Zur Homepage der Dissertation

Inhaltsverzeichnis

I Grundlagen **1**

1 Einleitung **3**

1.1 Motivation 6

1.2 Zielsetzung der Arbeit 9

1.3 Aufbau der Arbeit 9

2 Data Warehouse-Systeme **11**

2.1 Datenhaltungskomponenten 12

2.2 Werkzeuge der Integrationsschicht 14

2.3 Werkzeuge der Auswertungsschicht 21

2.4 Schichtenübergreifende DWS-Komponenten 21

2.5 Zusammenfassung und Bewertung des Status Quo 23

3 Datenqualität **25**

3.1 Analogie zur industriellen Fertigung 25

3.2 Merkmale von Datenprodukten 27

3.3 Datenqualitätsmängel 32

3.4 Zusammenfassung 36

4 Datenqualitätsmanagement **37**

4.1 Geschichtliche Entwicklung des Qualitätsmanagements 37

4.2 Exkurs: Normenfamilie ISO 9000:2000 ff. 38

4.3 Adaption von QM-Konzepten an die Datenverarbeitung 40

4.4 Datenqualitätsplanung 40

4.5 Datenqualitätslenkung und -verbesserung 44

4.6 Datenqualitätssicherung 50

4.7 Zusammenfassung 50

5 Verwandte Ansätze **51**

5.1 AJAX: Deklaratives Data Cleaning (INRIA) 51

5.2 Data Quality Mining (DaimlerChrysler AG) 55

5.3 Ascential Quality Manager 57

5.4 Weitere Ansätze 62

5.5 Zusammenfassung und Bewertung 64

II Datenqualitätsmanagement in Data Warehouse-Systemen **67**

6 Datenqualitätsmetriken, Mess- und Verbesserungsverfahren **69**

6.1 Metriken und Messverfahren nach Datenqualitätsmerkmalen 69

6.2 Verbesserungsmaßnahmen nach Datenqualitätsmerkmalen 86

6.3 Domänenspezifische Datenqualitätsindikatoren 89

6.4 Zusammenfassung 90

7 Das CLIQ-Vorgehensmodell zur inkrementellen Datenintegration in DWS **91**

7.1 Überblick 91

7.2 Grundlegende Definitionen 93

7.3 Phase 1: Prüfung von Schlüssel- und referentieller Integrität 94

7.4 Phase 2: Vereinheitlichung der Repräsentation 96

7.5 Phase 3: Statistische Prozesskontrolle 99

7.6 Phase 4: Domänenspezifische Konsistenzprüfung/Nachbearbeitung 102

7.7 Phase 5: Record Linkage 105

7.8 Phase 6: Merging 109

7.9 Phase 7: Qualitätsmessung und -analyse 111

7.10 Phase 8: Lenkung von Datenprodukten/Qualitätsverbesserung 115

7.11 Phase 9: Freigabe von Datenprodukten 118

7.12 Phase 10: Auswertung von Kundenfeedback/Datenrücknahme 119

7.13 Sonderfall: Behandlung von quellenseitig gelöschten Tupeln 120

7.14 Parallelisierung der Phasenausführung 122

7.15 Komplexitätsabschätzungen 122

7.16 DQM in OLTP-Systemen 126

7.17 Zusammenfassung 127

8 Anforderungen an die CLIQWorkbench 129

8.1 Überblick über die Anforderungen . . . 129

8.2 Objektorientierte Analyse des Problembereichs . . . 131

8.3 Zusammenfassung . . . 136

9 Softwaretechnischer Entwurf der CLIQWorkbench 137

9.1 Dem Entwurf zu Grunde liegende Annahmen . . . 137

9.2 Objektorientierter Entwurf . . . 138

9.3 DQM Controller . . . 139

9.4 Data Migrator . . . 154

9.5 Measurement Engines . . . 160

9.6 Data Linker . . . 165

9.7 Data Auditor (MEDAS) . . . 168

9.8 Release/Retraction Handler . . . 170

9.9 CLIQWorkbench Console . . . 171

9.10 Database/Repository Interface . . . 171

9.11 Metarelationen zur Unterstützung inkrementeller Updates . . . 172

9.12 Zusammenfassung . . . 176

10 Implementierung der CLIQWorkbench 177

10.1 Überblick . . . 177

10.2 Fremdkomponenten der CLIQWorkbench . . . 178

10.3 Die CLIQWorkbench aus Sicht der komponentenbasierten Softwareentwicklung . . . 185

10.4 Umfang der Implementierung . . . 187

10.5 Zusammenfassung . . . 187

III Evaluierung 189

11 CLIQ@EKN: Eine Fallstudie 191

11.1 Das Epidemiologische Krebsregister Niedersachsen . . . 191

11.2 Das Anwendungsszenario der Fallstudie . . . 196

11.3 Datenintegration im EKN mit Hilfe der CLIQWorkbench . . . 199

11.4 Zusammenfassung und Bewertung der Ergebnisse der Fallstudie . . . 214

12 Weitere Evaluierungsmaßnahmen **217**

12.1 Abgleich der CLIQWorkbench mit den Anforderungen 217

12.2 Einbindung der CLIQWorkbench in das SAP BW 219

12.3 Evaluierung des CLIQ-Vorgehensmodells durch DaimlerChrysler 223

12.4 Zusammenfassung 224

IV Zusammenfassung und Ausblick **225**

13 Zusammenfassung und Ausblick **227**

13.1 Zusammenfassung 227

13.2 Ausblick 229

Anhänge und Verzeichnisse **235**

A Marktübersicht zu Softwarewerkzeugen für das DQM **235**

A.1 Tabellarische Charakterisierung der Werkzeuge 236

A.2 Abschließender Überblick 245

Glossar **247**

Abkürzungsverzeichnis **257**

Abbildungsverzeichnis **259**

Tabellenverzeichnis **261**

Literaturverzeichnis **263**

Index **279**

Zur Homepage der Dissertation

Danksagung

Das Erstellen einer Dissertationsschrift ist naturgemäß mit erheblichem Aufwand verbunden. Das war bei der vorliegenden Arbeit, die im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am OFFIS (Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme) entstanden ist, nicht anders. Dennoch werde ich das Forschen und Schreiben an dieser Arbeit in sehr positiver Erinnerung behalten. Dies liegt zum einen sicherlich an dem aus meiner Sicht spannenden Forschungsthema, zum anderen aber auch wesentlich an dem Umfeld, in dem die Arbeit geschrieben wurde. Daher möchte ich mich an dieser Stelle ganz herzlich bei denjenigen Personen bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Zuallererst möchte ich Herrn Prof. Appelrath nennen, der mich als Doktorvater von Anfang an umsichtig betreut und gefördert hat. Herrn Prof. Hasselbring danke ich für die Übernahme der Zweitbegutachtung.

Bei meinem Bereichsleiter Wilfried Thoben bedanke ich mich für die Freiräume zum wissenschaftlichen Arbeiten, bei den OFFIS-Kollegen der Data Warehousing-Arbeitsgruppe und des CARLOS-Teams für den konstruktiven Ideenaustausch und die nette Arbeitsatmosphäre. Ganz besonderer Dank geht an Olaf Herden für alles, was wir in den vergangenen Jahren zusammen unternommen und erreicht haben. Fine work, go on!

Weiterhin möchte ich mich ausdrücklich bei Thomas Aden für sein großes Engagement bedanken, das er sowohl während seiner Tätigkeit als wissenschaftliche Hilfskraft als auch im Rahmen seiner Diplomarbeit sowie anschließend als wissenschaftlicher Mitarbeiter im Umfeld des CLIQ-Projekts gezeigt hat.

Für ihre Mitarbeit an Softwareentwurf und Implementierung geht mein Dank an Ina Wentzlaff, Jürgen Meister und Thomas Wilkens. Sehr gefreut habe ich mich über die Unterstützung von Udo Grimmer, Rainer Höltke und dem EKN-Team bei der Durchführung der Evaluierung. Auch die tapferen Korrekturleser Heiko Tapken und Ludger Zachewitz sind hier zu nennen. Jürgen Schlegelmilch danke ich für die hilfreichen LaTeX-Tipps.

Schließlich hat auch mein privates Umfeld erheblich zum Fortschritt dieser Arbeit beigetragen. Insbesondere möchte ich mich hier bei Kirsten Panienski und meinen Eltern für ihr Verständnis und ihre Unterstützung bedanken.

Holger Hinrichs

Oldenburg, im April 2002

Zur Homepage der Dissertation

Zusammenfassung

Taktische und strategische Entscheidungen in Unternehmen und anderen Organisationsformen werden immer häufiger auf der Basis von Datenanalysen durchgeführt. Die zunehmende Verbreitung von Data Warehouse-Systemen und darauf aufsetzenden Anwendungen wie Customer Relationship Management und Supply Chain Management bestätigt diese Entwicklung. Ein Data Warehouse-System erlaubt die flexible und effiziente Auswertung von aus heterogenen Quellen integrierten, möglicherweise historisierten Datenbeständen. Während sowohl in der Forschung als auch in der Praxis intensiv an Techniken zur Datenauswertung gearbeitet wurde (multidimensionale Datenmodellierung, spezielle Indizierungsverfahren und Analysemethoden), blieben Fragen der Datenintegration und Datenqualitätssicherung lange unberücksichtigt. Dabei ist leicht einzusehen, dass die Auswertung von qualitativ mangelhaften (d. h. inkonsistenten, unvollständigen, obsoleten etc.) Daten zwangsläufig zu qualitativ mangelhaften Ergebnissen führt, die möglichst nicht zur Entscheidungsunterstützung herangezogen werden sollten („Garbage in, garbage out“).

Die vorliegende Arbeit hat sich daher zum Ziel gesetzt, grundlegende Konzepte eines Datenqualitätsmanagements herauszuarbeiten und diese im Kontext des Data Warehousing zu konkretisieren. Den Kern der Arbeit bildet ein 10-phasiges Vorgehensmodell zur qualitätsgelenkten Datenintegration, welches auf den Anforderungen der Norm ISO 9001:2000 basiert und die methodische Grundlage für hochgradig automatisierte, reproduzierbare und wartbare Integrationsprozesse mit qualitätsgesicherten Ergebnissen schafft. Eine notwendige Voraussetzung für ein effektives Datenqualitätsmanagement ist die Messbarkeit von Datenqualität. Vor diesem Hintergrund werden Metriken für verschiedene Datenqualitätsmerkmale definiert und durch Messverfahren operationalisiert. Zur technischen Umsetzung des Vorgehensmodells wird ein als CLIQWorkbench bezeichnetes Softwaresystem entworfen und prototypisch implementiert, welches die Planung, Durchführung und Analyse von Datenintegrationsprozessen unterstützt. Schließlich werden die im Rahmen der Arbeit entwickelten Konzepte und deren Implementierung an Hand von mehreren Anwendungsszenarien evaluiert.

Zur Homepage der Dissertation

Abstract

Tactical and strategic decision making within enterprises and other organisations is often based on data analyses. The increasing use of data warehouse systems and attached applications like customer relationship management and supply chain management supports this assumption. A data warehouse system provides means to integrate data from heterogeneous sources and analyse them in a flexible and efficient way. Both research and practice have intensively worked on techniques for data analysis (multidimensional modelling, special indexing etc.). At the same time, questions of data integration and data quality assurance have been neglected. However, it is quite obvious that analysing defective (i. e. inconsistent, incomplete, obsolete etc.) data inevitably leads to questionable results which should not be used for decision making („garbage in, garbage out“).

This PhD thesis aims at developing fundamental concepts of data quality management and applying them to the data warehousing context. A ten phase process model for quality controlled data integration is introduced, based on the requirements specified in the ISO 9001:2000 standard. This process model makes it possible to establish highly automated, reproducible and easily maintainable data integration processes delivering quality assured results. Being able to measure data quality presents a necessary precondition to an effective data quality management. Against this background, key figures for various data quality characteristics are defined and operationalised by measuring methods. To implement the process model, a software system called CLIQWorkbench is being designed and implemented prototypically, supporting the planning, execution, and analysis of data integration processes. Finally, the approach and its implementation are being evaluated by means of several application scenarios.

Zur Homepage der Dissertation

Teil I

Grundlagen

Kapitel 1

Einleitung

Die Auswertung elektronisch verfügbarer Informationen spielt heute sowohl in betriebswirtschaftlichen als auch in wissenschaftlichen Anwendungsfeldern eine überaus große Rolle. So müssen beispielsweise Unternehmen Trends und Zusammenhänge sehr zeitnah erkennen, analysieren sowie in taktische und strategische Entscheidungen umsetzen, um wettbewerbsfähig zu bleiben. Vor diesem Hintergrund ist der Bedarf an effizienten und flexiblen Methoden zur Datenbereitstellung und -analyse immens angewachsen. Begünstigt durch die zunehmende Verfügbarkeit leistungsfähiger Rechnersysteme hoher Speicherkapazität werden in Unternehmen und anderen Organisationen seit einigen Jahren Datensammlungen aufgebaut, deren Volumina oft im Gigabyte- oder Terabyte-Bereich liegen. Zusätzlich zur reinen Unterstützung der taktischen und strategischen Planung werden diese Datensammlungen auch als Basis für neuartige Anwendungsfelder wie *Customer Relationship Management (CRM)* und *Supply Chain Management (SCM)* genutzt.

Die analyseorientierte Verarbeitung sehr großer Datenvolumina stellt erhebliche Herausforderungen an die zu Grunde liegende Informationstechnologie. Mit herkömmlichen *OLTP-Systemen* (OLTP = Online Transaction Processing), die in Organisationen zur Abwicklung der täglichen Geschäftsvorgänge eingesetzt werden (z. B. Buchungssysteme), lassen sich die neuen Aufgaben nicht bewältigen. OLTP-Systeme sind darauf ausgelegt, gut strukturierte Aufgaben, die als kurze, atomare und isolierte Transaktionen auf detaillierten, aktuellen Daten formulierbar sind, in hoher Anzahl effizient zu bearbeiten. Zur Unterstützung von taktischen und strategischen Entscheidungsfindungsprozessen sind jedoch üblicherweise historisierte, zusammengefasste und konsolidierte Daten erforderlich, auf denen komplexe Anfragen selbst bei sehr großen Datenvolumina hinreichend effizient ausgeführt werden müssen [CD97]. In der Praxis sind die dabei zu untersuchenden Daten oft auf diverse, typischerweise heterogene Quellsysteme verteilt, z. B. auf OLTP-Systeme einzelner Betriebe oder Abteilungen einer Organisation.

Data Warehouse-Systeme

Vor dem Hintergrund der genannten Anforderungen ist mit *Data Warehouse-Systemen (DWS)* [Inm96] eine neue Klasse von Informationssystemen entstanden. DWS bündeln eine Reihe von Technologien, die in (Datenbank-)Forschung und Industrie in den letzten 20 Jahren entwickelt wurden [CBS98, HS00]. Zentrale Komponente eines DWS ist eine physische Datenbank, die das eigentliche Data Warehouse darstellt. Diese Datenbank bietet eine einheitliche Sicht auf Daten, die ursprünglich aus heterogenen Quellen stammen und im Rahmen eines Integrationsprozesses konsolidiert wurden (z. B. Produkt-, Marketing- und Controlling-Daten). Als Datenquellen eines DWS kommen OLTP-Systeme, aber auch externe Datenlieferanten wie Meinungsforschungsinstitute oder statistische Ämter in Frage. Häufig, aber nicht notwendigerweise findet in einem DWS eine Historisierung von Daten statt, um eine Analyse von zeitlichen Trends zu ermöglichen. In OLTP-Systemen ist dies i. Allg. nicht

möglich, da Daten hier oft nur für einen kurzen Zeitraum (oft < 3 Monate) vorgehalten werden, bevor sie gelöscht bzw. archiviert oder überschrieben werden. Grundsätzlich steht in einem DWS – im Gegensatz zu einem OLTP-System – der Auswertungsaspekt im Mittelpunkt.

Das Charakteristikum der physischen (materialisierten) Datenintegration grenzt ein DWS von einem *föderierten Datenbanksystem (FDBS)* [Con97] ab, in dem zur Laufzeit integrierte Sichten auf heterogene Datenbestände verwaltet werden („virtuelle“ Datenintegration, siehe auch [LRO96, HZ96, PAG96, GS98, GBMS99]). Eine Gegenüberstellung beider Ansätze wird z. B. in [Hul97] vorgenommen. Ein FDBS ist für die Entscheidungsunterstützung i. Allg. nicht geeignet, da die erforderliche Effizienz von Analyseprozessen eine Integration zur Laufzeit nicht zulässt und außerdem eine Historisierung von Daten nicht möglich ist.

Klassische Architekturen von Systemen zur Entscheidungsunterstützung – oft als *Decision Support-Systeme (DSS)* oder *Managementinformationssysteme* bezeichnet [GGC97] – setzen typischerweise direkt auf den einzelnen OLTP-Systemen auf und stellen zusätzliche Funktionalität zum Berichtswesen bereit, entweder rein proprietär oder unter Verwendung von herkömmlichen Tabellenkalkulationswerkzeugen. Im Vergleich zu DWS sind derartige Systeme weniger flexibel (keine Datenintegration, keine Historisierung) und aufgrund der „Konkurrenz“ durch die auf demselben Datenbestand operierenden OLTP-Anwendungen weniger performant. Abbildung 1.1 stellt die Architekturen (in grober Form) einander gegenüber. Einen historischen Überblick über die Entwicklung von Systemen zur Entscheidungsunterstützung geben [GGC97] und [Kem99].

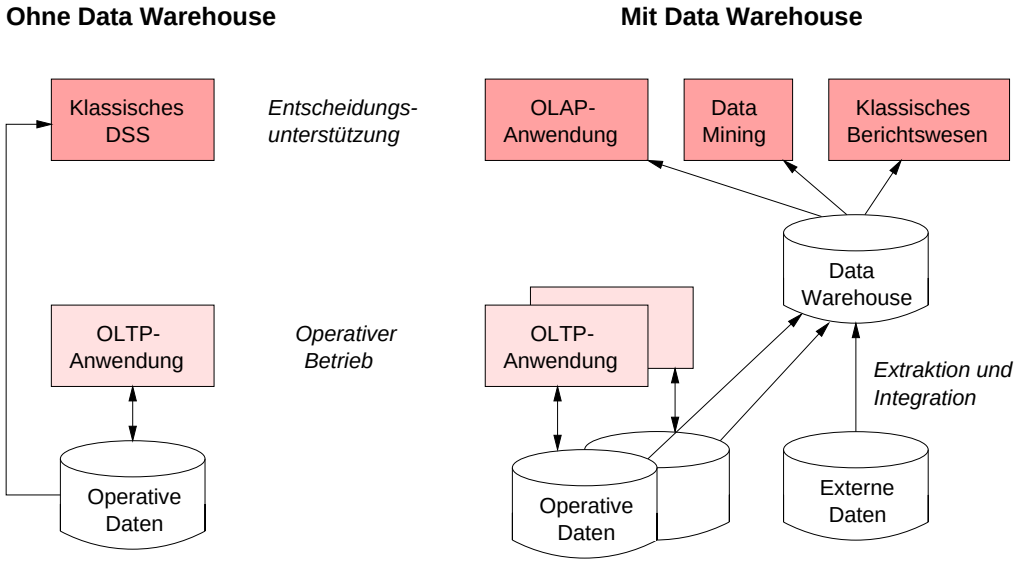


Abbildung 1.1: Entscheidungsunterstützung ohne und mit Data Warehouse

Die Anwendungsfelder von DWS sind nicht auf die Betriebswirtschaft (Produkt-, Kunden-, Segmentanalysen etc., siehe z. B. [Wes00]) beschränkt, sondern erstrecken sich u. A. auch auf den wissenschaftlichen Bereich (z. B. Klimaforschung [Mic91] und epidemiologische Krankheitsregister [AFH+97]) sowie öffentliche Einrichtungen (z. B. Einwohnermeldeämter).

Eine Auswertungsmethodik, die in vielen DWS zusätzlich zum klassischen Berichtswesen zum Einsatz kommt, ist das sog. *Online Analytical Processing* (OLAP). OLAP bietet die Möglichkeit, interaktiv, flexibel und effizient verschiedene multidimensionale Sichten auf einen Datenbestand zu erzeugen und zu analysieren. Dabei muss die zu Grunde liegende Datenbank nicht zwangsläufig physisch multidimensional modelliert sein. Verbreitet sind Ansätze wie *ROLAP* (Relational OLAP), bei dem multidimensionale Zusammenhänge auf relationale Schemata abgebildet werden. Bekannte Schematypen dieser Art sind die sog. *Sternschemata* (engl.: Star Schemas) und *Schneeflockenschemata* (engl.: Snowflake Schemas) [Pok99].

Einen weiteren Analyseansatz in DWS stellt das sog. *Data Mining* dar. Dabei handelt es sich um einen automatisierten Prozess, der große Datenbestände an Hand von statistischen Verfahren und Techniken des maschinellen Lernens nach signifikanten Mustern und Trends durchsucht, die sonst unerkannt geblieben wären. Data Mining kann als zentrale Phase eines als *Knowledge Discovery in Databases (KDD)* bezeichneten Prozesses aufgefasst werden. Dieser „nicht-triviale Prozess der Identifizierung gültiger, neuartiger, möglicherweise nützlicher und letztlich verständlicher Muster in Daten“ [FPS96] enthält weiterhin Phasen zur Datenvorverarbeitung sowie zur Interpretation und Konsolidierung von Data Mining-Ergebnissen [BA96].

Erfolgsbarrieren in DWS-Projekten

Wie die genannten Beispiele zeigen, sind die technischen Voraussetzungen zur Unterstützung des Data Warehousing – sowohl die Hardware als auch die Auswertungssoftware betreffend – durchaus gegeben. Laut einer Studie der Meta Group [Met99b] werden dennoch nur 23% aller DWS-Einführungen in Organisationen als erfolgreich eingestuft (siehe Abb. 1.2 (a)), eine vergleichbare Studie der Standish Group spricht gar von nur 17% [Sta99]. In nur 6% aller Fälle wurden laut [Met99b] die Erwartungen an den „Return on Investment“ (ROI) der DWS-Einführung erfüllt oder übertroffen (siehe Abb. 1.2 (b)). Eine weitere Studie, die von der Firma Data Mart Consulting in Zusammenarbeit mit der TU Darmstadt durchgeführt wurde [Dat99], kommt zu einem ähnlichen Ergebnis.

