das Javaprogramm zuerst durch einen Pre-Compiler vorübersetzen zu lassen. Während dieser Übersetzungsphase werden die eingebetteten SQLJ Anweisungen durch SQLJ-Laufzeitbibliotheken ersetzt. Das Ergebnis dieser Phase ist ein Javaprogramm, dass nun vom Java Compiler übersetzt werden kann. Der übersetzte Java-Bytecode wird durch die SQLJ-Laufbibliothek in ein JDBC-Programm umgewandelt, dass nun mit Hilfe des JDBC-Treibers Zugriffe auf die Datenbank gestattet.

Die Programmierumgebung des JDevelopers bietet dem Entwickler eine einfache Möglichkeit, komplette Datenbankpackages[66] mittels SQLJ in Java-Klassen zu

[66] Sammlung von SQL-Funktion die zu einem Package zusammengefasst werden können.

transformieren. Im System-Navigatorfenster des JDevelopers, lassen sich unter > Database > Benutzername > Packages diese über dem Kontextmenüpunkt „Generate Java“ erzeugen. Anschließend wird eine *.sqlj Datei angelegt, die später durch den SQLJ Compiler in eine Java Klasse umgewandelt wird.

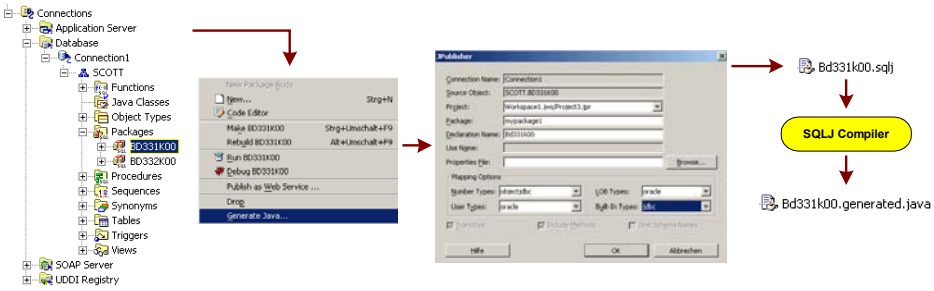

Abbildung 19 SQLJ Generierung im Oracle JDeveloper

Obwohl diese Technik der Transformierung einem Entwickler viel Arbeit abnehmen kann, war es dennoch nicht möglich sie für den Fax-Basisdienst zu verwenden:

- Übersetzte SQLJ Dateien in Java müssen im Konstruktor immer eine JDBC-Connection übergeben bekommen. Sie enthalten keinen Mechnismus (wie z.B. Connection-Pooling), der überprüft, ob eine Verbindung mit einer Datenbank überhaupt noch existiert.
- Java unterstützt in allen von Compiler generierten Methoden nur einen möglichen Rückgabewert. Die Sprache SQL bietet aber die Möglichkeit über den OUT-Parameter praktisch unendlich viele Rückgabewerte an. OUT-Parameter finden durchaus oft Verwendung in den Basisdiensten der MGI.
- SQLJ generiert alle Funktionen und Methoden auf der Basis eines Datenbankbenutzers (z.B. Scott), der auch später im Quellcode hinterlegt wird. > `BEGIN:1:=SCOTT.BD331K00.BD331P07_GET_DESNAME(…)` Dies ist generell eine schlechte Wahl. Sollte später der Benutzer einmal geändert werden, so muss das komplette Programm neu übersetzt werden.

Oracle-Berater bestätigten diese fehlende Flexibilität von SQLJ, und empfohlen in den Wrapperklassen ein Mapping von Hand per JDBC vorzunehmen. Trotz intensiver Fahndung nach ähnlichen Tools wie SQLJ im Internet blieb die Suche erfolglos. Natürlich ist durch die Nutzung von Java durch JDBC ein erheblicher Overhead entstanden. Die alten Fax-Dämonen liefen durch die datenbanklastige

Programmiersprache Oracle Forms mit Sicherheit weitaus performanter als die neue Java Lösung, aber bei einem täglichen Aufkommen von ca. 50 Faxen, macht sich dieser Nachteil jedoch nicht bemerkbar.

Abschließend sein noch erwähnt das weder SQLJ noch JDBC im Gegensatz zu Oracle-Forms in der Lage ist, ein fertiges Programm zur Kompilierzeit gegen eine Datenbank zu validieren; fehlende Methoden in der Datenbank werden erst zur Laufzeit der Fax-Dämonen bemerkt.

3. Kapitel – Fax Kommunikationsanbindung

3.1 Der TOPCALL-Server

Informationen können den Einzelnen heute auf vielen verschiedenen Wegen erreichen - als Telefonat, als Brief, als Fax und oder E-Mail. Alle diese verschiedenen "Messages" dem Betreffenden schnell, zentral und übersichtlich zu präsentieren, ist das Anliegen von "Unified Messaging". Durch die Nutzung von "Unified Messaging" ersparen sich manche Anwender überhaupt erst ein Faxgerät. So ist es zum Beispiel für einen Mitarbeiter möglich, eingehende Faxe als E-Mail zu erhalten. Mit seinem E-Mail-Account erhält er auch eine individuelle Faxdurchwahl. Alle hier ankommenden Faxe werden in ein Grafikformat umgesetzt und an die E-Mail Adresse des Betreffenden weitergeleitet. Er kann das Fax dann am Bildschirm seines PCs betrachten und bei Bedarf auch ausdrucken. Vor 25 Jahren entwickelte die Firma TOPCALL eine Software, mit der sich Faxe an Geschäftsanwendungen anbinden ließen. TOPCALL bietet eine Komplettlösung, bestehend aus dem Basissystem und einer wachsenden Zahl an optionalen Modulen. Durch ständige Weiterentwicklung ist eine All-In-One Lösung entstanden (vgl. Abb. 20), die es möglich macht, alle gängigen Geschäftsprozesse, die an CRM[67], ERP[68] Systeme gekoppelt sind, mit allen wichtigen Kommunikationsmedien zu verbinden. Zum Beispiel Email, Fax, Telex, SMS (MMS), und Voice (Festnetz, Mobilfunk, IP).

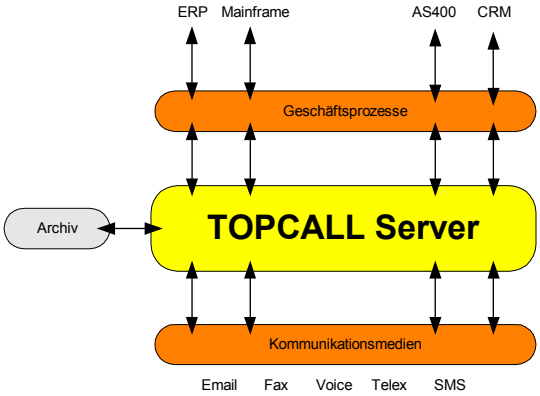

Abbildung 20 Der TOPCALL Communication Server

[67] CRM „Costumer Relationship Management“ umfasst alle Computer-Programme zur Unterstützung der Kundenbeziehungen in den Bereichen Marketing, Vertrieb, Service und Planung. CRM gibt es als selbständige Softwarelösungen und als Bestandteil von ERP-Systemen

[68] ERP „Enterprise Resource Planning“ löst heute immer mehr den herkömmlichen Begriff “kommerzielle betriebliche Datenverarbeitung” ab. Größter Anbieter von ERP ist SAP.

Auf Wunsch des Anwenders lassen sich alle Nachrichten in einem Archiv[69] dauerhaft speichern.

Später wurden Unified Messaging Lösungen für alle Protokolle der relevanten IT-Plattformen entwickelt. Die Metro Group entschied sich für dieses Produkt, weil ähnliche Lösungen anderer Hersteller dieses Ziel nur durch die Integration mehrerer Systeme für jeden Medientyp erzielten. TOPCALL läuft auf einem hoch skalierbaren Server, die TOC[70] ist daher wesentlich geringer als bei anderen Konzepten. Obwohl die Metro TOPCALL nur für das Empfangen und Versenden von Faxen benötigt, ist sie so für die Zukunft bestens gerüstet.

3.1.1 Die Funktionsweise von TOPCALL

Für den Aufbau einer TOPCALL-Lösung werden mindestens drei Serverkomponenten benötigt (vgl. Abbildung 21).

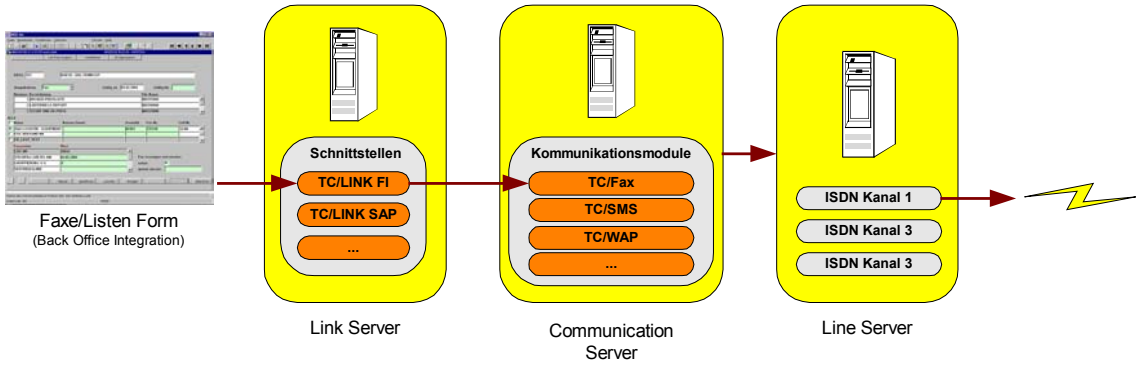

Abbildung 21 TOPCALL – Anbindung an die Faxe/Listen Form

- Der **Link Server** verbindet Geschäftsprozesse (SAP, Lotus Notes, Microsoft Exchange...) durch optional erhältliche TC/LINK-Schnittstellen mit einem TOPCALL Server. In der Metro findet derzeit nur die TC/Link-FI Schnittstelle[71] Verwendung. Sie kann bereits mit heutzutage minimalen

[69] Archivsysteme werden häufig mit Jukebox-Servern realisiert, die große Bestände von gebrannten CDs verwalten können
[70] Total Cost of Ownership bez. etwa nicht die Anschaffungskosten einer Lösung, sondern auch die anfallenden Gesamtkosten. Wie gut ein Serversystem wirklich ist, erkennt man daran, dass es sich langfristig rechnet.
[71] siehe Kapitel "Die TC/LINK-FI Schnittstelle"

Ressourcenanforderungen (Pentium I 133Mhz, 64MB RAM) unter Windows NT betrieben werden

- Der **Communication Server** ist das Herzstück des Systems, und verwaltet die gesamte Businesslogik zur Kommunikation. Hier lassen sich derzeit alle von TOPCALL möglichen Kommunikationsmedien (wie z.B. Fax, SMS, WAP u.v.m.) durch Module integrieren, die später von einer Geschäftslogik mit Daten über einen Link Server gespeist werden. Die Module bieten spezielle Aufgabe wie Text-To-Speech, Dokument- und Ausgabekonvertierungen.

- Der **Line Server** stellt die Kommunikationskanäle (Analog, ISDN oder Standleitung) zur Verfügung. Erst über diese Schnittstelle ist es später möglich mit der Außenwelt oder mit einem Netzwerk zu kommunizieren. (Sollte die Performance einmal unter der Datensatzrate leiden, so ist es möglich das System auf insg. 180 Voice/Fax/Telexkanäle zu erweitern.). Ein Line Server kann direkt vor Ort bei einer Niederlassung stehen, und durch ein leistungsfähiges Backbone Netzwerk innerhalb der Metro mit dem zentralen Fax-System in Düsseldorf verbunden werden. Somit können die Kosten für die einzelnen Stores getrennt voneinander abgerechnet werden, wenn für jeden Store zusätzlich auch eine eigener Link-Server[72] eingesetzt wird.

In der Metro gibt es jeweils ein Produktiv- und ein Testsystem von TOPCALL. Dafür sind keinesfalls mehrere Rechner nötig; es reicht wenn bereits mehrere Schnittstellen vom gleichen Typ auf dem Link Server installiert werden. Für die TC/Link-FI Schnittstelle müssen lediglich neue Verzeichnisse angelegt werden.

3.1.2 TC/LINK-FI Schnittstelle

TC/LINK-FI (FI = „File Interface“) ist eine Dateischnittstelle zum TOPCALL System. Sie ist eine Universallösung zur Anbindung fremder Back Office Anwendungen, für die

[72] in der Praxis erfolgt die Einsparung eines Link-Servers. Eine Konfiguration sog. Link-Verzeichnisse für jeden Store auf einen Link-Server erfüllt ebenfalls den gleichen Zweck.

es derzeit von TOPCALL noch keine entwickelten Standard TC/Link Module gibt. Für die hauseigene Entwicklung der Faxe/Listen-Form in der Metro ist diese Schnittstelle derzeit die einzige Lösung, einen Nachrichtenfluss an das TOPCALL System zu leiten. Die gesamte Kontroll- und Businesslogik zwischen der Faxe/Listen-Form und dieser Schnittstelle übernehmen die neu zu erstellenden Fax-Dämonen in Java.

3.1.2.1 Funktionsablauf

Der Benutzer erstellt in seiner Anwendung ein Dokument (z.B. ein Report in PDF Form) inkl. einer Transaktionsdatei, die alle Adressinformationen für mögliche Empfänger enthält TOPCALL zur Verfügung. Durch ein einfaches Laufwerksmapping mit der TC/LINK-FI Schnittstelle lassen sich diese zwei Dateien nun von einer Back Office Anwendung (Faxe/Listen-Form) zum TOPCALL System transferieren[73]. Das System konvertiert nun das Dokument in ein entsprechendes Ausgabeformat für die Kommunikationsmedien (Fax, Email, MMS...) und versendet dies mittels einer Transaktionsdatei. Eine entsprechende Empfangsbestätigung kann entweder per E-Mail am Client erfolgen oder mit einer Benachrichtigungsdatei (Rückmeldedatei), die wieder auf die TC/LINK-FI Schnittstelle vom Line Server zurückgeschrieben wird. Optional lassen sich beliebig viele TC/LINK Module an das System anbinden, die jeweils durch ein eigenes Verzeichnis angesteuert werden können.

Die Schnittstelle arbeitet pro Installation mit vier spezifischen Verzeichnissen:

- Systemmeldungen werden in ein `FI_TO_TC` Verzeichnis angelegt
- Rückmeldedateien zu Vorgängen gelangen in das `NOTIF` Verzeichnis
- Neue Steuer- und Transaktionsdateien einer Back Office Anwendungen müssen in `TC_TO_FI` abgelegt werden.
- `DIRSYNC` enthält alle Dokumente die von TOPCALL noch versendet werden müssen. Sollte im Communication Server ein Fax-Modul zu TC/LINK-FI konfiguriert sein, so wird ein Dokument in diesem Verzeichnis automatisch in ein Standardgrafikformat (TIFF, BMP, PCX ...) umgewandelt. Auf diese Weise lässt sich jedes Dokument als ein Fax versenden.

[73] bez. einen Prozess des Kopierens o. Verschiebens von Dateien

Die Verzeichnisse *DIRSYNC* und *TC_TO_FI* lassen sich optional zu einem zusammenfügen, ebenso *FI_TO_TC* und *NOTIF*. Hierdurch erreicht man eine Vereinfachung, und muss seine Back Office Module nicht auf mehreren Verzeichnisebenen arbeiten lassen. Diese Einstellungen gelten auch für die neu zu erstellenden Fax-Dämonen.

Transaktionsdateien können mehrere Empfänger für ein Dokument enthalten. Im folgenden werden die zwei Transaktionsformate

- TOPCALL Open Message Format (TOM)
- TOPCALL XML Message Format (TC/XML)

für die TC-LINK/FI Schnittstelle beschrieben. Eine Vermischung der zwei Formate ist zwischen Transaktions- und Rückgabedateien nicht möglich.

3.1.3 Das klassische TOM Format

Das TOM - TOPCALL Open Message Format gilt als Standardformat und lässt sich ohne große Mühe in jede Backoffice Anwendung einbauen, weil es auf einer einfachen Textebene basiert. Auch die alten Forms-Fax-Dämonen arbeiteten mit diesem Format. Listing 12 zeigt den Aufbau einer Steuerdatei für drei Empfänger die durch das Attribut „TO“ gekennzeichnet werden [vgl. Parameter N, Zeile 2-4]. Zu jedem dieser Empfänger wird ein Vorgang von der Faxe/Listen-Form zugewiesen [vgl. Parameter C3, Zeile 2-4] und natürlich ein generierter Report in Form einer PDF-Datei, der bereits im *TC_TO_FI* Verzeichnis existiert. Dieser wird durch den Parameter *ATT* [Zeile 7] identifiziert. Neben der Sollsendezeit eines Faxes, die durch *TIME* und *DATE* gekennzeichnet werden, sind alle restlichen Parameter TOPCALL spezifisch. Sie bleiben bei jedem Transaktionsvorgang identisch.

```
1  TIME=144932,DATE=19990331
2  TO:C3=200,SE=FAX,N=02119694903538
3  TO:C3=200,SE=FAX,N=02119694902338
4  TO:C3=200,SE=FAX,N=02119694903271
5  FROM:SN=oratopcall,SE=TOPCALL,N=oratopcall
6  TXT:ATT:NA=200_2.PD,APPL=200_2.PDF
```

Listing 12 Transaktionsdatei im TOM Format

Obwohl eine Transaktionsdatei im TOM Format schnell und einfach generiert werden kann, erweist sich dies leider als kleiner Nachteil in der Rückmeldedatei, die ebenfalls das gleiche Format haben muss. Listing 13 zeigt eine erfolgreiche Rückmeldungsdatei zu einem Empfänger mit der Telefonnummer *0211969490XXXX*. Später werden die enthaltenden spezifische Parameter vom Fax-Dämon BD314 geparst[74] und einem Vorgang in der Fax-Datenbank der Faxe/Listen-Form zugeordnet.

```
1   SUBJECT="TOPCALL delivery notification: ", TYPE=NOTIF, MCORR=00000035981
2   FROM: ACTIVE=YES, SERVICE=FAX, NUMBER=0211969490XXXX, ACTIVE=YES
3   TO: ACTIVE=YES, P=NORM, NF=NO, ARCHIVE=NO, RESOLUTION=NORM, SCOPY=YES, HLINE=YES, CCTR=2617,
    REMMSG=NO, C1=00008CA1, C2=00000012, SERVICE=TCFI, NUMBER=oratopcall, ACTIVE=YES
4   NFINFO: STATUS=DEL, TIME=990330:183700, LNOTE=492119694902338, COST=    48, MCORR=00000035981,
    C3=200
5   TXT:
6   TOPCALL Delivery Notification
7   ------------------------------------------------------------
8   Message 00000035981 sent to  02119694902338,-492119694902338
9   Time sent: 99-03-30 18:37:00
10  Subject:
11  Number of pages: 10
12  Costs:     48 for Costcenter 2617
13  ------------------------------------------------------------
```

Listing 13 Rückmeldedatei im TOM Format

Das wichtigste Attribut einer Rückmeldedatei ist sicherlich das *STATUS* Feld [Listing 13, Zeile 4]. Es zeigt durch die Werte *DEL* (delivered) oder *NONDEL* (non delivered) an, ob ein Vorgang in der Faxe/Listen-Form erfolgreich versendet wurde.

In Java lassen sich Textdateien mit Hilfe von String-Funktionen auslesen. Oft findet die Methode *substring()* hierfür Verwendung. Sie benötigt immer einen Start- und einen Zielpunkt, um ein Ergebnis zwischen diesen Punkten zu liefern. Nachstehende Abbildung 22 verdeutlicht in dieser Hinsicht die Probleme bei Verwendung des TOM Formates. Im Richtig-Beispiel ließe sich der Parameter *STATUS* durch den Startpunkt *STATUS=* und den Stoppunkt *TIME=* identifizieren. Leider kann nicht immer eindeutig gewährleistet sein, ob zwischen dem Attribut *STATUS* bzw. *TIME* nicht noch ein weiteres existiert.

[74] bez. ein gezieltes Auslesen nach Informationen

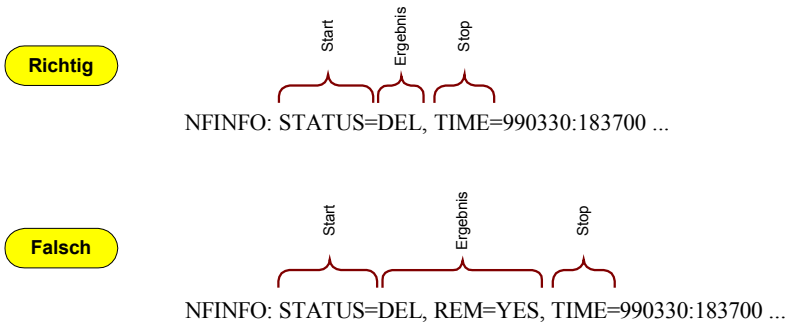

Abbildung 22 Ausleseprobleme in Java mit Substring()

Im „Falsch"-Beispiel erhält man nach dem gleichen Procedere plötzlich als Ergebnis: `DEL, REM`[75]`=YES`, obwohl hier auch der Wert `DEL` enthalten ist, lässt dieses Ergebnis später nicht mehr eindeutig von den Fax-Dämonen weiterverarbeiten, weil zwischen Start- und Stoppunkt noch ein zusätzliches REM-Attribut von TOPCALL hinzugekommen ist. Der Systemverwalter des TOPCALL-Servers optimierte in diesem Fall die Konfiguration des Systems, dies hatte zur Folge das noch andere Parameter außer REM an einer willkürlichen Stelle auftauchen können. Handbücher von TOPCALL geben leider keinen Aufschluss darüber, an welcher Stelle sich mögliche Wertepaare in der Rückmeldedatei befinden können. TOPCALL teilte auf der CeBit 2004 mit, dass dieses Problem erkannt wurde, und ausführlicher in den nächsten Handbüchern beschrieben wird. Bevor nun weitere Tests mit dem TOM-Format der Fax-Dämon erfolgten, musste zuerst sichergestellt werden, dass auch das Testsystem und Produktivsystem von TOPCALL in der Metro mit gleichen Konfigurationen laufen. Obwohl die alten Forms-Dämonen zuverlässig ihren Dienst mit dem klassischen TOM-Format tätigten, erschein diese Lösung auf Dauer eher ungeeignet und unzuverlässig zu sein. Recherchen aus den TOPCALL-Handbüchern ergaben, dass TC/LINK-FI auch eine XML-Darstellung unterstützt.

