# Graphbasiertes Reengineering von Telekommunikationssystemen

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften
der RWTH Aachen University zur Erlangung des akademischen
Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker

Christof Mosler

aus Ruda Slaska, Polen

Berichter: Universitätsprofessor Dr.-Ing. Manfred Nagl
Universitätsprofessor Dr. Uwe Aßmann

Tag der mündlichen Prüfung: 17. April 2009

Berichte aus der Informatik

**Christof Mosler**

# Graphbasiertes Reengineering von Telekommunikationssystemen

Shaker Verlag
Aachen 2009

**Bibliografische Information der Deutschen Nationalbibliothek**
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: D 82 (Diss. RWTH Aachen University, 2009)



Printed in Germany.

ISBN 978-3-8322-8240-0
ISSN 0945-0807

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen
Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

# Zusammenfassung

Bis zu 80 Prozent der Ressourcen im Lebenszyklus eines Softwaresystems entfallen auf die Wartung und das Reengineering. Zwar existieren viele Arbeiten, die sich mit der schwierigen Aufgabe des Reengineerings von komplexen Altsystemen beschäftigen, die Mehrzahl dieser Arbeiten betrachtet aber ausschließlich sequentielle und nicht zeitkritische Softwaresysteme. Meistens handelt es sich dabei um betriebswirtschaftliche Anwendungen, die in COBOL geschrieben sind und bei denen in der Regel die Neustrukturierung der Datenbasis am Anfang des Reengineering-Prozesses steht. Neuere Arbeiten hingegen befassen sich in erster Linie mit objektorientierten Programmiersprachen wie Java oder C++. Der hierfür repräsentative Begriff des Refactorings ist zu einem weit diskutierten Thema geworden.

Diese Arbeit beschäftigt sich mit dem Reengineering von Telekommunikationssystemen. Das Projekt ECARES (**E**ricsson **C**ommunication **AR**chitecture for **E**mbedded **S**ystems) ist eine Kooperation zwischen dem Lehrstuhl für Informatik 3 (Softwaretechnik), RWTH Aachen, und den EuroLabs der Firma Ericsson in Aachen. Im Rahmen des ECARES-Projekts werden Konzepte, Lösungen und Werkzeuge zum Reengineering des AXE10-Systems untersucht, der Ericsson-Implementation des Mobile-service Switching Centers (MSC) für das GSM-Netzwerk. Solche eingebetteten Systeme sind verteilt und zeitkritisch, was besondere Anforderungen mit sich bringt, beispielsweise im Hinblick auf die Systemperformace. Zur Realisierung des Systems wurde die hausinterne Programmiersprache PLEX verwendet. Diese prozessorientierte Sprache basiert auf dem Signalparadigma, was bedeutet, dass die verteilten Systemprozesse mit Hilfe von Signalen gesteuert werden. Diese Eigenschaften grenzen diese Arbeit von den meisten anderen Reengineering-Projekten ab.

In der ersten Phase des ECARES-Projekts stand das Reverse Engineering von Telekommunikationssystemen im Mittelpunkt [Mar04a]. Der Schwerpunkt lag also auf der Herleitung des aktuellen Zustands der Architektur sowie dem Verständnis des Systems. Ausgehend vom Quelltext wurde zur Systemrepräsentation ein sogenannter Strukturgraph erzeugt, an dem die unterschiedlichen Analysen durchgeführt werden konnten. Neben der statischen Betrachtung des Quelltextes wurden mit den zur Laufzeit erzeugten Trace-Dateien auch dynamische Aspekte berücksichtigt. Da Teile des untersuchten Telekommunikationssystems ursprünglich in Form von endlichen Zustandsautomaten spezifiziert wurden, hat auch eine Analyse der im aktuellen Quelltext implementierten Zustandsautomaten stattgefunden. In einem evolutionären Prozess konnten im Laufe der Zeit Lösungen und Werkzeuge zu unterschiedlichen Problemstellungen beim Reverse Engineering entwickelt werden.

In dieser Arbeit wird dieser graphbasierte Reverse Engineering-Ansatz zu einer vollständigen Reengineering-Lösung erweitert. Es steht also nicht mehr ausschließlich das

Verständnis des vorhandenen Systems im Vordergrund, es sollen vielmehr auch Modifikationen und Verbesserungen des Systems unterstützt werden. Der Reengineering-Ansatz läuft dabei auf zwei Abstraktionsebenen ab. Zum einen werden auf der Quelltextebene die Phasen des Reverse Engineering und des Forward Engineering durchgeführt. Während das Reverse Engineering das Parsen der existierenden Software und die Erzeugung des entsprechenden Strukturgraphen sowie seine Analyse beinhaltet, wird zum Forward Engineering die finale Erzeugung des neuen Quelltextes gezählt. Zum anderen findet zwischen diesen beiden Phasen mit dem Redesign die eigentliche Modifikation der Systemarchitektur statt, die auf der Ebene des Strukturgraphen vollzogen wird. Während auf der ersten Ebene die Verarbeitung des Quelltextes größtenteils von Werkzeugen automatisch durchgeführt wird, soll auf der abstrakteren Ebene des Strukturgraphen das eigentliche Redesign in einem interaktiven Prozess vom Ingenieur gesteuert werden.