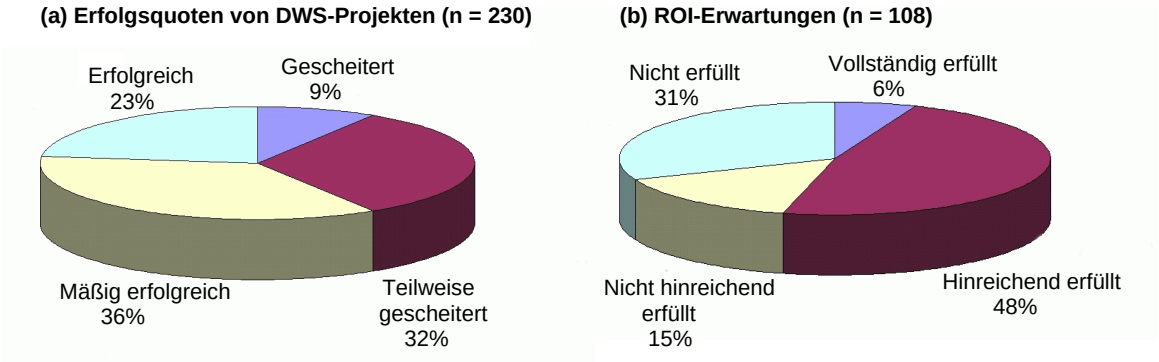


Abbildung 1.2: (a) Erfolgsbewertung und (b) ROI-Bewertung von DWS-Projekten [Met99b]

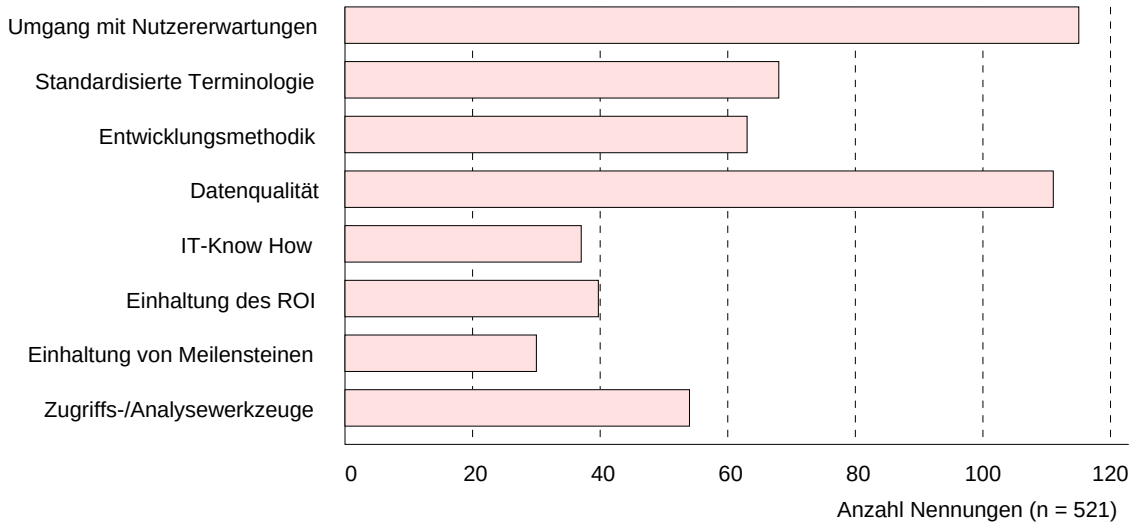


Abbildung 1.3: Erfolgsbarrieren in DWS-Projekten [Met99b]

Um den Ursachen dieser geringen Erfolgsquote nachzugehen, wurden die betroffenen Organisationen im Rahmen der o. a. Meta Group-Studie nach den drei ihrer Meinung nach signifikantesten Erfolgsbarrieren in ihren DWS-Projekten befragt. Abbildung 1.3 zeigt das Ergebnis: Die mit Abstand meistgenannten Antworten betreffen den Umgang mit den Erwartungen von Nutzern an das DWS sowie die Qualität der im DWS verwalteten Daten. Das erstgenannte Problem liegt vor allem darin begründet, dass sich die von OLTP-Systemen bekannten Entwurfsmethodiken für Data Warehouses nur unzureichend eignen und spezielle, auf die besonderen Bedürfnisse des Data Warehouse-Entwurfs zugeschnittene Methodiken (siehe z. B. [Har99, HH99, Her01a, Her01b]) noch nicht in der Praxis etabliert sind. Dadurch bleiben Anforderungen und Erwartungen von DWS-Nutzern oft unberücksichtigt.

Im Mittelpunkt der vorliegenden Arbeit steht die zweitgenannte Problematik, also die oft mangelhafte Datenqualität in DWS. Diese wird von der Gartner Group als „one of the greatest risks and causes of failure in data movement efforts“ [HL98] bezeichnet.

1.1 Motivation

Der Begriff „Qualität“ ist nach [Deu00a] definiert als „Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt“. Anforderungen können dabei von unterschiedlichen Beteiligten aufgestellt werden [MG94], im DWS-Kontext z. B. insbesondere von Nutzern eines DWS, aber auch vom DWS-Administrator oder von der Organisationsleitung. Im Folgenden werden einige Beispiele für typische Qualitätsmängel von Daten im Kontext eines DWS aufgeführt (siehe auch Abb. 1.4):

- Inkorrekte Angaben, verursacht durch Eingabe-, Mess- oder Verarbeitungsfehler (Merkmal *Korrektheit*)
- Logisch widersprüchliche Angaben (Merkmal *Konsistenz*)
- Fehlende Angaben (Merkmal *Vollständigkeit*)
- Duplikate im Datenbestand (Merkmal *Redundanzfreiheit*)
- Uneinheitlich repräsentierte Daten (Merkmal *Einheitlichkeit*).

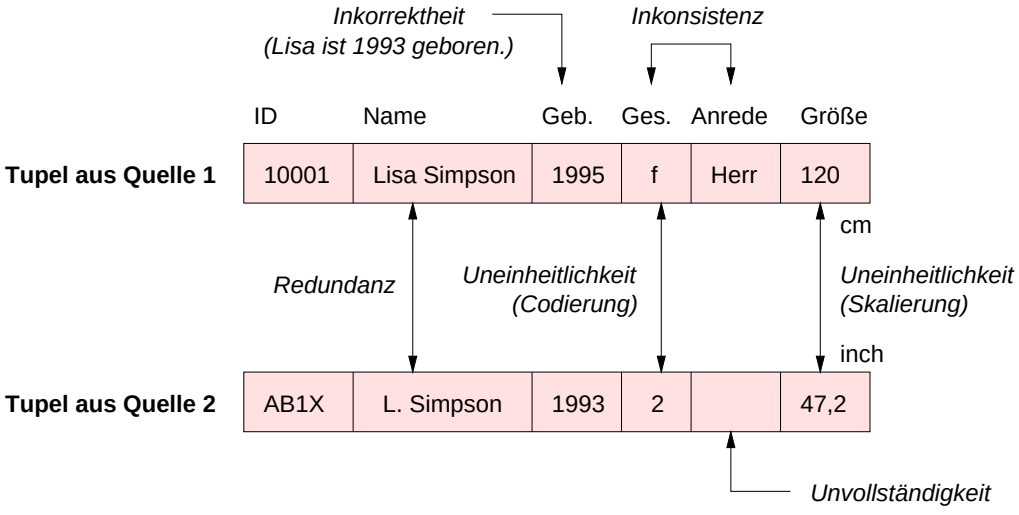


Abbildung 1.4: Beispiele für Datenqualitätsmängel

1.1.1 Folgen mangelhafter Datenqualität

In [DM92] wurde die Qualität von Daten bereits 1992 als entscheidender Faktor für den Erfolg eines Informationssystems identifiziert (siehe Abb. 1.5). Zusammen mit dem Faktor der Systemqualität habe die Datenqualität erheblichen Einfluss auf die Zufriedenheit der Nutzer mit dem Informationssystem und damit mittelfristig auf den Nutzungsgrad des Systems. Der Einfluss von IT-Systemen erstreckt sich allerdings nicht nur auf die Interna einer Organisation, sondern ist zunehmend auch nach außen, d. h. gegenüber Kunden und Partnern, sichtbar (z. B. Präsenz im World Wide Web). Damit wird Datenqualität zu einem signifikanten Faktor in Bezug auf die Reputation von Organisationen.

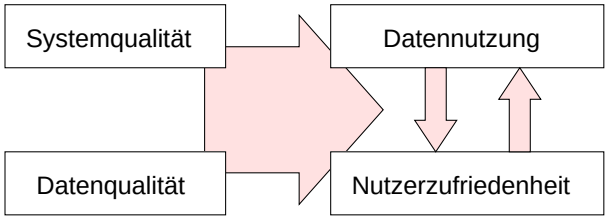


Abbildung 1.5: Ausschnitt aus dem Information System Success Model [DM92]

Die Qualität von Daten betrifft sowohl die strategische und taktische Entscheidungsunterstützung als auch den operativen Betrieb in Organisationen [Red98]. Strategische Entscheidungen, die auf der Basis qualitativ mangelhafter Daten gefällt werden, können für die Organisation schwerwiegende bis existenzbedrohende Folgen haben, z. B. den Verlust von Marktanteilen oder die Rücknahme einer kostenintensiven Marketing-Strategie. Auf taktischer Ebene können mangelhafte Daten der Erarbeitung von Ansätzen zur Geschäftsprozessoptimierung im Wege stehen. Schlecht aufeinander abgestimmte Geschäftsprozesse wiederum verringern die Produktivität und können negativen Einfluss auf die Motivation der betroffenen Mitarbeiter ausüben. Den unmittelbarsten Einfluss auf den Erfolg einer Organisation haben mangelhafte Daten, die den operativen Betrieb betreffen. So können z. B. veraltete oder unvollständige Artikelangaben zu einer Verprellung von Kunden führen oder redundante Kundenadressen die Wirksamkeit einer Marketing-Aktion schmälern. Schließlich darf nicht übersehen werden, dass die Qualität von Daten, die im Rahmen der Produktion (z. B. als Eingabe von Maschinen, Messgeräten etc.) bzw. Dienstleistungserbringung Verwendung finden, unmittelbar die Qualität der resultierenden Produkte und Dienstleistungen beeinflusst. Die genannten Beispiele lassen sich durch den Leitsatz „Garbage in, garbage out“ charakterisieren.

[Eng99] nennt eine Reihe von Praxisbeispielen für folgenschwere Datenqualitätsmängel:

- Falsche Preisangaben in Artikeldatenbanken des Einzelhandels kosten US-Konsumenten ca. 2,5 Mrd. Dollar jährlich. Datenaudits (objektive Analysen von Datenbankinhalten) zu Folge übersteigen vier von fünf über Barcode-Scanner ausgelesene fehlerhafte Preisinformationen den veröffentlichten Artikelpreis, d. h. in diesem Fall profitiert der Einzelhandel von den Datenqualitätsmängeln – auf Kosten der Konsumenten.

- In 1992 konnten 96.000 Steuererstattungsbescheide der US-Finanzbehörden aufgrund von fehlerhaften Adressangaben nicht zugestellt werden.

- Ein europäisches Unternehmen fand durch ein Datenaudit heraus, dass es vier Prozent aller Bestellungen nicht in Rechnung stellte, was einem Verlust von 80 Mio. Dollar entsprach.

- Im März 1997 deckte eine britische Bank den Verlust von 90 Mio. Pfund auf, der durch Datenfehler in einem Berechnungsmodell für Wertpapieranlagen verursacht worden war.

Wie diese Beispiele zeigen, können Datenqualitätsmängel erhebliche Kosten verursachen. [Hel00] identifiziert zusammenfassend folgende Kosten verursachende Faktoren:

- Zusätzlicher Aufwand zur nachträglichen Beseitigung von Qualitätsmängeln
- Unzufriedenheit von Kunden (im DWS-Kontext: Anwender des DWS) mit dem gelieferten Produkt (im DWS-Kontext: Datenbestand des DWS), in Folge dessen ggf. Kundenabwanderung
- Ineffiziente Unterstützung von operativen Geschäftsprozessen
- Fehleinschätzung von Kundensegmenten und Verkaufspotentialen und daraus folgende Fehlentscheidungen
- Kosten- und Rentabilitätsberechnungen auf falscher Zahlengrundlage.

1.1.2 Forschungsbedarf im Bereich des DQM

Es stellt sich nun die Frage, wie mit Datenqualitätsmängeln umzugehen ist. Im Einzelnen sind folgende Unteraspekte zu betrachten:

- Wie können Datenqualitätsmängel aufgedeckt werden?
- Wie können Datenqualitätsmängel beseitigt werden?
- Wie können Datenqualitätsmängel (a priori) vermieden werden?

Es liegt nahe, zur Beantwortung dieser Fragestellungen auf bereits etablierte Konzepte des Qualitätsmanagements (QM) zurückzugreifen. Diese haben, angefangen bei den ersten Ansätzen zur Qualitätskontrolle zu Beginn des 20. Jahrhunderts (Taylor, Ford) bis hin zu den modernen, auf eine kontinuierliche Qualitätsverbesserung ausgerichteten Total Quality Management-Konzepten ([Dem00], [Fei91] et al.), eine mehr als 100-jährige Entwicklung aufzuweisen. Dabei stand durchgängig die Qualität von Produkten und Dienstleistungen im Fokus des Interesses. Der Qualität von Daten wurde hingegen lange keine Beachtung geschenkt. Erst Wang et al. unternahmen 1992 mit der Einführung des Total Data Quality Management-Programms (TDQM) [WRK92] den Versuch, QM-Konzepte auf den Kontext der Datenqualität zu übertragen.

Bis heute jedoch ist Datenqualitätsmanagement (DQM) nicht über einen Forschungsstatus hinausgekommen. In der Literatur finden sich kaum Beispiele für DWS, in denen das Thema DQM angemessene Berücksichtigung findet. Oft sind sich die Nutzer eines DWS nicht einmal darüber bewusst, welche Qualität die von ihnen ausgewerteten Daten aufweisen. Dieses Problem wird dadurch verstärkt, dass – bedingt durch die Architektur von DWS – Erfasser und Auswerter von Daten üblicherweise nicht identisch sind, d. h. während bei der isolierten Auswertung homogener Daten in einem klassischen OLTP-System vorliegende Datenqualitätsmängel oft bekannt sind und entsprechend berücksichtigt werden können, ist dies bei der von den OLTP-Systemen entkoppelten Auswertung integrierter, ursprünglich heterogener Daten in einem DWS nicht mehr der Fall.

In der DQM-Forschung mangelt es insbesondere an Ansätzen, die die relativ allgemeinen QM-Ideen auf konkrete IT-Umgebungen anwenden und auf diese Weise die Voraussetzung dafür schaffen, ein softwaretechnisch gestütztes DQM in die Praxis hineinzutragen. Die wesentlichen Defizite lassen sich wie folgt charakterisieren:

- Es existieren praktisch keine domänenunabhängigen Metriken und Messverfahren, die eine Abschätzung der Qualität gegebener Daten im Hinblick auf Kriterien wie Konsistenz, Vollständigkeit, Redundanzfreiheit etc. erlauben. Die Messbarkeit von Datenqualität ist jedoch eine unabdingbare Prämisse für ein effektives DQM („You cannot control what you cannot measure“ [DeM82]).

- Die Integration von Daten in ein DWS erfolgt in der Praxis i. Allg. unsystematisch, was die Wartbarkeit und Reproduzierbarkeit von Integrationsprozessen negativ beeinflusst und außerdem keine Aussage über die Qualität der integrierten Daten zulässt. Es existiert bisher keine Methodik, die vorschreibt, welche Integrationsschritte in welcher Reihenfolge auszuführen sind, um den Grundsätzen des DQM zu genügen.
- Wie die Produktübersicht in Anhang A zeigt, sind bisher – mit wenigen Ausnahmen (z. B. Ascential Quality Manager [Asc01a], siehe dazu auch Abschnitt 5.3) – keine Softwarewerkzeuge verfügbar, die eine durchgängige, (teil-)automatisierte Durchführung von DQM-Maßnahmen bieten.

Diese Überlegungen verdeutlichen den dringenden Bedarf an Konzepten und Methoden für ein softwaretechnisch unterstütztes DQM. Die Ausführungen in [WB97, Bra96, Wid95, Mad93, Pat93] stützen diese These.

1.2 Zielsetzung der Arbeit

Die vorliegende Arbeit soll Ansätze aufzeigen, mit denen die Qualität von Daten IT-gestützt abgeschätzt werden kann und die es ermöglichen, Daten aus heterogenen Quellen „qualitätsgelenkt“ in ein Data Warehouse zu integrieren, d. h. Qualitätsmängel in den Quelldaten aufzuspüren und zu eliminieren sowie sicherzustellen, dass der resultierende integrierte Datenbestand nachweislich die Qualitätsanforderungen von DWS-Nutzern und anderen interessierten Parteien erfüllt.

Im Einzelnen sollen mit dieser Arbeit die folgenden Ziele erreicht werden:

- Untersuchung des State of the Art in den Bereichen Datenqualität und Datenqualitätsmanagement sowie der Anwendbarkeit der untersuchten Ansätze auf den DWS-Kontext
- Entwicklung von zumindest exemplarischen Metriken und Messverfahren zur Abschätzung der Qualität von Daten
- Entwicklung eines Vorgehensmodells zur qualitätsgelenkten Datenintegration in DWS
- Konzipierung und prototypische Implementierung eines Softwaresystems zum DQM in DWS unter Berücksichtigung der zuvor entwickelten Ansätze
- Evaluierung des implementierten Softwaresystems an Hand von Anwendungen aus der Praxis.

Die zu entwickelnden Lösungsansätze sollen dabei auf bereits etablierten QM-Konzepten sowie auf existierenden Standards wie der Normenfamilie ISO 9000:2000 ff. aufsetzen.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in vier Teile (plus Anhang): Teil I enthält neben dieser Einleitung die wesentlichen Grundlagen der für die Arbeit relevanten Themenbereiche DWS, Datenqualität und DQM. Des weiteren werden verwandte Ansätze vorgestellt und bewertet. Teil II widmet sich den im Rahmen dieser Arbeit entwickelten DQM-Konzepten (Metriken und Vorgehensmodell) und deren softwaretechnischer Umsetzung. Teil III dokumentiert die Evaluierung des gewählten Ansatzes an Hand unterschiedlicher Anwendungsszenarien. Zusammenfassung und Ausblick in Teil IV schließen die Arbeit ab.

Kapitel 2

Data Warehouse-Systeme

Ein DWS enthält neben der als Data Warehouse bezeichneten Datenbank „die Menge aller Komponenten (und deren Beziehungen untereinander), die für das Beschaffen, Aufbereiten, Speichern und Analysieren von Daten notwendig sind“ [BG01]. Obligatorisch sind dabei die in Abb. 2.1 dunkel eingefärbten Komponenten zur Datenintegration, Datenbereitstellung und Datenauswertung. Optionalen Charakter haben hingegen der DWS-Manager zur zentralen Steuerung aller Prozesse im DWS sowie das Metadaten-Repository zur zentralen Bereitstellung von Metadaten (zu diesen Komponenten siehe Abschnitt 2.4).

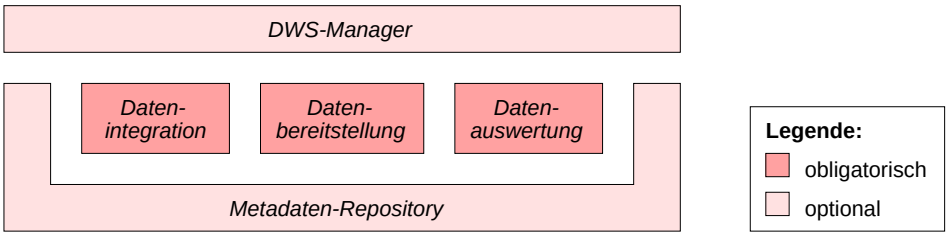


Abbildung 2.1: Grobarchitektur eines DWS (strukturorientierte Sicht)

Die Aufbau- und Ablauforganisation der einzelnen DWS-Komponenten kann sehr unterschiedlich gestaltet sein. So kann die Datenbereitstellung zentral (klassisches Data Warehouse) oder verteilt (sog. *Data Marts*) erfolgen und die Datenauswertung durch ein einzelnes Werkzeug oder eine breite Palette von Werkzeugen unterstützt werden. In der Datenintegrationskomponente reicht das Spektrum vom einfachen Kopieren von Quelldaten bis hin zu komplexen, qualitätsgelenkten Integrationsprozessen, wie sie in Teil II dieser Arbeit diskutiert werden.

Abbildung 2.2 zeigt eine verfeinerte DWS-Architektur, die als Referenzarchitektur für die vorliegende Arbeit dienen soll. Auf Grund der datenflussorientierten Sichtweise sind DWS-Manager und Metadaten-Repository in der Abbildung nicht dargestellt. Die einzelnen Komponenten der Referenzarchitektur werden in den nachfolgenden Abschnitten näher erläutert, gegliedert in Werkzeuge der Datenintegrationsschicht (Abschnitt 2.2), Werkzeuge der Datenauswertungsschicht (Abschnitt 2.3) und schichtenübergreifende Werkzeuge (Abschnitt 2.4, Schichten jeweils gemäß Abb. 2.2). Datenhaltungskomponenten werden separat in Abschnitt 2.1 behandelt. Die Ausführungen basieren auf den gängigen Grundlagenwerken zum Data Warehousing, insbesondere [Inm96, AM97, PR97, KRRT98, Mar98, MB98, Gar98, Kur99, JLVV00, BG01]. Evaluationen von DWS-Werkzeugen finden sich in [WH98].