3.1.4 Das TC/XML Format

XML „eXtensible Markup Language" ist zu einer bedeutenden grundlegenden Technologie für die Standardisierung von Daten- und Dokumentationsformaten

[75] REM(msg) ist ein genereller Message-Parameter von TOPCALL (vgl. Datei TOM_20604.PDF – S. 23 „General Message Parameters" auf der CD im Anhang). Jedes Email-Postfach eines Mitarbeiter kann durch eine persönliche Faxnummer Faxe empfangen, diese werden als Bilddatei einer Email angefügt. Durch Aktivierung dieses Parameters lassen sich temporär generierte Emails von TOPCALL nach einem erfolgreichen Versand direkt auf dem System löschen, dies spart Speicherressourcen auf einer Festplatte.

geworden. XML 1.0 wurde erstmals Februar 1998 vom W3C[76] verabschiedet. Weil XML sowohl im Browser angezeigt werden kann, als auch für die Kodierung von Objekten benutzt wird, findet es immer häufiger in Multi-Tier-Anwendungen[77] Verwendung. Auf dem Client wird es zur Anzeige und einfachen Verarbeitung von Dateninhalten verwendet. In der Mittleren Schicht wird es zur Serialisierung und zum Austausch von Daten angewendet, so auch bei der TC/Link-FI Schnittstelle. Listing 14 zeigt analog zum TOM-Format einen Auszug aus einer Rückmeldedatei im XML-Format.

```
1   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2   <NOTIFICATION xmlns=http://www.topcall.com/XMLSchema/2002/tc/xml>
3   <SUBJECT>TOPCALL delivery notification_ TOPCALL Open Message Format description</SUBJECT>
4   <FROM>
5   <ACTIVE>YES</ACTIVE>
6   <P>HIGH</P>
7   <SERVICE>FAXB</SERVICE>
8   <NUMBER>0219173366</NUMBER>
9   </FROM>
10  ….
11  </NOTIFICATION>
```

Listing 14 TC/Link-FI XML-Layout

Die zentrale Informationseinheit von XML ist immer ein XML-Dokument, dass mehrere Elemente sog. XML-Tags enthält. In obiger Abbildung ist z.B. *<NUMBER>* ein solches Tag[78], dass die Telefonnummer eines Lieferanten beschreibt. Die Informationen in den einzelnen Tags können später mit XML-Parsern in einer Baumstruktur[79] ausgelesen werden. Wie jedes andere Stück Software hat auch ein XML-Dokument einen physikalischen Inhalt und eine logische Struktur. In der Praxis ist es aber oft schwer über eine logische Struktur zu reden, ohne physikalische Einheiten wie Zeichen, Strings und Daten zu erwähnen. Ein allgemeiner Name für all diese Dinge ist in der XML-Spezifikation[80] unter „Entitiy“ zu finden.

[76] Die Standards für den korrekten Aufbau von Internetseiten werden vom W3-Consortium verabschiedet
[77] trennen die Präsentations-, Applikations- und Datenbankschicht. Vgl. hierzu Kapitel 2.1 „Oracle Forms“
[78] ein Tag bez. in XML einen Knoten der Daten enthält. Dieser kann wiederum durch andere Knoten erweitert werden.
[79] vgl. dazu Abbildung 21
[80] s. Extensible Markup Language (XML) 1.0 (Third Edition) <http://www.w3.org/TR/2004/REC-xml-20040204/> (02.03.2004)

Entities lassen sich zu jedem XML Dokument optional in DTD „Document Type Definition“ bzw. „Document Type Declaration“ Dateien beschreiben[81]. In ihnen wird definiert, wie die XML-Tags in der XML-Datei verwendet werden, und an welcher Stelle diese zu finden sind. In der DTD werden quasi die XML-Tags deklariert und ein XML-Parser prüft, ob die Tags in der XML-Datei mit der Definiton in der DTD übereinstimmen. Stimmen diese nicht überein, wird ein Fehler gemeldet, ansonsten ist die XML-Datei korrekt. Wird eine DTD verwendet, spricht man von einer gültigen XML-Datei (valid). Liegt keine DTD vor, spricht man von einer wohl-geformten XML-Datei (well-formed).

Sowohl für die Transaktionsdateien und die Rückgabedateien von der TC/Link-FI Schnittstelle gab es keine DTD Dateien die TOPCALL zur Verfügung stellen könnte. Lediglich einige Anwendungsbeispiele aus dem Handbüchern lieferten Aufschluss über den möglichen Aufbau dieser XML-Dateien. Der Gedanke selbst eine DTD-Datei aufgrund dieser Informationen zu schreiben entfiel schnell, da eine mögliche Validierung der XML-Rückgabedateien im Parser durch nicht bekannte neu hinzugekommene Attribute wie `REM` fehlschlagen könnte, und die Fax-Dämonen sich auf diese Weise beenden.

Welchen Nutzen brachte also das neue XML-Format im Gegensatz zum TOM-Format?

Bei XML ist es unerheblich welche Tags (vgl. Abb. 23 graue Knoten) zwischen den anderen neu hinzukommen. In Abbildung 23 ist zu erkennen, dass ab dem Root-Element[82] der Pfad zum Tag `NUMBER` eindeutig traversiert werden kann (roter Pfad, bzw. gelbe Knoten).

[81] Tools wie „XML Spy“ <http://www.altova.com/download.html> (23.01.2004) können autom. ein DTD zu einem XML Dokument generieren

[82] Ein Root Element ist der Anfangspunkt jeder XML-Traversierung

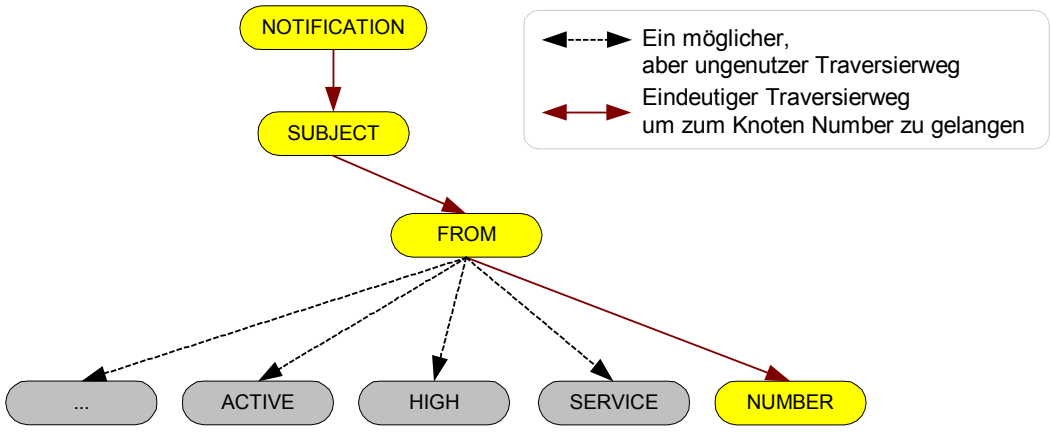

Abbildung 23 Baumstruktur einer TC/Link-FI Rückgabedatei im XML Format

Sollten neue Attribute z.B. ab dem Knoten `FROM` durch eine Konfigurationsänderung des TOPCALL-Servers neu hinzugekommen, so bleibt der Weg bis zum Knoten `NUMBER` immer noch der Selbe.

Für die neu zu programmierenden Fax-Dämonen begann nun die „Qual der Wahl“[83] nach der Suche eines geeigneten XML-Parsers in Java. Es gibt mindestens ein Dutzend nichtvalidierender Parser und ungefähr halb so viele validierende. Generell unterscheidet man zwischen zwei Arten von Parsern:

- **SAX-Parser** sind ereignisorientiert. Sie können beim Leseprozess eines XML-Dokuments über bestimmte Ereignisse benachrichtigen. Z.B ob eine Zeichenkette zwischen zwei Elementen gefunden wurde usw. SAX liefert eine Fülle von Standardmethoden für Rückruf-Funktionen, die von diesen Ereignissen ausgelöst werden können. Sollte SAX beim parsen auf Tags stoßen, die später benutzt werden sollen, so muss eine beständige Referenz darauf gespeichert werden, weil sie noch kein Bestandteil irgendeiner Struktur sind. Bei SAX-Parsern gibt es weder Eltern- noch Kinderknoten.

- **DOM-Parser** liefern Standardschnittstellen für die Datenstruktur von Dokumenten, die aus dem Parsen resultiert.

[83] XML Magazin Ausgabe 6.02, S. 78 „Übersicht XML-Parser“ – zeigt derzeit sämtliche verfügbaren Parser inkl. der Downloadadressen im Internet

Ein anschaulicher Vergleich dieser zwei Parsertypen lässt sich wie folgt beschreiben:

Stellen Sie sich vor, Sie wollen den Listenpreis von jedem Taschenbuch aus Ihrem Buchbestand um 10% senken. Mit SAX sagen Sie etwa Folgendes: >>Reduziere für jedes Buchelement, das vorbeikommt und dessen Typattribut den Wert >>Taschenbuch<< hat, den Wert des nächsten Preiselements um 10%.<< Mit DOM sagen Sie dagegen: >>Reduziere für jedes Buchelement, das ein Nachfahre der Wurzel ist, den Wert seines Kind-Preiselements um 10%, falls sein Typattribut den Wert >>Taschenbuch<< hat.[84]

Die TC/Link-FI Schnittstelle arbeitet mit einer eindeutigen Datenstruktur, d.h. sie enthält Knoten in Form einer geordneten Baumstruktur. Aus diesem Grunde machte es Sinn einen DOM orientierten Parser *JDOM*[85] im Fax-Dämon BD314 verwendet, der nun ohne Probleme Rückgabedateien nach Attributen parsen kann. Dieser Parser ermöglicht zugleich auch eine Validierung, die leider wegen fehlender DTD abgeschaltet werden musste. Sollte TOPCALL diese in nächster Zeit zur Verfügung stellen, so ist es ohne großen Zeitaufwand möglich das Feature wieder einzuschalten.

Zu guter Letzt sei noch angemerkt, das die TC/Link-FI Schnittstelle jede XML-Transaktionsdatei automatisch in ein internes XSLT[86]-Dokument umwandelt und anschließend verarbeitet. Theoretisch ließe sich so ein kleiner Geschwindigkeitsvorteil erreichen wenn der Fax-Dämon BD314 dies schon erledigen würde. Dieser Minimale Performancegewinn wird aber durch das Kopieren[87] von Dateien zum TOPCALL-Server wieder zunichte gemacht, weshalb von diesem Feature abgesehen wurde.

Transformierungen dieser Art arbeiten immer nach dem SAX-Parser Prinzip. Obwohl diese Dateien in Systemverzeichnissen des TOPCALL-Server zu finden waren, ist es dennoch nicht möglich von einer XSLT-Datei auf den Aufbau einer XML Datei zu schließen.

[84] s. “Java XML Programming” von Alexander Nakhimovsky und Tom Myers – S. 255
[85] download <www.jdom.org> (14.12.2004)
[86] XSLT steht fuer „Extensible Stylesheet Language for Transformation“. XSLT beschreibt ein Vokabular für die Transformation von XML-Dokumenten.
<http://www.fase4.com/de/tutorials/page,1,xslt.xml#i__138745856_132> (14.12.2004)
[87] vgl. dazu Kapitel 4 “MGI Fax-Basisdienst BD300”

3.2 Die Architektur des Fax-Basisdienstes

Bevor die neuen Fax-Dämonen produktiv zum Einsatz kommen konnten, musste zuerst die Systemarchitektur grundlegend überdacht werden. Abbildung 24 zeigt den kompletten Aufbau des alten Fax-Basisdienstes. Als erstes ist auffallend, dass kein Web Cache benutzt wurde, d.h. ein Anwender, der über das Intranet die Faxe/Listen-Form benutzen möchte, landet immer auf demselben Fax-Server. Dieser basierte noch auf einem herkömmlichen Windows System und funktionierte noch nach dem alten Client/Server Schema. Schedulingüberwachung der Forms-Dämonen BD313/BD314 stellte somit keinerlei Probleme dar.

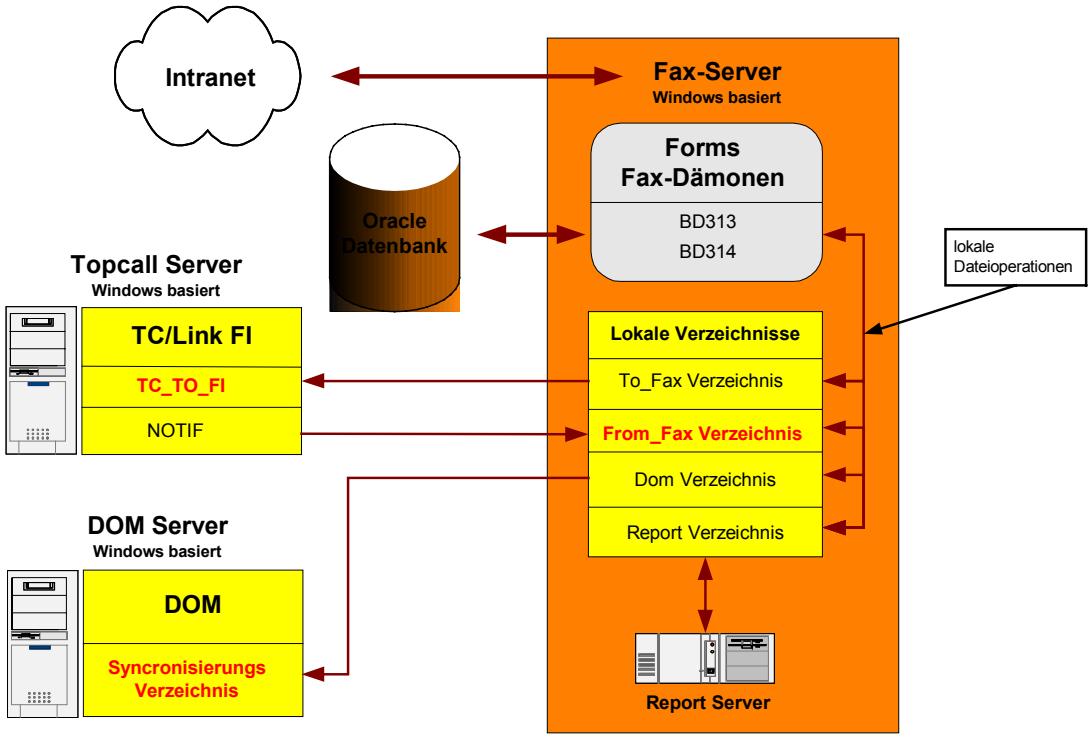

Abbildung 24 Alte Systemarchitektur des Fax-Basisdienstes

Sowohl der DOM Archivierungsserver als auch der TOPCALL-Server mussten nun mit dem Fax-Server verbunden sein. Problemlos konnte dies damals mit einem Laufwerksmapping[88] unter Windows erreicht werden, weil alle Systeme mit dem gleichen Betriebssystem arbeiteten. Alle rot markierten Verzeichnisse sind „gemappte“

[88] unter MS-Windows lassen sich Verzeichnisse oder ganze Laufwerke von einem Server auf einen anderen Client abbilden. Obwohl diese gar nicht physikalisch auf einem Client existieren, können Dateien oder Verzeichnisse soweit eine Berechtigung vergeben ist, wie lokale Dateien verwendet werden.

Laufwerke eines anderen Systems. Strategisch gesehen ist das Mappingverfahren eine gute Lösung gewesen, weil es fest im Betriebssystem verankert ist, zusätzliches Aufspielen einer Software oder eines Treibers entfällt somit völlig. Mitarbeiter von systemkritischen Anwendungen sind hier äußerst sensibel und möchten keinerlei möglicherweise „instabile Zusatzlösungen“ auf einem Serversystem installiert haben. Sollte dennoch ein Problem auftreten, so kann ggf. Hilfestellung von Microsoft durch Supportverträge gewährleistet werden. Diese durchaus einfache Lösung lief bis zum heutigen Tage ohne größere Probleme zuverlässig. Das neue Konzept der Java-Fax-Dämonen sieht von der Architektur aber völlig anders aus (vgl. Abb. 25).

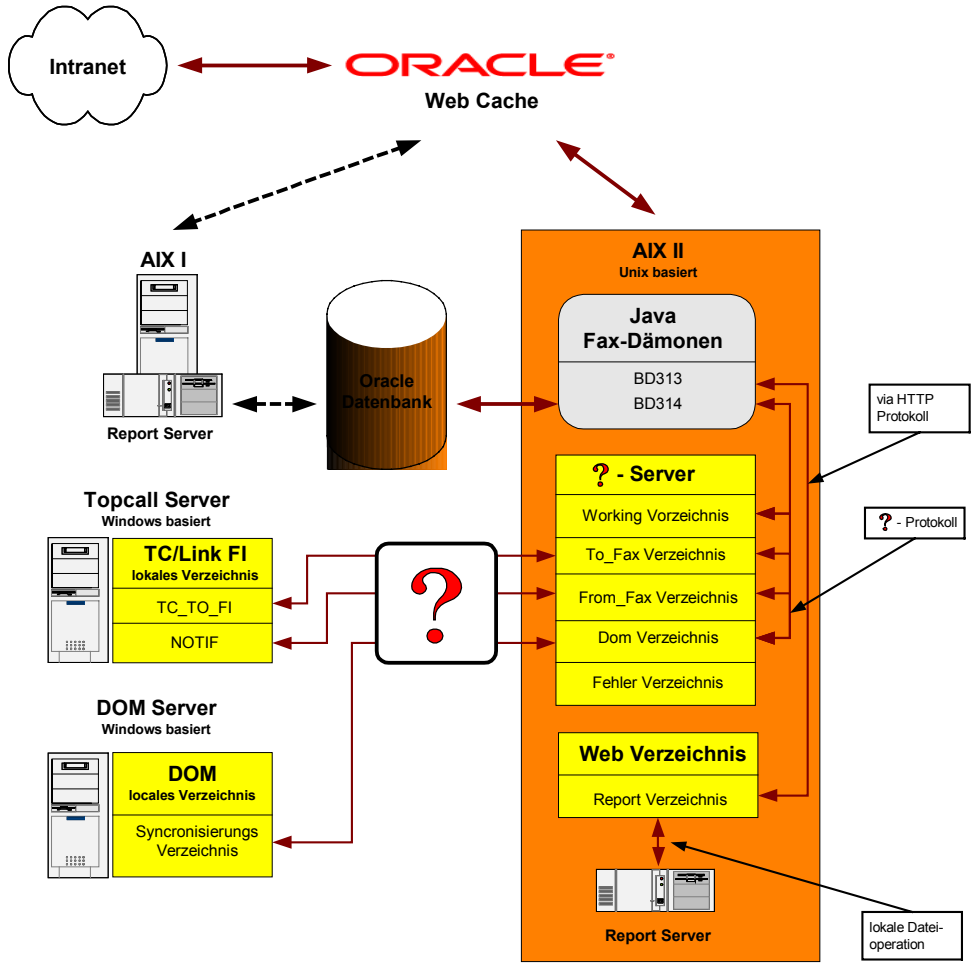

Abbildung 25 Neue Systemarchitektur des Fax-Basisdienstes

Mitarbeiter der Metro erkannten schnell, das ein Oracle Web Cache neben anderen webbasierten MGI-Anwendungen auch sinnvoll für Faxe/Listen-Form erscheint. Eine Anfragelast wird somit auf einzelne AIX-Rechner verteilt. Die AIX-Server haben leistungsstarke Prozessoren, sind mit hochsensiblen Daten unterwegs und laufen

ausschließlich unter dem Betriebssystem UNIX. Einer dieser AIX-Systeme wurde nun auserwählt um die neuen Java-Fax-Dämonen zu verwalten[89]. Die gesamte Businesslogik dieser neuen Dämonen blieb in ihren Grundzügen erhalten, Verzeichnisstrukturen blieben bis auf die temporäre Arbeitsverzeichnisse Error-Dir, Working-Dir gleich. Ein große rotes Fragezeichen in Abbildung 25 kennzeichnet nun ein großes Problem. Die Betriebssysteme sind nun von völlig unterschiedlicher Natur.

Mit welcher Technologie sollte der TOPCALL- inkl. des Archivierungs-Servers an den Fax-Basisdienst angekoppelt werden?

Aus Sicherheitsgründen war es auch zusätzlich nicht mehr möglich die Fax-Dämonen vorerst mit lokalen Dateioperationen direkt auf dem AIX-System arbeiten zu lassen; lediglich der Zugriff auf das Reportsverzeichnis das alle generierten Faxe enthält konnte durch einen einfachen HTTP-Protokoll Zugriff[90] geschehen. Um Manipulationen auszuschließen, müssen alle Verbindungen mit einen Passwortschutz versehen sein.

Zukünftig sollte später zu jedem Zeitpunkt die Möglichkeit bestehen, ohne großen Aufwand, die Fax-Dämonen und/oder die Verzeichnisse auf unterschiedlichen Rechnerarchitekturen zu verteilen[91]. Nachstehende Unterkapitel beschreiben nun mögliche „Mapping“-Szenarien. Alleine diese Konzeptplanung verschlang durch ständige Rücksprache mit Systemoperatoren und Verantwortlichen aller betroffenen Abteilungen gut 1/3 der Diplomarbeitszeit.

3.2.1 Network File System

Das Network File System (NFS) ist eine Entwicklung von Sun. Die erste Version erschien im Jahre 1985. Mittlerweile sind zwei Nachfolgeversionen veröffentlicht worden (1989 und 1995). 1998 übergab SUN alle Rechte an dem NFS Protokoll an die ISOC[92] zur Weiterentwicklung. NFS kommt aus der UNIX/Linux Welt und erlaubt das Verwalten von Dateien auf mehreren entfernten Hosts innerhalb eines Netzwerkes so als

[89] Vgl. zusätzlich Kapitel 4.6 „Ressourcenengpass“
[90] für HTTP-Zugriffe ist es notwendig unter UNIX ein Webverzeichnis zur Verfügung zu stellen. Lesezugriffe können dann dank des HTTP-Protokolls analog wie in einem Web-Browser geschehen.
[91] dank dieser Zusatzanforderungen konnte später auch das auftretende Ressourcenproblem s. Kapitel 4.2 „Ressourcenenpass“ behoben werden.
[92] Internet Society <http://www.isoc.org> (07.01.2004)

ob sie auf der lokalen Festplatte gespeichert wären. Für den Benutzer unterscheiden sich entfernte über NFS eingehängte Dateisysteme nicht von lokalen Dateisystemen. Dadurch ist es nicht notwendig zu wissen, wo die Daten tatsächlich physikalisch abgelegt werden.

NFS ist in der Lage, Homeverzeichnisse von Benutzern auf entfernten Servern anzulegen und diese dann nach dem Einloggen auf einem Client, wie jedes andere Filesystem, zu mounten. Ein Zugriff auf die persönlichen Daten kann danach über Laufwerkmapping vorgenommen werden. NFS läst sich relativ einfach konfigurieren und ist zudem noch ausgereifter und z.T. performanter als SAMBA[93]. Ein weiterer Vorteil ist der gemeinsame Dateizugriff. Mehrere Clients können von unterschiedlichen Rechnern aus auf dieselbe Datei, die sich auf einen NFS-Server befindet, zugreifen. Sollte eine spätere Aufsplittung der Fax-Dämonen (BD313/BD314) auf verschiedenen Rechnern gewünscht sein, so ist es möglich, explizit eine Sperre auf eine Datei zu vergeben, solange diese von einem Dämonen abgearbeitet und danach wieder freigegeben wird.

Ein großer Nachteil dieses Protokolls aber ist es, das eine Authentifizierung ausschließlich über den angemeldeten Benutzer des Rechners erfolgt. Eine Passwortvergabe durch die Fax-Dämonen - unabhängig vom eingeloggten Benutzer - auf dem Server ist nicht möglich. Weil FAX-Basisdienste aber später als Systemprozesse (Dämonen) auf einen Server läufen müssen, scheiterte dieser Lösungsansatz schon im Kern. Anschließende Auskünfte beim UHD[94] zeigten, dass NFS nicht strategisch von der Metro AG eingesetzt wird.

Derzeit existieren noch keine guten und zuverlässigen Windows-Implementationen, die für eine Anbindung des Archivierungssystems „DOM“ und des Kommunikationsservers TOPCALL notwendig wären.