Die Arbeit beschreibt zunächst alle nötigen Grundlagen und erläutert die Problemdomäne. Im Hauptteil wird der entwickelte graphbasierte Reengineering-Ansatz in allen Einzelheiten erklärt. Der Schwerpunkt wird dabei auf die Redesign-Phase gelegt, in der es gilt, den Ingenieur bei dem interaktiven Prozess möglichst gut zu unterstützen. Die hierfür entwickelten Algorithmen, die sowohl dynamische Aspekte ansprechen als auch die im System realisierten Zustandsautomaten berücksichtigen, werden ausführlich behandelt. Anschließend werden die im Rahmen des Projektes entwickelten Werkzeuge vorgestellt. Am Ende wird auch die Erweiterbarkeit bzw. die Übertragbarkeit des Ansatzes diskutiert.

Der wissenschaftliche Beitrag der Arbeit ist in drei Aspekten zu finden. *Erstens* unterscheidet sich der entwickelte Reengineering-Ansatz in vielerlei Hinsicht von den bisher verfügbaren Lösungen. Er arbeitet auf einer Graphstruktur, die einen vergleichsweise hohen Abstraktionsgrad bietet, und umfasst dennoch relativ viele spezifische Konzepte, die für ein erfolgreiches Reengineering wichtig sind. Dabei wird ein evolutionäres Vorgehen ermöglichst, bei dem die Systemmodifikationen in mehreren Schritten durchgeführt werden können. Entsprechende Werkzeuge unterstützen den Ingenieur bei der Durchführung seiner Aufgaben. Der *zweite* Beitrag dieser Arbeit richtet sich deshalb auf die prototypische Entwicklung solcher Werkzeuge. Hier steht vor allem die Beschreibung der eingesetzten modellbasierten, auf formalen Spezifikationen beruhenden Entwicklungstechniken und der dabei gewonnenen Erfahrungen im Vordergrund. Die entwickelten Konzepte und Werkzeuge wurden bei Ericsson zur Lösung konkreter Probleme beim Reengineering des AXE10-Systems eingesetzt. In den hierbei gewonnenen Erkenntnissen ist deshalb der *dritte* wichtige Beitrag der Arbeit zu sehen. Viele der beschriebenen Lösungen sind gerade wegen ihres systemspezifischen Charakters für die AXE10-Ingenieure besonders interessant.

# Danksagung

Die vorliegende Arbeit ist mit der Unterstützung einer Vielzahl von Menschen entstanden. An dieser Stelle möchte ich mich bei den wichtigsten von ihnen bedanken.

Mein erster Dank richtet sich an meinen Doktorvater, Prof. Dr.-Ing. Manfred Nagl, der mir die Gelegenheit gegeben hat, diese Arbeit anzufertigen. Er schaffte den Rahmen, der sowohl genügend Freiraum als auch eine umfangreiche Unterstützung bot, um ein Gelingen der Promotion zu ermöglichen.

Weiterhin möchte ich Herrn Prof. Dr. Uwe Aßmann von der TU Dresden danken, der sich zur Erstellung des Zweitgutachtens bereit erklärt hat.

Mein Dank geht auch an Prof. Dr. Otto Spaniol, der als Vorsitzender des Graduiertenkollegs *Software für mobile Kommunikationssysteme* meine Arbeit vor allem organisatorisch bestens unterstützte.

Ferner möchte ich den Mitgliedern des Ericsson-Teams der Aachener EuroLabs danken. Ohne Dietmar Wenninger, der seit fast einem Jahrzehnt die wichtigste Konstante des Projektes darstellt, wäre eine Weiterführung der Kooperation sicher nicht möglich gewesen. Durch sein Gespür für wissenschaftlich relevante Fragestellungen konnten wir stets neue interessante Brücken zu praktischen Problemen schlagen. Für die ausgezeichnete Unterstützung in technischen Fragen danke ich außerdem Helmut Seidler sowie Thomas Mansson, von dessen Begeisterungsfähigkeit meine Diplomanden regelmäßig profitiert haben. Ralf Empen und Max Bataillie danke ich vor allem für die organisatorische Unterstützung.