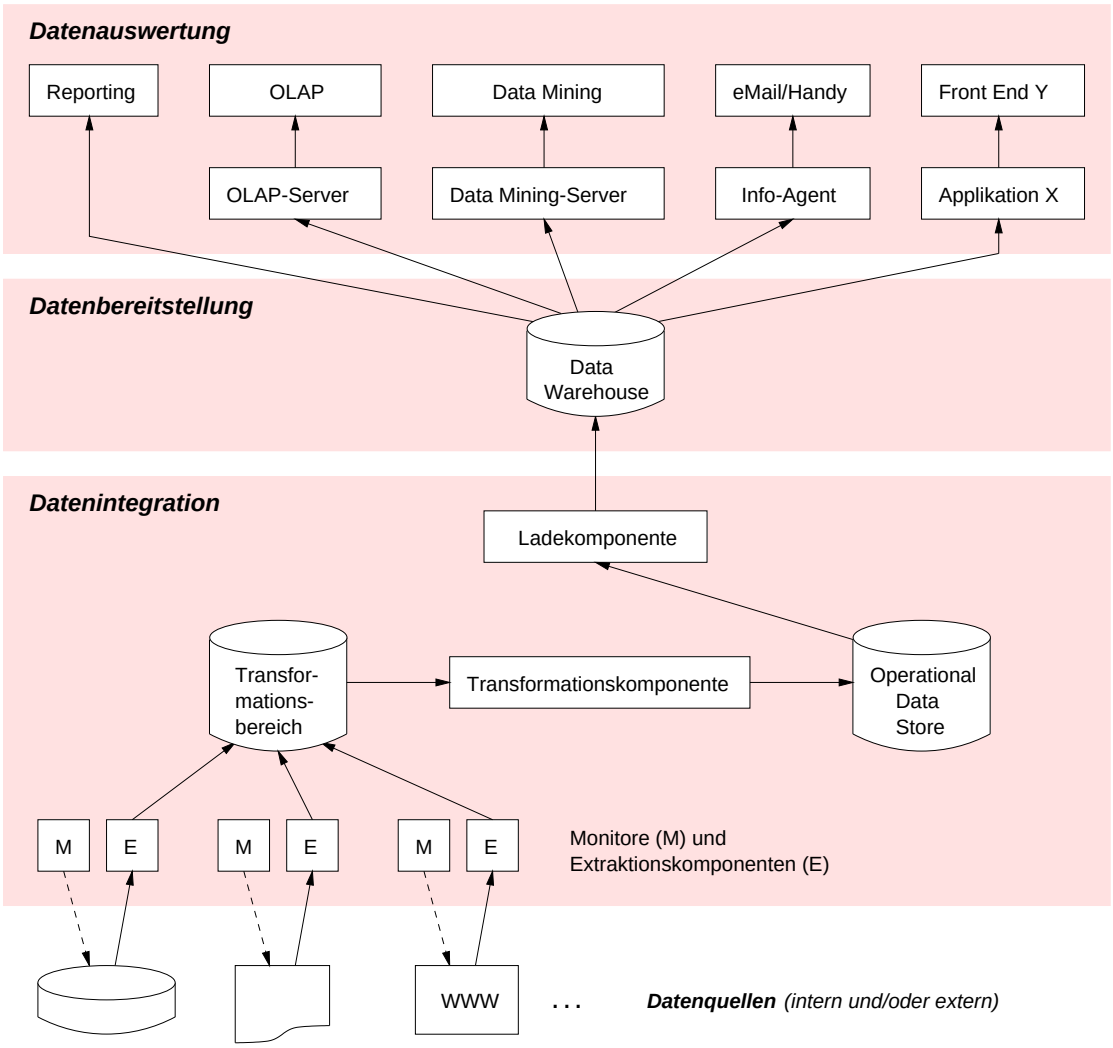


Abbildung 2.2: DWS-Referenzarchitektur (datenflussorientierte Sicht)

2.1 Datenhaltungskomponenten

Ein DWS enthält mehrere Datenhaltungskomponenten. Zusätzlich zum eigentlichen Data Warehouse (Abschnitt 2.1.1) sind dies der *Operational Data Store* (Abschnitt 2.1.2) sowie der *Transformationsbereich* (Abschnitt 2.1.3). Datenquellen (Abschnitt 2.1.4) sind streng genommen nicht Bestandteil eines DWS, wurden hier jedoch der Vollständigkeit halber in die Beschreibung aufgenommen.

2.1.1 Data Warehouse

Zentrale Komponente eines DWS ist das Data Warehouse, eine durch folgende Charakteristika gekennzeichnete Datenbank [BG01]:

- Die Datenbank ist physisch realisiert.
- Sowohl Schema als auch Daten sind integriert.
- Das Schema ist analyseorientiert.
- Daten werden i. Allg. nicht oder nur selten nachträglich modifiziert.
- Daten liegen in der Regel historisiert vor.

Tabelle 2.1 fasst die wesentlichen Unterschiede zwischen einer herkömmlichen operativen Datenbank und einem Data Warehouse zusammen [BS97a, BG01]. Diese Übersicht unterstellt die idealtypische Vorstellung einer weitgehenden „Orthogonalität“ von operativer Datenbank und Data Warehouse, wie sie in der Literatur oft zu finden ist. In einigen Quellen (z. B. [ZS99]) wird darauf hingewiesen, dass „die Welt in Wirklichkeit nicht so schwarzweiß ist“. So können bestimmte Anwendungsszenarien durchaus die Speicherung historisierter Daten in einer operativen Datenbank oder die Manipulation von Daten in einem Data Warehouse erfordern. Ebenso wird man in der Praxis in den operativen Systemen nicht durchgängig auf normalisierte Schemata treffen.

Im Hinblick auf die Analyseorientierung hat sich das multidimensionale Datenmodell für viele Problemstellungen als besonders zweckmäßig erwiesen [TKS01, KS99]. Die Implementierung von multidimensionalen Schemata erfolgt in der gängigen Praxis meist auf einem relationalen Datenbankmanagementsystem (RDBMS). Dazu müssen die multidimensionalen Strukturen auf relationale Strukturen abgebildet werden (Stern- oder Schneeflockenschemata, siehe dazu [Pok99]).

Kriterium	**Operative Datenbank**	**Data Warehouse**
Zweck und Anwendercharakteristika		
Verwendung	Transaktional, Unterstützung und Abwicklung von operativen Geschäftsprozessen	Analytisch, Informationsbereitstellung für Entscheidungsträger
Anwendertyp	Sachbearbeiter	Analysten, Manager
Anwenderzahl	Sehr viele	Relativ wenige
Zugriffscharakteristika		
Zugriffsart von Applikationen	Lesen, Hinzufügen, Modifizieren, Löschen	Lesen, periodisches Hinzufügen
Transaktionsdauer und -typ	Kurze Lese- und Schreibtransaktionen	Lange Lesetransaktionen
Abfragestruktur	Einfach	Komplex, rechenintensiv
Abfragetyp	Vorhersagbar	Häufig ad-hoc–Anfragen
Zugriffsart	Einzelne Datensätze	Bereichsanfragen
Ergebnisvolumen	Wenige Datensätze	Viele Datensätze
Erwartete Antwortzeiten	(Milli-)Sekunden	Sekunden bis Minuten
Updates	Laufend, konkurrierend	Ergänzend
Daten- und Schemacharakteristika		
Datenquellen	(Meistens) eine	(Fast immer) mehrere
Eigenschaften der Daten	Nicht abgeleitet, zeitaktuell, autonom, dynamisch	Abgeleitet, konsolidiert, historisiert, integriert, stabil
Granularität der Daten	Detaildaten	Detaildaten und Aggregate
Aktualität der Daten	Online, Realtime	Unterschiedlich (aufgabenabhängig)
Zeithorizont der Daten	Aktuelle Daten	Historisierte Daten
Datenschema	Normalisiert	Zweckgebunden, analyseorientiert
Datenvolumen	Megabytes bis Gigabytes	Gigabytes bis Terabytes

Tabelle 2.1: Gegenüberstellung von operativer Datenbank und Data Warehouse [Her01a]

2.1.2 Operational Data Store

Der Operational Data Store (ODS) – in [BG01] Basisdatenbank genannt – ist in der DWS-Referenzarchitektur zwischen dem Transformationsbereich (siehe Abschnitt 2.1.3) und dem Data Warehouse angesiedelt [KRRT98, Inm00]. Er dient der analyseneutralen, dauerhaften Verwaltung integrierter, konsolidierter, ggf. auch historisierter Daten feinster Granularität. Ihm kommt eine zentrale Sammel- und Verteilungsfunktion zu, indem aus heterogenen Quellen integrierte Daten – ausgehend vom ODS – für unterschiedliche Analyseanforderungen unterschiedlich aufbereitet (insb. aggregiert) und in das Data Warehouse geladen werden. Diese Struktur, die die Schnittstellen zwischen Datenquellen und Auswertungswerkzeugen minimiert, wird in [BG01] als „Hub-and-Spoke–Architektur“ bezeichnet. Bei Bedarf können Analyseverfahren, die auf dem Data Warehouse aufsetzen, mittels eines sog. *Drill Through* auf die (analyseunabhängig gespeicherten) Detaildaten im ODS zurückgreifen. In diesem Sinne hat der ODS also auch eine Auswertungsfunktion.

2.1.3 Transformationsbereich

Der Transformationsbereich (in [BG01] Arbeitsbereich, gelegentlich in der Literatur auch Staging Area genannt) stellt die zentrale Datenhaltungskomponente der Integrationsschicht (siehe Abschnitt 2.2) dar. Hier werden Daten auf ihrem Weg von den Datenquellen in den ODS temporär zwischengespeichert. Notwendige Datentransformationen können dann direkt auf diesem Zwischenspeicher ausgeführt werden, ohne dass die Datenquellen oder der ODS in ihrem laufenden Betrieb beeinträchtigt werden. Erst nach Abschluss aller Verarbeitungsschritte werden die Daten in den ODS übertragen und aus dem Transformationsbereich entfernt.

2.1.4 Datenquellen

Datenquellen sind der Ausgangspunkt einer datenflussorientierten Betrachtung der DWS-Architektur. Die typischerweise heterogenen Datenquellen können internen Ursprungs (operative Systeme der eigenen Organisation) oder aber extern sein, wobei zunehmend das Internet als Quelle in Frage kommt. Der Auswahl geeigneter Quellen zum Aufbau eines DWS kommt erhebliche Bedeutung zu. Bei dieser Auswahl spielen insbesondere die Faktoren Datenqualität, technische und organisatorische Verfügbarkeit sowie Kosten des Datenerwerbs eine Rolle.

2.2 Werkzeuge der Integrationsschicht

Als Werkzeuge der Integrationsschicht werden zusammenfassend diejenigen DWS-Komponenten bezeichnet, die funktional zwischen den Datenquellen und dem Data Warehouse angesiedelt und für die Integration heterogener Daten zuständig sind. Neben *Monitoren* (Abschnitt 2.2.1) sind dies vor allem Werkzeuge zur *Extraktion* (Abschnitt 2.2.2), zur *Transformation* (Abschnitt 2.2.3) sowie zum *Laden* von Daten (Abschnitt 2.2.4) [LZW+97, HGW+95]. Auf Grund der Anfangsbuchstaben dieser Werkzeugtypen werden die Werkzeuge der Integrationsschicht oft auch als *ETL-Werkzeuge* und der durch sie unterstützte Prozess als *ETL-Prozess* bezeichnet. Als physische Zwischenspeicher in der Integrationsschicht dienen Transformationsbereich (siehe Abschnitt 2.1.3) und ODS (siehe Abschnitt 2.1.2).

2.2.1 Monitore

Monitore sind für die Entdeckung von Datenänderungen (neue, geänderte oder gelöschte Tupel) in Quellsystemen zuständig. Um den ODS – und nachfolgend auch das Data Warehouse – aktuell zu

halten, müssen Updates in den Quellsystemen inkrementell in das DWS propagiert werden. Die konkrete Funktionsweise eines Monitors hängt unmittelbar von den Charakteristika des angeschlossenen Quellsystems sowie von den Anforderungen der Analysekomponenten ab. Dabei kommen als Quellsysteme nicht nur Datenbankmanagementsysteme (DBMS), sondern auch „flache“ Dateien, XML-Dokumente etc. in Frage. Folgende Monitoring-Strategien können unterschieden werden [VGD99]:

- *Trigger-basiert:* Handelt es sich bei der Datenquelle um ein Datenbanksystem (DBS), welches aktive Mechanismen in Form von Triggern unterstützt, kann ein Monitoring erfolgen, indem jede Datenmanipulation einen Trigger auslöst (z. B. Post Update), der das geänderte Tupel in eine Datei oder eine andere Datenstruktur schreibt.
- *Replikationsbasiert:* Moderne DBMS bieten Replikationsdienste an. Diese können so spezifiziert werden, dass sie geänderte Tupel in spezielle Tabellen schreiben.
- *Zeitstempelbasiert:* Jedem Tupel ist ein Zeitstempel zugeordnet, der im Falle einer Änderung des Tupels auf den Zeitpunkt der Änderung gesetzt wird. An Hand der Zeitstempel kann später entschieden werden, welche Tupel sich nach dem Zeitpunkt der letzten Extraktion geändert haben. Temporale DBMS [Sno95] bieten eine explizite Unterstützung der Zeitdimension an.
- *Log-basiert:* In diesem Fall nutzt man die Fähigkeit von DBMS aus, vorgenommene Transaktionen in einer Log-Datei zu protokollieren. Durch Analyse einer solchen Log-Datei kann ermittelt werden, welche Daten sich geändert haben.
- *Snapshot-basiert:* Bei dieser Variante wird der Datenbestand einer Quelle in periodischen Zeitabständen in eine Datei, den sog. *Snapshot*, geschrieben. Durch einen Vergleich von Snapshots (Delta-Berechnung) können Änderungen identifiziert werden.

Von den vorgestellten Monitoring-Strategien erfordert die Snapshot-basierte Variante den größten (Implementierungs-)Aufwand, da sie keine gegebenen Dienste nutzt. Gerade bei Altsystemen, die solche Dienste i. Allg. nicht anbieten, ist Snapshot-Monitoring jedoch häufig die einzig praktikable Technik zur Entdeckung von Änderungen im Datenbestand.

Bei dienstorientierten Quellen (z. B. Web-Services) lässt sich der bei Middleware-Technologien verbreitete *Interception*-Mechanismus zum Monitoring nutzen. Interceptions erlauben es dem Entwickler eines verteilten Anwendungssystems, in das Zusammenspiel zwischen Client, Middleware und Server einzugreifen und bei Eintritt bestimmter Ereignisse, wie z. B. dem Aufruf einer Methode eines Server-Objekts, zuvor bestimmte Operationen zur Ausführung zu bringen (im Kontext des Monitoring insbesondere Protokollierung von datenmanipulierenden Methodenaufrufen). Für nähere Informationen zu Interceptions siehe [Sta00] oder [MKR+00].

2.2.2 Extraktionskomponenten

Die Extraktionskomponente einer an ein DWS angebundenen Datenquelle ist für die Übertragung von Daten in den Transformationsbereich verantwortlich. Je nach verwendeter Monitoring-Strategie (siehe Abschnitt 2.2.1) gestaltet sich die Extraktion sehr unterschiedlich: Bei der Trigger-basierten Variante sind die geänderten Tupel aus den entsprechenden Dateien auszulesen, bei Verwendung der Replikationsdienste können sie per SQL-Anfrage aus den Replikationstabellen selektiert werden. Die zeitstempelbasierte Variante erfordert lediglich die Selektion von Tupeln an Hand ihres Zeitstempels. Bei der Log- bzw. Snapshot-Variante hängt das Vorgehen von der gewählten Umsetzung der Log-Analyse bzw. des Snapshot-Vergleichs ab. Werden die als geändert identifizierten Tupel beispielsweise in eine Datei eingetragen, so ist diese Datei zu importieren.

Eine grundlegende Entscheidung besteht in der Festlegung, welche Datenquellen bzw. Ausschnitte daraus in ein DWS zu integrieren sind. Diese Selektion hängt stark von der inhaltlichen Relevanz der Datenquellen für die geplanten Auswertungen sowie von der Qualität der Quelldaten ab.

Bei der Extraktion werden die selektierten bzw. durch das Monitoring als geändert identifizierten Daten aus den quellenspezifischen Datenstrukturen ausgelesen und in die Datenstrukturen des Transformationsbereichs überführt. Aufgrund der u. U. großen Datenvolumina kann eine Komprimierung der zu transferierenden Daten sinnvoll sein. Die Zeitpunkte, an denen eine Extraktion durchgeführt wird, können je nach Bedarf unterschiedlich gewählt werden. Prinzipiell werden folgende Vorgehensweisen unterschieden [KRRT98]:

- *Periodisch:* Die Extraktion wird in periodischen Zeitabständen durchgeführt, wobei die Periodendauer von der Dynamik der Daten bzw. von den gestellten Anforderungen an die Aktualität der Daten abhängt. So sind z. B. Börsenkurse oder Wetterdaten i. Allg. (mehrmals) täglich zu aktualisieren, während Daten über technische Spezifikationen von Produkten typischerweise beständiger sind und daher mit einer längeren Periodendauer auskommen.
- *Anfragegesteuert:* In diesem Fall wird die Extraktion durch eine explizite Anfrage angestoßen. Wenn beispielsweise eine Produktgruppe um einen neuen Artikel erweitert wird, so kann die Extraktionskomponente angewiesen werden, die in den operativen Quellen zu diesem neuen Artikel gespeicherten Informationen in das DWS zu übertragen.
- *Ereignisgesteuert:* Häufig ist es sinnvoll, einen Extraktionsvorgang durch ein Zeit-, Datenbank- oder externes Ereignis auszulösen. Ein typisches Datenbankereignis wäre beispielsweise der Aufruf einer SQL-Anweisung wie Insert, Update oder Delete. Ein externes Ereignis würde z. B. vorliegen, wenn ein Börsenindex eine bestimmte Marke über- oder unterschreitet. Streng genommen sind auch die periodische und die anfragegesteuerte Extraktion ereignisgesteuert, da sie einem speziellen Zeitereignis bzw. einem durch den Anwender ausgelösten Ereignis entsprechen.
- *Sofort:* Bei besonders hohen Anforderungen an die Aktualität von Daten kann es erforderlich sein, Änderungen in den operativen Quellen unmittelbar in den ODS zu propagieren. Die Daten im ODS sind damit „praktisch“ immer genauso aktuell wie die Daten in den operativen Quellen.

Die technische Realisierung der Extraktion erfolgt typischerweise mit Hilfe von Gateways (Schnittstellen zwischen Netzwerken) und Standard-Datenbankschnittstellen (z. B. ODBC oder OLE DB [Woo99]).

2.2.3 Transformationskomponente

Ein DWS wird i. Allg. von mehreren heterogenen Datenquellen versorgt, die Daten mit disjunkten oder sich überschneidenden Inhalten in unterschiedlichen Repräsentationsformen liefern. Bevor die aus den Quellen extrahierten Daten in einen ODS bzw. ein Data Warehouse geladen werden können, müssen sie daher in einen Zustand einheitlicher Repräsentation überführt werden. Transformationen, die dem Zweck der Vereinheitlichung dienen, werden unter dem Begriff *Data Migration* zusammengefasst.

Die transformierten Daten werden schließlich im ODS abgelegt. Dabei ist im Falle inkrementeller Updates ggf. eine Historisierung vorzunehmen. Ein im DWS gespeichertes Tupel, zu dem es eine Änderung in einer Datenquelle gegeben hat, darf daher nicht grundsätzlich mit dem geänderten Tupel überschrieben werden. Stattdessen ist das geänderte Tupel ggf. mit einem Zeitstempel zu versehen und zusätzlich zu den bereits „veralteten“ Tupeln im DWS abzulegen.

Aufgrund ihrer besonderen Bedeutung im Kontext dieser Arbeit werden die einzelnen Transformationstypen der Data Migration-Phase im Folgenden an Hand von Beispielen vorgestellt. Dabei geht es ausschließlich um die Transformation von Daten, nicht um die Transformation von Datenstrukturen bzw. Schemata. Eine abgeschlossene Schemaintegration ist allerdings eine entscheidende Voraussetzung für den Einsatz einer Transformationskomponente. Nähere Ausführungen zur Schemaintegration finden sich z. B. in [SL90, Con97, OV99].

Schlüsselbehandlung

Bei der Übertragung von Daten aus heterogenen Quellen in ein DWS können Schlüssel von Quelltupeln i. Allg. nicht übernommen werden, da sie nicht global eindeutig sind. Im Rahmen der Transformation werden Quellschlüssel auf Surrogate, also künstlich erzeugte Schlüssel, abgebildet. Die Erzeugung von Surrogaten kann in Oracle z. B. mit Hilfe des SQL-Kommandos `CREATE SEQUENCE` vorgenommen werden. Die Abbildung zwischen Schlüsseln sollte in einer Zuordnungstabelle gespeichert werden, um Updates den richtigen DWS-Tupeln zuordnen zu können (siehe Tab. 2.2). Diese Maßnahme kann ggf. entfallen, wenn das DWS (dauerhaft) nur von einer einzigen Datenquelle „gespeist“ wird bzw. wenn die Verträglichkeit (Disjunktheit und Übereinstimmung des Datentyps) von Schlüsseln aus den angebundenen Datenquellen explizit sichergestellt ist.

Quelle	Relation	Attribut	Lokaler Schlüssel	Globales Surrogat
system1	kunde	kunden_nr	12345	66
system1	kunde	kunden_nr	*44444*	*69*
system2	client	client_id	*44444*	71
system3	customer	customer_id	A134	*69*
system3	customer	customer_id	B777	72

Tabelle 2.2: Zuordnung von lokalen zu globalen Schlüsseln

Ein weiteres Problem besteht in der Schwierigkeit, erkennen zu müssen, ob sich zwei Tupel aus unterschiedlichen Quellen auf dieselbe Entität der realen Welt (hier: denselben Kunden) beziehen oder nicht. Solche Entscheidungen können oft nicht an Hand von Schlüsseln getroffen werden, da diese – wie oben erwähnt – i. Allg. nicht global eindeutig sind. In Abschnitt 6.2 wird näher auf diese Problematik eingegangen.

Die fehlende Eindeutigkeit ist jedoch nicht das einzige Defizit von Quellschlüsseln. Hinzu kommt, dass diese häufig eine implizite Semantik besitzen, die bei der Übernahme ins DWS berücksichtigt werden muss. So könnte z. B. in die Schlüssel einer Artikeldatenbank das Jahr der Einführung des jeweiligen Artikels hineincodiert worden sein („ABC1999“). Die im Schlüssel enthaltenen Informationen müssen daher ggf. extrahiert und in separaten Attributen abgelegt werden.

Behandlung von fehlenden Werten

Im DWS sollten fehlende Werte einheitlich repräsentiert sein. Moderne DBMS bieten dazu einen speziellen Nullwert an, unabhängig vom jeweiligen Datentyp des Attributs. In Legacy-Systemen (Altsystemen) ist dies jedoch häufig nicht der Fall. Stattdessen werden fehlende Werte hier oft durch spezielle Werte des jeweiligen Datentyps dargestellt, z. B. -1 bei numerischen Werten, „ “ bei Zeichenketten oder „9.9.9999“ bei Datumsangaben. Kenntnis über die jeweils verwendeten Werte ist essentiell für deren korrekte Umsetzung in Nullwerte. Ein weiteres Problem besteht in der unterschiedlichen

Semantik von fehlenden Angaben. Folgende drei Arten der Interpretation sind denkbar (siehe auch [Atz00]):

1. „Es gibt in der realen Welt keinen Wert für das Attribut.“ (z. B. Name des Ehepartners bei ledigen Personen)
2. „Es gibt in der realen Welt einen Wert für das Attribut, aber dieser Wert war zum Erfassungszeitpunkt entweder nicht bekannt oder wurde aus bestimmten Gründen nicht erfasst.“ (In operativen Systemen werden z. B. optionale Angaben zu Kunden oft nicht erfasst, weil sie dort nicht notwendig sind. Analysen im DWS können jedoch durchaus auf solche Angaben – z. B. die Geburtsdaten von Kunden – angewiesen sein.)
3. „Es ist nicht bekannt, ob es in der realen Welt einen Wert für das Attribut gibt.“ (z. B. Faxanschluss einer Person)

Im zweiten Fall kann versucht werden, durch Ableitung aus anderen Attributwerten einen Wert zu schätzen. Dieser sollte speziell gekennzeichnet werden. Bestimmte Analyseverfahren – z. B. im Data Mining – erfordern solche Schätzungen, da sie auf vollständige Tupel angewiesen sind. Im ersten Fall wäre eine solche Schätzung jedoch unangebracht, da de facto kein Wert existiert und damit jede Schätzung falsch wäre. Eine Feststellung, welcher Fall jeweils vorliegt, ist schwierig und kann nur durch Ausnutzung von Domänenwissen geschehen.