[93] vgl. Kapitel 3.2.2 „Server Message Block Protokoll“
[94] UHD – “User Help Desk”, bietet einen allumfassenden IT-Support für alle Anwender der Metro Group

3.2.2 Server Message Block Protokoll

Das SMB Protokoll ist das Standardprotokoll bei Microsoft Windows und IBM OS/2 Rechnern. SMB ist mit vielen darunter liegenden Protokollen implementiert worden. Dies umfasst XNS, NBT, IPX, NetBEUI und TCP/IP. Es legt fest, wie Windows Computer im Netzwerk kommunizieren. SMB ermöglicht den Zugriff auf Festplatten, Drucker, serielle Schnittstellen und Kommunikationskanäle. Im Gegensatz zu anderen Lösungen wie NFS wird hierbei auf den Client-Rechnern keinerlei Software installiert. Die Administration beschränkt sich daher auf die Installation der SAMBA-Software auf den UNIX-Host und die Erstellung und Pflege eines Konfigurationsfiles, welches den Zugriff auf den SAMBA-Server regelt.

Samba[95] ist nun ein Paket von Programmen die das SMB Protokoll benutzen, somit können auch UNIX Rechner über das Microsofteigene Protokoll NETBIOS mit anderen Rechnern "sprechen". Die von UNIX-Host aus freigegebenen Ressourcen erscheinen dem Anwender transparent in der "Netzwerkumgebung" und können nun auf beliebige Laufwerksbuchstaben gemappt werden.

Samba wurde unter den GPL[96] veröffentlicht und ist damit kostenlos und im Quelltext erhältlich, häufig gilt es als bestes Mittel der Wahl um Unix bzw. Linux mit Microsoft Windows zu verbinden, weil es inzwischen sehr ausgereift ist und es ständig neue Updates und Versionen im Internet gibt. Eine zusätzliche Java-Bibliothek für SMB aufzusuchen entfällt, weil bereits die Java-SDK 1.3 entsprechende APIs für den lokalen Dateizugriff bereitstellt. Eine entfernte Festplatte auf einen Remote-Host lässt sich über ein virtuelles Laufwerksmapping wie eine Lokale ansprechen. Für einfache Dateioperationen ließe sich das `Java.io.*` Package im SDK nutzen.

Eigentlich die besten Voraussetzungen, um es auch in der MGI für den Fax-Basisdienst einzusetzen. Leider darf dieses Protokoll nicht genutzt werden, da es im Problemfall keinerlei Support gibt. Fälschlicherweise wird SAMBA auch oft in diesem Zusammenhang als Bastlerprodukt für Linux-Freaks angesehen.

---

[95] download <http://samba.org> (09.01.2004)
[96] General Public License. Lizenz für den Vertrieb von Open-Source Software. Betroffene Programme werden vom Händler gegen Ersatz der Vertriebs- und Supportkosten vertrieben. Die eigentliche Software ist kostenlos und wird samt des Quellcodes zur Verfügung gestellt

3.2.3 File Transfer Protokoll

Das File-Transfer-Protocol kurz FTP, wurde 1985 definiert und findet noch millionenfach Anwendung. Neben E-Mail ist FTP eines der ältesten und am meisten genutzten Dienstprotokolle im Internet. FTP ist ein Standard zur Übertragung von Dateien. Dieses Protokoll wird aufgrund seiner hohen Performance häufig genutzt, um mehrere und vor allem größere Datenmengen mit Hilfe eines FTP-Clients auf einen Internet-Server heraufzuladen, oder von diesem herunterzuladen. Die Kommunikation zwischen einen Client und einem FTP-Server läuft über immer über 2 Ports[97] ab:

- Datenkanal – Kanal für die Datenpakete einer Datei
- Befehlskanal – empfängt und sendet Kommandos für die nächste Operation

Um Daten zu versenden, öffnet der Client in einer FTP-Sitzung eine Kontrollverbindung, bestehend aus einem Befehlskanal und einem Datenkanal zu einem Server, die während der gesamten Sitzungsdauer bestehen bleibt. Über diese Kontrollverbindung werden Befehle und Antworten zwischen Client und Server übertragen. In der Metro ist FTP leider nicht anzutreffen, und wird auch nicht strategisch verfolgt. Gründe liegen insbesondere in der IT-Sicherheit: Über einen Befehlskanal eines FTP-Servers lassen sich nicht nur FTP-spezifische Kommandos absetzen, sondern auch Systemkommandos, die es einen Hacker ermöglichen, Information über die interne Netzwerkstruktur der Metro zu sammeln, um somit später einen gezielten Angriff auf mögliche Schwachstellen im System vorzunehmen.

3.2.4 Hypertext Transfer Protokoll

Das Hypertext Transfer Protokoll (HTTP) ist ein Kommunikationsprotokoll, das speziell für den Zweck entwickelt wurde, Multimedia-Dokumente (HTML-Seiten genannt) zwischen WWW[98]-Servern und WWW-Clients auszutauschen. Am häufigsten wird es für die Übermittlung von Daten im Internet verwendet. Es kann ohne jede Einschränkung beliebige Dateiformate übertragen. Strategisch gesehen ist dieses

[97] universale Kommunktions-Schnittstelle zu einem Anwenderprogramm
[98] World Wide Web = Internet

Protokoll in der Metro erlaubt und Web Server[99], die eine Kommunikation mit HTTP zulassen, existieren bereits. Die Java-Fax-Dämonen könnten somit nach einer Freigabe eines Webverzeichnisses auf diesem von einer zentralen Stelle aus operieren. Leider entfiel auch diese Lösung, nachdem die RFC[100] des HTTP-Protokolls genauer studiert wurden. Das Protokoll sendet nach jeder Operation (wie z.B. anlegen, kopieren) einen Status an den Client zurück, aber speziell bei der DELETE-Operation ist dieser Status nicht aussagekräftig:

"The client cannot be guaranteed that the operation has been carried out, even if the status code returned from the origin server indicates that the action has been completed successfully. However, the server SHOULD NOT indicate success unless, at the time the response is given, it intends to delete the resource or move it to an inaccessible location."[101]

Die Java-Fax-Dämonen sollten aber in ihren Anforderungen zuverlässig und sicher ablaufen. Ein Status, der mit einem „OK" gekennzeichnet ist, obwohl ein Fehler aufgetreten sein könnte, war keine akzeptable Lösung. Obwohl auch das HTTP-Protokoll für das Abholen von Reports aus einen Web-Verzeichnis eingesetzt wird (s. Abb. 25), war es für diesen Zweck ausreichend, weil ein „Aufräumen" dieses nicht durch die DELETE Operation vorgenommen wird, sondern dieser Job in regelmässigen Abständen von Job-Scheduling Systemen übernommen wird. Standardmäßig kommt es auch eher selten vor, dass DELETE-Operation über das HTTP-Protokoll ausgeführt werden. Anwender, die im Internet arbeiten und Dateien austauschen, benutzen meist das schnellere FTP-Protokoll.

3.2.5 WebDAV Protokoll

Die Abkürzung DAV steht für „Distributed Authoring and Versioning". Mit dieser Technik sollte sich Ende 1998 das Publizieren von Internet-Auftritten vereinfachen. Das WebDAV Protokoll ist eine Erweiterung des HTTP-Protokolls und erlaubt Anwendern eine gemeinsame Nutzung von Dateiressourcen auf einem Server.

[99] Die MGI verwendet den Apache Web Server <http://www.apache.org> (20.01.2004)
[100] Request for Comments - RFCs sind eine Form von Diskussionspapieren, in denen technische Komponenten erschöpfend erklärt werden.
[101] vergleiche RFC <ftp://ftp.isi.edu/in-notes/rfc2616.txt> (20.01.2004)

Es enthält neben den normalen Funktionsumfang eines HTTP-Protokolls zusätzliche Methoden, die mit den Systemkommandos eines Unix-Systems vergleichbar sind:

- **MKCOL** - mkdir Verzeichnisse erstellen
- **PROPFIND** - ls, find Verzeichnisse, Dateien auflisten
- **PROPWATCH** - chmod, chtime Properties einer Datei ändern
- **COPY/MOVE** - cp, mv Dateien kopieren bzw. löschen
- **LOCK/UNLOCK**- flock, lockf Dateien sperren, entsperren

Obwohl das WebDAV Protokoll noch eine Fülle von zusätzliche Methoden zur Versionierung[102] zur Verfügung stellt, war diese Erfordernis für Businesslogik der Fax-Dämonen nicht notwendig; die Aufgabe der Archivierung übernimmt das systemeigene DOM System im Hause der Metro.

Der Entschluss die WebDAV Technologie zu benutzen erfolgte aus insgesamt vier Gründen:

- Es enthält sehr detailreiche Fehlerinformation, falls eine Dateioperation einmal fehlschlagen sollte. Im Gegensatz zum normalen HTTP-Protokoll erfolgt die Statuscodeausgabe eindeutig.
- Möglichkeit eines Passwortschutzes.
- Dank einer Erweiterung des HTTP-Protokolls muss kein zusätzlicher Port[103] in einer Firewall freigeschaltet werden
- Alle AIX-Systeme besitzen einen WebServer der von Hause aus die Funktionalität eines WebDAV-Servers in einem Modul zur Verfügung stellen kann.

Die Suche nach WebDAV-Server entfiel somit, lediglich für die Java-Dämonen müsste eine entsprechende API gefunden werden. Trotz der großen Masse an möglichen Client-Implementationen für unterschiedliche Programmiersprachen, zeigte sich ein nüchternes

[102] Die Versionierung ermöglicht die kompakte Verwaltung von unterschiedlichen Versionen einzelner Dateien bzw. vorgegebener Teile des Dateibaumes. Eigenständige Versionsverwaltungsprogramme werden oft in der Programmentwicklung eingesetzt.
[103] Ports sind Känale eines Programmes über die kommuniziert werden kann

Ergebnis an. Für Java sind derzeit leider nur zwei ausgereifte Bibliotheken im Netz verfügbar:

- **SLIDE**, ein Jakarta Slide Projekt, beinhaltet auch einen WebDAV Server <http://jakarta.apache.org/slide/index.html> (19.04.2004)
- **Websphere DAV for Java** - Eigenentwicklung von IBM Alphaworks <http://www.alphaworks.ibm.com/tech/DAV4J> (19.04.2004)

"Websphere DAV for Java" eignete sich als bester Lösungsansatz, weil es insbesondere von IBM auf einer Rechnerarchitektur getestet wurde, die den AIX Servern der Metro entspricht. Auch der gleiche WebDAV Server den IBM für seine Testzwecke auserwählt hatte, entsprach des gleichen Types wie in der Metro. Als WebDAV Server fungiert nun eine auserwählte AIX-Maschine die für alle Fax-Dämonen im Netzwerk zugänglich ist (vgl. Abb. 25).

Durch Lockingmechanismen die durch `LOCK/UNLOCK`[104] realisiert werden könnten, stießen beim Fax-Dämon BD314 (Rückmeldeverarbeitung) auf großen Interesse, denn sollten mehrere Fax-Dämonen dieser Art, die für unterschiedliche Stores eingesetzt werden, so könnte zuerst ein zentraler Lock auf eine Rückmeldedatei gesetzt werden bevor sie verarbeitet wird, ohne dass sich evtl. andere Dämonen bei der Verarbeitung in die Quere kommen. Schade nur, dass die Businesslogik es vorschreibt, diese Datei bei einer erfolgreichen Rückmeldung von TOPCALL diese in ein anderes Verzeichnis (DOM-System) zu verschieben. Ein sog. „moven" beschreibt den Prozess des Kopierens und des anschließendem Löschens.

Mit einem `LOCK` versehende Dateien lassen sich aber prinzipiell nicht löschen, auch wenn ein Prozess, der einen Lock selbst vergeben hat, dies tun möchte. Nachstehendes Listing 15 zeigt nun dennoch eine Lösung dieses Problems.

[104] Ein LOCK sperrt eine Datei oder ein Verzeichnis für andere Prozesse, solange dieser nicht wieder durch ein UNLOCK aufgehoben wird. Unbefugte Dateioperationen wie löschen, umbenennen u.s.w. lassen sich dadurch unterbinden.

```
1   //--- check if MSG belongs to these process
2      try {
3              belongsToMe = topcallXML.MSGbelongsToMandant(BD314_FROM_FAX, newWebDAVFile, MANDANT);
4            } catch(Exception ex) {
5              if(!c99fileAccess.resourceExists(BD314_FROM_FAX + newWebDAVFile.getNameOfFile())) {
6                bd330.bd330j02_setDaemonInfo(DAEMON, "file " + "(" +
7                newWebDAVFile.getNameOfFile()  + ")" + " was proceeded by another daemon.");
8              else
9                throw ex;
10           }
```

Listing 15 Lösung des LOCK Problems

Sollte eine neue Rückgabedatei in der TC/LINK-FI Schnittstelle vom Fax-Dämon BD314 gefunden werden, so muss dieser zuerst feststellen ob sie für seinen Store bestimmt ist (Listing 15, Zeile 3).

Pech hat dieser während des Auslesevorganges[105] dieser Datei, wenn ein anderer bereits eine gültige Zuordnung zu einem Store gefunden hat. Sollte dies der Fall sein, so wird diese Datei infolge der DOM-Archivierung unter den Füssen des anderen Prozesses entrissen und gelöscht, obwohl der Auslesevorgang evtl. noch gar nicht beendet worden war. Mittels eines Try & Catch Blockes (Zeile 2-4) wird sofort eine Fehlerbehandlung „File not found“ ausgelöst. Diese stellt in Zeile 5 zusätzlich fest, ob die Rückmeldedatei wirklich nicht mehr existiert. Sollte dies der Fall sein, so müsste diese von einen anderen Fax-Dämonen ordnungsgemäß abgearbeitet worden sein.

[105] der Auslesevorgang bez. das Feststellen der Zugehörigkeit eines Metro-Stores

4. Kapitel - MGI Fax-Basisdienst BD300

Unter den Begriff Basisdienst BD300 sind alle hauseigenen entwickelten Module der MGI zu verzeichnen, die für eine Fax-Kommunikation notwendig sind. Sie unterteilen sich in die folgenden drei Bereiche:

- **Faxe/Listen-From**, dient als Applikationsplattform für die Erstellung von Kommunikationsvorgängen
- **Fax-Dämon BD313** zur Bearbeitung von Kommunikationsaufträgen
- **Fax-Dämon BD314** zur Bearbeitung von Rückmeldungen

die alle zusammen auf das Tabellenschema des Faxdienstes BD300 zugreifen (siehe Abbildung 26).

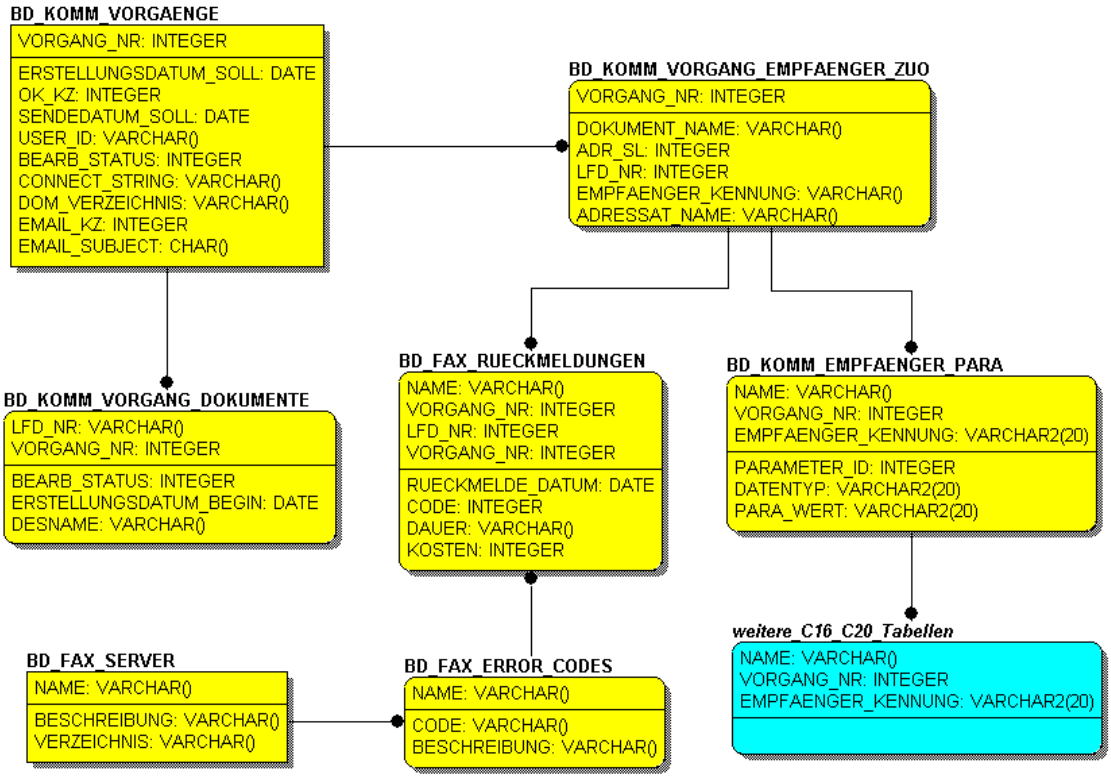

Abbildung 26 Tabellenschema des BD300 Faxdienstes

Obwohl dieses Schema im Umfang mehr als 25 Tabellen vorweist, die über mehrere Services verteilt sind, wird hier nur der relevante Teil für die Fax-Dämonen abgebildet. Einige Attribute die systemtechnisch vorhanden sind, wurden wegen der Übersichtlichkeit weggelassen. Auch wenn man manchmal angeregt wurde dieses

Datenmodell aufgrund doppelter Datensatzeinträge abzuändern, durfte dies auf keinen Fall erfolgen, weil bereits die Fax-Applikation auf diesem Modell aufbaut. Ein Füllen dieser Tabellen übernimmt im ersten Schritt immer die Faxe/Listen Form. Sollte ein Auftrag vom Anwender aus angelegt worden sein, so verarbeiteten die Fax-Dämonen diesen und tragen bei erfolgreichem Versand eines Auftrages anschließend eine Bestätigung ins Tabellenmodell ein. Zum weiteren Verständnis wird kurz die Bedeutung der einzelnen Tabellen erklärt.

BD_KOMM_VORGAENGE

Ist der zentrale Ausgangspunkt jedes Kommunikationsauftrages. Jeder Auftrag wird mit einen Vorgang assoziiert und ist durch ein Statusfeld gekennzeichnet, das ständig durch die Fax-Dämonen BD313 und BD314 überwacht wird und ggf. gesetzt wird.

BD_KOMM_VORGANG_DOKUMENTE

Referenziert zu jedem Kommunikationsauftrag mindestens ein Dokument in Form eines Reports. Pro Report wird eine Transaktionsdatei erstellt. Ebenfalls kennzeichnen hier die Fax-Dämon BD313 und BD314 einen Vorgang in einem Statusfeld. Dieser kann ggf. erfolgreich bei einem Faxversand sein, oder durch einen TOPCALL- bzw. Reportfehler entsprechend deklariert werden.

BD_KOMM_EMPFAENGER_ZUO

Zu jedem Vorgang wird mindestens eine Email oder einer Faxnummer zu einem Empfänger zugeordnet. Sollte zu einem Auftrag eine erfolgreiche Rückmeldung von der TC/Link-FI Schnittstelle erfolgen, so wird für die spätere Archivierung die DOM-Verzeichnisstruktur ausgelesen.

BD_FAX_RUECKMELDUNGEN

Diese Tabelle wird jedes Mal durch den Fax-Dämon BD314 gefüllt, wenn eine Rückmeldedatei eingetroffen ist. In der Faxe/Listen-Form lässt sich später der Erfolg eines Faxes im Detail zu jedem Empfänger erkennen (vgl. dazu Abb. 8 im Kapitel 1).

BD_KOMM_EMPFAENGER_PARA

Definiert alle reportspezifischen Parameter für einen Reportauftrag zu einem Lieferanten. Diese Tabelle verweist auch auf generelle „Common Service“-Tabellen für einen Reportprozess, und wird ausschließlich von der Faxe/Listen-Form verwendet.

BD_FAX_SERVER

Diese Tabelle listet alle derzeit aktiven TOPCALL-Fax-Systeme der Metro auf. In jeder Transaktionsdatei die später von der zentralen TC/Link-FI Schnittstelle verarbeitet wird, ist es möglich explizit einen anderen TOPCALL-Server anzusprechen. Derzeit wird der gesamte Faxverkehr von nur einem Fax-Server in Düsseldorf verwaltet. Zu einem späteren Zeitpunkt erfolgt aber die Einbindung aller Store-Datenbanken (insg. 60) an dieses IT-System, welches zentral am Standort der Metro Group Düsseldorf operiert. Um nun eine problemlose Kostenaufsplittung zu einen Store zu gewährleisten, sollte ein Fax auch von einem TOPCALL-System der entsprechenden Niederlassung (des Stores) versendet werden. Beispiel: „Ein in Russland aufgegebenes Fax sollte ohne Kommunikationsumwege direkt an einen Lieferanten in Russland gesendet werden“. Würde für diesen Fall der Fax-Server in Düsseldorf eingesetzt, so wären die Telefonkosten höher, als wenn diese Aufgabe direkt vor Ort (Russland) ein Fax-Server übernimmt.

BD_FAX_ERROR_CODES

Sollte die Versendung eines Faxes einmal fehlschlagen, so findet sich hier zu jedem TOPCALL-Codekürzel in einer Rückgabedatei, eine ausführliche Beschreibung des Fehlers, die ein Benutzer später in der Faxe/Listen-Form einsehen kann.

4.1 Konfiguration der Fax-Dämonen

Eine Konfiguration der Fax-Dämonen BD313 und BD314 sollte soweit wie möglich dynamisch erfolgen, d.h. es ist auch möglich, während der Laufzeit in den Programmablauf einzugreifen und evtl. Einstellungen vorzunehmen.

Für die Initialisierung sämtlicher Programmmodule in der MGI steht der Common Service C02 - „Verwaltung von Anwendungsparametern“ zur Verfügung. Er stellt SQL-Methoden zur Verfügung, die auf eine „C02 Parameter Tabelle“ verweisen. In dieser lassen sich alle Parameter für den Start eines Moduls registrieren, so auch für die Fax-Dämonen. Typischerweise lassen sich hier alle erforderlichen WebDAV-Verzeichnisse wiederfinden und ein Parameter „Sleeptime“ für jeden Fax-Dämon. Er definiert (in Sekunden), wie lange sich ein Dämon schlafen legen soll, bevor er wieder nach neuer

Arbeit sucht. Mitarbeiter der Basisdienst-Abteilung haben somit durch die C02 Tabelle die Möglichkeit, Programme von einer zentralen Stelle aus zu konfigurieren, egal auf welchen unerreichbaren Rechner sie auch laufen mögen. Durch eine massive temporäre Erhöhung des „Sleep-Time Parameters“ ließe sich ein Dämon zwischenzeitlich „deaktivieren“, ohne ihn jedoch tatsächlich zu beenden. Der CPU des Servers erhält somit wieder mehr Ressourcen für wichtige Aufgaben. Auch eine komplette Umkonfigurierung des WebDAV-Servers ließe sich während des Betriebes durchführen.