Auch möchte ich dem ECARES-Team danken. Mein Vorgänger André Marburger unterstützte mich in der Anfangsphase und gab mir auf der Suche nach dem Promotionsthema die entscheidenden Hinweise. Ich danke allen meinen Diplomanden: Linlin Xu, Wei Zhao, Andrey Falaleev, Tobias Zeuch, Yingying Zhang, André Kamp und Tobias Walter. In unzähligen Gesprächen haben wir uns das breite Thema Stück für Stück gemeinsam erarbeitet. Erwähnt werden sollte auch Marcel Pettau, der als Forschungsstudent besonders in den letzten Monaten eine unersetzliche Hilfe war.

Allen meinen Kollegen möchte ich für die unvergesslichen Jahre am Lehrstuhl danken. Meinem alten Studienkollegen und Nachbar(ski) danke ich für das Korrekturlesen dieser Arbeit und für die vielen gemeinsam durchlebten kleinen Abenteuer, die den nötigen Ausgleich zum wissenschaftlichen Ernst lieferten. Auch Ibrahim Armac, das dritte Mitglied der „Ruderetage“, war mit seinen täglich neuen Ideen immer eine Bereicherung des Lehrstuhlalltags. Simon Becker und Theresa Körtgen danke ich für die vielen interessanten Ge-

spräche und die zusammen gelaufenen Kilometer, denen oft ein geselliger Abend folgte. Für so manchen Tipp möchte ich mich außerdem bei den alten Lehrstuhlhasen Marcus Heller, Thomas Haase und Christian Fuss bedanken. Und auch einige Kollegen aus der verbleibenden Generation waren öfters mit ihrer praktischen Hilfe zur Stelle: René Wörzburger, Erhard Weinell, Thomas Heer und Cem Mengi. Ihnen allen wünsche ich weiterhin viel Erfolg.

Besonderen Dank schulde ich meinem ehemaligen Betreuer und späteren Kollegen Ulrich Norbisrath, der maßgeblich für mein Interesse an der Softwaretechnik verantwortlich war. In meinen ersten Monaten als Wissenschaftler stand er mir stets mit wertvollen Tipps zur Seite und war nicht nur auf den gemeinsamen (Konferenz-)Reisen als „wahrer“ Mensch eine freundschaftliche Begleitung.

Meiner Frau Stephanie gilt ebenfalls ein großer Dank. Sie brachte die Geduld und das Verständnis für meine Pläne mit und war gleichzeitig für mein persönliches Glück in den letzten Jahren verantwortlich.

Ein ganz herzlicher Dank geht an meine Eltern, Barbara und Donat Skrzypczyk, die mir immer geholfen haben, im Leben den richtigen Weg zu finden. Ohne ihre Unterstützung wäre diese Arbeit nicht möglich gewesen.

Christof Mosler

# Inhaltsverzeichnis


**1 Einführung** **1**

1.1 Wartung von Softwaresystemen . . . . . 1

1.2 Motivation . . . . . 3

1.3 ECARES-Projekt . . . . . 5

1.4 Zielsetzung . . . . . 10

1.5 Lösungsansatz . . . . . 12

1.6 Gliederung der Arbeit . . . . . 14

**2 Grundlagen** **17**

2.1 Softwarewartung und -Reengineering . . . . . 17

2.2 Mobile Telekommunikationssysteme . . . . . 20

2.2.1 GSM-Netz . . . . . 21

2.2.2 AXE10-System . . . . . 23

2.2.3 Betrachtetes Teilsystem . . . . . 24

2.3 Einführung in PLEX . . . . . 27

2.3.1 Struktur eines PLEX-Programms . . . . . 28

2.3.2 Datenstrukturen . . . . . 29

2.3.3 Kontroll- und Datenfluss . . . . . 34

2.3.4 Communication Buffer . . . . . 40
2.3.5 Ausführungsmodell . . . . . 43
2.4 Bearbeitete Reengineering-Szenarien . . . . . 46
2.4.1 Restrukturierung von Subsystemen . . . . . 46
2.4.2 Restrukturierung von PLEX-Blöcken . . . . . 47
2.4.3 Analyse von Communication Buffern . . . . . 48
2.5 Zusammenfassung . . . . . 50

**3 Methodischer Ansatz 53**

3.1 Reverse Engineering . . . . . 53
3.1.1 Statisches Reverse Engineering . . . . . 54
3.1.2 Dynamisches Reverse Engineering . . . . . 58
3.1.3 Graphschema . . . . . 63
3.2 Redesign . . . . . 70
3.2.1 Redesign-Prozess . . . . . 71
3.2.2 Redesign-Operationen . . . . . 74
3.2.3 Inkosistenzen und korrigierende Transformationen . . . . . 80
3.3 Forward Engineering . . . . . 86
3.3.1 Überblick . . . . . 86
3.3.2 Erzeugung des annotierten Codes . . . . . 92
3.3.3 Quelltexttransformationen . . . . . 95
3.4 Verwandte Arbeiten . . . . . 99
3.5 Zusammenfassung . . . . . 103