Anpassung von Datentypen

Falls der Datentyp eines Quellattributs nicht mit dem des korrespondierenden Zielattributs übereinstimmt, ist eine Konvertierung der Attributwerte erforderlich, z. B.:

Character	$\rightarrow$	*Date*	(oft bei Datumsangaben)
Character	$\rightarrow$	*Number*	(oft bei numerischen Schlüsseln)

Konvertierung von Codierungen

Bei Attributen, die codiert vorliegen, ist eine Konvertierung erforderlich, wenn in Quelle und Ziel unterschiedliche Codierungsstandards verwendet werden. Des weiteren kann der Fall auftreten, dass Klartexte zu codieren sind oder – eher selten – Codes in Klartexte aufgelöst werden müssen. Bei der Konvertierung können Lookup-Tabellen hilfreich sein.

Einkaufspreis $\rightarrow 1$, *Verkaufspreis* $\rightarrow 2$	(Klartext $\rightarrow$ Codierung)
$E \rightarrow 1$, $V \rightarrow 2$	(Codierung A $\rightarrow$ Codierung B)

Vereinheitlichung von Zeichenketten

Zur Vereinheitlichung von Zeichenketten können diese transformiert werden, z. B. indem Umlaute oder andere sprachspezifische Sonderzeichen durch eine alternative, i. Allg. mehrbuchstabige Schreibweise ersetzt werden oder indem alles einheitlich klein oder groß geschrieben wird. Weitere Transformationen auf Zeichenketten umfassen die Eliminierung von White Spaces, z. B. führende und nachfolgende Leerzeichen, und die Vereinheitlichung von Trennsymbolen und Abkürzungen.

Die Vereinheitlichung von Zeichenketten senkt beim Record Linkage (siehe Abschnitt 6.2) die Wahrscheinlichkeit von Synonymfehlern (Nichterkennung der Identität zweier Entitäten), erhöht allerdings die Gefahr von Homonymfehlern (Zusammenführung zweier nicht zusammengehöriger Tupel),

da u. U. charakteristische Merkmalsausprägungen verloren gehen können. Entscheidend ist jedoch, dass ein einheitliches Vokabular von Fachbegriffen etabliert wird. Dieses Vokabular sollte zentral im Metadaten-Repository (siehe Abschnitt 2.4.2) verwaltet werden.

Küchengerät	→	*Kuechengeraet*
Kuechengeraet	→	*KUECHENGERAET*
' Videorecorder '	→	*'Videorecorder'*
VCR	→	*Videorecorder*
Klient	→	*Kunde*

Bei multinationalen Data Warehouse-Projekten ist typischerweise eine Abbildung von lokal verwendeten Zeichensätzen auf einen einheitlichen Zeichensatz wie UNICODE [Uni00] erforderlich, z. B. zur Erstellung eines zentralen Produktkatalogs.

Darüber hinaus sind Transformationen denkbar, die Domänenwissen bzw. Wissen über die Semantik von Attributen voraussetzen. Ein Beispiel dafür ist die einheitliche Notation eines Attributs *name* in der Form „<Nachname>, <Vorname>“. Falls Wissen darüber vorliegt, dass alle Einträge der Quelle die Form „<Vorname> <Nachname>“ haben, kann die Transformation auf einfache Weise vorgenommen werden. Ansonsten müsste auf zusätzliches Domänenwissen zurückgegriffen werden, z. B. auf eine Liste aller zulässigen Vornamen. Kritisch sind natürlich Namen, die gleichzeitig Vor- und Nachname sein können, z. B. „Frank“ oder „Claus“.

Kurt Meyer	→	*Meyer, Kurt*	
Meyer Kurt	→	*Meyer, Kurt*	(da „Kurt“ als zulässiger Vorname erkannt wird)
Frank Claus	→	*?*	(unklar, was Vor- und was Nachname ist)

Um Schreib- und Hörfehler zu kompensieren, kann es zusätzlich sinnvoll sein, phonetische Codes zu bilden, z. B. gemäß Kölner Phonetik [Pos75]:

VIDEORECORDER	→	*FEDEUREKURDER*
VIDEOREKORDER	→	*FEDEUREKURDER*

Derartige Operationen können auch der Kategorie „Berechnung abgeleiteter Werte“ (siehe unten) zugeordnet werden.

Vereinheitlichung von Datumsangaben

Moderne DBMS wie Oracle unterscheiden bei Datumsangaben zwischen einer internen und einer externen Repräsentation. Während die interne Repräsentation statisch ist, kann die externe Repräsentation an die jeweiligen Anwenderbedürfnisse angepasst werden, z. B. an ein landesspezifisches Datumsformat. Falls das Ziel-DBS eine solche Trennung unterstützt, ist keine Konvertierung von Datumsangaben erforderlich, da die gewünschte (externe) Repräsentation auf Zielseite jederzeit angepasst werden kann. Erwartet das Ziel-DBS hingegen ein ganz bestimmtes, proprietäres Datenformat, so ist i. Allg. eine Konvertierung unumgänglich, z. B.:

MM-DD-YYYY → *DD.MM.YYYY*

In Altsystemen sind Datumsangaben häufig als Zeichenketten modelliert, so dass vor der eigentlichen Vereinheitlichung des Datumsformats eine Konvertierung des Datentyps erforderlich ist (siehe oben). Des weiteren kann es vorkommen, dass Datumsangaben in Quelle oder Ziel als einzelne numerische Angaben für Tag, Monat und Jahr repräsentiert sind. In diesem Fall ist eine Kombination bzw. Separierung von Attributwerten bei der Konvertierung erforderlich (siehe unten).

Problematisch ist die Situation, wenn Datumsangaben in der Quelle nicht vollständig vorliegen, z. B. mit nur zweistelliger Jahresangabe oder ohne Tagesangabe (nur Monat und Jahr). Im ersten Fall müssen Jahresangaben auf vier Stellen erweitert werden, wobei jeweils auf die Wahl des korrekten Jahrhunderts (z. Zt. meist 19xx oder 20xx) zu achten ist, im zweiten Fall müssen ggf. Dummy-Werte für Tagesangaben eingefügt (und als solche gekennzeichnet) werden, z. B. 1 (Monatsanfang) oder 15 (Monatsmitte). Bei international tätigen Organisationen kann zusätzlich zur Konvertierung von Datumsformaten eine Umrechnung von Zeitzonen notwendig sein. Dazu muss allerdings bekannt sein, auf welche Zeitzone sich eine gegebene Zeitangabe bezieht.

Umrechnung von Maßeinheiten und Skalierungen

Numerische Merkmalsausprägungen besitzen i. Allg. eine Maßeinheit, es sei denn, sie beschreiben eine Anzahl (von Jahren, Artikeln etc.) oder eine Codierung. Beispiele dafür sind Angaben zu Länge, Gewicht etc. Dabei kann die in einer Quelle für ein Attribut verwendete Maßeinheit von der im Ziel benötigten abweichen. Selbst wenn Quelle und Ziel einem Attribut dieselbe Maßeinheit zu Grunde legen, so kann immer noch eine unterschiedliche Skalierung vorliegen. In beiden Fällen sind Umrechnungen zur Angleichung erforderlich, die jedoch bei Kenntnis der vorliegenden Maßeinheiten und Skalierungen relativ trivial sind.

$10\ \textit{inch} \rightarrow 25,4\ \textit{cm}$
$10\ \textit{km} \rightarrow 10.000\ \textit{m}$
$30\ \textit{mph} \rightarrow 48,279\ \textit{km/h}$

Kombination und Separierung von Attributwerten

Manchmal ist es notwendig, mehrere Attributwerte einer Quelle zu einem Attributwert des Ziels zusammenzufassen, beispielsweise bei Datumsangaben mit separat gespeichertem Tag, Monat und Jahr. Eine solche Kombination von Attributwerten kann an Hand einer a priori definierten Berechnungsvorschrift erfolgen. Umgekehrt kann es aber auch erforderlich sein, einen Attributwert der Quelle in mehrere Attributwerte des Ziels zu zerlegen. Beispiele hierfür sind Zerlegungen komplexer Bezeichnungen oder Auflösung zusammengesetzter Schlüssel bzw. Codes.

$\textit{tag} = 18, \textit{monat} = 5, \textit{jahr} = 2001 \rightarrow \textit{datum} = 18.05.2001$
$\textit{bezeichnung} = \text{'SchokoMax, 100g'} \rightarrow \textit{typ} = \text{'SchokoMax'}, \textit{gewicht} = 100$

Es werden hier lediglich Attributwerte (Datenebene) kombiniert bzw. separiert und nicht die zu Grunde liegenden Attribute (Schemaebene). Letzteres ist der Schemaintegration zuzurechnen (siehe z. B. [KS96]). Diese muss abgeschlossen sein (DWS-Entwurf), bevor mit der Datenintegration (DWS-Betrieb) begonnen werden kann.

Berechnung abgeleiteter Werte

Aus zwei Gründen kann es sinnvoll sein, aus bestimmten Attributwerten der Quelle neue Werte abzuleiten, die dann im Ziel gespeichert werden: Entweder werden die Quelldaten in ihrer originären Form im Ziel nicht benötigt – dies wird von Geschäftsregeln festgelegt – oder aber die abgeleiteten Werte werden zusätzlich zu den originären Werten gespeichert, um Effizienzsteigerungen bei Auswertungen zu erzielen. So erfordern z. B. Data Mining-Algorithmen eine spezielle Aufbereitung der Basisdaten [FPS96]. Diese Aufbereitung kann weitaus komplexer sein, als die unten angegebenen Beispiele vermuten lassen.

$\textit{einführ_datum} = \ldots \rightarrow \textit{verkaufszeitraum} = \textit{tagesdatum} - \textit{einführ_datum}$
$\textit{preis_exkl_mwst} = \ldots \rightarrow \textit{preis_inkl_mwst} = \textit{preis_exkl_mwst} * 1.16$

2.2.4 Ladekomponente

Die Ladekomponente ist für die Übertragung von konsolidierten Daten aus dem ODS in die analyseorientierten Strukturen des Data Warehouse zuständig. Zur technischen Durchführung dient dabei oft das Ladewerkzeug des jeweils verwendeten DBMS, z. B. der Oracle SQL*Loader [DG98].

Es kann zwischen Online- und Offline-Ladevorgängen unterschieden werden [AM97]: Beim Online-Laden steht das Data Warehouse auch während des Ladens für Anfragen zur Verfügung, beim Offline-Laden ist dies nicht der Fall. Üblicherweise findet nur das initiale Laden offline statt. Inkrementelle Updates sollten hingegen nur online durchgeführt werden, um den laufenden Betrieb des DWS nicht unterbrechen zu müssen. Das Zeitfenster für den Ladevorgang sollte so gewählt werden, dass die Beeinträchtigung der Benutzung minimiert wird, z. B. nachts oder an Wochenenden.

Während des Ladens von Daten in ein Data Warehouse findet häufig eine Aggregation numerischer Kennzahlen statt. Dabei werden feingranulare Daten bzgl. analyserelevanter Kriterien (Dimensionen, z. B. Zeit, Ort oder Produkt) in eine gröbere Granularität überführt und so im Data Warehouse abgelegt. Die einfachste Form der Aggregierung ist die Summierung, z. B.:

$$\textit{monatsumsatz} = \text{Summe}(\ \textit{tagesumsatz}, \textit{monat}\ )$$

Aufgrund der möglicherweise sehr großen Datenvolumina können spezielle Maßnahmen zur Effizienzsteigerung erforderlich werden, z. B. durch Parallelisierung der Ladevorgänge (siehe dazu [DG98]). Weiterhin ist analog zur Extraktionskomponente eine angemessene Ausnahme- und Fehlerbehandlung unerlässlich.

2.3 Werkzeuge der Auswertungsschicht

Die Auswertungswerkzeuge eines DWS stellen Dienste zur Datenabfrage und -analyse sowie zum Berichtswesen bereit. Als Analysemethodiken sind OLAP [Mar98, Wie00, TKS01] und Data Mining [FPS96, BS97a, WK97, BHM98] verbreitet. Obwohl Data Mining auch ohne Data Warehouse möglich ist, bietet das Aufsetzen auf einem Data Warehouse erhebliche Vorteile. So können viele Zusammenhänge erst durch eine integrierte, historisierte Sicht entdeckt werden. Auf der anderen Seite sei aber auch auf die Nachteile von Data Mining in einem DWS hingewiesen. So können z. B. für OLAP optimierte Schemastrukturen sich als ungeeignet zum Data Mining erweisen. Eine Gegenüberstellung der Vor- und Nachteile findet sich in [Gün00]. Neben OLAP- und Data Mining-Systemen sind weitere Auswertungswerkzeuge bzw. -endgeräte denkbar, z. B. auf Basis von mobilen Diensten (Mobiltelefon etc.).

Da Auswertungswerkzeuge nicht im Fokus dieser Arbeit stehen, wird nicht näher darauf eingegangen. Für Details sei auf [BG01] verwiesen.

2.4 Schichtenübergreifende DWS-Komponenten

Die in diesem Abschnitt erläuterten Komponenten DWS-Manager (Unterabschnitt 2.4.1) und Metadaten-Repository (Unterabschnitt 2.4.2) übernehmen schichtenübergreifende Aufgaben der Steuerung, Kontrolle und Metadatenverwaltung (vgl. Abb. 2.1).

2.4.1 DWS-Manager

Der DWS-Manager ist für die zentrale Steuerung und Überwachung der einzelnen, im DWS stattfindenden Prozesse zuständig. Dazu koordiniert er insbesondere das Zusammenspiel der beteiligten

Komponenten, inklusive einer angemessenen Ausnahme- und Fehlerbehandlung. Da Daten üblicherweise im Batch-Betrieb integriert werden, wäre es unpraktikabel, den gesamten Integrationsprozess bei Vorliegen einer Ausnahme oder eines Fehlers anzuhalten und erst nach Behebung des Problems fortzusetzen – zumindest, solange das Problem lokal begrenzt ist, z. B. im Falle einer Integritätsverletzung eines Tupels. Stattdessen wird eine Protokollierung des Problemfalls vorgenommen und der Integrationsprozess fortgesetzt. Datentupel, bei denen Probleme aufgetreten sind, können dann später manuell nachbearbeitet werden. Nur für den Fall, dass die Anzahl der Ausnahmen bzw. Fehler ein a priori definiertes Limit übersteigt oder der Schweregrad des Problems eine Fortsetzung der Integration nicht zulässt (z. B. Hardwarefehler oder fehlerhaftes Datenformat), wird der Prozess abgebrochen. Dabei muss der DWS-Manager ein geregeltes Anhalten und Wiederaufsetzen von Prozessen sicherstellen.

Für die Konfigurierung der an dem DWS beteiligten Komponenten sowie für den reibungslosen Betrieb des DWS ist ein (menschlicher) DWS-Administrator zuständig. Dieser muss dazu mit dem DWS-Manager (Softwarekomponente) interagieren, um z. B. Protokollinformationen entgegenzunehmen und Änderungen in der Prozessplanung durchzuführen [TS97].

2.4.2 Metadaten-Repository

Eine Grundvoraussetzung für den effektiven Umgang mit Daten ist das Vorhandensein von Wissen über deren Syntax und Semantik (inkl. Qualitätseigenschaften). Sind diese „Daten über Daten“ (Metadaten) formal repräsentiert und zentral verfügbar, können Datenverarbeitungsprozesse einerseits hochautomatisiert, andererseits sehr flexibel gestaltet werden. Im DWS ist das sog. *Metadaten-Repository* für die zentrale Verwaltung und Bereitstellung von Metadaten zuständig.

Die gebräuchlichste Klassifikation unterscheidet *technische* und *domänenspezifische* Metadaten (engl.: Business Metadata). Domänenspezifische Metadaten werden in der Literatur auch als *Geschäftsmetadaten* bezeichnet. Den technischen Metadaten werden dabei (im Kontext des Data Warehousing) insbesondere zugerechnet:

- *Schemaspezifikationen:* Diese umfassen zumindest die Angaben eines klassischen Data Dictionary (im relationalen Datenmodell: Relationen- und Attributschemata, Fremdschlüsselbeziehungen etc.), ggf. aber zusätzlich weitere Informationen, z. B. Maßeinheiten von Attributen.
- *Spezifikationen von Datenintegrationsprozessen:* Datenintegrationsprozesse lassen sich mit gerichteten Graphen beschreiben, indem Datenquellen und Datensenken als Knoten und die Datenflüsse zwischen ihnen als mit Transformationsoperatoren attributierte Kanten aufgefasst werden.
- *Statistiken aus dem bzw. für den operativen Betrieb:* Hierzu gehören z. B. Statistiken über Verteilungen von Datenzugriffen, Qualitätsmängel von Daten sowie Systemauslastung und -verfügbarkeit.

Domänenspezifische Metadaten umfassen unter Anderem:

- *Geschäftsregeln:* Diese können insbesondere zur Prüfung der Konsistenz von neu zu integrierenden Daten, aber auch zu anderen Zwecken hinzugezogen werden, z. B. zur Festlegung, wie Werte eines Attributs aggregiert werden dürfen (Summenbildung etc.).
- *Organisationsstrukturen:* Die Modellierung der Aufbauorganisation erlaubt es, organisatorische Einheiten in anderen Metadatenmodellen zu verwenden (z. B. bei der Zuordnung von Ressourcen zu Geschäftsprozessen) und dabei Beziehungen zwischen Einheiten auszunutzen (z. B. Weisungsbefugnisse).

- *Spezifikationen von Geschäftsprozessen:* Diese sind sowohl zu Dokumentationszwecken als auch zur Ausführung durch ein Workflow Management System (WfMS) [JBS97] verwendbar.
- *Begriffstaxonomien:* Für den erfolgreichen Betrieb eines DWS ist es unerlässlich, dass alle Beteiligten auf einer gemeinsamen begrifflichen Grundlage kommunizieren. Hierfür kann eine formal repräsentierte, organisationsweit zugreifbare Begriffstaxonomie bzw. Ontologie hilfreich sein, bestehend aus einem Vokabular und einer Menge von logischen Aussagen zur Einschränkung der Menge der möglichen Interpretationen für Worte (allg.: Symbole) und deren Beziehungen [MSS01, Fer99, LJ99, MKIS96, Gru95]. Beispiele für Taxonomien im Bereich medizinischer Informationssysteme gibt [Has99].

Das Metadaten-Repository kommuniziert mit den anderen DWS-Komponenten, die entweder Metadaten anfordern (z. B. Schemabeschreibungen) oder selbst erzeugte Metadaten ablegen möchten (z. B. Zugriffsstatistiken). Mit dem Management von Metadaten in DWS beschäftigen sich u. A. [SR96, Rot96, Wie98, MSR99, SVV01, NR99].

2.5 Zusammenfassung und Bewertung des Status Quo

In diesem Kapitel wurden die einzelnen Bestandteile eines DWS charakterisiert. Ein Schwerpunkt lag dabei auf den Komponenten der Integrationsschicht, die in ihrer Gesamtheit den sog. *ETL-Prozess* (Extraktion – Transformation – Laden) realisieren. Zur Unterstützung des ETL-Prozesses existieren zahlreiche kommerzielle Softwarewerkzeuge (siehe Anhang A).

Die im Zusammenhang mit der Transformationskomponente diskutierte Funktionalität betrifft in erster Linie syntaktische Aspekte der Datenintegration, insbesondere die Vereinheitlichung der Repräsentation von Daten. Wie in Kapitel 3 erläutert wird, ist „Einheitlichkeit“ jedoch nur eines von diversen Datenqualitätsmerkmalen. Andere Merkmale wie Konsistenz, Redundanzfreiheit, Vollständigkeit und Zeitnähe, die eher auf die Semantik und Pragmatik (Verwendung) von Daten ausgerichtet sind, werden im ETL-Prozess oft vernachlässigt, obwohl sie entscheidenden Einfluss auf die Eignung von Daten für vorgesehene Auswertungen haben. Gängige ETL-Werkzeuge bieten nur einen Bruchteil der erforderlichen Funktionalität, was zur Folge hat, dass Daten in der Praxis oft ohne jegliche Qualitätskontrolle in ein DWS geladen und dann ausgewertet werden.

Kapitel 3

Datenqualität

Dieses Kapitel geht vertiefend auf einzelne Merkmale von Datenqualität (Abschnitt 3.2) sowie auf Datenqualitätsmängel und deren Ursachen ein (Abschnitt 3.3). Zur Veranschaulichung wird in Abschnitt 3.1 zunächst eine Analogie zwischen einem Datenintegrationsprozess und einem Fertigungsprozess gezogen.

3.1 Analogie zur industriellen Fertigung

Die Konzepte des QM wurden zwar ursprünglich für die industrielle Fertigung bzw. das Dienstleistungsgewerbe entwickelt, lassen sich jedoch auch auf den Kontext des Data Warehousing übertragen, da der Prozess der Datenintegration grundsätzlich mit einem Fertigungsprozess vergleichbar ist (siehe Abb. 3.1). Dabei entsprechen die von einer Datenquelle (Lieferant) bereitgestellten heterogenen, unbereinigten Daten den Rohmaterialien, die einen Integrations- bzw. Aufbereitungsprozess durchlaufen. Das Ergebnis dieses Prozesses sind „Datenprodukte“, die dann Kunden (hier: Datennutzer) zu bestimmten (Auswertungs-)Zwecken zur Verfügung gestellt werden. Analog zur industriellen Fertigung sollte der gesamte „Produktionsprozess“ qualitätsgelenkt, also auf die Erfüllung von Kundenanforderungen ausgerichtet sein.