Für die Fax-Dämonen sollte dieses Konzept sinnvollerweise übernommen werden. Insbesondere bei Java macht es aber keinen Sinn, bestimmte Parameter wie z.B. Passwort und Benutzername für eine Datenbankverbindung in der Datenbank selbst zu hinterlegen. Generell werden sicherheitskritische Daten in Startskripten abgelegt und dort an ein Programm weiterübergeben. Diese Startskripte liegen meist auf großen Servern, die in einer Sicherheitszone liegen, auf die nur Systemoperatoren Zugriff haben. Listing 17 zeigt ein solches Startskript für den Fax-Dämon BD313:

```
1   . ./connect.env
2   # Initial configuration settings
3   # Display start
4   echo ------------------------------------------------------------
5   echo STARTING DAEMON BD313
6   echo ------------------------------------------------------------
7   java -jar -Dlog4jPropertyFileAndPath=$log4jPropertyFileAndPath -DTimezone=$TIMEZONE -
    DDBConnect=$DBCONNECT -DDaemon=BD313 -DWebdavuser=$WEBDAVUSER -DWebdavpasswort=$WEBDAVPASSWORT
    -DSMTPMailServer=$SMTPMailServer -DMandantname=$MANDANTNAME -DMandantpasswort=$MANDANTPASSWORD
    -DproxyHost=$PROXYHOST -DproxyPassword=$PROXYPASSWORD -DproxyUser=$PROXYUSER -
    DproxyPort=$PROXYPORT BD313.jar
8   RET=$?
9   echo ------------------------------------------------------------
10  echo Programmende Daemon BD314
11  if [ $RET -eq 0 ] ; then echo Programm wurde normal beendet
12  else echo Programm wurde mit Fehlerstatus beendet, Fehlercode = $RET
13  fi
14  echo ------------------------------------------------------------
15  #exit $RET
```

Listing 17 Ein Startskript für den Fax-Dämon BD313

In Zeile 7 werden alle Systemproperties an die Java Runtime Engine übergeben, gleichzeitig erfolgt der Start des Fax-Dämons (BD313.jar). Um zusätzlichen

Konfigurationsaufwand zu mindern, wurden Parameter die von beiden Dämonen benutzt werden, nur einmalig in der Datei connect.env deklariert. (vgl. Listing 18)

```
1   # settings for both daemons (BD313/BD314)
2   export log4jPropertyFileAndPath=$HOME/?????/c98.properties
3   export WEBDAVUSER=XXXXXXXXXX
4   export WEBDAVPASSWORT=XXXXXXXXXX
5   export DBCONNECT=XXXXXXXXXXXXXX:1234:MANDANT
6   export MANDANTNAME=XXXXXXXXX
7   export MANDANTPASSWORD=XXXXXXXXXX
8   export PROXYHOST=proxy.XXXXXXXXX.de
9   export PROXYPASSWORD=XXXXXXXXXX
10  export PROXYUSER=XXXXXXXXXX
11  export PROXYPORT=1234
12  # settings only used for daemon BD314
13  export TIMEZONE=ECT
14  # settings only used for daemon BD313
15  export SMTPMailServer=XXXXXXXXXX
```

Listing 18 Propertiedatei „connect.env“ für die Fax-Dämonen

Der Systemoperator braucht sich ebenfalls nicht um die Existenz evtl. Verzeichnisse auf einem WebDAV-Server zu kümmern. Diese werden bei Bedarf automatisch angelegt.

4.2 Der Fax-Dämon BD313

Dieser Dämon ist für die Bearbeitung aller angelegten Kommunikationsaufträge im Fax-Basisdienst zuständig. Abbildung 27 zeigt diesen in seiner alten Form. Damals existierte

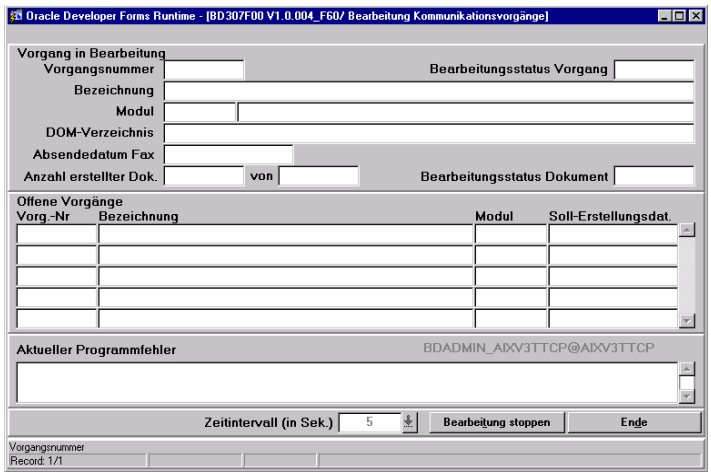

Abbildung 27 Alter Dämon BD313 in Oracle Froms

noch eine GUI-Oberfläche, in der sich der aktuell bearbeitete Prozess live beobachten lies. Diese Funktionalität war für die neuen Fax-Dämonen nicht mehr erwünscht. Sie werden zukünftig im Hintergrund als Systemprozess auf einem Server laufen. Die einzige Ausgabe, die ein Server erhält, ist der Rückgabewert[106] des Dämons, der durch ein Job-Scheduling-System aufgefangen werden kann. Aktuelle Informationen zur Laufzeit des Dämons (z.B. welcher Kommunikationsvorgang gerade bearbeitet wird) können nun flexibler durch die Entwicklung eines zusätzlichen Watcher-Tools[107] von jedem Arbeitsplatz überwacht werden. Der aktuelle Verarbeitungsstatus der Fax-Dämonen kann natürlich immer noch mit der Faxe/Listen-Form eingesehen[108] werden; dieser wird bei jeder Aktion in die Statusfelder der Tabellen `BD_KOMM_VORGAENGE` und `BD_KOMM_VORGANG_DOKUMENTE` eingetragen.

Die gesamte Businesslogik beider Fax-Dämonen läuft in einem globalen Thread in Java. Bei vielen Programmierprojekten gibt es Aufgaben, die gleichzeitig erfüllt werden müssen: Server müssen mehrere Anfragen parallel bearbeiten können; die Verarbeitung von großen Datenmengen kann auf Mehrprozessor-Maschinen schneller geschehen, wenn beide Prozessoren unabhängig voneinander arbeiten. Die Fax-Dämonen operieren mit einer sequenziellen Abarbeitung aller Kommunikationsvorgänge im Basisdienst. Eine Verteilung dieser einzigen Aufgabe in mehreren Threads wäre unsinnig und würde zusätzlich einen zu hohen Overhead verursachen. Threads bieten zudem noch zusätzlich die Möglichkeit, Ressourcen durch ihre Inaktivität wieder an den CPU[109] zurückzugeben. Diese Inaktivität lässt sich in Java nach jeder Abarbeitung eines Fax-Auftrages durch einen Sleep-Time Wert[110] (in Millisekunden) vergeben. Bei einem größeren Fax-Aufkommen sollte dieser für eine zügigere Verarbeitung verkleinert werden.

[106] sog. „Exit-Code“, er kennzeichnet einen Abbruch oder ein ordnungsgemäßes Beenden eines Programms
[107] s. Kapitel 5.2 „Die Admin GUI“
[108] vgl. dazu Kapitel 1.5.1 Abb. 8 - Punkt 1
[109] „Central Processing Unit“ - Basissteuereinheit eines Computers, die für die Verarbeitung von Daten, und Ausführung von Programmen zuständig ist.
[110] vgl. SDK Doc <http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Thread.html#sleep(long, int)> (21.04.2004)

4.2.1 Funktionsablauf

Erfolgt nun ein Start des Dämons, so verbindet sich dieser zuerst mit einem Store und initialisiert seine Startparameter (die nicht im Startskript hinterlegt wurden) durch den Common Service. Anschließend erfolgt eine kurze visuelle Ausgabe dieser Parameter auf der Konsole (vgl. Abb. 28), sicherheitsrelevante Daten werden durch ein „invisible“ gekennzeichnet.

```
* Configuration Settings ********************************** DAEMON BD313 ***
*
* TIMEZONE              : ECT
* DBCONNECT             : [redacted].mgi.de:1526:[redacted]
* DAEMON                : BD313
* WEBDAVUSER            : webdavtest
* WEBDAVPASSWORT        : invisible
* SMTPMAILSERVER        : [redacted].metro-dus.de
* MANDANTNAME           : [redacted]
* MANDANTPASSWORT       : invisible
* PROXYHOST             : proxy.mgi.de
* PROXYPASSWORD         : invisible
* PROXYUSER             : u[redacted]
* PROXYPORT             : 3128
*
*******************************************************************************
*                  Metro MGI Informatik GmbH, Duesseldorf                     *
*                     Copyright (c) 2004 by Manuel Jung                       *
*******************************************************************************
```

Abbildung 28 Neuer Fax-Dämon BD313 in Java

Die Parameter im Common Service können auch während der Laufzeit des Dämons ggf. verändert werden, kurz vor jeder Schlafphase erfolgt ein erneutes Auslesen aus der Datenbank. Im Folgenden wird zuerst die Bedeutung der einzelnen Parameter näher beschrieben. Zu diesen zählen:

EXCEPTION_LANGUAGE[111]

Für jeden Store lassen sich beliebig viele Fehlermeldungen in einer Ländersprache im Common Service hinterlegen. Sollte das Softwareprodukt einmal im Ausland eingesetzt werden, so ist es sinnvoll Fehlermeldungen der Fax-Dämonen in der entsprechenden Landessprache auszugeben. (Einzige globale Einstellung für beide Dämonen)

SLEEP_TIME_IN_SEC

Definiert in Sekunden die Sleep-Time, in der sich der Thread schlafen legen soll, bevor er die Tabelle *BD_KOMM_VORGAENGE* nach neuen Kommunikationsaufträgen suchen soll.

[111] vgl. Kapitel 2.4.1 „Logging Bibliotheken“ – s. Mehrsprachenfähigkeit

TO_FAX

Enthält die Einstellungen für den Verzeichnispfad auf einen WebDAV-Server, in dem später das TC/Link-FI Verzeichnis `TC_TO_FI` angebunden[112] wird. Der Administrator braucht sich zum Glück nicht mehr um die Existenz von erforderlichen Verzeichnissen kümmern, diesen Aufgabe übernimmt der Dämon automatisch zur Laufzeit.

WORKING_DIR

Ein temporäres WebDAV-Verzeichnis, in dem alle versandten Kommunikationsaufträge (Report als PDF-Datei) zwischengespeichert werden. Der Fax-Dämon BD314 archiviert diese Dateien nach einem erfolgreichen Versand in eine DOM-Verzeichnisstruktur auf dem WebDAV-Server. Solange keine Rückmeldedateien vom TOPCALL-Server eintreffen, müssen diese Dateien zwischengelagert werden. Um eine Mandantenfähigkeit für die einzelnen Dämonen (Stores) zu gewährleisten, wird jeweils immer ein entsprechendes Mandanten-Unterverzeichnis erstellt. Topcallspezifische Verzeichnisse bleiben in ihrer globalen Form erhalten.

EMAIL_TEXT

Sollte der Kommunikationsvorgang eine Email darstellen, so lässt sich hier ein Emailtext hinterlegen. In allen Emails des Fax-Basisdienstes werden ausschließlich Reports als Dateianhänge versendet. Aus diesem Grunde ist es nützlich, hier einen Hinweistext in der Form `please see attachment` zu hinterlegen.

EMAIL_SUBJECT[113]

Falls ein Anwender keinen Betreff für einen Email-Kommunikationsvorgang vergeben hat, so benutzt BD313 eine Standardbetreffzeile in der Form

```
%DESNAME% - (%VORGANGNR%/%LFDNR%).
```

EMAIL_DEFAULT_SENDER

Ist dieses Attribut gesetzt, so werden alle erstellten Kommunikationsvorgänge mit einer generellen Standardemailadresse versendet, und nicht mehr mit dem jeweils angemeldeten Anwender, der in der Faxe/Listen-Form einen Auftrag erstellt hat.

---

[112] eine Anbindung erfolgt derzeit durch sog. Kopierdämonen, vgl. Kapitel 4.7 „Kopierdämonen“
[113] s. Kapitel 4.2.3 „Zusatzfeature Emailversand“

FST_MSG_LAYOUT

Definiert ein Dateiformat für die TC/Link-FI Schnittstelle; sowohl Rückgabe- als auch Transaktionsdateien werden entweder im XML oder TOM Format verarbeitet.

Nach dem Start des Dämons wird automatisch mit der Bearbeitung von offenen Kommunikationsvorgängen begonnen, die anhand eines Status identifiziert werden können. Entdeckt der Dämon einen neuen Kommunikationsvorgang, so wird dieser auf den Status *Auftrag erkannt* gesetzt. Nun wird die Verfügbarkeit der jeweiligen Reports zu einem Vorgang überprüft[114]; sind Fehler schon bei der Reportsgenerierung aufgetreten, so wird nicht der komplette Vorgang beendet, sondern ausschließlich der Report in der Fax-Datenbank als fehlerhaft gekennzeichnet. Sollte sich der Report noch in der Generierungsphase befinden, so wird dieser übersprungen und es wird zwischenzeitlich mit anderen fortgefahren. Bei einem fertigen Report wird der Status *Report fertig* gesetzt, und anhand des Email-Kennzeichens in der Tabelle *BD_KOMM_VORGAENGE* wird zwischen einem Fax- oder einem Email-Kommunikationsvorgang unterschieden.

Fax-Kommunikationsvorgang

Sollte ein Vorgang als Fax gekennzeichnet sein, so wird eine Transaktionsdatei im festgelegten Layout (TOM oder XML Format) anhand der *BD_EMPFAENGER_ZUO* Tabelle erstellt und das dazugehörige Attachment (Report als PDF-Datei) des Dokumentenvorganges ins temporäre Working-Verzeichnis des WebDAV-Servers kopiert. Anschließend wird der Vorgang mit dem Status *Steuerdatei fertig* gekennzeichnet. Der zugehörige Report zu diesem Vorgang ist der Tabelle *BD_KOMM_VORGANG_DOKUMENTE* im Feld *DESNAME* hinterlegt, und lässt sich durch die *C16_URL_QUEUE* Tabelle in Common Service auf einem Report-Server leicht lokalisieren[115]. Dieser wird ebenfalls in das Verzeichnis mittels des HTTP-Protokolls kopiert. Erst wenn die Transaktionsdatei inkl. des zugehörigen Reports auf dem WebDAV-Server existieren, werden diese in das Faxausgangsverzeichnis *TO_FAX* verschoben und somit der TC/Link-FI Schnittstelle zur Verfügung gestellt. Den Fax-Versand übernimmt nun ein TOPCALL-Fax-Server. Die Versendung der Faxe richtet

[114] vlg. Kapitel 2.2 „Oracle Reports“ Abb.8 Ablaufschema eines Faxes
[115] Funktionsprinzip ist im Kapitel 2.3 „Der Oracle Web Cache“ beschrieben

sich nicht nur nach den in der Steuerdatei angegeben Parametern, sondern ist auch von anderen Faktoren, wie z. B. der Anzahl der freien Leitungen, der Anzahl der unter TOPCALL eingestellten maximalen Wählversuche usw., abhängig. Unmittelbar nachdem die Dateien ins Faxausgangsverzeichnis kopiert wurden, wird der Bearbeitungsstatus des Dokumentenvorganges in `BD_KOMM_VORGANG_DOKUMENTE` auf `an_fax_server_uebertragen` gesetzt.

Email-Kommunikationsvorgang

Sollte ein Vorgang als Email gekennzeichnet sein, so ist die Verfahrensweise analog zu einem Fax-Kommunikationsvorgang, lediglich die Generierung einer Transaktionsdatei für die TC/Link-FI Schnittstelle entfällt. Die Versendung von Emails erfolgt direkt über einen internen SMTP-Server der Metro, der je nach Konfiguration auch eine Rückmeldung[116] in ein Email-Postfach eines Mitarbeiters (Anwender der Faxe/Listen-From) der Metro Group senden kann.

Sind alle Dokumente des Kommunikationsvorganges in `BD_KOMM_VORGANG_DOKUMENTE` erstellt und versendet worden, so wird der Bearbeitungsstatus des Gesamtvorganges in `BD_KOMM_VORGAENGE` auf `an Fax-Server übertragen` gesetzt. Obwohl in diesem Falle keine Faxe verschickt worden sind, sollte aus historischen Gründen dieser Status auch für Email-Vorgänge weiter genutzt werden.

Nach Ablauf des angegebenen Zeitintervalls wird erneut nach offenen Vorgängen gesucht.

[116] vgl. Kapitel 4.3.1 „Zusatzfeature Emailversand“

4.2.2 Programmablaufplan

Der Funktionsablauf des Fax-Dämonen BD313 wird an dieser Stelle nochmals detailreicher in einem Flußdiagramm verdeutlicht. Grundfunktionalitäten wie das Anlegen eines JDBC-Connection-Pools, Fehlerbehandlungsroutinen, Initalisierung des Dämons etc. würden von eigentlichen Funktionsumfang ablenken und werden deshalb nicht mit abgebildet.

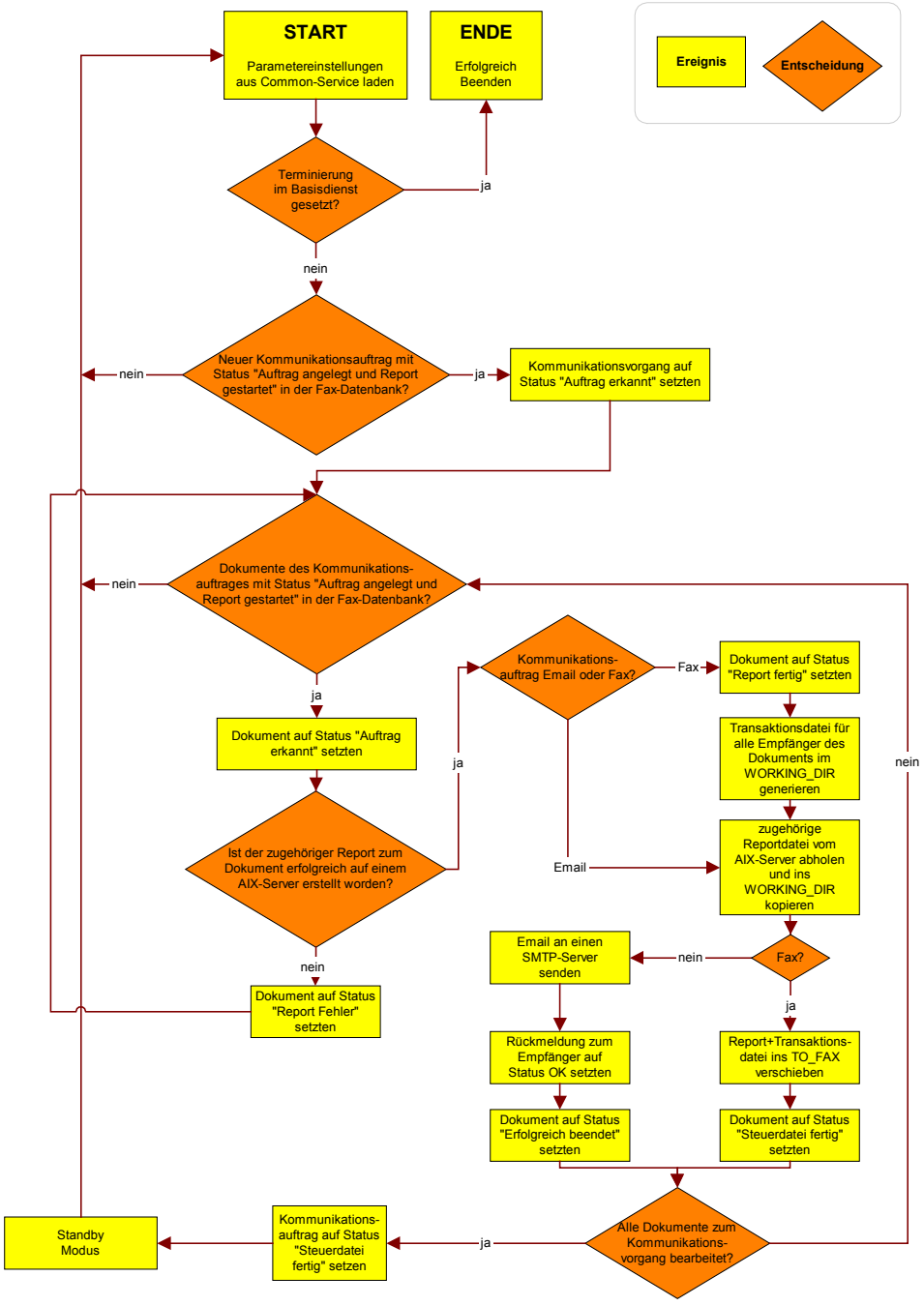

Abbildung 29 Programmablaufplan Fax-Dämon BD313

4.2.3 Zusatzfeature Emailversand

Als zusätzliches Feature sollte in der Faxe/Listen-Form die Möglichkeit bestehen, neben Faxen auch Emails zu versenden. Theoretisch könnte diese Funktionalität auch der TOPCALL-Server mittels der TC/LINK-FI Schnittstelle zur Verfügung stellen, eine Freischaltung des erforderlichen Moduls würde aber zusätzlichen Support und somit Kosten verursachen. Das Thema wurde aus diesem Grund auf die Fax-Dämonen verlagert. Prinzipiell ist es nur möglich, eine Ausgabeform in der Faxe/Listen-Form pro Vorgang auszuwählen (Fax oder Email)[117]. Eine Unterscheidung des Sendeformates erfolgt später mit einem Kennzeichen in einer Auftragstabelle. Optional lässt sich für jede Email eine Betreffzeile in der Faxe/Listen-Form vergeben. Sollte diese Eingabe einmal nicht erfolgen, so werden alle Emails mit einen Defaultemailsubject in der `C02Parameters` Tabelle vom Fax-Dämon BD313 versendet:

```
%DESNAME% - (%VORGANGNR%/%LFDNR%)
```

- Der Desname beschreibt den generierten Reportnamen
- Die Vorgangsnummer bezeichnet den aktuell angelegten Faxvorgang
- Findet mehr als eine Reportausprägung in einem Vorgang Verwendung, so werden diese durch eine Laufnummer unterteilt.

Die Zustellung einer Mail übernimmt eine Standard Mail API[118] Bibliothek von Sun, die in eine universelle Mail-Provider-Klasse des Basisdienstes eingebunden wurde und vom Fax-Dämon BD313 benutzt wird. Ein Nachteil von Emails im Gegensatz zu Faxen ist, dass keine Bestätigung des Versands zuverlässig erfolgt. Oft bestätigen SMTP-Server einen Zustellungsfehler mit einer Rückgabemail an den Absender oder speichern diese separat in ein Fehlerpostfach ab. Ist letzteres der Fall, so liese sich jeder Status zu einer Mail auch dort vom Fax-Dämon BD313 abholen, auswerten und in die Fax-Datenbank hinterlegen. Der „OK“-Statuseintrag[119] in der Faxe/Listen-Form hätte somit die selbe Bedeutung wie bei Faxen. Leider musste diese durchaus sinnvolle Lösung verworfen werden, denn je nach Konfiguration eines SMTP-Servers muss dieser nicht zwangsläufig eine Rückmeldung ausgeben.

[117] s. Abbildung 6 „Faxe/Listen-Form“ – Markierung 2
[118] verwendet wurde die JavaMail Bibliothek (akt. Version derzeit 1.2) - download <http://java.sun.com/products/javamail/1.2/docs/javadocs/javax/mail/package-use.html> (12.03.2004)
[119] vgl. Abb. 7 “Faxe/Listen-Form Historie“

Dem „OK“ Status in der Faxe/Listen-Form, der einen erfolgreichen Versand einer Email kennzeichnen sollte, musste eine andere Bedeutung zugewiesen werden. Sobald der Fax-Dämon BD313 einen STMP-Aufruf an einen Mail-Server verschickt, so wird die Mail als abgesendet (`an_fax_server_übertragen`) deklariert.