**4 Reengineering-Szenario Blockdekomposition 105**

4.1 Dekomposition des Blocks TRACON . . . . . 106

4.2 Terminologie . . . . . 108
4.2.1 Bindung und Kopplung . . . . . 108
4.2.2 Direkte und indirekte Vergleiche . . . . . 109
4.2.3 Clustering . . . . . 110
4.3 Clustering-basierte Ansätze . . . . . 111
4.3.1 Bekannte Heuristiken . . . . . 112
4.3.2 Heuristiken zur Blockdekomposition . . . . . 113
4.3.3 Weitere Varianten des Verfahrens . . . . . 117
4.3.4 Kombination verschiedener Dekompositionstechniken . . . . . 122
4.3.5 Cluster Dependency Graph-Ansatz . . . . . 122
4.4 Musterbasierter Ansatz . . . . . 124
4.4.1 Muster in Blöcken . . . . . 125
4.4.2 Parametrisierung des Pattern Matching . . . . . 127
4.4.3 Einbettung von Pattern Matching in den clusterbasierten Ansatz . 129
4.5 Transformationen bei Blockdekomposition . . . . . 131
4.6 Evaluierung der Ansätze . . . . . 135
4.7 Zusammenfassung . . . . . 140

**5 Realisierung** **143**

5.1 Übersicht . . . . . 143
5.2 Werkzeuge zur Quelltextanalyse . . . . . 145
5.3 Redesign-Werkzeug . . . . . 147
5.3.1 Graphersetzungssysteme . . . . . 148
5.3.2 Modellerzeugung mit Fujaba . . . . . 151
5.3.3 RePLEX-Editor . . . . . 156

5.4 Werkzeuge zur Quelltextgenerierung . . . . . 165
5.4.1 Quellcodetransformation . . . . . 165
5.4.2 Übersicht TXL . . . . . 166
5.4.3 PLEX-Grammatikdefinition mit TXL . . . . . 170
5.4.4 PLEX-Transformationen mit TXL . . . . . 172
5.5 Entwicklungsaufwand . . . . . 176
5.6 Zusammenfassung . . . . . 179

**6 Erweiterungen** **181**

6.1 Zustandsautomaten . . . . . 181
6.1.1 Extraktion . . . . . 183
6.1.2 Überprüfung der Zustandsautomaten . . . . . 187
6.1.3 Analyse der Zustandsautomaten . . . . . 188
6.1.4 Restrukturierung von Zustandsautomaten . . . . . 189
6.1.5 Dekomposition von Zustandsautomaten . . . . . 197
6.1.6 Realisierung . . . . . 202
6.1.7 Evaluierung der Ergebnisse . . . . . 205
6.1.8 Verwandte Ansätze . . . . . 207
6.2 Komponierte Redesign-Regeln . . . . . 210
6.2.1 Kompositionssprache . . . . . 211
6.2.2 Realisierung . . . . . 214
6.2.3 Verwandte Ansätze . . . . . 217
6.2.4 Fazit . . . . . 220
6.3 Weitere Programmiersprachen: Fortran 95 . . . . . 220
6.3.1 Projektbeschreibung . . . . . 221

6.3.2 Ziele der Erweiterung . . . . . 222
6.3.3 RePLEX-Erweiterung . . . . . 222
6.3.4 Fazit . . . . . 224
6.4 Zusammenfassung . . . . . 224

**7 Werkzeugunterstützung 227**
7.1 Werkzeuganwendung aus Benutzersicht . . . . . 227
7.1.1 Reverse Engineering . . . . . 229
7.1.2 Interaktives Redesign . . . . . 231
7.1.3 Forward Engineering . . . . . 236
7.1.4 Extraktion von Zustandautomaten . . . . . 237
7.1.5 Komponierte Redesigns . . . . . 240
7.1.6 Fortran-Editor . . . . . 244
7.2 Zusammenfassung . . . . . 247

**8 Schlussbemerkungen 249**
8.1 Zusammenfassung . . . . . 249
8.2 Ausblick . . . . . 251
8.3 Fazit . . . . . 254

**A Quelltexttransformationen 259**
A.1 Erstellung neuer Programmartefakte . . . . . 259
A.1.1 Erstellung neuer Code-Blöcke . . . . . 259
A.1.2 Erstellung neuer Sende-/Aufruf-/Sprung-/Zugriffs-Anweisungen und der Empfangsanweisungen . . . . . 260
A.1.3 Erstellung neuer Datenstrukturen . . . . . 263
A.2 Änderung eines Attributwertes . . . . . 264

A.2.1 Umbenennungen . . . . . 264
A.2.2 Neubelegung eines Aufzählungstyps . . . . . 264
A.3 Umwandlungen von Anweisungen . . . . . 265
A.4 Löschen von Zeilen . . . . . 268
A.5 Manuelle Transformationen . . . . . 268