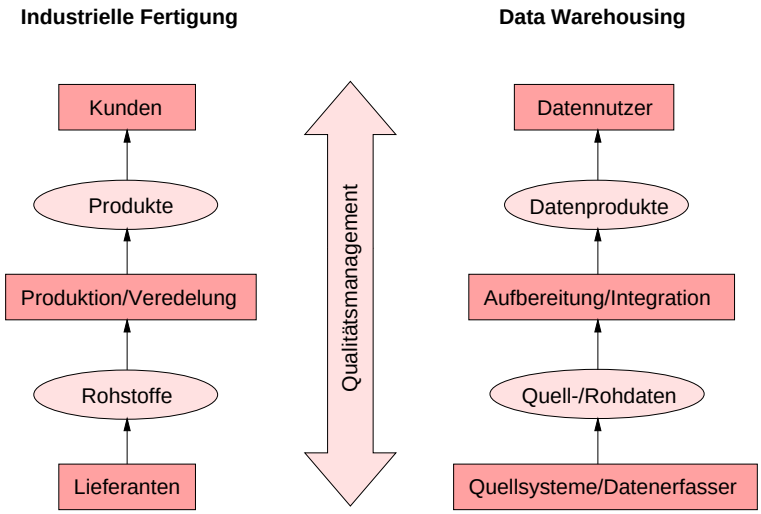


Abbildung 3.1: Analogie zur industriellen Fertigung

Der Hauptunterschied der beiden Sichtweisen liegt im Betrachtungsgegenstand und in dessen (Qualitäts-)Merkmalen: Im industriellen Fertigungsprozess werden physische Produkte bzw. Güter

produziert (z. B. Bauteile für die Automobilindustrie), die Merkmale wie Belastbarkeit, Länge, Motorleistung, Haltbarkeit etc. aufweisen. Im Kontext der Datenverarbeitung entspricht dem „Produkt“ ein bestimmter Ausschnitt eines (ggf. integrierten) Datenbestands, im Folgenden *Datenprodukt* genannt. Ein Datenprodukt kann ein einzelner Wert, ein Datensatz oder eine beliebige Menge von Datensätzen sein.

In Anlehnung an die Definition des Begriffs „Qualität“ (siehe Abschnitt 1.1) wird „Datenqualität“ wie folgt definiert:

Definition 3.1 (Datenqualität): Grad, in dem ein Satz inhärenter Merkmale eines Datenprodukts Anforderungen erfüllt. □

Die Merkmale eines Datenprodukts unterscheiden sich zwangsläufig von den Merkmalen eines physischen Produkts. Beispiele für derartige Merkmale sind Vollständigkeit, Konsistenz (im Sinne logischer Widerspruchsfreiheit) und Zeitnähe. Nutzeranforderungen an Datenprodukte können über diese Merkmale spezifiziert werden. Allerdings ergibt sich dabei das Problem der Quantifizierung des Erfüllungsgrades von Anforderungen und der Messbarkeit von Merkmalsausprägungen (siehe dazu Abschnitt 6.1).

Beispiel 3.1: Anforderungen bzgl. physischer Produkte lassen sich meist auf einfache Weise mittels etablierter Maße spezifizieren. So könnte eine Anforderung an das Merkmal „Länge“ eines Produkts „Schraube“ lauten: „Die Länge darf um maximal 0.5 mm von der geplanten Länge 10 cm abweichen.“ Die Messung des Erfüllungsgrades dieser Anforderung an einem gegebenen Produkt ist ebenfalls problemlos möglich – das Vorhandensein geeigneter Messinstrumente (hier: zur Längenmessung) vorausgesetzt. Wie jedoch spezifiziert man eine Anforderung an das Merkmal „Konsistenz“ eines Datenprodukts, und wie misst man den Erfüllungsgrad dieser Anforderung? □

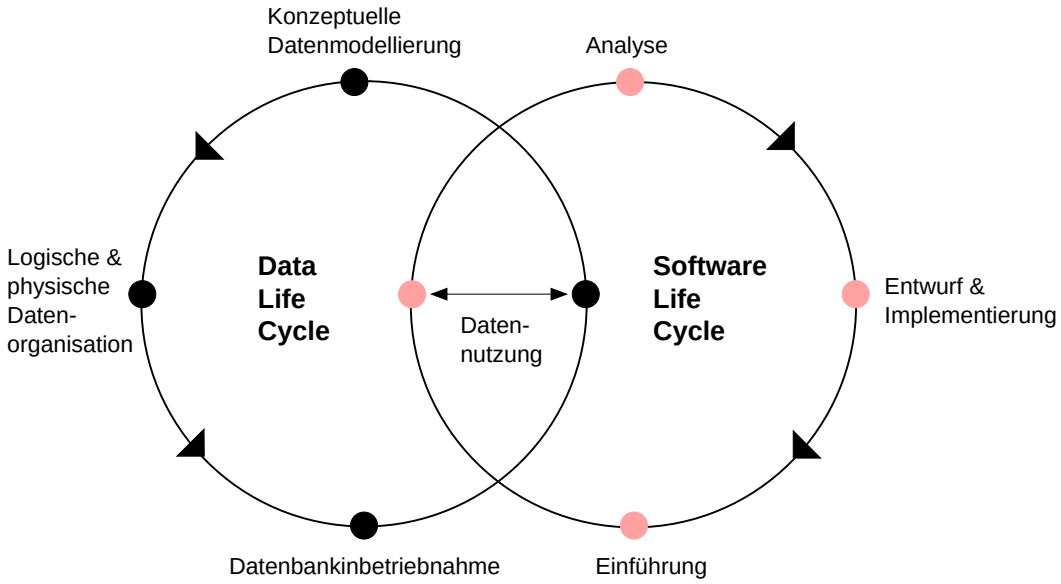


Abbildung 3.2: System-Bi-Cycle (frei nach [Ort95])

Die Qualität von Daten betrifft ihren gesamten „Lebenszyklus“, von der konzeptuellen Datenmodellierung über die logische und physische Datenorganisation und die Verwaltung in einem DBMS bis hin zur Datennutzung. Dabei dürfen Daten allerdings nicht isoliert betrachtet werden. Vielmehr sind sie immer im Zusammenhang mit der Software (und deren Nutzern) zu sehen, mit Hilfe derer sie verwaltet, manipuliert oder analysiert werden. [Ort95] definiert daher einen sog. *System-Bi-Cycle*, der die Verzahnung der Lebenszyklen von Daten und Software verdeutlicht (siehe Abb. 3.2). In der Da-

tennutzungsphase wirkt sich die Qualität der Software (bzw. die Qualität der Bedienung der Software durch den Anwender) unmittelbar auf die Qualität der bearbeiteten Daten aus.

3.2 Merkmale von Datenprodukten

Bevor näher auf Merkmale von Daten(-produkten) eingegangen wird, wird zunächst geklärt, was Daten sind und wie der Begriff „Daten“ von den verwandten Begriffen „Information“ und „Wissen“ abgegrenzt werden kann.

3.2.1 Daten, Information, Wissen

In der Literatur finden sich verschiedene Definitionen des Begriffs „Daten“ (u. A. [Pfe74, Hau84, Fis92, Kuh91]). Dieser Arbeit soll eine Definition zu Grunde liegen, die aus der Semiotik (Zeichentheorie) stammt. Semiotik setzt sich aus den Gebieten Syntaktik, Semantik und Pragmatik zusammen, welche zueinander in einer Dreiecksbeziehung stehen (siehe Abb. 3.3): Während sich die Syntaktik mit der Struktur von Zeichenfolgen befasst, beschäftigt sich die Semantik mit der (Be-)Deutung und die Pragmatik mit der Verwendung solcher Zeichenfolgen. Analog zu dieser Dreiteilung lassen sich nun die Begriffe „Daten“, „Wissen“ und „Information“ definieren:

- *Daten* sind Zeichenfolgen, die Wissen repräsentieren (vgl. Syntaktik).
- *Wissen* ist eine Menge von Aussagen über einen Weltausschnitt, die als wahr angenommen werden und die tatsächlich wahr sind (vgl. Semantik).
- *Information* entsteht aus Daten, wenn diese in einen Kontext gesetzt werden, der vor dem Hintergrund gegebenen Wissens eine Interpretation der Daten erlaubt (vgl. Pragmatik).

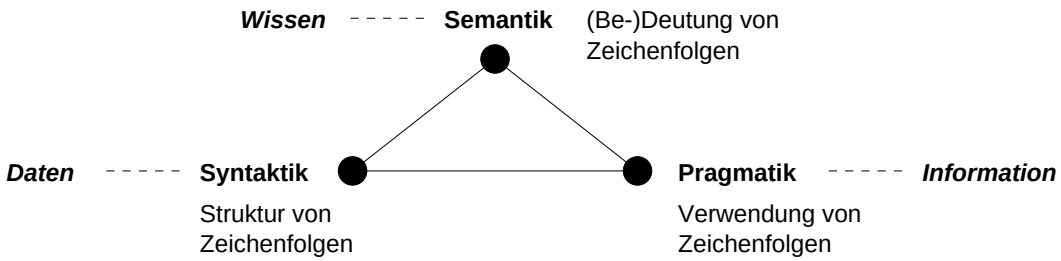


Abbildung 3.3: Semiotisches Dreieck (frei nach [Lew90])

Der in dieser Arbeit durchgängig verwendete Begriff „Datenqualität“ bezieht sowohl die syntaktischen Eigenschaften (z. B. Einheitlichkeit der Repräsentation) als auch die semantischen Eigenschaften (z. B. Korrektheit, Zeitnähe) von Daten ein. In der Literatur wird der Begriff „Datenqualität“ meist synonym zum Begriff „Informationsqualität“ genutzt.

3.2.2 Ansätze zur Identifizierung und Klassifizierung von Datenqualitätsmerkmalen

Die Definition von (Daten-)Qualität als „fitness for use“, wie man sie in der Fachliteratur häufig findet [Wan98], ist zwar prägnant, hilft aber in der Praxis kaum weiter. Vielmehr ist es entscheidend, allgemeingültige (Qualitäts-)Merkmale von Datenprodukten zu identifizieren und – soweit möglich

– zu klassifizieren [NR00]. Die Merkmale solcher immateriellen Objekte unterscheiden sich zwangsläufig von typischen Merkmalen physischer Rohmaterialien oder Produkte (Ausdehnung, Gewicht, Belastbarkeit etc.). In der Literatur werden Qualitätsmerkmale von Datenprodukten oft als Datenqualitätsdimensionen, vereinzelt auch als Datenqualitätsfacetten bezeichnet. In dieser Arbeit wird durchgängig der an die Norm ISO 9000:2000 [Deu00a] angelehnte Begriff (Daten-)Qualitätsmerkmal verwendet. Folgende Ansätze zur Bestimmung von Datenqualitätsmerkmalen sind bekannt:

- *Intuitiver Ansatz*, z. B. [WRK92]

  Festlegung von Merkmalen aufgrund des intuitiven Verständnisses eines Experten.

- *Empirischer Ansatz*, z. B. [GH93, WSG94, SLW97b]

  Identifizierung von Merkmalen, Beziehungen zwischen Merkmalen sowie Bestimmung der Bedeutsamkeit von Merkmalen und Beziehungen auf der Basis von Kundenbefragungen. Die Anzahl der aus empirischen Studien hervorgehenden Datenqualitätsmerkmale kann sehr groß sein ([WSG94] nennt z. B. 118 Merkmale), so dass aus Gründen der Handhabbarkeit eine Gruppierung, Hierarchisierung und Priorisierung von Merkmalen notwendig wird. [SLW97b] nimmt eine Gruppierung in die Klassen „intrinsisch“, „kontextabhängig“, „repräsentationsorientiert“ und „zugriffsorientiert“ vor.

- *Dienstleistungsorientierter Ansatz*, z. B. [KSW97]

  Identifizierung von Merkmalen auf der Basis der Anwendung von Konzepten der Dienstgüte (Quality of Service) auf den Kontext der Lieferung von Informationen an „Informationsverbraucher“ (intakte, nützliche, nutzbare und effektive Information).

- *Semiotischer Ansatz*, z. B. [SC99, Sha99, Kra91]

  Identifizierung von Merkmalen auf der Basis einer Auswahl der in [Sta92] definierten semiotischen Ebenen (vgl. Abschnitt 3.2.1), nämlich „syntaktisch“ (Struktur und technische Verfügbarkeit von Daten), „semantisch“ (inhaltliche Bedeutung von Daten), „pragmatisch“ (Verwendungseignung von Daten) und „sozial“ (Verständnis von Daten).

- *DBS-orientierter Ansatz*, z. B. [Mün95]

  Identifizierung von Merkmalen auf der Basis der klassischen 3-Ebenen-Architektur eines DBS. Hierbei wird unterschieden zwischen einem zentralen logischen (bzw. konzeptuellen) Schema, das von jeglichen Implementierungsdetails abstrahiert, einem internen Schema, das das logische Schema in einem konkreten DBMS implementiert, sowie verschiedenen externen Schemata, die nutzerspezifische Sichten auf das logische Schema darstellen. In [Mün95] werden die technischen Merkmale Verfügbarkeit, Zugänglichkeit, Darstellungsform und Datensicherheit mit dem internen Schema (physische Datenqualität) assoziiert, die inhaltsbezogenen Merkmale Korrektheit, Präzision, Zuverlässigkeit, Vollständigkeit und Konsistenz mit dem logischen Schema (logische Datenqualität) sowie die kontextabhängigen Merkmale Relevanz und Aktualität/Rechtzeitigkeit mit den externen Schemata (fachliche Datenqualität). Bei Rechtzeitigkeit wird unterschieden zwischen „zu früh“, „zu spät“ und „passend“. Dabei kann „passend“ im anwendungsspezifischen Einzelfall bedeuten: „frühestmöglich“, „just in time“, oder „fristgerecht“.

- *Systemorientierter Ansatz*, z. B. [WW96]

  Identifizierung von Merkmalen auf der Basis der Betrachtung von Zuständen in der Diskurswelt und korrespondierenden, ggf. falsch zugeordneten Zuständen im Informationssystem (unvollständige Repräsentation, falsche Repräsentation, bedeutungslose Zustände, mehrdeutige Zustände).

Wenngleich sich die oben genannten Ansätze in der Methodik zur Bestimmung von Qualitätsmerkmalen unterscheiden, so sind die jeweiligen Ergebnisse doch zumindest teilweise vergleichbar. Tabelle 3.1 ordnet einzelne Kategorien einander zu.

Empirisch [SLW97b]	**Dienstleistungs-orientiert [KSW97]**	**Semiotisch [SC99]**	**DBS-orientiert [Mün95]**
intrinsisch	intakt	semantisch	logisch
kontextabhängig	nützlich	pragmatisch	fachlich
repräsentationsorientiert	nutzbar	syntaktisch	physisch
zugriffsorientiert	effektiv		physisch
		sozial	

Tabelle 3.1: Gegenüberstellung von Ansätzen zur Merkmalsklassifizierung

3.2.3 Definition von Datenqualitätsmerkmalen

Aus QM-Sicht ist der empirische Ansatz zu bevorzugen, da er die Erfordernisse und Erwartungen von Kunden in den Mittelpunkt stellt. Abbildung 3.4 zeigt eine empirisch ermittelte Klassifikation von Datenqualitätsmerkmalen.

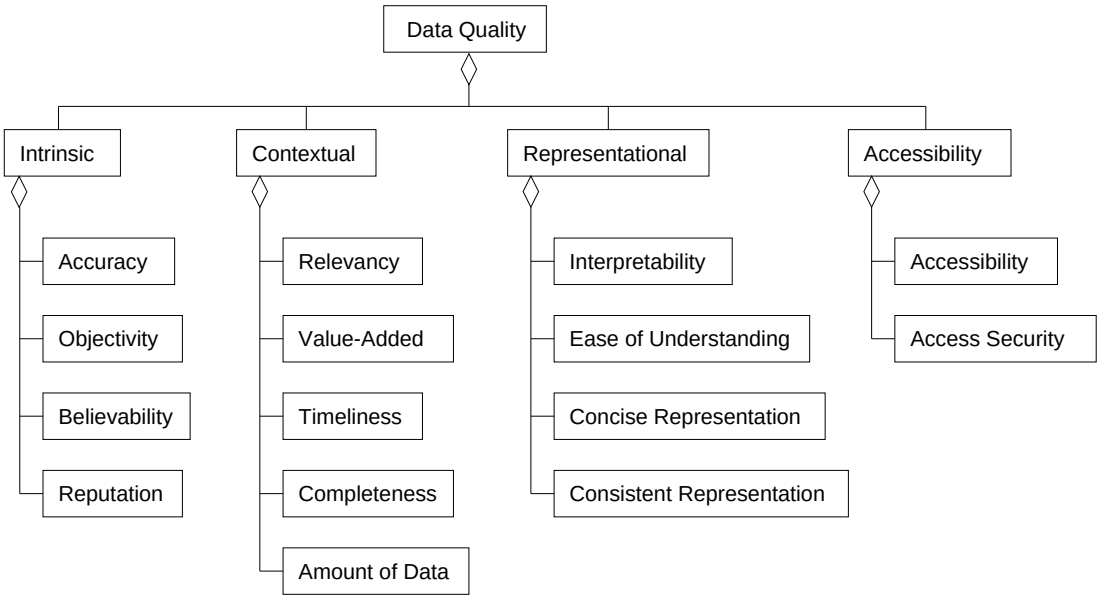


Abbildung 3.4: Taxonomie von Datenqualitätsmerkmalen nach [SLW97b] (UML-Notation)

Folgende Kritikpunkte können gegen diese Klassifizierung vorgebracht werden:

- Merkmalsbezeichner bleiben undefiniert und werden nicht klar voneinander abgegrenzt, so bleibt z. B. der Unterschied zwischen „Believability“ und „Reputation“ unklar.
- Wichtige Merkmale wie z. B. „Absence of Redundancy“ fehlen.
- Die Kategorie „Accessibility“ beschreibt keine Merkmale von Daten, sondern Merkmale des umgebenden Informationssystems.

- Die Begriffe „Intrinsic“ (kontextunabhängig) und „Contextual“ (kontextabhängig) sind komplementär, so dass man auf dieser Ebene keine weiteren Kategorien erwarten würde.
- Die Zuordnung von Merkmalen zu Kategorien ist nicht immer eindeutig, so ließe sich z. B. „Completeness“ auch durchaus als intrinsisches Merkmal auffassen.

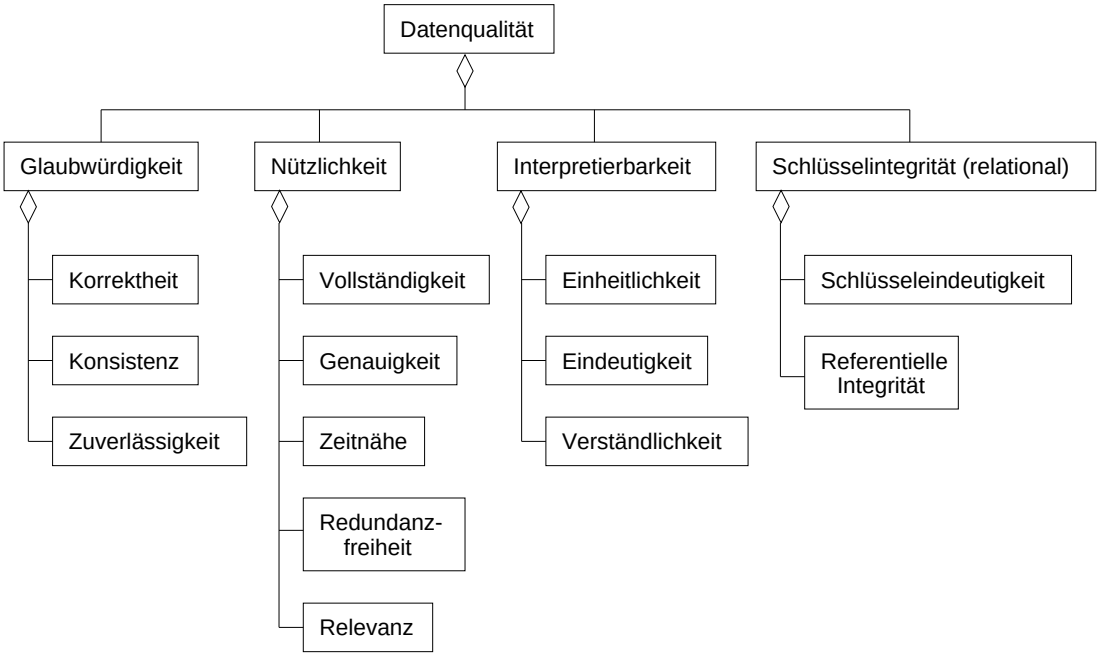


Abbildung 3.5: Taxonomie von Datenqualitätsmerkmalen nach [Hin01a] (UML-Notation)

Aufgrund dieser Kritikpunkte wurde die Taxonomie aus Abb. 3.4 im Rahmen der vorliegenden Arbeit grundlegend überarbeitet [Hin01a] (siehe Abb. 3.5). Allerdings ist die Zugehörigkeit von Merkmalen zu Kategorien auch in der überarbeiteten Taxonomie nicht immer eindeutig, da Merkmale wie „Konsistenz“ oder „Vollständigkeit“ letztendlich sowohl die Glaubwürdigkeit als auch die Nützlichkeit von Daten beeinflussen. In der revidierten Taxonomie wird zwischen den vier Kategorien Glaubwürdigkeit, Nützlichkeit, Interpretierbarkeit und Schlüsselintegrität (speziell für das in den späteren Kapiteln dieser Arbeit im Mittelpunkt stehende relationale Datenmodell, logische Ebene) unterschieden. Jeder Kategorie wurden verschiedene Merkmale zugeordnet. Da sich in der Literatur praktisch keine Definitionen von Datenqualitätsmerkmalen finden lassen, musste für die Arbeit neu definiert werden, was unter den einzelnen Merkmalen konkret zu verstehen sein soll:

- *Korrektheit:* die Eigenschaft, dass die Attributwerte eines Datenprodukts (im Informationssystem) denen der modellierten Entitäten (in der Diskurswelt) entsprechen.
- *Konsistenz:* die Eigenschaft, dass die Attributwerte eines Datenprodukts keine logischen Widersprüche aufweisen.
- *Zuverlässigkeit:* die Eigenschaft, dass Attributwerte nicht mit einem Unsicherheitsfaktor belegt, d. h. nicht vage oder unsicher sind (entspricht damit Robustheit, optimierbar durch Ausschaltung von subjektiven Ermessensspielräumen und Festlegung eindeutiger Verfahrensvorschriften [Mün95]).
- *Genauigkeit:* die Eigenschaft, dass die Attributwerte eines Datenprodukts in dem jeweils „optimalen“ Detaillierungsgrad – abhängig vom jeweiligen Anwendungskontext – vorliegen.