Glücklicherweise sendet der derzeitig eingesetzte SMTP-Server in der Metro im Fehlerfalle eine Bestätigung an den Absender zurück. Bei jedem angelegten Fax-Vorgang wird die User-ID des angemeldeten Benutzers in der Fax-Datenbank mitgespeichert. Für den Versand einer Email könnte man als Absenderadresse die Emailadresse über die User-ID herausfinden. (Optional lässt sich jede Email vor der Versendung mit einer Standardemailadresse füllen, falls ein Eintrag für die Dämonen in den Common Service hinterlegt wurde.)

Weil sämtliche Accounts in der Metro zentral auf LDAP-Servern verwaltet werden, und kein Ort besser für aktuellere Emailadressen erscheint, kam der Gedanke LDAP als Lösung zu verwenden.

4.2.3.1 LDAP

LDAP bedeutet „Leightweight Directory Access“ und definiert ein Protokoll für den Zugriff auf ein Verzeichnis – ein sog. Nachschlagewerk. Dahinter steht eine Datenbank, die ein oder mehrere Verzeichnisse beinhaltet, die hierarchisch aufgebaut sind. LDAP arbeitet durch spezielle Suchalgorithmen sehr performant auf Daten, die wenig verändert, aber häufig abgefragt werden. Leider macht es LDAP möglich, mehrere Emailadressen zu einem Account zu hinterlegen. Dies würde zu der Tatsache führen, dass man nicht wüsste, welche eigentlich für den Fax-Dienst vorgesehen ist. Auch eine automatische Emailadressen-Generierung durch den Fax-Dämon BD313 nach folgendem Muster `Vorname.Nachname@Metro.de` scheiterte, denn wie sollte der Fall abgedeckt werden, wenn es mehrere Benutzer mit gleichem Vor- und Nachnamen gibt ? Die Lösung dieses Problems durch Generierung der Emailadresse mit einer zusätzlichen Laufnummer in der Form `Vorname.Nachname2@Metro` setzt auch tatsächlich voraus, dass diese existiert, aber evtl. macht der Systemoperator hier Sonderausnahmen von Fall zu Fall.

Leider gibt es nicht eine zentrale LDAP-Datenbank in der Metro. Die Betriebssysteme Microsoft Windows NT und Windows XP sind so konfiguriert, dass die Benutzerverwaltung über unterschiedliche LDAP-Datenbanken verwaltet wird. Sollte ein Benutzer über seinen NT-Client von der Faxe/Listen-Form über das Intranet Gebrauch machen, so müsste die jeweilige LDAP-Datenbank zur Emailadressensuche verwendet werden. Über die Form ließe sich ein solches Betriebsystem mittels Systembefehlen leicht herausfinden, aber eine kontinuierliche Pflege der NT-Benutzerdaten wird in der Metro Group nicht mehr vorgenommen, weil derzeit eine komplette Umstellung auf Windows XP Rechner im Hause erfolgt. Alte Einträge könnten somit evtl. gar keinen Emaileintrag vorweisen. Emails ohne einen Absender machen aber keinen Sinn...

Insgesamt also ein schwieriges Unterfangen, eine simple Emailadresse herauszubekommen. Anstelle von LDAP, werden personenbezogene Daten für Forms Anwendungen in der Metro zusätzlich in einer speziellen Tabelle `C20_Personell` im Common Service hinterlegt, die meist in jeder Datenbank zu finden ist. Der Fax-Dämon BD313 kann dort nun eine Verknüpfung mittels der User-ID zu einer Emailadresse vornehmen.

Die sorgfältige Pflege dieser Daten erfolgt ausschließlich durch die Administration der hauseigenen MGI-Benutzerverwaltung (ebenfalls in Oracle Forms realisiert). Um einen Missbrach in der Faxe/Listen-Form mit Emailadressen der Metro vorzubeugen, dürfen einfache Anwender von MGI-Anwendungen ihre Email nicht selber pflegen. Emailvorgänge könnten dann z.B. unter einem falschen Absender verschickt werden, anstatt unter der üblichen Emailsyntax der Metro.

4.3 Fax-Dämon BD314

Dieser Dämon beinhaltet den Prozess zum Auslesen der Rückmeldungen der TC/Link-FI Schnittstelle des TOPCALL-Fax-Servers. Der Vorgänger dieses Dämons teilte in der alten Oracle Forms Lösung ebenfalls seine Arbeitsweise direkt auf einer Console mit.

Abbildung 30 Alter Fax-Dämon BD314 in Oracle Froms

Hierzu zählen die angelegten, verarbeiteten und gefaxten Dokumente. Auch bei diesem Dämon diese Art der Ausgabe nicht mehr erforderlich, aktuell verarbeitete Kommunikationsaufträge lassen sich ebenfalls mit der Admin GUI überwachen.

Die Auslesung der Initialisierungsparameter erfolgt auf die identische Weise wie beim Fax-Dämon BD313. Der Einsatz der Parameter *SLEEP_TIME*, *WORKING_DIR* und *FST_MSG_LAYOUT* ist bei beiden gleichbedeutend, darüber hinaus verwendet dieser folgende Parameter:

FROM_ FAX

Mapping zum TC/Link-FI Verzeichnis *NOTIF* auf einem WebDAV-Server. In diesem Verzeichnis stehen die von TOPCALL erzeugten Rückmeldungsdateien zu einem angelegten Fax-Vorgang.

ERROR_DIR

Sollte einmal ein Zustellungsfehler für ein Fax aufgetreten sein, so wird die Rückgabedatei inkl. Reports direkt von *FROM_FAX* in dieses Verzeichnis verschoben. Dank der Aufsplittung der Stores in einzelne Mandantenverzeichnisse kann somit schnell eine Zuordnung im Fehlerfalle vorgenommen werden. Fehlerhafte Faxe werden

natürlich nicht im DOM-System archiviert. Für Emails ist dieses Verzeichnis leider völlig unnütz, weil ein Dämon niemals feststellen kann, ob wirklich durch einen SMTP[120]-Server ein Auftrag verschickt wurde.

CLEANUP_INTERVAL

Für die automatische Säuberung des Error-Verzeichnisses ist nicht etwa ein Job-Scheduling-System verantwortlich, sondern der Dämon selbst. Mit diesem Parameter lässt sich das maximale Alter von Reports und Rückgabedateien in Tagen definieren. Überschreiten Dateien diese Grenze werden sie automatisch gelöscht.

DOM_DIR

Archivierungsverzeichnis auf einem WebDAV-Server, welches später an das DOM-System angekoppelt[121] wird.

4.3.1 Funktionsablauf

Nachdem der Dämon gestartet worden ist, werden analog zum `FROM_FAX` Verzeichnis zusätzlich alle Emailaufträge überwacht, die bereits mit dem Status `an Fax-Server übertragen` gekennzeichnet sind. Diese können getrost[122] auf den Status `Erfolgreich beendet` gesetzt werden. Eine erfolgreiche Versendung eines Faxes kann über das `FROM_FAX` Verzeichnis festgestellt werden. Stehen in diesem Verzeichnis eine oder mehrere Rückmeldedateien von TC/Link-FI, so werden diese nach Ablauf des Zeitintervalls `SLEEP_TIME` von einer Prozedur ausgelesen und inhaltlich bearbeitet. Das bedeutet, dass der entsprechende Vorgang, der zu der Rückmeldung gehört, mitsamt des Dokumentes geprüft wird. Im Erfolgsfall werden die entsprechenden Statuswerte für das Dokument und den Vorgang in der Tabelle `BD_FAX_RUECKMELDUNGEN` gesetzt. Die Rückmeldung und das Dokument werden (falls erwünscht) im vorläufigen DOM-Verzeichnis auf einem WebDAV-Server im `DOM_DIR` Verzeichnis archiviert.

Zum Schluss werden die entsprechenden Stati gesetzt, falls sich welche geändert haben.

[120] SMTP „Simple Mail Transport Protocol“, das gängigste Protokoll zum Versenden von Emails Abkürzung
[121] für den Anbindungsprozess s. Kapitel 4.7.4 „DOM/File.Net Archivierung“
[122] vlg. Kapitel 4.1.3 „Zusatzfeature Email“

4.3.2 Programmablaufplan

Analog zum vorherigen Dämon erfolgt nun ein Flussdiagramm für den Fax-Dämon BD314.

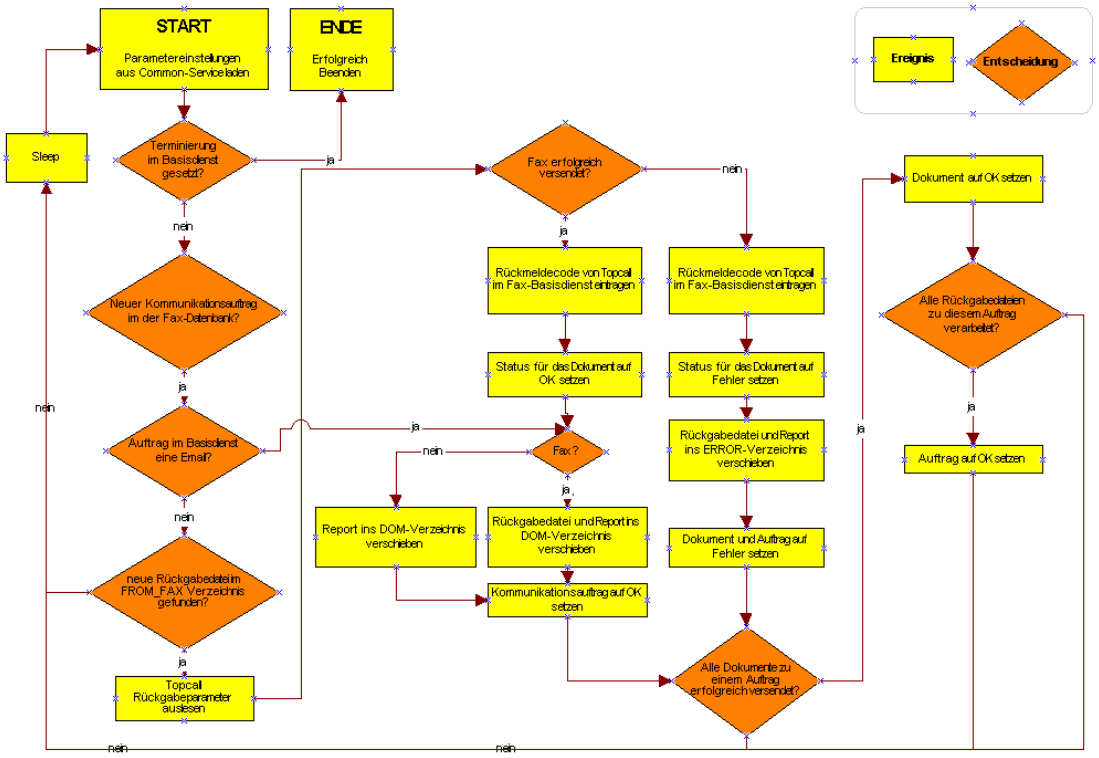

Abbildung 31 Programmablaufplan Fax-Dämon BD314

4.4 Mandantenfähigkeit

Bisher war es ausschließlich möglich, nur einen Metro-Store mit Hilfe der Fax-Dämon zu versorgen. Die neuen Dämonen ermöglichen nun auch eine Mandantenfähigkeit. Mit dieser Technik können sich mehrere Mandanten eine gemeinsame Datenbank teilen. Datenbanken sind meist sehr speicherhungrig; nicht selten werden pro Installation mindestens 512MB Arbeitsspeicher benötigt. Komplettinstallationen mit Geschäftsdaten benötigen in der Metro mindestens ca. 20 Gigabyte. Mit Hilfe der Mandantenfähigkeit können nun übergreifende Tabellen gemeinsam genutzt werden, die jeweils unterschiedlichen Geschäftdaten werden in einem eigenen Tabellenschema verwaltet. Dieses Schema bezeichnet man als eine Instanz einer Datenbank, einen sog. „Mandanten“. Insgesamt können bei voller Ausreizung des Fax-Basisdienstes 60 Mandanten (Stores) angelegt werden. Würde pro Store eine komplette

Datenbankinstallation erfolgen, so stößt man schnell an die Ressourcengrenzen eines Servers. Aus diesem Grunde wird für jeden Store eine eigene Datenbankinstanz in der Fax-Datenbank erstellt, auf die sich mit entsprechenden Verbindungsparametern[123] über die Faxe/Listen-Form zugreifen lässt

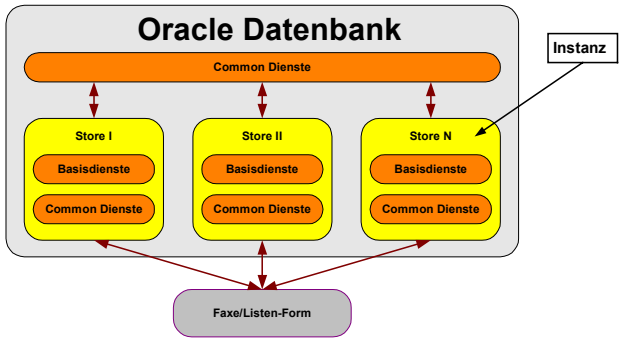

Abbildung 32 Realisierung der Mandantenfähigkeit

Sowohl die Basis- als auch die Commondienste sind für jeden Store unterschiedlich konfiguriert. Gleiches gilt auch für Fax-Dämonen, lediglich gemeinsam genutzte feste Variablen im Common-Service wie z.B. Standard-Report-Parameter lassen sich global in der Datenbank festlegen und gemeinsam von allen Stores nutzen. Abbildung 32 zeigt, dass es mit Hilfe der Faxe/Listen-Form möglich ist, für jeden Store eigene Faxe in der Datenbank zu hinterlegen. Dies ist sinnvoll, weil jeder Store andere Lieferanten und Konditionen haben kann.

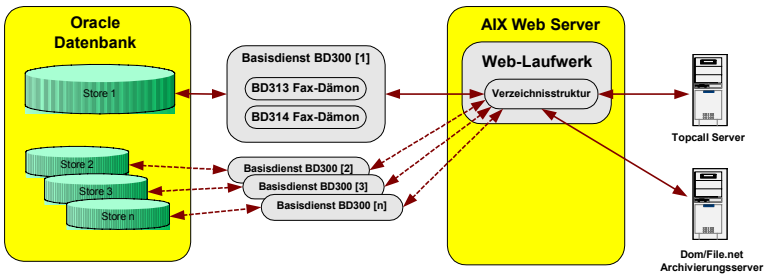

Abbildung 33 Anbindung der Fax-Dämonen an die Stores

Natürlich müssen alle diese Stores auch von den Fax-Dämonen versorgt werden, für jede Instanz wird daher ein BD313- und BD314 Fax-Dämon benötigt[124]. Mittels JDBC lässt sich die Mandantenfähigkeit[125] einfach realisieren, und dank eines zentralen

[123] s. Kapitel 1.5.1 „Ein Fax geht auf Reisen“ Abbildung 4, Login-Fenster
[124] vgl. Kapitel 4.6 „Ressourcenengpass“
[125] Connectstring für Oracle Datenbanken in Java: jdbc:oracle:thin:@Host:Port:Mandant

WebDAV-Servers auf einem AIX-Server und nur einer TC/Link-FI Schnittstelle konnten weitere Ressourcen eingespart werden. Für die Mandantenfähigkeit einigte man sich auf eine eindeutige Namenskonvention für die Reports und die Transaktionsdateien für TOPCALL nach dem Schema

```
1   BD300_<Datenbankname_Mandant>_<Faxvorgang>_<Fax-Laufnummer>.<PDF bzw. FST>
```

Listing 16 Namenskenvention für Reports u. Transaktionsdateien

Der Vorteil hierbei ist, dass sich Fax-Vorgänge jederzeit eindeutig identifizieren lassen, und der Fall einer Dateiüberschreibung durch die Dämonen ausgeschlossen werden kann. Der Fax-Dämon BD313 hatte somit keinerlei Probleme seine Arbeit aufzunehmen.

Wie sollte jedoch eine Rückmeldedatei von der TC/Link-FI Schnittstelle einen Mandanten im Fax-Dämon BD314 zugeordnet werden?

Der TOPCALL-Server verwendet hierfür sein eigenes Namensschema für eine Datei in Form einer Sequenz:

[1] **00000001a.msg**
[2] **00000002a.msg**
[3] **00000009a.msg**
[4] **00000001b.msg**
[5] **0000000.....msg**

Glücklicherweise gibt es für die TC/Link-FI Schnittstelle frei definierbare Variablen die in jeder Transaktionsdatei mitgegeben werden können, diese werden ohne jegliche Überprüfung und mit gleichen Inhalt wieder von TOPCALL in eine Rückgabedatei geschrieben. Benutzt man einer dieser Variablen für die Zuweisung eines Datenbankmandanten, so wird nun ebenfalls Fax-Dämon BD314 seine Aufgabe erfüllen können. Der Prozess der Mandantenüberprüfung in BD314 wurde bereits im Listing 15 - Zeile 3 dargestellt.

4.5 Datenbankbackup

Die Erfordernis eines Backups einer Datenbank mit sensiblen Daten bedarf keinerlei Begründung. Fraglich ist nur, wie die Fax-Dämonen mit einem radikalen herunterfahren einer Datenbank zurechtkommen würden. Hierbei werden alle offenen Sessions sofort geschlossen, egal ob ein Anwender oder ein Dämon seine Arbeit erledigt hat. Obwohl die Test-Fax-Datenbank hiervor weitestgehend unberührt blieb, ist es bei Produktivdatenbanken üblich diese am Wochenende zu sichern. Je nach Konfiguration kann dies im Online- oder im Offline-Betrieb geschehen. Für den letzteren Fall musste eine Lösung für die Fax-Dämonen gefunden werden. In einer zusätzlichen angelegten Tabelle *BD_DAEMON_STATUS* im Common Dienst werden ständig die aktuellen Laufzeitstati der Fax-Dämonen eingetragen und selbst wieder ausgelesen, ebenfalls wird hier auch der aktuelle Arbeitsvorgang hinterlegt. Zu den möglichen Stati der Fax-Dämonen BD313 und BD314 zählen:

- 1 läuft - wird beim Starten eines Dämonen gesetzt
- 0 erfolgreich gestoppt - der Dämon wurde vom Benutzer erfolgreich gestoppt
- 2 Dämon herunterfahren - Dämon soll seine Verarbeitung beim aktuellen Vorgang abschließen und sich beenden.
- 9 Fehler aufgetreten - Dämon wurde aufgrund eines Fehler beendet

Sollte nun die Verbindung zu einer Datenbank abreißen, so ist es möglich mit Job-Scheduling-Systemem ein Stopscript vorher zu starten, dass einen Statuseintrag *dämon_beenden* für beide Dämonen setzt. In diesem Falle werden die letzten Aufträge von BD313 bearbeitet und BD314 beendet ebenfalls seine Arbeit nachdem die nächstanstehende Rückmeldedatei verarbeitet worden ist. Listing 19 zeigt einen Auszug eines Stopskriptes, welches ins Job-Schedulingsystem für den BD314 eingebunden wird.

```
1  # Initial configuration settings
2  DAEMON=BD314
3  PRGNAME=stopBD314
4  JAVA_DAEMONNAME=BD314
5  DB_NAME=xxxxxx
6  USERNAME=xxxxx/xxxxx@$DB_NAME
7  sqlplus -s <<EOF
```

```
8   $USERNAME
9   DECLARE
10    STOPSTATUS VARCHAR2(2) := bd330k00.bd330p08_statusStopDaemon();
11    timestamp VARCHAR2(40);
12
13  BEGIN
14    DBMS_OUTPUT.PUT_LINE('trying to stop daemon...');
15    timestamp := TO_CHAR(SYSDATE,'HH24MISS') || TO_CHAR(SYSDATE, 'YYYYMMDD');
16    bd330k00.bd330p01_setDaemonStatus ('$JAVA_DAEMONNAME', STOPSTATUS, timestamp);
17
18  EXCEPTION
19    WHEN OTHERS THEN
20      RAISE_APPLICATION_ERROR(-20001,'cannot stop Daemon...');
21  END;
22  /
23  EXIT;
```

Listing 19 Stopskript für den BD314 Fax-Dämon

Dieses Shellskript liest einen Defaultstopwert aus der Datenbank und trägt ihn in den Common Service ein. Man sollte bei einem Backup der Datenbank den Fax-Dämonen ein genügend großes Zeitpuffer mitgeben um die letzen Fax-Aufträge zu verarbeiten und um anschließend ihre Verbindung zu beenden.

Ein Zeitintervall von 30 min ist hierfür ein guter Kompromiss.

4.6 Ressourcenengpass

Die JVM[126] wird als das Herz von Java gesehen und muss auf jedem Rechner vorhanden sein, wenn Java-Anwendungen ausführt werden müssen. Die virtuelle Maschine stellt eine Ausführungsschicht zwischen dem kompilierten Programm, der Hardware bzw. dem Betriebssystem dar und realisiert zur Laufzeit alle Java-Klassen in ausführbare Prozessorbefehle. Dies ist einer der großen Pluspunkte für Java, denn hierdurch wird erst die Plattformunabhängigkeit erreicht, des weiteren kümmert sich die JVM um grundlegende Java-Operationen wie die Objekterstellung und die automatische Garbage Collection[127]. Obwohl große Mühe auf eine ressourcensparende Programmierweise gelegt worden war, belegt die JVM immer eine gewisse „Grundlast“ an einen Rechner. Mittels des „ps“[128] Befehls auf einer AIX lässt sich testweise ein Verbrauch der laufenden Fax-Dämonen BD313 und BD314 ermitteln. Weil jeder Dämon in einen eigenen Prozess für ein Job-Scheduling-System gestartet wird, muss selbstverständlich auch jedes Mal eine JVM mitgestartet werden. Diese belegt inkl. der Fax-Dämonen im Schnitt pro Mandanteninstallation (im Leerlauf) auf einem normalen AIX Rechner der Metro ca.

- 12 MB Arbeitsspeicher und 0.6 % CPU Last

Rechnet man dies für alle 60 Stores zusammen, so beträgt die Minimalkonfiguration eines Servers min. 1.6 Gigabyte Arbeitsspeicher, der CPU dieses Servers wäre mit der Arbeit sichtlich überfordert. Aus diesem Grunde entschied man sich diese auf einer der leistungsfähigsten Rechner der Metro einzusetzen, einen großen Datenbankserver mit insgesamt 26 CPUs und reichlich Arbeitsspeicher. Es wurde im Anschluss natürlich reichlich nachgedacht ob es evtl. besser gewesen wäre, alle Fax-Dämonen in einen zu verpacken, sollte dieser aber aus welchen Gründen auch immer abstürzen, so wird kein einziger Markt der Metro mit Faxen versorgt. Vielleicht wäre eine Lösung mit der maschinenabhängigen Programmiersprache C++ effektiver gewesen, weil keine JVM mehr gebraucht würde. Der Vorteil der einfachen Portabilität durch Java auf andere Systeme wäre damit zu Nichte gemacht.

[126] Java Virtuel Maschine
[127] „Müllbeseitigung“ - Objekte auf die kein Referenz mehr bestehen werden automatisch aus dem Arbeitsspeicher entfernt. Im Gegensatz zu anderen Programmiersprachen wie in C++ oder Delphi hat ein Programmierer in Java generell keinen Einfluss dies zu handhaben
[128] UNIX Systembefehl, ps=Prozess-Status ermitteln

4.7 Kopierdämonen

Die Systemarchitektur des Fax-Basisdienstes BD300 wurde bereits im Kapitel 3.2 ausführlich erläutert. Durch die WebDAV-Technologie einigte man sich auf ein einheitliches Protokoll zur Kommunikation zwischen den Fax-Diensten. Die Fax-Dämonen sind bereits durch die neue DAV4J Bibliothek an einen WebDAV-Server auf einem AIX Rechner angebunden. Fraglich ist nun, wie die windowsbasierten Systeme DOM und TOPCALL dort angebunden werden können. Bevor man sich in eine Java-Lösung verrennt und Räder wieder neu erfindet, sollte zuerst die Möglichkeit untersucht werden, kommerzielle Mapping-Tools für WebDAV einzusetzen. Generell sollte jedoch wegen Supportverträgen keinerlei Installationen auf diesen Systemen vorgenommen werden. Kurzzeitig kam daher die Idee, die alte Windows-Share-Lösung weiterzuverwenden und einen sog. Mapping-Gateway, der WebDAV-Verzeichnisse von einem AIX-Rechner abbildet und diese mittels einer Dateifreigabe in Windows diesem Systemen zur Verfügung stellt, (vgl. Abbildung 34) einzusetzen.

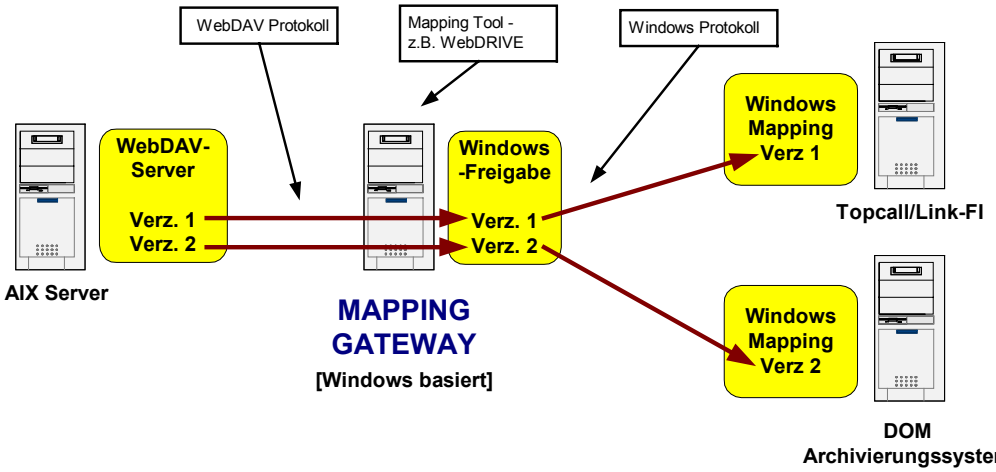

Abbildung 34 Einsatzszenario eines Mapping-Gatewayes

Die finanzielle Ressourcenbereitstellung eines Mapping Gateways in der Metro war eher das kleinere Übel. Technisch gesehen ist es gar nicht möglich, „gemappte“ Verzeichnisse in Windows weiter an andere Rechner „freizugeben“. Hinzu kommen deutlich hohe TOC[129]-(kosten), und fehlendes Wartungspersonal in der Metro. Diese

[129] Total Cost of Ownership – betrifft in diesem Falle die Support- und Personalkosten für den Mapping Gateway

Lösung scheiterte also schon im Ansatz, und setzte somit doch eine Mapping-Tool-Installation auf dem DOM-System und der TC/Link-FI Schnittstelle voraus.

4.7.1 WebDRIVE & Co

Nach längeren Recherchearbeiten im Internet eignete sich WebDRIVE 6.0[130] von Southrivertech für eine Lösung des Problems. Es integriert FTP, SFTP[131] oder WebDAV-Server durch ein Mapping in einen Laufwerksbuchstaben im Windows-Datei-Explorer. Dateien können somit direkt ohne zusätzliche Installation eines Clients per „drag & drop“, oder mit dem Systembefehlen in Windows transferiert werden (vgl. Abbildung 35).

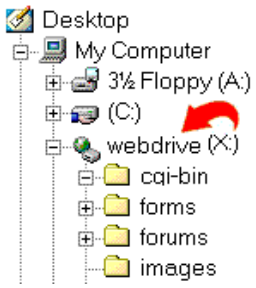

Abbildung 35 WebDRIVE Mapping im Datei Explorer

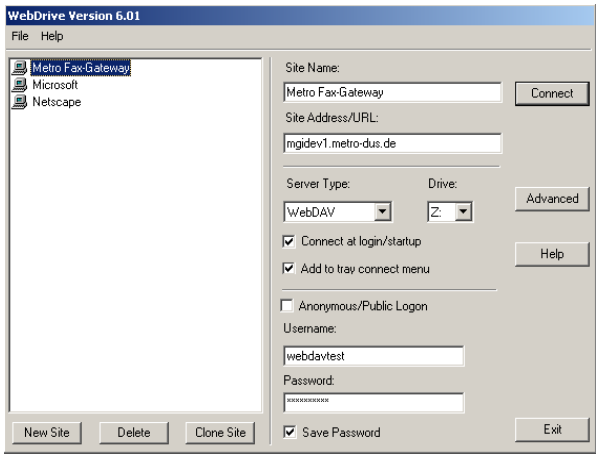

Abbildung 36 Konfiguration von WebDRIVE

WebDRIVE arbeitet als Systemdienst im Hintergrund; Benutzer müssen sich nicht explizit am System anmelden. Die Konfiguration erwies sich auch für Laien als äußerst simpel (s. Abbildung 36). WebDRIVE erstellt in einstellbaren Zeitperioden ein Abbild des Serververzeichnisses in Form eines Caches, der Zugriff auf diesen Cache erfolgt über ein normales Verzeichnis. Schade nur, dass dieses Tool auf dem Topcallserver nicht zum Einsatz kommen konnte. Die Topcall/Link-FI Schnittstelle arbeitet im OSI-Modell[132] eine Schicht tiefer, d.h. bevor überhaupt irgendein Mappingtreiber geladen werden konnte, musste bereits ein Mapping eines Verzeichnisses vorhanden sein.

[130] download unter <http://www.southrivertech.com/download/index.html> (08.02.2004)
[131] verschlüsseltes FTP
[132] Das OSI-Modell teilt die Problembereiche der Netzwerkkommunikation in 7 Schichen auf die aufeinander aufsetzten. Näheres s. <http://de.wikipedia.org/wiki/OSI-Modell> (08.02.2004)

Derzeit lässt sich die TC/Link-FI Schnittstelle ausschließlich über Einstellungen der Registriereinträge in Windows konfigurieren. Dort hat man entweder die Möglichkeit, einen direkten Pfad auf die lokale Festplatte zu vergeben oder sich eines UNC-Pfades zu bedienen. UNC ist eine Eigenentwicklung von Microsoft und wird für Dateifreigaben in Windows-Netzwerken genutzt. Freigeschaltete Verzeichnisse lassen sich somit auch ohne einen Laufwerksbuchstaben in der Form `\\<Rechnername>\<Verzeichnis>` im Explorer direkt einbinden. Diese Funktionalität wurde bereits in der alten Fax-Basisdienstlösung verwendet. Optional bietet ein Windows 2000 Server die Möglichkeit über den Systembefehl „Netuse“[133], ebenfalls auch WebDAV-Verzeichnisse anzubinden. Leider ist dieses Feature noch nicht vollständig von Microsoft implementiert worden. Es ist zwar möglich einen Web-Folder im Windows Datei Explorer an einen Web-Server zu mappen, dieser wird aber nicht in Form eines Laufwerksbuchstabens angegeben. Dieser wird als eine Art temporäres Verzeichnis im Explorer angelegt, und kann nur von Microsoft Applikationen intern verwendet werden.

Mapping-Tools wie „WebDRIVE“ gibt es viele, leider unterstützen die wenigsten das WebDAV Protokoll, und weisen dieselbe Problematik wie oben beschrieben auf. Die Lösung dieser Aufgabe erfolgte durch die Entwicklung sog. Kopierdämonen in Java. Diese erstellen kein Abbild einer Verzeichnisstruktur, sondern verschieben relevante Dateien auf direktem Wege auf den WebDAV-Server oder auf die lokale Festplatte eines Rechners. Alle Kopierdämonen laufen unter dem gleichen Betriebssystem wie auch die Dienste operieren; in diesem Falle ist dies für das DOM- und des TOPCALL-Systems „Microsoft Windows 2000“.

4.7.2 Kopierdämon FST/PD

Dieser Kopierdämon stellt eine Datei-Verbindung zwischen dem TC/Link-FI Verzeichnis `TC_TO_FI` auf dem TOPCALL-Server und einem WebDAV-Server-Verzeichnis `TO_FAX` her. Nach der Installation und dem Start präsentiert sich dieser Dämon mit seinen Einstellungen auf einer Konsole und zeigt seine aktuellen Kopier-operationen an.

[133] <http://www.microsoft.com/resources/documentation/WindowsServ/2003/enterprise/proddocs/en-us/Default.asp?url=/resources/documentation/WindowsServ/2003/enterprise/proddocs/en-us/net_use.asp> (12.04.2004)

In Abbildung 37 ist zu sehen, das dieser einen Report (XXX.PD) inkl. einer Transaktionsdatei (XXX.FST) an die TC/Link-FI Schnittstelle verschiebt.

Abbildung 37 Kopierdämon FST/PD

Eine Anbindung zur Fax-Datenbank musste für diese Aufgabe nicht bestehen, dadurch konnte dieser Prozess auch nicht von einer zentralen Stelle aus durch die „Admin GUI“ überwacht werden, weil ein Logging in die Datenbank vom Dämon nicht durchgeführt wird.

4.7.2.1 Funktionsablauf

Das Prinzip des Dämons war also recht einfach, neue Transaktionsdateien mussten inkl. eines Reports auf die TC/Link-FI Schnittstelle verschoben werden. Doch wie lässt sich die Existenz einer neuen Datei von einer alten prinzipiell feststellen?

Oft werden diese anhand des Erstellungsdatum erkannt. Sollte das aktuelle Dateidatum älter als die aktuelle Systemzeit sein, so ist dies ein Merkmal für eine neue Datei innerhalb einer Verzeichnisses. Fraglich ist nun folgender Sonderfall: Eine Transaktionsdatei und ein Report zu einem Auftrag werden innerhalb der Winterzeit z.B. um 3:00 Uhr vom Fax-Dämon BD313 auf dem WebDAV-Server erstellt, danach erfolgt auf diesem Server eine Sommerzeitumstellung auf 2:00. Nun startet der Kopierdämon seinen Zeit-Vergleich und stellt fest, dass ein Dateidatum in der Zukunft liegt, obwohl die Datei in der Vergangenheit erstellt worden ist. Der Kopierdämon würde nach diesem Prinzip die Dateien, die vor einer Zeitumstellung generiert worden waren, erst nach einer Zeitdifferenz von einer Stunde bearbeiten. Obwohl man denken

würde, dass um 3:00 nachts sicherlich keiner ein Fax verschicken wird, ist dies ein Trugschluss. Alle Fax-Dämonen arbeiten zusammen auf einem Server in Düsseldorf mit der gleichen Systemzeit. Durch die spätere Anbindung der einzelnen Stores (z.B. in Russland) kann aufgrund der Zeitverschiebung durchaus ein Fax in Düsseldorf zu dieser eher unüblichen Zeit verarbeitet werden und über die TC/Link-FI Schnittstelle wieder an den TOPCALL-Line-Server in Russland versendet werden. Insgesamt also keine gute Lösung für zeitkritische Faxe.

Um nun vollständig unabhängig von irgendeinen Datum zu sein, wurde ein spezielle Scantechnik in allen Kopierdämonen entworfen. Im ersten Schritt generiert der Kopierdämon intern alle Dateien in Form einer Liste, die derzeit in einem WebDAV-Verzeichnis vorhanden sind [vgl. Abb. 38, 1].

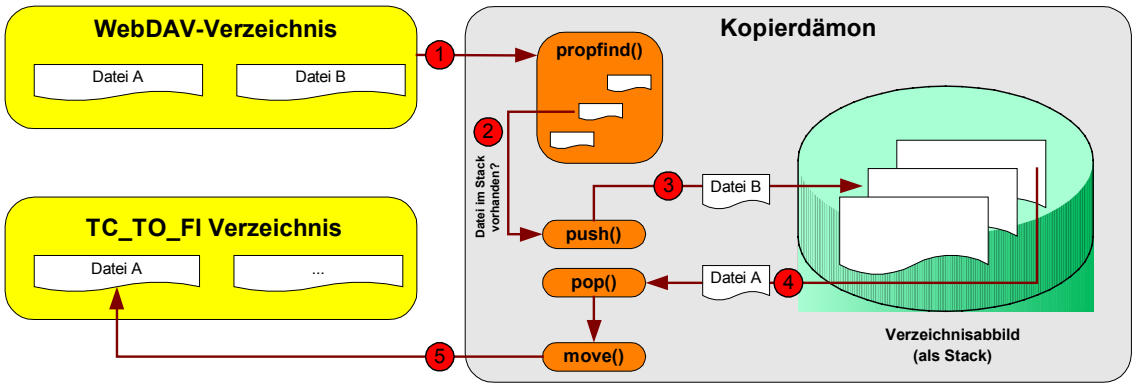

Abbildung 38 Scantechnik der Kopierdämonen

Diese Liste wird nun sequenziell abgearbeitet [2], und ständig mit einen Stack überprüft, der dauerhaft alle neuen Dateinamen inkl. des WebDAV-Pfades speichert; sollte z.B. die Datei „B“ noch nicht im Stack existieren, so wird diese mittels eines `push()` in Java hinzugefügt [3]. Ist die interne Liste durch den WebDAV-Befehl `propfind()` verbraucht, so beginnt der Kopiedämon den Abbildungs-Stack nach dem FIFO-Prinzip mittels eines `pop()` zu entleeren [4], dabei verschiebt er die zurückgelieferten Dateien entgültig auf ein anderes System [5]. Für den FST/PD Dämon ist dies das `TC_TO_FI` Verzeichnis auf dem TOPCALL-Server.

Diese Funktionsweise ist bei allem Kopierdämonen inkl. den Fax-Dämonen prinzipiell gleich.

Leider gibt es aber beim `move()` für diesen Dämon eine Sonderausnahme durch TOPCALL. Die TC/Link-FI Schnittstelle identifiziert Transaktionsdateien immer anhand einer dreistelligen Dateiendung, aus diesem Grunde erhalten diese die Endung `xxx.FST`, und PDF-Reports werden durch den Fax-Dämon BD313 zuvor schon in `xxx.PD` umbenannt. Sollte nun in der internen Liste, die im Kopierdämon durch `propfind()` aufgebaut wird, zuerst eine FST-Datei liefern, so darf diese erst verschoben werden, wenn der entsprechende Report bereits auf dem TOPCALL-Server kopiert worden ist, sonst läuft das TOPCALL-System bei der Generierung eines Faxes auf einen internen Fehler[134].

Zusätzlich listet der `propfind()`-Befehl der WebDAV-Bibliothek in Java je nach verwendeten Betriebssystem Dateien in einen Verzeichnis unterschiedlich auf. Vergleichbar ist dies mit dem MS-DOS Befehl `c:\>dir` in MS-Windows auf einer Konsole. Dieser Systembefehl listet Dateien immer nach den Namen auf, andere Betriebssystem können hier variieren, z.B. kann auch nach einem Erstellungsdatum, oder ggf. erst nach Verzeichnissen sortiert die Liste ausgegeben werden. Um nun sicherzustellen das PD-Dateien als erstes zur TC/Link-FI kopiert werden, war es notwendig gewesen zuerst eine interne Liste durch `propfind()` aufzubauen, und diese sequienziell abzuarbeiten.

4.7.3 Kopierdämon MSG

Um das TOPCALL-System nun vollständig an den Fax-Basisdienst anzubinden, mussten zuletzt natürlich noch die Rückmeldedateien (MSG = Message) auf dem WebDAV-Server im `FROM_FAX` Verzeichnis für den Fax-Dämon BD314 bereitgestellt werden. Hierfür ist ebenfalls wieder ein Kopierdämon zuständig, der auf dem TOPCALL-Server installiert wird.

4.7.3.1 Funktionsablauf

Ein erster Test des Kopierdämons zeigte in Zusammenspiel mit dem Fax-Dämonen BD314 zunächst keinerlei Probleme. Seltsam war es aber, dass einige

[134] die Transaktionsdatei enthält eine Angabe der des entsprechenden Reports. Bei der Verarbeitung dieser ‚wird ein Fax mit hilfe des Reports erstellt – vgl. Listing 12 („ATT:“-Attribut)

Kommunikationsaufträge anscheinend niemals als erfolgreich gekennzeichnet wurden, obwohl Rückmeldedateien von der TC/Link-FI Schnittstelle erzeugt wurden und bereits in das `FROM_FAX` Verzeichnis erfolgreich kopiert wurden.

Folgendes Szenario beschreibt und zeigt zugleich die Lösung dieses Problems

Rückmeldedateien werden in TOPCALL oft „schubweise“ generiert, je nachdem wie schnell Transaktionsdateien im `TC_TO_FI` Verzeichnis auf der TC/Link-FI Schnittstelle bereitgestellt und abgearbeitet werden. Eine Pollrate die durch einen Administrator vergeben wird, bestimmt (ähnlich wie bei dem Fax-Dämonen der Sleep-Time-Wert) in welchen Zeitintervallen dieses Verzeichnis bearbeitet werden soll. Häufig sind zu diesem Zeitpunkt schon mehrere Transaktionsdateien durch den Fax-Dämonen BD313 generiert worden. Angenommen der Fax-Dämon BD314 der später die Rückgabedateien verarbeiten wird, verweilt genau zu diesem Zeitpunkt im „Standbymodus“.

Nun tritt der MSG Kopierdämon in Aktion, und verschiebt alle gefundenen Rückmeldedateien für den Fax-Dämon BD313 (vgl. Abb. 39 – hellgraue Rückmeldedateien) vom `NOTIF` Verzeichnis in das `FROM_FAX` Verzeichnis auf einem WebDAV-Server - Dämon BD314 schläft immer noch, im Gegensatz zu BD313, der immer noch neue Kommunikationsauftrage in der FAX-Datenbank vorfindet und Transaktionsdateien TOPCALL bereitstellen wird.

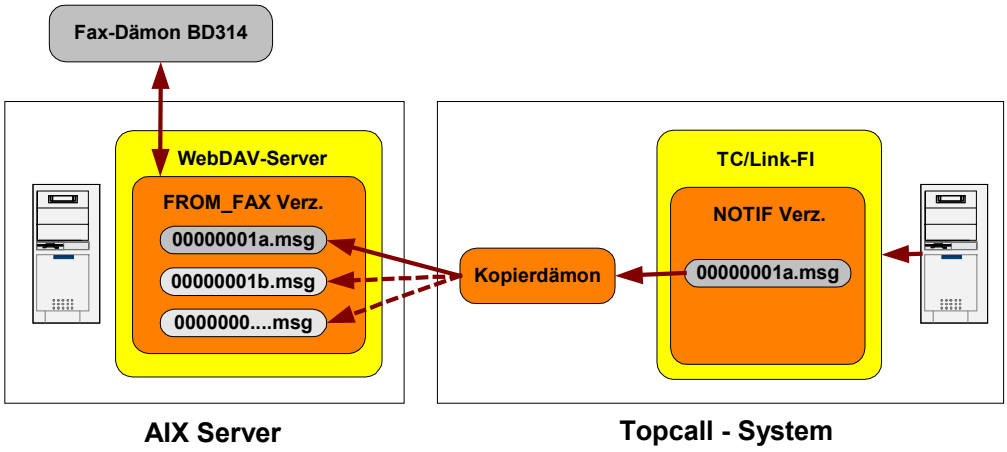

Abbildung 39 Sequenzprobleme der TC/Link-FI Schnittstelle

Zu den neuen Transaktionsdateien gibt es wiederum neue Rückmeldedateien von TOPCALL. Die TC/Link-FI Schnittstelle wird nun ihre Namenskonvention in Form einer Sequenz für die Rückmeldedateien wieder vom Anfang an im *NOTIF* Verzeichnis vergeben (00000001a.msg). Dieser Vorgang TOPCALLs wird sich ständig durch den MSG Kopierdämon wiederholen, denn er sorgt dafür, dass dieses Verzeichnis ständig leer wird. *FROM_FAX* auf dem WebDAV-Server enthält aber bereits eine Rückmeldedatei mit der selben Namensgebung, die nun durch den MSG Kopierdämon überschrieben werden würde (vgl. Abb. 39 – s. dunkelgraue Rückmeldedatei).

Die alte Rückmeldung zu einem Fax-Empfänger wäre in diesem Falle verloren. Zur Lösung musste eine eindeutige Namensgebung der Dateien während eines Kopiervorganges vergeben werden. Optimal geeignet erschien hierfür die Systemzeit des TOPCALL-Servers in Millisekunden. Neue Rückmeldedateien werden nun durch einen Zeitstempel der Form:

```
[Zeitstempel]_[TopcallSequenz].MSG
```

eindeutig dem Fax-Dämon BD314 zur Verfügung gestellt. Unkritisch wird somit der Zeitpunkt der Bearbeitung von Rückmeldungen.

4.7.4 DOM / File.Net Archivierung

Jeden Tag werden viele Entscheidungen getroffen; jede beeinflusst über kurz oder lang den Erfolg eines Unternehmens. Die Qualität einer Entscheidung hängt jedoch sehr oft davon ab, ob die dafür benötigten Informationen auch genau zu dem richtigen Zeitpunkt zur Verfügung stehen. In der Praxis liegen Informationen oft in unterschiedlichen Formen im ganzen Unternehmen verstreut vor. In Gegensatz zu früher gibt sie in den verschiedensten Formaten: Bilder, Audiodokumente, Emails, eingescannte Dokumente, Videodateien u.v.m. All dies wird unter den Begriff Content zusammengefasst. Zur zentralen Verwaltung dieser Dokumente werden Content Management Systeme eingesetzt. In der Metro Group verwendete man bis vor kurzem das DOM-System (Document Output Managment) zur Archivierung von Dokumenten. Dieses System unterstützt das sog. Collaboration Management, d.h. Informationen werden an Benutzergruppen (z.B. Buchhaltung, Einkauf u.s.w.) gekoppelt und können gemeinsam

von diesen verwaltet werden. Ein Client enthält die Funktionalität der Archivierung und einer erweiterten Suche nach diesen Informationen. DOM wurde damals von einer kleinen Softwarefirma entwickelt und enthält speziell angepasste Features eigens für die Metro Group. Infolge der Insolvenz des Unternehmens und des damit nun fehlenden Supports begann die Metro, nach kommerziellen Content Management Produkten zu suchen.

Die Wahl fiel hierbei auf den Content Manager von FileNet[135]. Die zusätzliche Web-Integration ermöglicht zudem eine Archivierung von jedem Arbeitsplatz aus; eine Installation von Clientsoftware entfällt völlig. Leider bot das neue System nur in seinen Grundzügen die Funktionalität des DOMs. Beispielsweise können eingescannte Dokumente (z.B. Rechnungen, Mahnungen, Bestellungen) von einen Netzwerkscanner aus nicht benutzerfreundlich ins System archiviert werden. FileNet bietet aber die Möglichkeit, mit der Open Source Programmierschnittstelle „Open File Net“ hauseigene Features nachzuprogrammieren. Der größte Nachteil vom Content Manager von FileNet im Gegensatz zu DOM ist das Fehlen einer Gruppenzugehörigkeit und bei der Nutzung der Web-Komponente macht sich eine bedeutend schlechte Performance bei der Pflege von vielen Dokumenten bemerkbar. Durch die freien Softwaremodule von „Open FileNet“, wurde das System weitestgehend wieder an das alte DOM angepasst. Abbildung 40 zeigt einen Auszug aus der neu entwickelten Web-Komponente „DMS“. Von hier aus lassen sich alle Dokumente einer Benutzergruppe archivieren, verwalten, ausdrucken oder weiterleiten.