- *Vollständigkeit:* die Eigenschaft, dass die Attribute eines Datenprodukts mit Werten belegt sind, die semantisch vom Wert *NULL* abweichen. Dabei sind die in Abschnitt 2.2.3 vorgestellten Nullwertsemantiken zu beachten. Eine weitere, in dieser Arbeit nicht näher betrachtete Charakterisierung von Vollständigkeit ist die Eigenschaft, dass alle im modellierten Diskursbereich vorkommenden Entitäten im Informationssystem repräsentiert sind.

- *Zeitnähe:* die Eigenschaft, dass die Attributwerte bzw. Tupel eines Datenprodukts jeweils dem aktuellen Diskursweltzustand entsprechen, d. h. nicht veraltet sind.

- *Redundanzfreiheit:* die Eigenschaft, dass innerhalb eines Datenprodukts keine Duplikate vorkommen. Als Duplikate werden in diesem Kontext Tupel verstanden, die dieselbe Entität der Diskurswelt beschreiben, aber – insbesondere aufgrund von Qualitätsmängeln – nicht notwendigerweise in allen Attributwerten übereinstimmen.

- *Relevanz:* die Eigenschaft, dass sich der Informationsgehalt eines Datenprodukts bzgl. eines gegebenen Kontextes mit dem Informationsbedarf einer Anfrage bzw. Auswertung deckt.

- *Einheitlichkeit:* die Eigenschaft, dass ein Datenprodukt eine einheitliche Repräsentationsstruktur aufweist.

- *Eindeutigkeit:* die Eigenschaft, dass ein Datenprodukt eindeutig interpretiert werden kann, d. h. zum Datenprodukt Metadaten hoher Qualität vorliegen, die dessen Semantik festschreiben. Metadatenqualität ist wiederum im Hinblick auf die genannten Datenqualitätsmerkmale zu bewerten.

- *Verständlichkeit:* die Eigenschaft, dass ein Datenprodukt so repräsentiert ist, dass es in seiner Begrifflichkeit und Struktur mit der Vorstellungswelt eines Fachexperten übereinstimmt bzw. (im weiteren Sinne) maschinell verarbeitet werden kann.

- *Schlüsseleindeutigkeit:* die Eigenschaft, dass Primärschlüssel einer Relation eindeutig sind.

- *Referentielle Integrität:* die Eigenschaft, dass zu jedem Fremdschlüssel ein Primärschlüssel in der referenzierten Relation existiert und dass die im Schema spezifizierte Multiplizität der Beziehung (1:0..1, 1:1, 1:1..*, 1:* in UML-Notation) eingehalten wird.

Es muss an dieser Stelle betont werden, dass die Menge der in dieser Arbeit definierten Datenqualitätsmerkmale keinen Anspruch auf Vollständigkeit oder Alleingültigkeit erhebt. Alternative Klassifizierungen, wie sie in der Literatur zu finden sind, sollen damit nicht obsolet gemacht werden. Vielmehr soll eine formale Grundlage geschaffen werden, auf denen die in dieser Arbeit entwickelten Konzepte aufbauen können. Ein kanonischer, allgemein etablierter Satz von Datenqualitätsmerkmalen wäre zwar wünschenswert, eine Entwicklung in diese Richtung ist jedoch in der aktuellen Forschung nicht erkennbar.

Es ist leicht einsehbar, dass der Erfüllungsgrad einer Anforderung an ein Merkmal eines gegebenen Datenprodukts variieren kann und dass ein Erfüllungsgrad von 100% unter Praxisbedingungen einerseits oft nicht erreicht werden kann, andererseits von den auf dem Datenprodukt arbeitenden Auswertungsverfahren aber auch nicht gefordert wird bzw. werden darf. Vielmehr müssen je nach Auswertungsverfahren Sollwerte spezifiziert werden, die einer notwendigen „Mindestqualität“ entsprechen. Metriken, auf deren Basis Anforderungen (im Rahmen einer Qualitätsplanung) spezifiziert und Erfüllungsgrade von Anforderungen gemessen werden können, sind Thema von Kapitel 6. Im folgenden Abschnitt werden Ursachen für Datenqualitätsmängel untersucht.

3.3 Datenqualitätsmängel

Bei Datenqualitätsproblemen lassen sich Single Source-Probleme (eine Datenbasis) und Multi Source-Probleme (mehrere heterogene, sich ggf. inhaltlich überlappende Datenbasen) unterscheiden. Innerhalb beider Problembereiche kann eine Klassifizierung in schema- und instanzbezogene Probleme vorgenommen werden (siehe Abb. 3.6). In der vorliegenden Arbeit werden Multi Source-Probleme auf Schemaebene ausgeklammert, da sie nicht ohne Schemaänderungen lösbar sind.

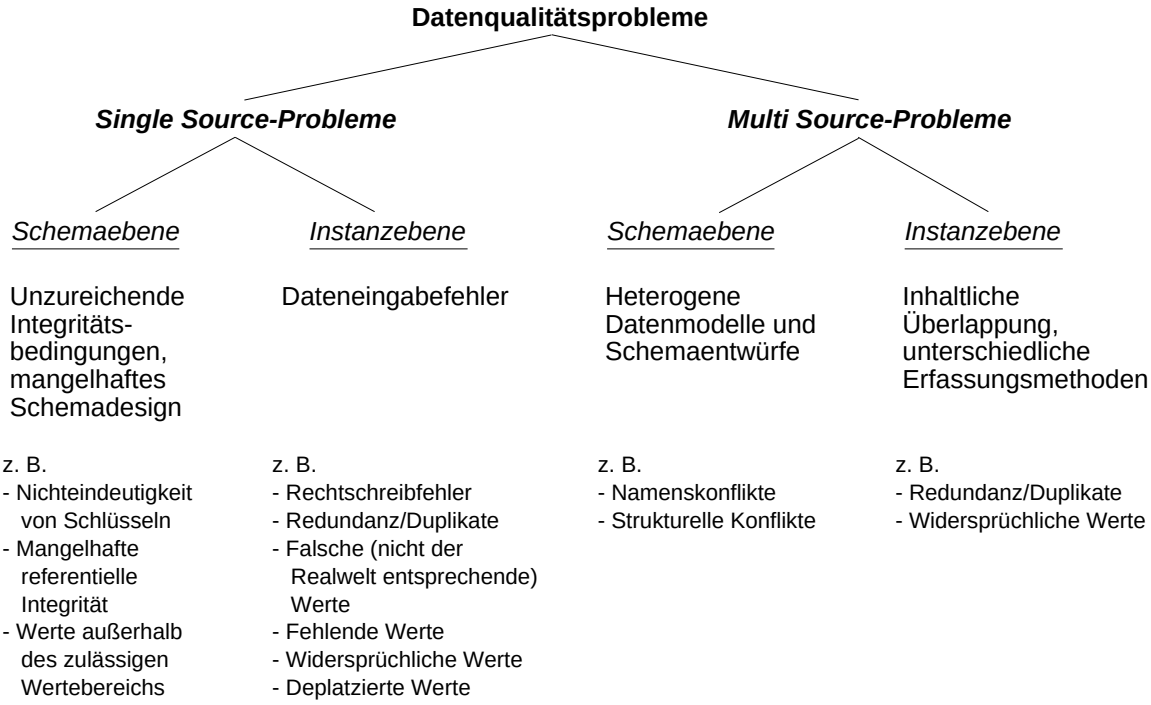


Abbildung 3.6: Klassifikation von Datenqualitätsproblemen nach [RD00]

Das Erkennen von Daten, die den gestellten Qualitätsanforderungen nicht genügen, und das Identifizieren der Ursachen bzw. der Verursacher stellt eine wesentliche Grundlage für Maßnahmen zur Verbesserung der Qualität von Daten dar. Dem Begriff „Fehler“ wird in diesem Abschnitt die nachfolgende, prozessorientierte Definition zu Grunde gelegt:

Definition 3.2 (Fehler): Ungewünschte Abweichung vom angestrebten Verarbeitungsprozess, ausgelöst durch fehlerhafte Bedienung, Software- oder Hardwareprobleme. □

Fehler machen sich in Form von Mängeln bemerkbar, so kann z. B. ein Fehler in einem Produktionsprozess daran erkannt werden, dass das resultierende Produkt den Anforderungen nicht genügt. Fehler verursachen also (Qualitäts-)Mängel.

3.3.1 Ursachen für Datenqualitätsmängel

Um die Ursachen für Datenqualitätsmängel zu ergründen, ist es sinnvoll, ausgehend von der Entwicklung der verwendeten IT-Systeme über die Erfassung der Daten bis zu den auf Datenbeständen durchgeführten Analysen alle Verarbeitungsphasen auf mögliche Ursachen mangelhafter Qualität zu untersuchen. Grob kann dabei zwischen Entwicklungs- und Betriebsphasen des datenverarbeitenden Systems unterschieden werden:

1. *Entwicklungsphasen* (gemäß den gängigen Vorgehensmodellen des Software Engineering)

   (a) *Systemanalysephase*
   In dieser Phase geht es um die Erfassung und Bewertung von Anforderungen an das datenverarbeitende System bzw. an dessen Handhabung. Es ist zu unterscheiden zwischen Kundenanforderungen, gesetzlichen und behördlichen sowie weiteren durch die Organisation festgelegten Anforderungen [Deu00b].

   (b) *Entwurfsphase*
   Auf der Basis der zuvor ermittelten Anforderungen werden in dieser Phase die Datenschemata der beteiligten Datenhaltungskomponenten definiert und die zur Erfassung und Verarbeitung von Daten zu verwendenden Softwarekomponenten entworfen.

   (c) *Implementierungs- und Softwareauswahlphase*
   In dieser Phase werden die zuvor entworfenen Systembestandteile implementiert bzw. – soweit sie als Komponenten bereits verfügbar sind – ausgewählt. Dabei gilt, dass Software, die speziell auf die geplanten Datenverarbeitungsprozesse zugeschnitten wurde, i. Allg. umfassendere Eingabekontrollen (Vollständigkeit auf Datensatzebene, Abhängigkeiten zwischen Attributen, Kontrolle von Wertebereichen etc.) gemäß der gestellten Anforderungen bieten kann als Standardsoftware.

   Nachträgliche Änderungen am Datenschema bzw. Softwaresystem, die aufgrund von veränderten Anforderungen (z. B. neuen Geschäftsregeln) vorgenommen werden, sind ebenfalls der Implementierungs- und Softwareauswahlphase zuzurechnen.

   (d) *Testphase*
   Mittels systematisch zu entwickelnder Testszenarien können in den vorangegangenen Entwicklungsphasen begangene Fehler identifiziert und daraufhin Nachbesserungen veranlasst werden, so dass ein iterativer Entwicklungsprozess entsteht.

2. *Betriebsphasen*
   Im Folgenden werden die typischen Betriebsphasen des Data Warehousing betrachtet. Andere Systemumgebungen weisen möglicherweise abweichende Phasen auf. Grundsätzlich gilt jedoch immer das klassische EVA-Prinzip (Eingabe – Verarbeitung – Ausgabe).

   (a) *Datenerfassungsphase*
   In dieser Phase werden Daten manuell (mit Hilfe bereitgestellter Softwarekomponenten) oder automatisiert (durch Messgeräte) erfasst, ggf. innerhalb eines operativen Systems weiterverarbeitet und persistent gespeichert.

   (b) *Datenübertragungsphase* (DWS-spezifisch)
   Diese Phase umfasst die Extraktion von Daten aus den angeschlossenen operativen Datenquellen (bei inkrementellen Updates: nur neue oder geänderte Daten) und deren Übertragung in den Transformationsbereich des DWS. Das zur Datenübertragung verwendete Medium ist dabei prinzipiell beliebig (CD-R, Diskette, Netzwerk etc.).

   (c) *Datenintegrationsphase* (DWS-spezifisch)
   Diese Phase umfasst alle Aktivitäten der Integration von Daten innerhalb des DWS, also insbesondere die Transformation und Bereinigung von Daten sowie das Laden von konsolidierten Daten in die analyseorientierten Strukturen des Data Warehouse.

   (d) *Datenhaltungs- und Analysephase*
   Diese Phase umfasst alle Aktivitäten zur Datenhaltung (z. B. physikalische Speicherverwaltung, Datenreplikation etc.) sowie alle (vom Nutzer initiierten) Aktivitäten zur Datenanalyse.

Im Folgenden werden die genannten Phasen im Hinblick auf potentielle Fehler (als Ursachen für Datenqualitätsmängel) analysiert. Die Entwicklungsphasen werden dabei gemeinsam betrachtet, da

sie nicht im Mittelpunkt dieser Arbeit stehen. Es wird grundsätzlich zwischen *menschlichen Fehlern* (Entwickler-, Administrator-, Erfasser- und Nutzerfehler) und *technischen Fehlern* (Hardware- und Softwarefehler) unterschieden.

Mängelursachen in den Entwicklungsphasen

Eine fehlerhafte Modellierung von Datenschemata kann dann zu einer Minderung der Datenqualität führen, wenn gestellte Anforderungen nicht hinreichend im Modellierungsprozess berücksichtigt werden (z. B. weil sie nicht erfasst oder nicht angemessen bewertet worden sind). Die daraus entstehenden Mängel sind entweder dem Entwickler (Nichtbeachtung vorgegebener Spezifikationen bzgl. der notwendigen Daten bzw. Metadaten) oder den Kunden (lückenhafte Anforderungsspezifikation) zuzuschreiben.

Vergleichbare Probleme treten auf, wenn Daten zwar korrekt modelliert, aber falsch in den Metadaten beschrieben werden (z. B. Währung in DM statt in Euro). Der Übergang zwischen Datenmodellierung und Datenbeschreibung ist hier fließend.

Bei der Implementierung bzw. Auswahl von Softwarekomponenten können weitere Mängel induziert werden, wenn bei der technischen Realisierung des vorgegebenen Entwurfs unsauber gearbeitet wird (z. B. Programmierfehler, Auswahl ungeeigneter Komponenten, falsche Konfigurierung von Komponenten, zweckentfremdete Nutzung von Datenbankattributen).

Die Testphase induziert i. Allg. zwar keine neuen Mängel in das System, eine unsachgemäße Durchführung kann jedoch der Erkennung von in vorangegangenen Phasen entstandenen Mängeln hinderlich sein.

Bei der Softwareentwicklung wird i. Allg. darauf geachtet, dass das resultierende Softwaresystem ausschließlich konsistente Eingaben des Erfassers zulässt, z. B. regelbasiert per Trigger oder Constraint. Im Gegensatz dazu weisen Legacy-Systeme häufig nur unzureichende Eingabekontrollen auf, was die Eingabe inkonsistenter oder lückenhafter Daten ermöglicht. Dieser Effekt tritt ebenso bei Softwarekomponenten auf, die nicht speziell zur Erfassung der modellierten Daten implementiert bzw. konfiguriert wurden (z. B. Standard-Tabellenkalkulationen).

Mängelursachen in der Datenerfassungsphase

In dieser Phase ist ein weitgehend übereinstimmendes Verständnis von Datenerfasser und Datennutzer über die Semantik der zu erfassenden Daten und deren Relevanz für Auswertungen entscheidend, beispielsweise sollte ein Datennutzer über Charakteristika des Datenerfassungsprozesses informiert sein (Messinstrumente mit begrenzter Auflösung, zeitlich verzögerte Dateneingabe etc.). Da im Data Warehousing Daten oft über die Zeit fortgeschrieben werden, besteht hier die Gefahr einer zweckentfremdeten Datennutzung, da zum Zeitpunkt der Erfassung möglicherweise andere Anforderungen an die Daten vorgelegen haben und die späteren Analysevorhaben nicht bekannt waren. Unterscheiden sich die Auffassungen von Erfasser und Nutzer signifikant, was i. Allg. nur durch eine intensive Kommunikation zwischen den beteiligten Parteien aufgedeckt werden kann, so resultiert dies ggf. in einer Erfassung irrelevanter Daten bzw. in einer (zu) unsauberen Erfassung relevanter Daten.

Beispiel 3.2: Die von einem Nutzer gestellten Anforderungen bzgl. des Datenqualitätsmerkmals Genauigkeit seien sehr hoch (z. B. auf vier Nachkommastellen genaue Erfassung eines Attributwerts), dem Erfasser der Daten jedoch nicht bekannt. Dann könnte der Erfasser, unbewusst der Tragweite seiner Entscheidung, z. B. eine Rundung auf zwei Nachkommastellen vornehmen. □

Folgende weitere Ursachen für Erfassungsfehler sind denkbar:

- Fehlinterpretation des Diskursweltzustands
- Mangelnde Konzentration
- Mangelnde Fertigkeiten
- Demotivation/Gleichgültigkeit
- Bewusste Verfälschung.

Unzureichende Eingabekontrollen leisten den folgenden typischen Eingabefehlern Vorschub:

- Nichterfassung einer Entität
- Fehlende Attributwerte
- Ungenaue Attributwerte
- Veraltete Attributwerte
- Falsche (der Diskurswelt nicht entsprechende) Attributwerte
- Unzulässige (dem Wertebereich nicht entsprechende) Attributwerte
- Inkonsistente (in sich widersprüchliche) Attributwertkombinationen
- Redundante Datenerfassung.

Zu den klassischen „Tippfehlern“, die zu falschen, unzulässigen oder inkonsistenten Attributwerten führen können, gehören (im weitesten Sinne):

- Buchstabierfehler (z. B. bedingt durch Phonetik bei Eigennamen)
- Zeichentranspositionen („Zahlendreher“ etc.)
- Worttranspositionen
- Verwechslung von Eingabefeldern
- Verwendung von Synonymen (inkl. Abkürzungen und unterschiedlichen Sprachen)
- Präfix- und Suffixvariationen um einen stabilen Wortstamm herum.

In analoger Weise können technische Fehler zu einer Minderung der Datenqualität führen. So können im Falle der automatischen Erfassung durch defekte oder unzureichend justierte Messgeräte falsche Werte geliefert werden. Die Verantwortung hierfür liegt beim Administrator des datenerfassenden Systems.

Schließlich können natürlich auch mangelhafte Datenverarbeitungsprozesse, die der Erfassung nachgeschaltet sind, Mängel der o. a. Arten in den Datenbestand induzieren.

Mängelursachen in der Datenübertragungsphase (DWS-spezifisch)

Bei der Übertragung von Datenbeständen von den Datenquellen zum DWS können technische Fehler, z. B. in Form von Netzwerkfehlern oder fehlerhaften Datenträgern, auftreten. Fehlerhafte Datenverarbeitungsprozesse zur Vor- bzw. Nachbereitung der Übermittlung (z. B. Export aus einer Datenbank) bzw. deren fehlerhafte Bedienung können wiederum Mängel der oben genannten Arten zur Folge haben. Die Verantwortung liegt hier bei den Administratoren der Datenquellen und des DWS.

Mängelursachen in der Datenintegrationsphase (DWS-spezifisch)

In der Datenintegrationsphase werden i. Allg. eine Reihe von Transformations- und Bereinigungsprozessen zur Vereinheitlichung und Konsolidierung von Daten durchgeführt. Die Konfigurierung dieser Prozesse obliegt dem DWS-Administrator (vgl. Abschnitt 2.4.1) bzw. dem „Datenverantwortlichen“ des DWS (nach [Imh98]) und muss in angemessener Weise erfolgen, um ein Auftreten der oben beschriebenen Mängel zu vermeiden.

Mängelursachen in der Datenhaltungs- und Analysephase

Der laufende Betrieb eines DBMS ist insbesondere durch die Aufrechterhaltung der Verfügbarkeit der Datenbasis sowie durch die Bereitstellung von Verfahren zur Datenanalyse geprägt. Bei letzterer stehen lesende Datenzugriffe im Vordergrund, die die Datenqualität nicht beeinflussen. Eine Ausnahme stellt das Qualitätsmerkmal Zeitnähe dar: Da Daten ständig altern (Ausnahmen: invariante Angaben wie Geburtsdaten und Vornamen sowie historisierte Daten mit explizitem Ende des Gültigkeitszeitraums), nimmt der Erfüllungsgrad von Anforderungen an die Zeitnähe zwangsläufig stetig ab. Dieser Effekt kann nur durch regelmäßige inkrementelle Updates neuer/aktualisierter Quelldaten kompensiert werden.

Wenngleich Datenanalysen die Datenqualität nicht direkt beeinflussen, so bleibt doch die Gefahr bestehen, dass Nutzer gegebene Analyseverfahren in unzulässiger Weise anwenden bzw. unzulässige Schlüsse aus Analyseergebnissen ziehen. In diesem Fall ist selbst eine optimale Datenqualität kein Garant mehr für eine hohe Qualität strategischer und taktischer Entscheidungen.

Fehler in der Datenhaltungs- und Analysephase umfassen weiterhin technische Fehler, die die Integrität der gespeicherten Daten gefährden (Festplattendefekte, Sicherheitslücken, Datenreplikation mit unvollständiger Nachführung von Updates etc.).

3.3.2 Weitere Klassifizierungen von Datenqualitätsmängeln

Die im vorigen Abschnitt vorgestellte Klassifizierung von Ursachen für Datenqualitätsmängel beruht auf der Betrachtung der logischen Abfolge von Datenverarbeitungsphasen, betrachtet also die Entstehungsorte von Mängeln. Es sind jedoch auch weitere Klassifizierungen denkbar, die an dieser Stelle nur kurz erwähnt werden sollen:

- hinsichtlich der Erkennungsorte von Mängeln (hängt von der Umsetzung des DQM ab, siehe Kapitel 4)
- hinsichtlich der Typen von Mängeln (systematisch oder sporadisch)
- hinsichtlich der Typen von Fehlern als Ursachen für Mängel (technische Fehler, Modellierungsfehler, Erfassungsfehler, Nutzungsfehler)
- hinsichtlich der Datenqualitätsmerkmale bzw. der Kategorien von Merkmalen (vgl. Abb. 3.5).

3.4 Zusammenfassung

Aufbauend auf einer Analogie zwischen dem Prozess der Datenintegration (in einem DWS) und einem industriellen Fertigungsprozess wurde in diesem Kapitel der Begriff des „Datenprodukts“ eingeführt. Die Qualität eines Datenprodukts wird bestimmt durch seine Merkmale. In der Literatur finden sich unterschiedliche Ansätze zur Identifizierung und Klassifizierung von Datenqualitätsmerkmalen. Ausgehend von einer Untersuchung dieser Ansätze wurde ein Satz von Merkmalen festgelegt, der als Grundlage für die in dieser Arbeit zu entwickelnden Konzepte dienen soll. An Hand eines typischen Datenintegrationsprozesses wurden schließlich Ursachen für Datenqualitätsmängel klassifiziert.