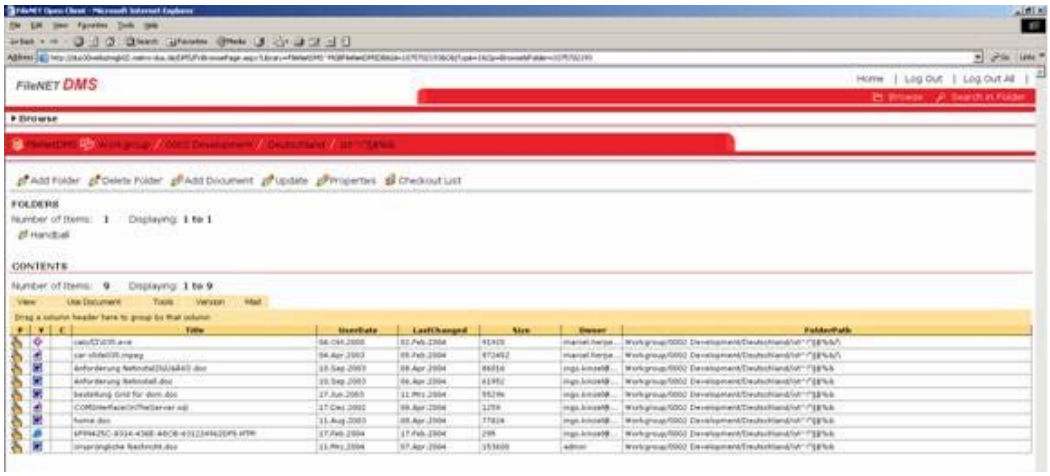

Abbildung 40 File.Net - Archivierungskonsole im Browser

[135] Homepage: <http://www.filenet.com> (16.04.2004)

Warum wird nun ein solches System auch für den Fax-Basisdienst gebraucht?

Kommunikationsaufträge werden durch den Fax-Dämonen BD314 versendet. Dieser gibt zwar eine Auskunft in Form eines einfachen Sendejournals, welches eine Zustellungszeit oder einen Statuscode enthält, jedoch lässt sich mit der Faxe/Listen-Form zu keiner Zeit der Inhalt eines versandten Dokumentes feststellen. Schnell kann es vorkommen, dass versehentlich ein falscher Lieferant in der Faxe/Listen-Form ausgewählt wurde, und einer anderer vergeblich auf seine Bestellungen wartet. Diese Fehler lassen sich aufgrund eines Anlegedatums und der Gruppenzugehörigkeit (Käse-, Fleisch-, Fisch...) der Lieferanten im Archivierungs-System schnell aufspüren. Alleine das Finanzamt schreibt eine lückenlose Protokollierung aller Geschäftsprozesse der Metro Group vor.

4.7.4.1 Kopierdämon DOM/File.Net

Mit Hilfe eines Erweitungsmoduls von File.Net dem „Enterprise Content Management“ – ECM lassen sich Dokumente nicht nur „per Hand“ über einen Browser archivieren, sondern ähnlich wie beim TOPCALL-System mittels optional erhältlicher Transfermodule an ERM-Systeme anbinden. File.Net bietet eine ereignisorientierte Architektur, genannt „Active Content“. Durch diesen ist es möglich, Inhalte zu aktivieren und deren kompletten Bearbeitungszyklus zu steuern. Für den hauseigenen Fax-Basisdienst gibt es diese Transfermodule natürlich nicht. Mit Hilfe von „Open File Net“ Treibern ist es aber damals möglich gewesen, einen Universal-File-Importer in Visual Basic[136] zu schreiben, der Kommunikationsaufträge automatisch ins System einspeist. Um diese alte Lösung weiter zu verwenden, ist es damals wie heute notwendig gewesen, eine Gruppierung der Lieferanten in Form einer Verzeichnisstrukturebene lokal auf der Festplatte des File-Import-Rechners zu speichern. D.h. diese Struktur[137] wurde automatisch auf einem Windows-Server (auf dem auch der Fax-Dämon BD314 lief) zu jedem erfolgreichen Kommunikationsauftrag angelegt, sofern dieser auch archiviert werden sollte. Diese Verzeichnisstruktur konnte damals dank gleicher Betriebssysteme (Microsoft Windows) per Laufwerksmapping auf dem DOM-System abgebildet werden. Dieses Verzeichnis wurde anschließend vom Universal-File-Importer ständig nach neuen Dateien

[136] Programmiersprache von Microsoft
[137] vgl. Abb. 26 Tabelle BD_KOMM_VORGAENGE, Attribut Dom_Verzeichnis (wird durch die Faxe/Listen-Form eingetragen)

durchsucht. Dabei geben der Verzeichnispfad und die Namenskonvention Auskunft, an welcher Stelle ein Kommunikationsauftrag[138] im DOM archiviert werden kann (Abb. 41).

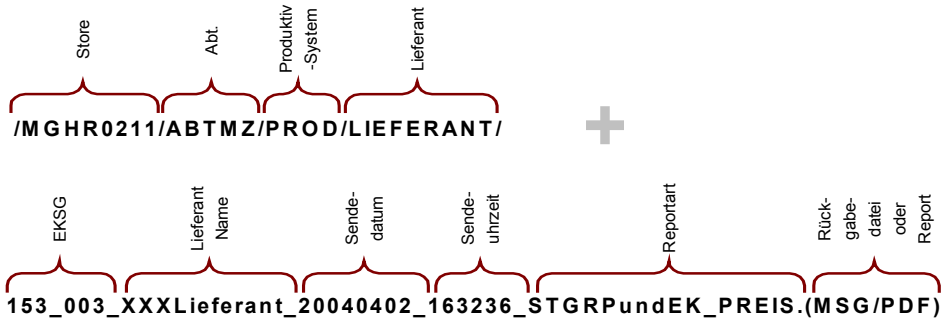

Abbildung 41 Archivierungskonvention für den Universal File Importer

Das Archivierungsprinzip sollte auch für die neuen Fax-Dämonen erhalten bleiben. Diese operieren jedoch nun an zentraler Stelle mit einem WebDAV-Server (Abb. 42).

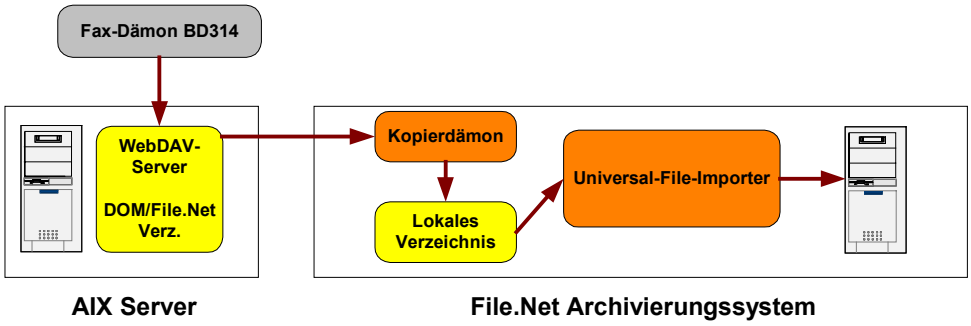

Abbildung 42 Anbindung des Kopierdämons an File.Net

Ein Laufwerksmapping zu FileNet gibt es nicht mehr. Obwohl Mapping-Tools wie WebDRIVE im Gegensatz zu TOPCALL durchaus auf dem System funktionierten würden, sollte auch hier eine einheitliche Lösung mittels eines Kopierdämons in Java entwickelt werden.

4.7.4.2 Funktionsablauf

Der DOM bzw. File.Net-Kopierdämon läuft auf dem Archivierungssystem als Hintergrundprozess. Er stellt eine Dateischnittstelle zu einem WebDAV-Server her, und untersucht das Archivierungsverzeichnis `TO_DOM` rekursiv nach neuen Dateien. Wird

[138] Zu einem erfolgreichen Kommunikationsauftrag werden immer zwei Dateien archiviert: Rückmeldedatei vom TOPCALL-System und der Report selbst.

dieser fündig, so werden Reports und Rückmeldedateien zu einen Kommunikationsprozess inkl. des gesamten Dateipfades verschoben und auf der lokalen Festplatte des File.Net Systems gespeichert.

Abbildung 43 Kopierdämon DOM/File.Net

In Abbildung 43 ist dieser Dämon mit einem Report in Aktion zu sehen. Alle Dateiaktionen (kopieren bzw. löschen von Dateien) werden auf einer Konsole inkl. einer URL[139] ausgegeben. Um eine Konfrontation mit dem Fax-Dämon BD314 auszuschließen, bleiben leere Verzeichnisstrukturen zu Lieferanten auch nach einem Kopiervorgang auf dem WebDAV-Server erhalten.

Folgendes Szenario einmal angenommen

Der DOM/File.Net Kopierdämon hat alle Reports in einem Lieferantenverzeichnis vom WebDAV-Server auf das Archivierungssystem verschoben. Nun wird eigentlich das gesamte Verzeichnis zum Lieferanten auf dem WebDAV nicht mehr gebraucht, weil es bereits leer ist. Sollte in diesem Falle der Kopierdämon versuchen, das Verzeichnis zu löschen, so könnte zur selben Zeit der Fax-Dämon BD314 einen erfolgreichen Kommunikationsauftrag zum gleichen Lieferanten in diese Struktur zu speichern. Dafür erfolgt zunächst eine Abfrage, ob das Verzeichnis schon existiert. Ist dies der Fall so wird sofort mit einer Archivierung begonnen, andernfalls werden die benötigten Lieferantenverzeichnisse erstellt.

[139] Uniform Resource Locator – eindeutige Adresse in einem Netzwerk um eine Datei oder einen Recher ausfindig zu machen.

Dem Fax-Dämon BD314 nun seine Verzeichnisstruktur während eines Kopiervorganges unter den Händen vom Kopierdämon „wegzulöschen“ ist generell eine unsaubere Lösung. Technisch lässt sich dieser Prozess durch Vergabe sog. „LOCK“s auf Verzeichnisse und Dateien vermeiden. „LOCK“s können im WebDAV-Protokoll benutzerabhängig oder global vergeben werden. Fraglich ist nur, wann ein solcher LOCK wieder durch ein „UNLOCK“ freigegeben werden soll. Alle Fax-Dämonen, die zu einem Store angebunden sind, arbeiten mit dem gleichen WebDAV-Benutzer.

Wie sollte ein Fax-Dämon nun wissen, ob nicht bereits ein anderer auf dem gleichen Lieferantenverzeichnis versucht, seine Kommunikationsaufträge zu speichern?

Eine Kommunikation zwischen den Fax-Dämonen ist durch Verwendung einzelner Datenbankinstanzen für jeden Store leider auch nicht möglich gewesen. Die Kopierdämonen enthalten Aufgrund von strategischen Anforderungen nicht mal eine Anbindung an eine Datenbank.

Nach reichlicher Überlegung entschied man sich dazu, leere Lieferantenverzeichnisse nicht durch diesen Kopierdämon zu löschen. Die Performance konnte dadurch ebenfalls verbessert werden – denn Verzeichnisse die bereits existieren müssen nicht noch einmal vom Dämon BD314 angelegt werden.

4.8 Job-Scheduling-Systeme

Job Scheduling ist als Teilbereich des System Managements zu verstehen. Im Kern geht es darum, Jobs unter Einhaltung vorbestimmter Abhängigkeiten zu bestimmten Terminen auf einem oder mehreren Rechnern zu starten und deren Ausführung automatisiert zu überwachen. Jobs können Programmaufrufe, Scriptaufrufe oder ein Betriebssystem-Kommando enthalten. Diese Art der Verarbeitung wird auch als Batch-Produktion oder Stapelverarbeitung bezeichnet. Jobs können auch als Jobströme zusammengefasst werden, die je nach Funktion oder Auftrag an einem bestimmten Datum zu einer bestimmten Uhrzeit gestartet werden können. Verschiedene Hersteller bieten für diese Funktionen spezielle Job-Scheduling-Systeme an, wie z.B. den Workload Scheduler von Tivoli (IBM), der auch in der Metro eingesetzt wird.

Die komplette Funktionalität dieses Scheduling-Systems ist enorm umfangreich, daher werden nur die drei Hauptmerkmale beschrieben - detailreichere Informationen sind dem aktuellen Tivoli-Handbuch[140] (Version 8.0) zu entnehmen.

Tivoli Workload Scheduler-Steuerkomponente

Hierbei handelt es sich um eine Steuerkomponente, die für die Terminierung von Prozessen zuständig ist. Bei großen Konzernen sind mehrere tausend Batch Jobs keine Seltenheit. Als Strukturierungsmittel können einzelne Jobs zu sog. Jobnetzen zusammengefasst werden, die in der Praxis auf mehrere Rechner verteilt werden. Dieses Netz wird in der Praxis als Workload Scheduler-Netzwerk bezeichnet. Die Trivoli Scheduler-Steuerkomponente wird auf jedem Computer eines Workload-Netzwerks ausgeführt und arbeitet einen Workload innerhalb einer verteilen Umgebung ab. Bei der Installation innerhalb eines Tivoli Netzwerkes muss die jeweilige Aufgaben konfiguriert werden, die die Funktionsstelle innerhalb des Terminierungsnetzes an einem Rechner übernehmen soll. Diese kann neben der Terminierung von Jobs auf einer Maschine auch als sog. Agent tätig sein, der Jobstati aufgrund von Prozessabhängigkeiten an andere Steuerkomponenten innerhalb eines Netzwerkes weiter verteilt. (z.B. kann ein

[140] <http://publib.boulder.ibm.com/tividd/td/TWS/GH19-4539-00/de_DE/PDF/GH19-4539-00.pdf> (09.04.2004)

Rechnungsausdruck erst erfolgen, wenn die notwendigen Kalkulationsdaten übertragen worden sind). Alle existierenden Steuerkomponenten werden mit einem einzigen Steuerpunkt der sog. zentralen Master-Steuerkomponente verbunden, die eine Menge an benutzerdefinierten Jobs innerhalb einer Datenbank bereitstellt und durch die Job Scheduling Console verwaltet wird.

Job Scheduling Console

Die Job Scheduling Console (s. Abb. 44) ist eine javabasierte grafische Benutzerschnittstelle, die als Framework für die Erstellung und Verwaltung von Job-Aufträgen innerhalb einer Datenbank dient. Außerdem ermöglicht sie das Steuern und Überwachen von Jobs.

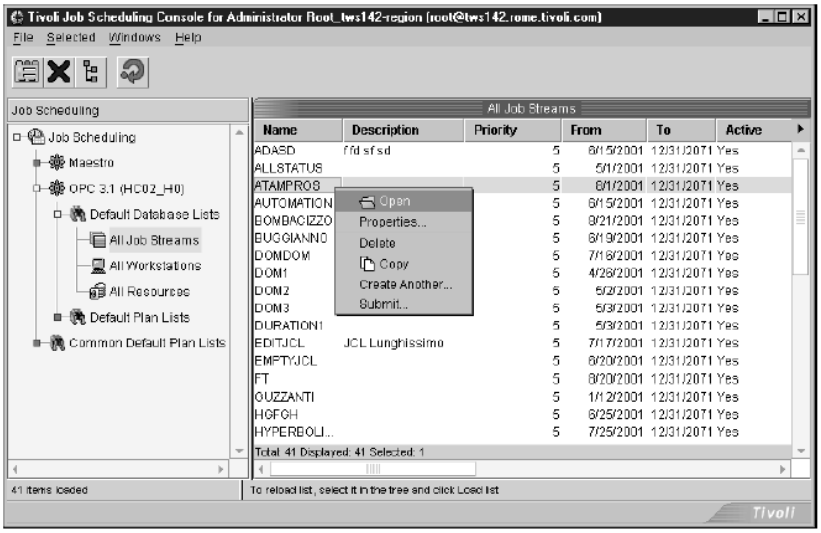

Abbildung 44 Auflistung von Job-Strömen innerhalb einer Datenbank

Sie muss nicht auf derselben Maschine wie die Tivoli Workload Scheduler-Steuerkomponente oder der Tivoli Workload Scheduler-Connector installiert werden.

Tivoli Workload Scheduler-Connector

Der Datenverkehr zwischen der Scheduler-Steuerkomonente und der Job Scheduling Console wird von einem sog. Connector übernommen. Er ordnet der Tivoli Workload Scheduler-Steuerkomponente die Befehle von der Job Scheduling Console zu. Der

Connector wird an der Master-Funktionsstelle und auf allen zu überwachenden Maschinen ausgeführt.

4.8.1 Anbindung der Fax-Dämonen

In der Metro werden mit dem Tivoli-System hauptsächlich kritische Prozesse in Produktivsystemen überwacht. Hierzu zählen insb. Dämonen, die verkaufte Artikelpositionen eines Marktes innerhalb eines Stores transferieren. Dieser Vorgang geschieht meist erst nach Geschäftschluss und kann mehrere Stunden andauern.

Der Tivoli Workload Scheduler kann neben vielen Komponenten eines ERP-Systems auch eigenentwickelte Programme anhand eines Rückgabecodes überwachen. Dieser wird in Form eines Exit-Status von einem Prozess vergeben, der bei seiner Beendigung an den aufrufenden Prozess (meist ein Startskript-Skript, vgl. dazu RET Parameter Zeile 15 in Listing 17) zurückgibt. Beendet man ein Programm ohne die Rückgabe eines Exit-Status, so ist dieser undefiniert, was andere Prozesse wie z.B. den Tivoli Workload Scheduler, die sich auf diesen verlassen, in Schwierigkeiten bringen kann. In der Vorversion von Tivoli Workload Scheduler wurde ein Job als erfolgreich ausgeführt betrachtet, wenn er mit dem Rückkehrcode Null beendet wurde, und als fehlgeschlagen, wenn er mit einem Rückkehrcode ungleich Null zurückgegeben wurde. In der neuen Version ist es jetzt möglich, einen logischen Ausdruck zu definieren, um darzustellen, welche Rückkehrcodes als erfolgreich betrachtet werden sollen. Dadurch, dass der Job als erfolgreich ausgeführt oder fehlgeschlagen definiert werden kann, kann der Ausführungsablauf[141] des Jobs, abhängig vom Ergebnis der Jobausführung, flexibler gesteuert werden. Die neuen Fax-Dämonen werden durch Shellskripte gestartet, die nur zwei Rückgabewerte kennen:

- „0“ eine erfolgreiche Terminierung oder mit
- „1“ das Auftreten eines Fehlers.

Theoretisch ließe sich jeder Fehler individuell durch einen Rückgabecode beschreiben.

[141] z.B. autom. Neustart bei auftretenen Fehlern, oder alternativ Fehlermitteilung an einen Systemadministrator durch ein Job-Scheduling-System

Leider sind je nach verwendetem Betriebssystem einige schon vorbelegt, so z.B. bei MS-Windows.

Status-Code	Fehlerbeschreibung
2	File not found
3	Path not found
5	Access denied
6	Invalid handle
8	Not enough memory
10	Invalid environment
11	Invalid format
18	No more files

Listing 22 Mögliche Rückcodes in Microsoft Windows

Obwohl die Fax-Dämonen derzeit auf einem AIX-Rechner unter UNIX operieren, könnten diese später aufgrund der Plattformunabhängigkeit praktisch auf jedem anderen Betriebssystem laufen, das wiederum andere vordefinierte Rückgabecodes aufweist. Aus diesem Grund wird die eigentliche Ursache eines auftretenden Fehlers in einer Fehlertabelle im Fax-Basisdienst (vgl. Seite 48 – Tabelle `C98_INFO_MESSAGES`) oder in einer Log-Datei festgehalten.

Im Workload-Scheduler werden später vom Administrator pro Store 4 Jobs angelegt,

- Start-/Stopskript für Fax-Dämon BD313
- Start-/Stopskript für Fax-Dämon BD314

die jeweils zu festgelegten Datenbankbackup-Zeiten ausgeführt und überwacht werden.

Das Job-Scheduling-System von Tivoli ist ein sinnvolles Tool zur Überwachung einzelner Prozesse, es kann jedoch nicht den aktuellen Zustand der einzelnen Fax-Dämonen erkennen. Im Fehlerfalle oder zur Laufzeit ist es aber durchaus von Interesse festzustellen, mit welchen Fax-Kommunikationsvorgang die Dämonen beschäftigt sind; zu diesem Zwecke wurde eine grafische Benutzeroberfläche, die oben bereits erwähnte Admin-GUI entwickelt.

4.9 Die Admin GUI

Sie ist ein eigenständiges Zusatzprogramm und überwacht die aktuellen Arbeitsschnitte der Dämonen zur Laufzeit. Durch eine Entwicklung in Java lässt sie sich ebenfalls auf jedem Rechner einsetzten, und ist damit so flexibel wie die Fax-Dämonen. Der Name „Admin“ entstand ursprünglich gesehen aus dem schon teilweise realisierten Funktionsumfang für die Administration der Initialisierungsparameter der Fax-Dämonen in der GUI; dieses Feature sollte leider nach Absprache mit den Basisdienst-Mitarbeitern wieder entfernt werden, um einen evtl. Missbrauch zu vermeiden. D.h. ein direkter Schreibzugriff im Common Dienst auf systemkritische Parameter ist durch die Admin GUI nicht mehr möglich. Die erforderlichen Parametereinstellungen für die Dämonen werden ausschließlich in einem Installationsskript[142] in die Fax-Datenbank eingespielt. Diese Patchenspielungen müssen zuvor immer von der Qualitätssicherung der Metro freigegeben werden. Evtl. Änderungen für eine MGI-Applikation erfolgen spätere durch das gleiche Procedere.

4.9.1 Bedienung

Die Admin GUI überwacht die globalen Initialisierungsparameter der Fax-Dämonen die für jeden Mandanten im Common Service hinterlegt sind. Gleichzeitig untersucht sie die Basisdienst Tabelle `BD_Daemon_Status` in dem von jedem Dämon der aktuelle Arbeitseintrag geloggt wird. Nach dem Start der Admin GUI ist es zuerst erforderlich

Abbildung 45 Admin GUI - Anmeldung

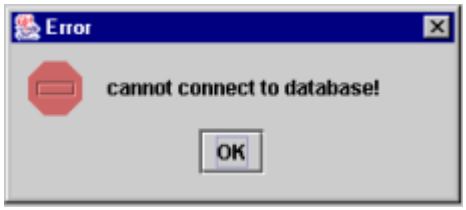

Abbildung 46 Admin GUI - Fehlerhafte Anmeldung

[142] eine Einspielung erfolgt mittels normaler Dateien die eine SQL-Syntax enthalten.

sich mit einer Mandantendatenbank (Store) eines Fax-Basisdienstes zu verbinden. Hierfür erscheint als erstes ein Loginfenster (vgl. Abb. 45) indem alle notwendigen Eingaben erfasst werden müssen. Nach einer erfolgreichen Anmeldung öffnet sich die Admin GUI; in Form von Karteireitern stehen einem Administrator nun folgende Funktionalitäten zur Verfügung:

- **Watcher**, protokolliert alle Arbeitsvorgänge eines Fax-Dämons (wahlweise einzeln oder gleichzeitig)
- **Processes**, ermöglich eine manuelle Terminierung der Dämonen
- **Configuration**, zeigt die aktuellen Initalisierungsparameter der Dämonen an

Jeder Fax-Dämon BD313 o. BD314 lässt sich gleichzeitig, durch eine Unterteilung im Fenster beobachten.

Um mögliche fehlerhafte Einstellungen zu lokalisieren, öffnet sich zu Beginn direkt der „Configuration"–Reiter, (s Abb. 47) indem sich alle Initialisierungsparameter[143] des Common Service für die Fax-Dämonen wiederfinden. Wohlgemerkt sollten die erforderlichen Shellskripte zum Start eines Dämons im Fehlerfalle mit überprüft werden, denn nur dort werden Autorisierungsparameter wie Benutzername und Passwort hinterlegt – diese können und sollen nicht von der Admin GUI ausgelesen werden

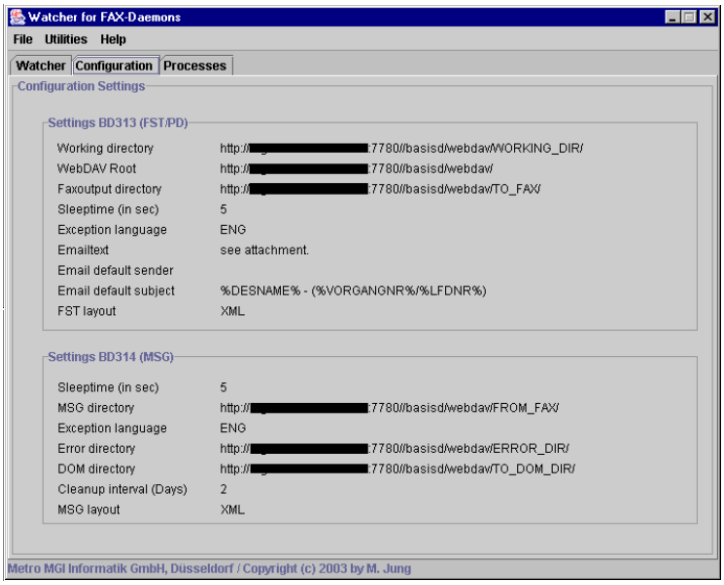

Abbildung 47 Admin GUI - Reiter „Konfiguration"

[143] vgl. Kapitel der Fax-Dämonen

Der interessanteste und wichtigste Teil bei einer Fehlersuche oder zur Laufzeitüberwachung ist der sog. „Watcher“-Reiter. Dieser ließt anhand der globalen `BD_Daemon_Status` Tabelle im Basisdienst, ununterbrochen den aktuellen Verarbeitungsrozess ab.

Abbildung 48 zeigt nun den Watcher-Reiter für den Store `v1t` in Aktion. Mittels eines Start/Stop-Knopfes lässt sich die Protokollierung für jedem Fax-Dämon (BD313, BD314) getrennt starten oder stoppen - bei einem Stopp und erneutem Start wird zusätzlich der Inhalt des entsprechenden Log-Fenster gelöscht; falls der Inhalt mehr als 5000 Zeichen enthält, wird dieser auf die Hälfte (5000/2) reduziert, um somit die Speicherressourcen des Rechners zu sparen und die Admin GUI auch tagelang durchlaufen zu lassen.

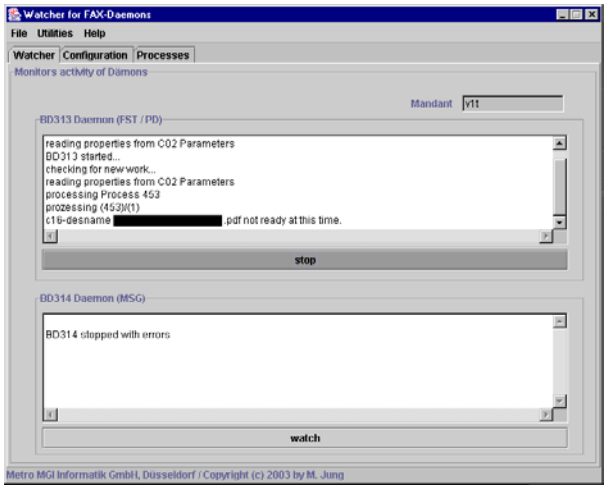

Abbildung 48 Admin GUI - Reiter „Watcher“

In der Beispielabbildung 48 ist nun zu sehen, dass der Fax-Dämon BD313 gestartet worden ist `BD313 started` und seine Initialisierungsparameter `reading properties from C02 Parameters` aus dem Common Service ausliest. Als nächstes findet dieser einen Kommunikationsauftrag aus der Fax-Datenbank `prozessing Process 453` und verarbeitet hierzu das erste Dokument `... (453/1)` . Leider ist der Reportserver noch mit der Generierung des Reports beschäftigt oder ist evtl. sogar ausgefallen, aus diesem Grunde kann der BD313 den Auftrag noch nicht abarbeiten `c16-desname XXX.pdf not ready at this time.` Sollten mehrere Reports für diesen Auftrag zu unterschiedlichen Lieferanten eingetragen sein, so wird zuerst versucht diese zuzustellen. Sollte auch dieser Versuch fehlschlagen, so wird der nächste Kommunikationsauftrag aufgegriffen.

Im Fenster des „Watchers“ ist zu erkennen, dass ein Fehler in BD314 aufgetreten ist. Die Folge ist nun, dass die Protokollierung automatisch abgebrochen wird und direkt ein Fehler gemeldet wird (s. Abb. 49), mit dem Hinweis diesen genauer in der Fehlertabelle *C98_INFO_MESSAGES*[144] zu lokalisieren.

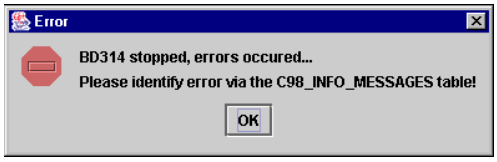

Abbildung 49 Admin GUI - Fehlermeldung

In diesem Beispiel wurde eine Transaktionsdatei im ungültigen Format von der TC/Link-FI Schnittstelle in Form einer ungültigen[145] Rückmeldedatei zurückgeweisen. Dieser Fall zeigt eindeutig den Nachteil eines einigen TOPCALL-Testsystems in der Metro Group. Die Schnittstelle kann immer nur ein Format XML/TOM verarbeiten[146], jedoch nicht zwei gleichzeitig (Fax-Dämon-BD313 und zusätzlich einen Qualitätssicherungsbeauftrageten der Metro der eine alte MGI Applikationen dort testet) Des weiteren ist es mit der Admin GUI zusätzlich möglich, die Fax-Dämonen auf einem Server zu terminieren. Der Funktionsablauf ist analog zu den Start- und Stop Shellskripten[147], die bei den Dämonen verwendet werden. Dies eröffnet die Möglichkeit beim Ausfall oder der Fehlkonfiguration eines Job-Scheduling-Systems im Notfall zu umgehen und eine Terminierung der Fax-Dämonen von Hand zu starten. Hierfür ist der „Processes“-Reiter in der GUI verantwortlich (s. Abb. 50).

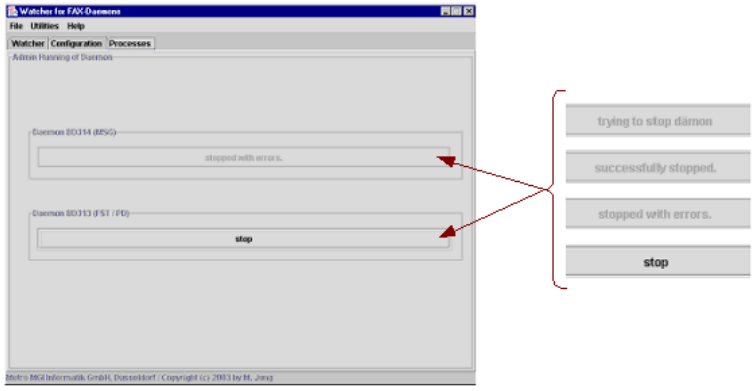

Abbildung 50 Admin GUI - Reiter „Processes“

[144] vgl. C98_INFO_MESSAGE Tabelle in “Logging Bibliotheken”
[145] diese Rückmeldedatei kann keinen Kommunikationsvorgang in der Fax-Datenbank zugeordnet werden
[146] durch Konfigurationsparameter im TOPCALL-System festgelegt
[147] vgl. Shellskripte und Stati in 4.2 “Datenbankbackup“

Nach einem erfolgreichen Start der Dämonen, erscheinen für jeden (in schwarzer Schrift hinterlegt), sog. „Stop“-Buttons. Auf Wunsch eines Administrators kann nun eine Terminierung durch ein Anklicken dieser erfolgen. Der jeweilige Button wird danach in seiner Funktion inaktiv und zeigt zugleich die letzten Laufstatus der Dämonen in Form eines grauen Textes in folgender Reihenfolge an:

- **“trying to stop dämon”**, wird unmittelbar nach dem Terminierungswunsch (durch ein Job-Scheduling-System oder per Hand mittels der Admin GUI) eingeblendet. In dieser Phase verarbeitet der jeweilige Dämon die letzten Kommunikationsaufträge und versucht sich danach zu beenden.
- **“successfully stopped”**, gibt Auskunft über die fehlerfreie Beendigung eines Dämons. Der Erfolg wird zusätzlich mit einen Meldungsfenster bekannt gegeben (s. Abb. 51).

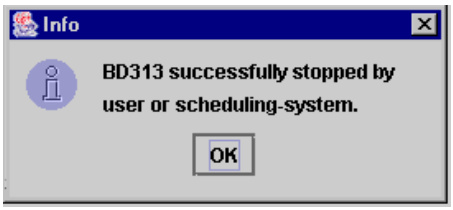

Abbildung 51 Admin GUI - Fehlerfreie Terminierungsmeldung

- **“stopped with errors”**, zeigt an, dass sich ein Dämon aufgrund eines Fehlers nicht ordnungsgemäß beendet hat. Es erfolgt anschließend ein Meldungsfenster (vgl. hierzu Abb. 49)

Mit der Admin GUI kann eine schnelle und einfache Überwachung der Fax-Dämonen vorgenommen werden, auch wenn sich keine Client-Datenbank-Applikationen wie dem SQL-Navigator[148] oder TOAD[149] auf einem Rechner befindet. Oft wurde behauptet, eine Protokollierung könne doch mit Hilfe dieser Software und einen simplen Quick-Browse[150] der `BD_DAEMON_STATUS` Tabelle erfolgen. Dieses Vorgehen zeigt jedoch nur einen Schnappschuss der aktuelle Sachlage, und verschafft im Gegensatz zur „Admin GUI“ keinen Gesamtüberblick aller aufgetretenen Log-Nachrichten.

[148] download unter <www.quest.com/sql_navigator> (19.04.2004)
[149] download unter < www.quest.com/toad> (19.04.2004)
[150] zeigt den gesamten Inhalt einer Tabelle an und ermöglicht ein benutzerfreundliches „durchblättern“ (browsen) der Daten.

Resümee

Das Ziel der Arbeit war die Neuentwicklung der Fax-Dämonen für den bereits bestehenden Fax-Basisdienstes BD300. Anstatt Kommunikationsaufträge ausschließlich per Fax an einen Lieferant zu versenden, wird nun auch eine Email-Variante unterstützt. Die Entwicklung und entsprechende Tests der Dämonen durch die Qualitätssicherung sind bereits beendet; der neu implementierte Fax-Dienst kann also Anfang Juli 2004 in der Metro in die Produktion gehen.

Durch die bereits existierenden Fax-Dämonen die in Oracle Forms realisiert wurden und bestehende Fax-Datenbankschemata, konnte schnell ein „Roter Faden“ in der Businesslogik des Fax-Basisdienstes BD300 aufgefunden werden, der eine Neuentwicklung der Module in Java zum Teil unterstütze. Wer jedoch meint, es wäre problemlos gewesen, die gleiche Logik in andere Programmiersprachen zu übertragen, hat weit gefehlt. Hauseigene Modulentwicklungen der MGI, die bereits für eine Grundstruktur in Oracle Forms Applikationen benutzt werden, gab es natürlich noch nicht in Java. In erster Linie zählen hierzu die Basis- und Commondienste, deren Anbindung in Wrapper-Klassen gekapselt wurden mussten. Die SQLJ Technologie sollte diese einfältige Arbeit schnell und kostengünstig meistern. Leider zeigte sie gravierende Schwächen, die einen Einsatz nicht zuließen. Diese fehlenden Funktionalitäten komplett „von Hand“ zu programmieren, beanspruchte viel Zeit. Bereits bewährte Lösungen wie die Fax-Datenbank mussten durch die Faxe/Listen-Form beibehalten werden, obwohl sie nach vielen Konferenzgesprächen an einigen Stellen hätte optimiert werden könnten. Viel Entfaltungsmöglichkeiten für neue Ideen gab es aufgrund eines bereits bestehenden Anwendungssystems[151] leider nur sehr wenig.

Die Herausforderung dieser Arbeit bewegte sich in der Konzeptplanung und der Einarbeitung in die bereits entwickelten Softwaremodule der MGI. Wöchentliche Meetings waren unbedingt für das Verständnis der Basis- bzw. Commondienste und der Konzeptplanung des neuen Fax-Dienstes notwendig. Bis zur ersten neuen Quellcodezeile in Java verstrich somit mehr als ein Monat – Zeitintervalle, die bei großen Softwareentwicklungen durchaus üblich sind. Schwierig gestaltete sich auch

[151] vgl. Kapitel 1.5 „Die Faxe/Listen-Form der MGI“

immer ein Testen der gesamten Fax-Applikation. Neue Softwareversionen[152] der Basis- und Commondienste werden oft ohne böse Absichten eines Entwicklers in Testdatenbanken eingespielt, auf denen ebenfalls die neu entwickelten Fax-Dämonen operieren. Durch die unterschiedlichen Versionsstände bekamen die Fax-Dämonen z.T. unterschiedliche Rückgabewerte in Methoden zurück, die sie laut Konzept korrekt verarbeiteten. Faxe wurden in diesem Falle aber dennoch nicht erzeugt. Eine anschließende Fehlersuche und einen Verantwortlichen hierfür zu finden war oft schwierig, weil Einspielungsinformationen in der Fax-Datenbank nicht benutzer-abhängig mitgespeichert werden.

Die provisorische Entwicklung der Kopierdämonen wird in Zukunft durch die Integration eines WebDAV-Protokolls im TOPCALL-System abgelöst, dadurch lässt sich der Supportvertrag auch auf die WebDAV-Anbindung mit übertragen. Die Entwicklungszeit wird hierfür voraussichtlich 2 Monate betragen, in dieser Zeit müssen die Kopierdämonen ihren Dienst beweisen. Leider kommen die besten Ideen immer zum Schluss, denn eine durchaus praktikable Alternative zu WebDAV bietet neben der TC/Link-FI Schnittstelle auch der Link-Server. Es besteht die Möglichkeit über ein sog. SOAP[153] Protokoll zu kommunizieren. SOAP ist ein XML-basiertes Protokoll und stellt einen einfachen Mechanismus zum Austausch strukturierter und typisierter Information zwischen den Rechnern in einer verteilten Umgebung dar. Es beschreibt unter Verwendung von Kodierungsregeln die innere Struktur einer Anwendung und kann in einer Vielzahl von Systemen bis hin zu Remote Procedure Calls[154], eingesetzt werden. Um diese Technik zu nutzen muss hierfür zuerst ein sog. SOAP-Server entwickelt werden, der über dieses Protokoll generierte Reports zu Kommunikationsaufträgen von einem zentralen WebDAV-Server auffindet und durch SOAP weiter an den TC/LINK-Server leitet. Leider konnte zur Konzeptplanungszeit der Fax-Dämonen diese Technologie nicht berücksichtigt werden, denn entsprechende TOPCALL-Dokumentationen lagen nicht vor.

Zuletzt möchte ich noch erwähnen, dass diese Diplomarbeit bewusst in einem etwas anderen Stil verfasst wurde als üblich. Die Strukturierung, Erklärweise und Techniken

[152] bzw. die MGI als sog. „Releasestände“, die in Versionsmanagmentsystemen wie PVCS verwaltet werden.
[153] Einführung unter <http://swt.wi-inf.uni-essen.de/~msze/Material/SOAP_Folien33.pdf> (13.04.04)
[154] Remote Procedure Calls sog. „RPCs“ werden eingesetzt um entfernte Funktionsaufrufe auf fremden Rechnern über ein Netzwerk auszuführen.

wurden absichtlich einfach gestaltet, um sowohl einem Laien als auch einem Informatiker ein Verständnis dieser Arbeit zu geben.

Schade nur, dass der gesamte Fax-Basisdienst inkl. der neuen Fax-Dämonen eine komplette Eigenentwicklung der Metro Group ist und sich daher niemals auf andere Unternehmen übertragen lässt. Möglicherweise ist aber das Konzept für andere Firmen ein Anreiz, analoge Faxgeräte aus den Büros durch integrierte Systeme wie TOPCALL abzulösen.

Manuel Jung

Anhang

Die entwickelte Software in dieser Arbeit unterliegt dem Urheberrecht der Firma :

METRO Group - Information Technology GmbH
Metro-Strasse 12
40235 Düsseldorf

A Inhalt der CD-ROM

Die beigefügte CD-ROM enthält ergänzende Unterlagen zur Diplomarbeit. Hierzu zählen projektbezogenen Sourcen, die in der Entwicklungszeit gelesen, weiterentwickelt und fertiggestellt wurden. Der Inhalt ist in folgender Verzeichnisstruktur gegliedert:

🗁 BD300 Fax-Dienst

Enthält alle Projektbezogenen Dateien des Fax-Basisdienst BD300 in Form eines MGI-Releasebaumes. Diese enthalten für eine Datenbankinstallation sowohl eine Komplett- als auch eine Patch-Installation in Form einer Verzeichnisstruktur. Die Entwicklung und Anpassung dieser Dienste erfolgte analog zu den Java-Fax-Dämonen.. Jedes Common- und Basisdienstemodul lässt sich in die folgende Ordnungsstruktur aufsplitten :

	Ordner	Script / Sourcen
🗁	admin	Erstellt einen Datenbankbenutzer
🗁	chk	Erzeugt Unique-Constraints für Datenbanktabellen
🗁	client	Oracle Forms Module für die Client-Seite
🗁	data	füllt Tabellen mit Inhalten
🗁	doc	Dokumentationen zum Dienst/Modul
🗁	grn	Berechtigungen des Modules
🗁	msg	Fehler- und Informationsmeldungen für den Dienst
🗁	pck	Modul Packages
🗁	pky	Primärschlüssel von Datenbanktabellen
🗁	script_nt	Startskripte für einen Windowsbasierten Rechner
🗁	script_unix	Startskripte für einen Unixbasierten Rechner
🗁	syn	Synonyme für Datenbankpackages
🗁	tab	Anlegeskript Datenbanktabellen
🗁	uky	Constraints für Datenbanktabellen

Datenschutzrechtliche Gründe der METRO Group untersagen leider eine vollständige Auflistung des gesamten Fax-Basisdienst Sourcecodes. Teilmodule und komplexe Forms Anwendungen wurden daher im Projektbaum entfernt.

🗁 Java

In diesem Verzeichnis sind die Sourcen inkl. aller erforderlichen Bibliotheken der neu entwickelten Java-Fax-Dämonen in einer MGI-Packagestruktur wiederzufinden.

🗁 WebDRIVE

Installationsdatei (Microsoft Windows) für WebDRIVE von South River Technologies, welches im Kapitel 4.7 “WebDRIVE & Co” leider nicht zu seinem Einsatz kommen konnte. Aktuellere Versionsstände sind im Internet unter <http://www.webdrive.com/products/webdrive/index.html> (20.04.04) erhältlich.

🗁 Java WebDAV dav4j

Java Bibliothek für die Anbindung der Fax-Dämonen zu einem WebDAV-Server.

🗁 Topcall

Um eine vollständigen Überblick der Funktionalität und Features des TOPCALL-Systems zu bekommen, sind hier alle Handbücher wiederzufinden die während der Arbeit zur Verfügung standen. Messestandinformationen der CeBit `04 sind in einem Unterverzeichnis enthalten.

🗁 JDeveloper Ver_9040

Umfangreiche Entwicklungsumgebung für Java™ Programme von Oracle, die fast keine Wünsche offen lässt. Inzwischen ist die aktuellere Version 10g auf dem Markt erhältlich, die auch zur Entwicklung in der MGI eingesetzt werden sollte. Leider scheint diese Version z.Z. noch nicht voll ausgereift. Kleinere Bugs behinderten oft die Arbeit, die Vorgängerversion 9.0 zeigte diese Schwächen jedoch nicht.

Literaturverzeichnis

Gedruckte Literatur

[1] **Graham Hamilton, Rick Cattell, Maydene Fisher** – JDBC Database Access with Java, Addison-Wesley Verlag, 1998.

[2] **Charles F. Goldfarb, Paul Prescod** – Das XML-Handbuch Addison-Wesley Verlag, 2000.

[3] **Christian Ullenboom** – Java ist auch eine Insel, Galileo Computing Verlag, 2001

[4] **Guido Krüger** – GoTo Java 2 - Addison-Wesley Verlag, 2000

[5] **TOPCALL** – TC/LINK-FI Technical Manual

Version 2.06.04, 2003

[6] **TOPCALL** – Open Message Format Technical Manual Version 2.06.04, 2003

[7] **TOPCALL** – TC/SOAP Version 1.00.03, 2003

[8] **Kevin Loney, George Koch** – ORACLE9i Die umfassende Referenz, Hanser Verlag, 2002

[9] **Oracle University Schulungsunterlagen** – Oracle Forms Developer, K1500 V1.0, Oracle, 2001

[10] **Jan Holz** – „Samba“ - Das Einsteigerseminar, bhv Verlag, 2002

[11] **Helmut Holz, Bernd Schmitt, Andreas Tikart** – Linux Server, bhv Verlag 2001

Sonstige Quellen

[1] **CeBit 23. März 2004** - TOPCALL Messestand (Halle 13, Stand C58)

Internetquellen

Alle Quellenangaben aus dem Internet waren zum Abgabezeitpunkt dieser Arbeit gültig.

[1] **Wikipedia** – Freie Enzyklopädie im Internet – http://www.wikipedia.de

[2] **Tivoli** - Produktpalette Überblick <http://www.electronicoffice.de/pdf/IBM/IBM_SW_Guide_Tivoli.pdf> (03.04.2004)

[3] **Tivoli** - Workload Schedulers Produktbeschreibung <http://as400bks.rochester.ibm.com/tividd/td/TWS/SC32-1256-00/de_DE/HTML/eqqg1mst02.htm#ToC_61> (10.04.2004)

[4] **Tivoli** - Workload Scheduler Handbuch <http://publib.boulder.ibm.com/tividd/td/TWS/GH19-4539-00/de_DE/PDF/GH19-4539-00.pdf> (09.04.2004)

[5] **FileNet** - Content Manager Broschüre <http://www.filenet.com/deutsch/Produkte/datenblatt/030420005.pdf> (13.04.2004)

[6] **o.V.** – JDBC Connection Pooling <http://www.datadirect-technologies.com/products/jdbc/docs/ConnectionPoolingTB-.asp> (20.02.2004)

[7] **Torsten Horn** - WebServices mit SOAP <http://www.torsten-horn.de/techdocs/soap.htm> (19.02.2004)

[8] **o.V.** - Was ist SOAP? <http://www.xml-rpc.de/soap/index.xhtml> (12.04.2004)

[9] **John Campbell** – SQLJ Transformierung **- <**http://www-106.ibm.com/developerworks/db2/library/techarticle/0202campbell/0202campbell.html?open&l=904,t=grdb> (10.03.2004)

[10] **o.V.** – WebDAV Resources <http://www.webdav.org> (26.04.2004)

[11] **Mela Eckenfels** – WebDAV <http://www.mela.de/Vortraege/WebDAV> (26.04.2004)

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Seiten, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Bergisch Gladbach, den 28. April 2004

Manuel Jung