Kapitel 4

Datenqualitätsmanagement

In der industriellen Fertigung und im Dienstleistungsgewerbe haben sich im Laufe des 20. Jahrhunderts unterschiedliche, mit sich ändernden Anforderungen (Massenproduktion, Kundenorientierung) gewachsene Konzepte des QM etabliert. Wie in Kapitel 3 gezeigt wurde, lassen sich diese Konzepte auf den Kontext der Datenintegration übertragen. Dabei stellen aufbereitete Daten Produkte dar, die Kundenanforderungen genügen müssen. In diesem Kapitel werden Grundlagen eines Qualitätsmanagements von Daten behandelt. Nach einem kurzen geschichtlichen Überblick in Abschnitt 4.1 folgt in Abschnitt 4.2 ein Exkurs über die Normenfamilie ISO 9000:2000 ff., den heute wohl bedeutendsten QM-Standard. Ansätze zur Anpassung der etablierten QM-Konzepte an den Kontext der Datenverarbeitung werden in Abschnitt 4.3 vorgestellt, gefolgt von einer detaillierten Betrachtung einzelner Maßnahmen zur Datenqualitätsplanung, -lenkung und -sicherung in den Abschnitten 4.4 bis 4.6.

4.1 Geschichtliche Entwicklung des Qualitätsmanagements

Die Entwicklung des QM seit den Anfängen um ca. 1900 bis heute lässt sich nach [Red99] in vier Stufen einteilen:

- *Sortierend*, um 1900 (F. W. Taylor, H. Ford)
  Qualitätskontrolle. Ansätze dieser Stufe sind durch eine klare Trennung von Produktion und Kontrolle gekennzeichnet. Die Qualitätskontrolle beschränkt sich im Wesentlichen auf eine Endkontrolle.
- *Steuernd*, um 1930 (W. A. Shewhart)
  Statistische Qualitätskontrolle. Charakteristisches Merkmal dieser Stufe ist die Einführung von Teilkontrollen auf Basis statistischer Verfahren, um die Leistungsfähigkeit und den Durchsatz der Prüfabteilungen zu erhöhen.
- *Vorbeugend*, um 1960 (G. Tagushi, W. E. Deming, A. V. Feigenbaum et al.)
  Qualitätsmaßnahmen in allen Organisationsbereichen. Ansätze dieser Stufe zeichnen sich durch die Ausweitung des QM auf die Entwicklung und Produktion aus. Ziel ist es, Mängel so früh wie möglich ursachenorientiert zu beseitigen, d. h. dort, wo sie entstehen, und nicht erst dort, wo sie entdeckt werden.
- *Integrierend*, um 1990 (W. E. Deming, A. V. Feigenbaum, W. Masing et al.)
  Ständige Qualitätsverbesserung. Diese Ansätze sind vor allem dadurch gekennzeichnet, dass sie eine ständige Qualitätsverbesserung als von der Organisationsleitung vorzugebendes und zu verantwortendes Organisationsziel definieren, das im Rahmen des QM verfolgt werden muss.

Zu den wichtigsten Wegbereitern und Mitbegründern der modernen QM-Philosophien gehören Deming und Feigenbaum. Deming entwickelte eine organisationsweite Qualitätsphilosophie, deren Kern vierzehn Managementprinzipien zur ständigen Qualitätsverbesserung bilden. Diese Prinzipien wurden in Japan unter dem Begriff „Kaizen“ umgesetzt und weiterentwickelt [Red99]. Der sog. *Plan-Do-Check-Act–Zyklus (PDCA-Zyklus)* (siehe Abb. 4.1) beschreibt Demings Vorstellungen als Regelkreis.

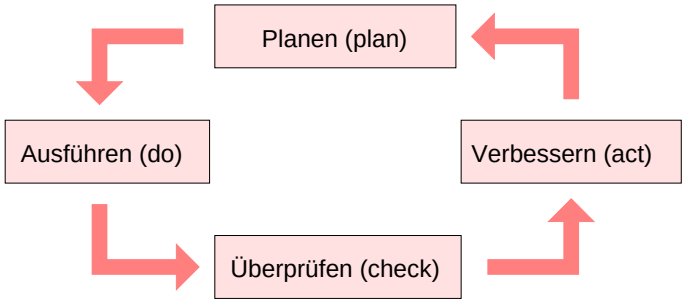


Abbildung 4.1: Plan-Do-Check-Act–Zyklus [Red99]

Feigenbaum formulierte 1961 das Konzept des *Total Quality Control (TQC)* [Fei91]. TQC beschreibt eine organisationsumfassende Qualitätsstrategie, die sich an Kundenbedürfnissen ausrichtet. Angelehnt an das in Japan verwendete TQC wurde in den USA das *Total Quality Management (TQM)* entwickelt, welches zunehmend auch in Europa Anwendung findet [Dem00, Mal99b].

4.2 Exkurs: Normenfamilie ISO 9000:2000 ff.

Im europäischen Wirtschaftsraum stellt die seit ihrer Einführung 1987 mehrfach revidierte internationale Normenfamilie ISO 9000 ff. einen wichtigen Standard für Qualitätsmanagementsysteme (QMS) dar. Auch außerhalb Europas nimmt die Bedeutung der Normenfamilie stetig zu [Joh00, Mal99b]. Dies schließt auch solche Länder ein, in denen QM-Konzepte eine sehr viel längere Tradition haben als in Europa, insbesondere Japan (TQC) und die USA (TQM). Dieser Abschnitt beschäftigt sich ausschließlich mit der im Dezember 2000 veröffentlichten Revision ISO 9000:2000 ff. Wenn im Folgenden von der ISO 9000 gesprochen wird, ist diese Revision gemeint.

Ziel der ISO 9000-Normenfamilie ist es, Organisationen beim Umsetzen und Betreiben von effektiven QMS zu unterstützen. Die Normenfamilie besteht aus folgenden Standards:

- *ISO 9000:* Grundlagen und Begriffe
- *ISO 9001:* Anforderungen
- *ISO 9004:* Leitfaden zur Leistungsverbesserung
- *ISO 19011:* Leitfaden für das Auditieren von Qualitäts- und Umweltmanagementsystemen.

In diesem Abschnitt steht die Norm ISO 9001 [Deu00b] im Vordergrund, da deren Anforderungen sich auch auf das Management der Qualität von Daten anwenden lassen, wie an späterer Stelle gezeigt werden wird.

Die ISO 9001 fordert einen prozessorientierten Ansatz beim Entwickeln, Umsetzen und Verbessern eines QMS, um die Zufriedenheit von Kunden durch Erfüllung von Anforderungen zu steigern [Wag01]. Eine Organisation muss dazu Aktivitäten der Produktrealisierung identifizieren, diese miteinander verknüpfen, Ressourcen zuordnen und so ein System kommunizierender Prozesse aufbauen. Abbildung 4.2 illustriert solch ein prozessorientiertes QMS. Dabei ist die Organisationsleitung für das Management von Ressourcen im Produktrealisierungsprozess verantwortlich. Der Prozess

selbst wird ständig überwacht und analysiert, wobei aus den Analyseergebnissen Verbesserungsmaßnahmen abgeleitet werden. Man erkennt hier deutlich die Verwandtschaft zu Demings PDCA-Zyklus (vgl. Abb. 4.1). Kunden spielen eine wichtige Rolle in diesem Modell, da ihre Anforderungen als Eingaben in den Produktrealisierungsprozess genutzt werden und ihre Zufriedenheit mit dem gelieferten Produkt ständig analysiert wird.

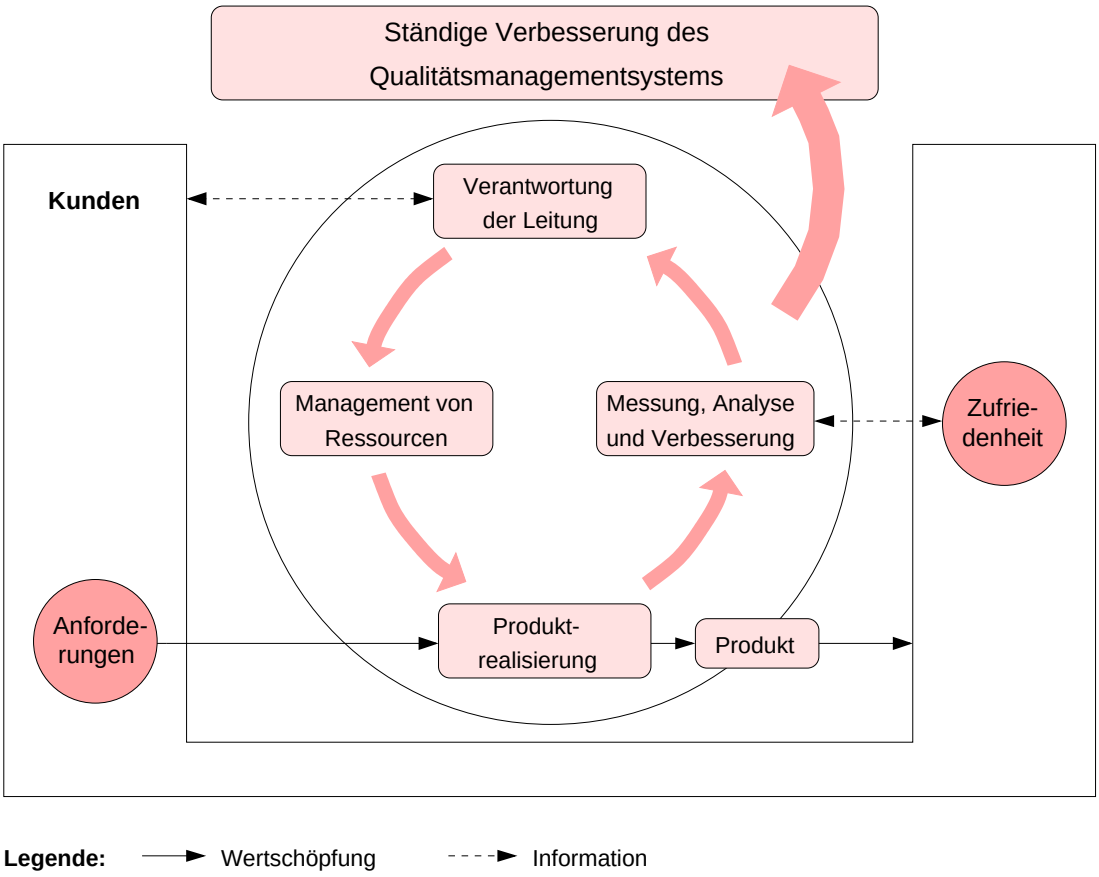


Abbildung 4.2: Modell eines prozessorientierten QMS (frei nach [Deu00b])

Die ISO 9001 besteht aus acht Abschnitten, von denen die ersten drei allgemeine Angaben über den Anwendungsbereich der Norm, normative Verweise sowie Begriffe enthalten. Die Abschnitte 4 bis 8 beschreiben die eigentlichen Anforderungen an ein QMS und seine Komponenten (vgl. Abb. 4.2). Die interne Struktur dieser Abschnitte zeigt Abb. 4.3.

QM im Sinne der ISO 9000-Normenreihe umfasst nach [Deu00b] die Funktionen Qualitätsplanung, Qualitätslenkung, Qualitätssicherung und Qualitätsverbesserung (siehe Abb. 4.4). In den folgenden Abschnitten werden diese Funktionen im Kontext eines QM von Daten beschrieben. Dies ist sinnvoll, da sich die Anforderungen der ISO 9001 mit Hilfe der in Abschnitt 3.1 gezogenen Analogie größtenteils auf das DQM übertragen lassen. In Anlehnung an [Deu00a] werden die Begriffe „Datenqualitätsmanagement“ und „Datenqualitätsmanagementsystem“ wie folgt definiert:

Definition 4.1 (Datenqualitätsmanagement, DQM): Aufeinander abgestimmte Tätigkeiten zum Leiten und Lenken einer Organisation bzgl. Datenqualität. □

Definition 4.2 (Datenqualitätsmanagementsystem, DQMS): System zur Durchführung des DQM. □

Ein DQMS sollte – soweit möglich – softwaregestützt sein (siehe dazu Kapitel 7 ff.).

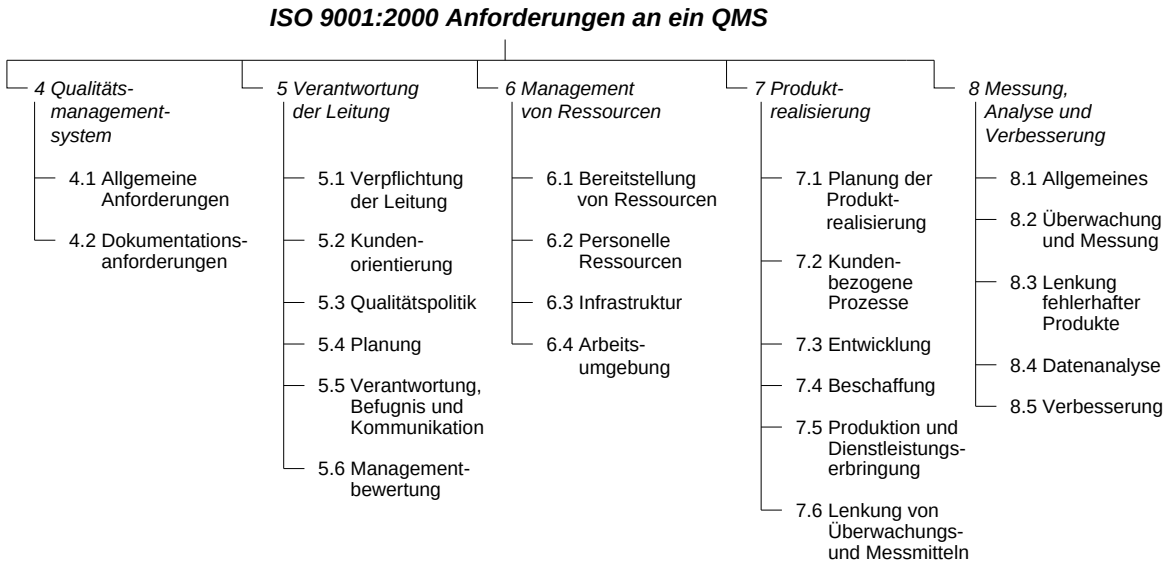


Abbildung 4.3: Interne Struktur der ISO 9001:2000-Abschnitte 4 bis 8 (ohne dritte Ebene)

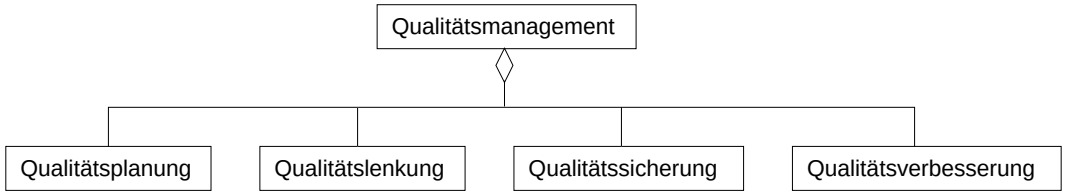


Abbildung 4.4: Funktionen des QM

4.3 Adaption von QM-Konzepten an die Datenverarbeitung

In der Literatur existieren bereits einige Ansätze zur Adaption von QM-Konzepten an die Datenverarbeitung. So nimmt [Wan98] unter der Bezeichnung *Total Data Quality Management (TDQM)* eine Anpassung von TQM vor, die den PDCA-Zyklus in den Datenverarbeitungsprozess integriert und die (Daten-)Qualitätsüberzeugung in der Organisationskultur verankert. Die TDQM-Konzepte wurden in [Eng99] um eine Vorgehensweise zur kontinuierlichen Datenqualitätsverbesserung erweitert. Diese wird als Regelkreis beschrieben, der neben einer Qualitätsmessung (auf Schema-/System- und Datenebene) auch Kostenbewertungen umfasst und nicht nur auf eine (symptomorientierte) Datenbereinigung setzt, sondern auch eine (ursachenorientierte) Prozessoptimierung einschließt (siehe Abb. 4.5). [Hel00] schließlich ergänzt die bei [Eng99] im Mittelpunkt stehende Ablauforganisation um eine Aufbauorganisation, einzuhaltende Standards und Normen sowie festgelegte Methoden, Maßnahmen, Werkzeuge und Hilfsmittel zu einem sog. *methodenbasierten DQM* (siehe Abb. 4.6). Ansatzpunkte zur kontinuierlichen Qualitätsverbesserung werden über ein Modell identifiziert, welches ausgehend von Erscheinungsformen von Datenqualitätsproblemen deren Ursachen und Auswirkungen analysiert und daraus Maßnahmen ableitet.

4.4 Datenqualitätsplanung

Die Qualitätsplanung als Funktion des QM ist auf das „Festlegen der Qualitätsziele und der notwendigen Ausführungsprozesse sowie der zugehörigen Ressourcen zur Erfüllung der Qualitätsziele“

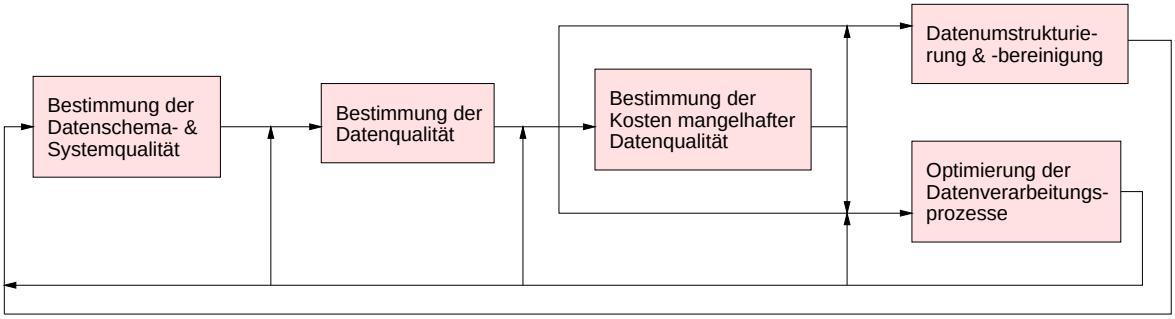


Abbildung 4.5: Total Quality data Management nach [Eng99]

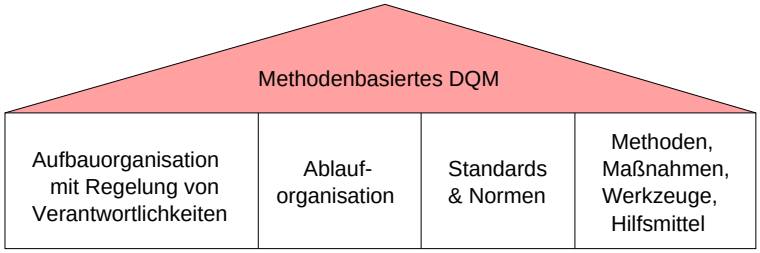


Abbildung 4.6: Methodenbasiertes DQM nach [Hel00]

gerichtet [Deu00a] und resultiert in einem QM-Plan. Analog dazu umfasse ein *DQM-Plan* die an ein Datenprodukt gestellten Anforderungen, die zu seiner Realisierung notwendigen Prozesse und Ressourcen, die erforderlichen Qualitätsmessmittel und -methoden sowie die erforderlichen Aufzeichnungen zum Nachweis der Konformität des Datenprodukts.

Das Festlegen von Qualitätszielen im Rahmen des DQM sollte sich i. Allg. an den für Datenprodukte charakteristischen Qualitätsmerkmalen (siehe Abschnitt 3.2) orientieren. Zur Formulierung von Qualitätszielen müssen alle an einem Datenprodukt interessierten Parteien (also insbesondere Kunden, d. h. Datennutzer/-auswerter, ggf. klassifiziert nach Nutzergruppen bzw. -rollen) nach ihren jeweiligen Anforderungen an den Erfüllungsgrad von Datenqualitätsmerkmalen befragt werden. Dies gestaltet sich in der Praxis oft schwierig (siehe dazu Abschnitt 4.4.1). Anforderungen unterschiedlicher Parteien an dasselbe Datenprodukt (oder zwei Datenprodukte mit überlappenden Extensionen) können konkurrieren.

Beispiel 4.1: Partei A stelle hohe Anforderungen an die Zeitnähe von Marketing-Daten, Partei B hingegen lege mehr Wert auf deren Konsistenz. Da die Beseitigung von Inkonsistenzen zeitaufwändig ist, liegt ein Konflikt zwischen Anforderungen vor, der aufgelöst werden muss. □

Interessenskonflikte treten auch dann auf, wenn zwei oder mehr Anforderungen unterschiedlicher interessierter Parteien ein und dasselbe Datenqualitätsmerkmal betreffen, jedoch verschiedene Erfüllungsgrade als Annahmekriterium genannt werden. Diese Einzelanforderungen können zu einer gemeinsamen Anforderung zusammengefasst werden, die der strengsten Einzelanforderung entspricht.

Die Spezifizierung von Anforderungen kann nominal (z. B. mit Abstufungen „sehr wichtig“, „wichtig“, „unwichtig“) oder numerisch (Angabe eines Zahlenwerts aus einem vorgegebenen Intervall) erfolgen. Anschließend ist ggf. eine Überführung der verwendeten Repräsentation in eine objektiv messbare Größe erforderlich – das Vorhandensein einer angemessenen Metrik vorausgesetzt.

4.4.1 Exkurs: Quality Function Deployment (QFD)

Die Bestimmung von Anforderungen an Produkte bzw. Dienstleistungen stellt eine der schwierigsten Aufgaben in einer Organisation dar, bedingt vor allem durch zwei Ursachen: Zum einen können Kunden oft nicht genau artikulieren, was sie wollen bzw. zur Durchführung ihrer Aufgaben benötigen (siehe untenstehende Beispielaussagen). Dieser Umstand wird noch erschwert durch die hohe Dynamik moderner IT-gestützter Umgebungen (technischer Fortschritt, Markteinflüsse etc.). Zum anderen sind zur Bestimmung von Anforderungen bestimmte Fähigkeiten erforderlich, insbesondere zur Überbrückung der Kluft zwischen dem (anwendungsorientierten) Problemverständnis des Kunden und der (eher technischen) Sicht der Organisation.

Beispiele für subjektive Kundenanforderungen im Datenqualitätskontext sind Aussagen wie:

- „Daten sollen richtig sein.“
- „Daten sollen möglichst schnell verfügbar sein.“
- „Diese Daten sollen zu jenen Daten passen.“
- „Daten sollen vor unbefugtem Zugriff geschützt sein.“
- „Daten sollen in ansprechender Weise dargestellt werden.“

Quality Function Deployment (QFD) [Aka90] stellt ein Hilfsmittel für die genannte Problematik dar. Es handelt sich dabei um ein in der Analyse- und Designphase von Produkten (bzw. Dienstleistungen) eingesetztes Verfahren, das subjektive Kundenanforderungen in objektive technische Spezifikationen umwandelt. QFD beruht auf einem Verfeinerungsprozess einer Reihe von Matrizen, deren Elemente Beziehungen zwischen Produkteigenschaften, Kundenanforderungen und den Prozessen zur Herstellung des Produktes beschreiben.

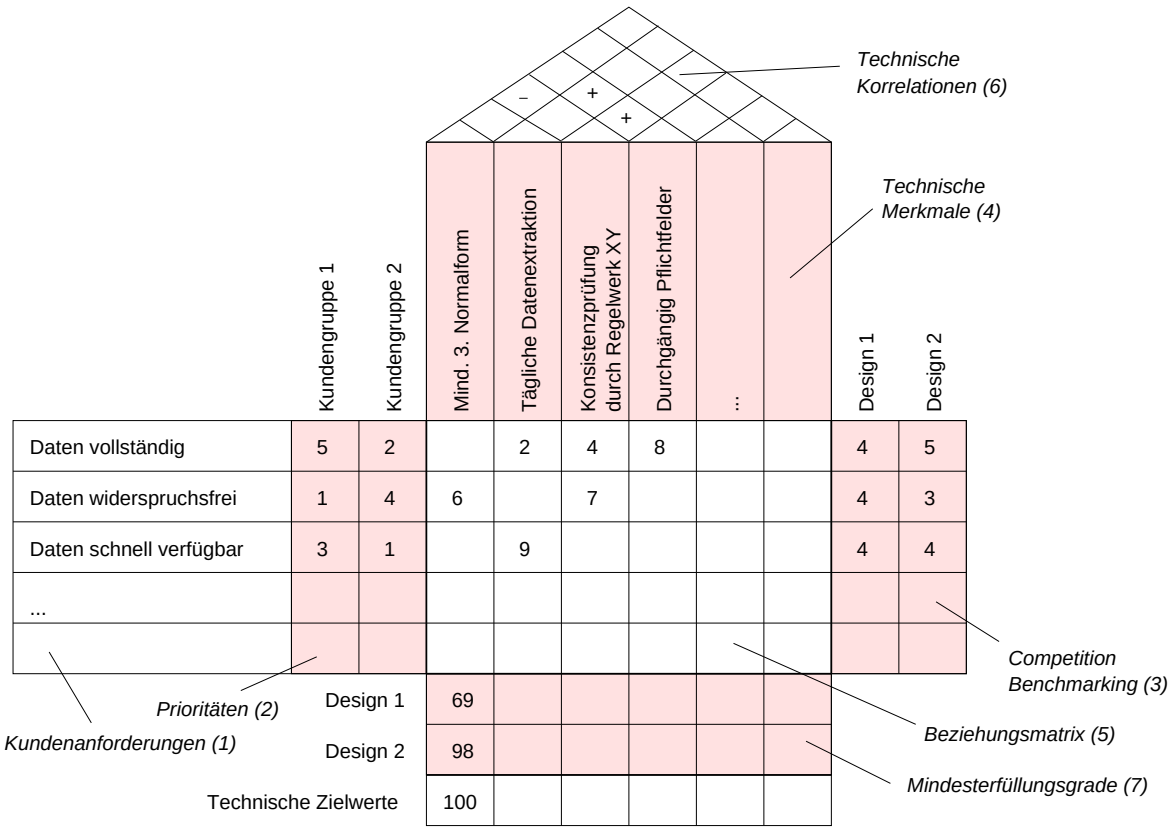


Abbildung 4.7: House of Quality im DQM-Kontext

Das bekannteste Element des QFD ist die sog. *House of Quality-Matrix.* Es existieren eine Reihe von Varianten dieser Matrix, die jeweils auf bestimmte Problem- oder Nutzertypen zugeschnitten sind. Die im Folgenden beschriebene und in Abb. 4.7 dargestellte Variante nach [HC88] besteht aus sieben Komponenten, die im Laufe eines QFD-Projektes „gefüllt“ werden:

1. *Kundenanforderungen:* Aus Kundenaussagen wird eine Liste von Kundenanforderungen abgeleitet und in die „House of Quality“-Matrix eingetragen. Im Datenqualitätskontext sind dies – vereinfacht dargestellt – z. B. Anforderungen wie „Daten müssen vollständig sein.“ oder „Daten müssen schnell verfügbar sein.“

2. *Prioritäten von Anforderungen:* Die Kundenanforderungen werden hier auf ihre relative Wichtigkeit für verschiedene Kundengruppen untersucht und dementsprechend gewichtet.

3. *Competition Benchmarking:* Auf der Basis von Marktforschungsanalysen wird ermittelt, wie zufrieden Kunden mit Konkurrenzprodukten sind (falls vorhanden) und wo Verbesserungspotential besteht.

4. *Technische Merkmale:* Aus den Kundenanforderungen werden technische Parameter abgeleitet, im Datenqualitätkontext z. B. Angaben zur Schemamodellierung, zur Extraktionsstrategie oder zu Konsistenzprüfungen.

5. *Beziehungsmatrix:* Zwischen Kundenanforderungen und technischen Parametern werden Beziehungen hergestellt und jeweils Beeinflussungsgrade quantifiziert. So existiert z. B. ein enger Zusammenhang zwischen der Anforderung einer schnellen Datenverfügbarkeit und der gewählten Strategie zur Datenextraktion.

6. *Technische Korrelationen:* Die (synergetischen oder konkurrierenden) Abhängigkeiten zwischen den einzelnen technischen Parametern werden analysiert und in der Matrix festgehalten.

7. *Mindesterfüllungsgrade:* Schließlich werden technische Zielwerte festgelegt und mit Angaben über den Schwierigkeitsgrad ihrer Umsetzung versehen.

Das Ergebnis der Matrixbearbeitung ist damit eine Menge von im Produktentwurf zu berücksichtigenden Zielwerten für jeden technischen Parameter. Die Vorteile eines Einsatzes von QFD liegen laut [Aka90] in einer kürzeren Time-to-Market, geringeren Entwicklungskosten, einer verbesserten Produktqualität und damit letztlich einer höheren Kundenzufriedenheit. In [Red96] findet sich ein durchgängiges Beispiel zur Anwendung von QFD im Datenqualitätskontext.

4.4.2 Planung von Ausführungsprozessen

Im DQM können hinsichtlich der Festlegung von Ausführungsprozessen im Sinne von Produktrealisierungsprozessen, Überwachungs- und Prüfmaßnahmen sowie von zugehörigen Ressourcen im Rahmen der Qualitätsplanung zwei Fälle unterschieden werden:

1. *Der Auswahl von Prozessen und Ressourcen zur Produkterstellung* gemäß [Deu00b] entspricht im DQM die Auswahl geeigneter (in Software umgesetzter) Methoden zur Verarbeitung von Daten, aber auch die Auswahl geeigneter Daten bzw. Datenquellen sowie der notwendigen Hardware zur Verarbeitung und persistenten Speicherung von Daten.

2. *Der Auswahl von Prozessen und Ressourcen zur Qualitätslenkung* gemäß [Deu00b] entspricht im DQM die Auswahl geeigneter (in Software umgesetzter) Methoden zur Messung und Bewertung von Datenqualität sowie der Auswahl dafür zuständiger (ggf. personeller) Ressourcen und die Festlegung von Messpunkten im Verarbeitungsprozess.

Die Festlegung von Punkten im Datenverarbeitungsprozess, an denen Qualitätsmessungen durchzuführen sind, muss sorgfältig geplant werden. Insbesondere kommen dazu Datensenken (hier: Datenhaltungskomponenten, in die Ergebnisse von Verarbeitungsprozessen geschrieben werden) in Frage, die kritischen (schreibenden) Verarbeitungsschritten nachgeschaltet sind (siehe auch Abschnitt 4.5.3).

4.5 Datenqualitätslenkung und -verbesserung

Grundvoraussetzung für die Etablierung einer Qualitätslenkung ist die Fähigkeit, Qualitätsmerkmale und damit den Erfüllungsgrad von Kundenanforderungen messen zu können („You cannot control what you cannot measure.“ [DeM82]). Ein ideales Qualitätsmaß sollte nach [Jur93] folgende Eigenschaften aufweisen:

- eine anerkannte Basis für Entscheidungen bieten
- allgemeinverständlich sein
- einheitlich interpretierbar sein
- allgemein anwendbar sein
- wirtschaftlich anwendbar sein
- mit den vorhandenen Sensoren (Messgeräte oder -prozesse) vereinbar sein.

Während es bei (materiellen) Produkten i. Allg. etablierte Metriken und Messverfahren für die verschiedenen Qualitätsmerkmale gibt (Länge, Gewicht, Leistung etc.), so ist dies bei (immateriellen) Datenprodukten bisher nicht der Fall. Wie in Kapitel 3 beschrieben, besteht hier nicht einmal Einigung über zu betrachtende Qualitätsmerkmale. In der Literatur finden sich kaum Vorschläge für Datenqualitätsmetriken. Es gibt zwar Ausnahmen wie z. B. [Kao94], diese haben aber weitgehend trivialen Charakter.

Das Fehlen von angemessenen Metriken hat zur Folge, dass die in der Literatur beschriebenen Verfahren zur Datenqualitätsmessung zwangsläufig eher informalen Charakter haben. So schlägt z. B. [HLW98] ein dreiteiliges Messsystem vor, das aus einem Fragebogen zur Einschätzung der subjektiven Wahrnehmung der Datenqualität durch Datennutzer, einem System zur objektiven, anwendungsunabhängigen Messung an Hand quantifizierbarer Variablen und einem System zur anwendungsabhängigen Messung unter Hinzunahme von Domänenwissen besteht. [JJQV99] nimmt eine Adaption des aus dem Software Engineering stammenden Goal-Question-Metric–Ansatzes (GQM) [BW84] vor. Die Idee besteht dabei darin, zunächst Qualitätsziele (Goals, vgl. Qualitätsplanung) zu spezifizieren und diesen (Ab-)Fragen (Questions) zuzuordnen, deren Ergebnisse (Antworten) dann – angewandt auf eine Metrik – eine Einschätzung der Datenqualität ermöglichen sollen. Qualitätsziele und Metriken werden in diesem Ansatz zwar formal repräsentiert (verwaltet in einer Metadatenbank), allerdings stellt die Trivialität der vorgeschlagenen Metriken auch hier die Praxistauglichkeit des Ansatzes in Frage. Diese konnte bisher nur für einfache Beispiele nachgewiesen werden.

Laut [Hel00] ist es nicht sinnvoll, ein (allgemeingültiges) System zur Datenqualitätsmessung zu definieren, da eine Einschätzung von Datenqualität stets anwendungsabhängig sein müsse. Alternativ schlägt der Autor eine Methodik zur Erstellung anwendungsabhängiger Messsysteme vor (siehe auch Abschnitt 4.5.2). Diese (weit verbreitete) Sichtweise ist jedoch falsch: anwendungsabhängig (und z. T. auch subjektiv) sind „nur“ die Qualitäts*anforderungen*, die Metriken zur Qualitätsmessung können hingegen durchaus anwendungsunabhängig und objektiv sein. Dies trifft auf die industrielle Fertigung genauso zu wie auf den Kontext der Datenverarbeitung: Die Messung von Längen kann objektiv vorgenommen werden (z. B. mittels Zentimetermaß), die Anforderungen an die Varianz von Längen (Fehlertoleranz) sind hingegen subjektiv und können sich je nach Kunde unterscheiden. Analog in der Datenverarbeitung: Vollständigkeit kann objektiv gemessen werden (entweder liegt ein Wert vor oder

nicht), während Anforderungen an die Vollständigkeit subjektiv sind und sich in unterschiedlichen, von den Datennutzern spezifizierten Sollwerten widerspiegeln können.

In Kapitel 6 dieser Arbeit werden objektive Metriken für ausgewählte Datenqualitätsmerkmale definiert. Für diese werden Messverfahren entwickelt, mit denen für ein gegebenes Datenprodukt der Erfüllungsgrad von Anforderungen berechnet werden kann. Je nach Messergebnis können dann unterschiedliche Maßnahmen ergriffen werden, z. B. Freigabe, Aussonderung oder Neueinstufung des Datenprodukts.

4.5.1 Data Profiling vs. Data Mining

[RD00] unterscheidet zwei Ansätze zur Einschätzung von Datenqualität: Data Profiling und Data Mining. Beide Ansätze werden im Folgenden kurz skizziert:

Data Profiling untersucht die Werte einzelner Attribute in einem Datenbestand und liefert Angaben wie:

- Genutzte Wertebereiche
- Mittelwert und Varianz numerischer Werte
- Häufigkeit diskreter Werte
- Häufigkeit von Nullwerten.

Die Idee des Einsatzes von *Data Mining* (vgl. Kapitel 1) zu Zwecken des DQM besteht darin, zunächst automatisiert nach Regelmäßigkeiten im Datenbestand zu suchen und von diesen Regelmäßigkeiten abweichende Datenprodukte als „verdächtig“ einzustufen. Die verdächtigen Datensätze können dann – falls sich z. B. eine Inkonsistenz bestätigt – ggf. an Hand der gefundenen Regelmäßigkeiten korrigiert bzw. – bei fehlenden Werten – vervollständigt werden [DJK00, HW00, HGG01]. Im Gegensatz zum Data Profiling kann das wesentlich aufwändigere Data Mining auch komplexere, attributübergreifende Muster in Datenbeständen erkennen (funktionale Abhängigkeiten etc.). Dies kann auch als Reengineering von Geschäftsregeln aufgefasst werden.

Beispiel 4.2: Eine Assoziationsregel *Gesamtpreis* = *Einzelpreis* $*$ *Anzahl* mit einer Konfidenz von 99% weist darauf hin, dass 1% der betrachteten Datensätze dieser Regel nicht genügen und damit potentiell, aber nicht zwangsläufig (z. B. Mengenrabatte) inkonsistent sind. □

Mit Hilfe einer solchen Regel ist eine automatisierte Korrektur von inkonsistenten Datensätzen prinzipiell möglich. Es muss dazu allerdings aus der konkreten Situation heraus bekannt sein, in welchem der beteiligten Attribute der Fehler aufgetreten ist. Ansonsten besteht die Gefahr der Induktion sog. *Typ I-Fehler* (Korrektur vormals korrekter Werte, d. h. deren Verfälschung) in den Datenbestand. Aufgrund dieser Schwierigkeiten sollten automatisiert vorgenommene Korrekturen grundsätzlich so ausführlich dokumentiert werden, dass bei Bedarf eine Umkehroperation durchgeführt werden kann.

4.5.2 Analyse gemessener Datenqualität

Die Datenqualitätsanalyse soll Aufschluss über den Qualitätszustand von Datenprodukten geben. Grundlage der Analyse sind Messergebnisse, die zu den einzelnen Qualitätsmerkmalen von betrachteten Datenprodukten berechnet wurden (Istwerte), sowie die Qualitätsanforderungen aus der Qualitätsplanung (Sollwerte). Werden Messergebnisse über die Zeit fortgeschrieben, können Trends in der Qualitätsentwicklung (ggf. quellenspezifisch) verfolgt werden. Weiterhin können Metainformationen über Datenquellen (Erfassungsmethode etc.) sowie über verwendete Messverfahren und Messpunkte zur Bewertung hinzugezogen werden.

Durch Vergleich von Ist- und Sollwerten lassen sich Aussagen darüber treffen, ob die spezifizierten Anforderungen erfüllt oder ggf. weitere Maßnahmen zu treffen sind. Diese Maßnahmen werden in Abschnitt 4.5.3 im Detail beschrieben. Die Aufzeichnung von vorgenommenen Maßnahmen über die Zeit ermöglicht deren Rückverfolgbarkeit bis hin zu den Datenquellen und erlaubt somit mittelfristig – zusammen mit neuen Messergebnissen – eine Bewertung der Wirksamkeit von Maßnahmen. Ziel muss es sein, eine ständige Verbesserung des DQM und der beteiligten produktrealisierenden Prozesse zu erreichen.

In der Literatur finden sich kaum Ausführungen zur Analyse von Datenqualität. Eine Ausnahme bilden [Hel00, HM01], wo ein sog. *Analyserahmen* verwendet wird, um Zusammenhänge zwischen Erscheinungsformen mangelhafter Datenqualität (Symptome), ihren Ursachen und ihren Auswirkungen herzustellen. Dabei werden Erscheinungsformen mittels Messverfahren bestimmt. Die Ermittlung von Ursachen dient zur Erkennung von Lösungsansätzen im Hinblick auf eine ursachenorientierte Qualitätsverbesserung. Eine Abschätzung von Auswirkungen kann als Basis für Wirtschaftlichkeitsuntersuchungen herangezogen werden, z. B. Kosten-/Nutzenanalysen zur Beantwortung der Frage, ob sich die Durchführung einer bestimmten (selbst mit Kosten verbundenen) Maßnahme, die ein gegebenes Qualitätsdefizit beheben soll, rentieren würde.

4.5.3 Reaktionen auf Datenqualitätsmängel und Datenqualitätsverbesserung

Im Rahmen eines qualitätsgelenkten Datenverarbeitungsprozesses kann – in Abhängigkeit von den Ergebnissen einer zuvor durchgeführten Datenqualitätsanalyse – in unterschiedlicher Weise auf entdeckte Datenqualitätsmängel reagiert werden. Im Einzelnen können die in der folgenden Aufzählung (unter Verwendung der in Abschnitt 3.3.1 vorgestellten phasenorientierten Klassifikation von Mängelursachen) aufgeführten Kategorien von Reaktionen unterschieden werden, die jeweils auf verschiedene Klassen von Mängelursachen anwendbar sein können:

- *Aussortieren und Neuanfordern*
  Werden Daten als nicht verwendbar (für einen hinreichend großen Anteil der geplanten Auswertungen) eingestuft, so müssen sie als Ausschuss deklariert und gelöscht werden. Metadaten, die die Ursachen der mangelhaften Qualität der betroffenen Datenprodukte analysieren helfen, sollten hingegen erhalten bleiben. Dazu gehören Angaben zur Datenquelle, Messergebnisse sowie Angaben zu den bis zum Zeitpunkt der Aussortierung durchgeführten Maßnahmen. Nach Beseitigung der Fehlerursachen (im operativen System bzw. im DWS) ist eine Neuanforderung der gelöschten Daten erforderlich, soweit sie nicht mehr in Zwischenspeichern, die sich „vor“ der Fehlerquelle befinden, vorliegen.

- *Einschränken der Nutzung*
  Existieren unterschiedliche Anforderungen an ein Datenprodukt, so kann es vorkommen, dass die gemessene Qualität des Datenprodukts nur einige dieser Anforderungen hinreichend erfüllt. Das Datenprodukt kann dann gezielt für bestimmte Auswertungen freigegeben und für andere gesperrt werden. Solche Attributierungen von Datenprodukten sind in den Metadaten abzulegen und bei Zugriffen auf das Datenprodukt auszuwerten.

- *Verbessern der Datenqualität (symptomorientiert)*
  Eine Verbesserung der Qualität von Daten kann in verschiedenen Phasen des Datenverarbeitungsprozesses vorgenommen werden. Es gilt: „Je früher Qualitätsmängel erkannt und behoben werden, desto besser.“ Schon in der Erfassungsphase können durch den Erfasser selbst oder eine nachgeschaltete Kontrollinstanz – ggf. mit Softwareunterstützung – einfache Fehler (Tippfehler, fehlende Werte etc.) korrigiert werden. Andere (oft komplexere) Datenqualitätsmängel können erst in späteren Phasen der Verarbeitung behandelt werden. Als Beispiel

sei die Beseitigung von unbeabsichtigter Redundanz aus einem aus heterogenen Quellen integrierten Datenbestand genannt. Diese kann erst erfolgen, wenn grundlegende Bereinigungen auf den Quelldaten (z. B. Vereinheitlichung der Repräsentation) abgeschlossen sind.

Die zu dieser Kategorie gehörenden Maßnahmen sind allesamt symptomorientiert. In der Literatur werden sie unter dem Begriff *Data Cleansing bzw. Data Cleaning* zusammengefasst. In Abschnitt 6.2 wird eine Klassifizierung von Data Cleansing-Verfahren nach Datenqualitätsmerkmalen vorgenommen.

- *Verbessern der (Daten-)Verarbeitungsprozesse (ursachenorientiert)*
  In dieser Kategorie lassen sich zwei Klassen von Reaktionen unterscheiden:

  1. *Prozessverbesserung*
     Optimierung des Prozessablaufs oder der Prozessparameter
  2. *Prozessredesign*
     Ersetzung des „alten“ Prozesses durch einen von Grund auf neu modellierten Prozess.

  Zur Prozessverbesserung sind dabei auch die Erweiterung eines DWS um geeignete Datenquellen sowie die Abkopplung von ungeeigneten Datenquellen an Hand von Kriterien wie inhaltlicher Relevanz, Reputation, Verfügbarkeit, Aktualität, Zugangskosten etc. zu zählen. Die Eignung von Datenquellen ist regelmäßig zu prüfen. Weiterhin gehören zur Prozessverbesserung: Schulung und Motivation von Datenerfassern, -verarbeitern und -nutzern, Optimierung und Debugging von Software etc.

  Ein komplettes Redesign von Prozessen ist zwar oft wünschenswert, um fundamentale Probleme anzugehen, muss aber in der Praxis in vielen Fällen einer Prozessverbesserung weichen, um (kurz- bis mittelfristig) Kosten zu sparen und etablierte Abläufe in der Organisation nicht zu beeinträchtigen. In der langfristigen strategischen Planung einer Organisation sollte ein Prozessredesign aber auf jeden Fall eine Rolle spielen, da es i. Allg. kostengünstiger ist als eine entsprechend hohe Zahl von Prozessverbesserungsschritten. Wichtig ist, dass die Neuordnung eines Prozesses von den betroffenenen Mitarbeitern auch getragen wird (soziale Komponente).

  Prozessverbesserung und -redesign sind den ursachenorientierten Reaktionen zuzurechnen, da sie auf die Vermeidung von Datenqualitätsmängeln in zukünftigen Datenverarbeitungsprozessen ausgerichtet sind. Ursachenorientierte Maßnahmen sind kurz- und mittelfristig kostenintensiver als symptomorientierte Maßnahmen, rentieren sich aber langfristig durch die Senkung von Data Cleansing-Kosten.

- *Verbessern der Prüfprozesse (ursachenorientiert)*
  Für diese Kategorie gelten die Aussagen zu voriger Kategorie, nur dass hier nicht die Datenverarbeitungsprozesse verbessert bzw. neu modelliert werden, sondern die Prozesse zur Qualitätsprüfung, insbesondere zur Qualitätsmessung. Ziel muss es sein, eine zunehmend exaktere Einschätzung der tatsächlichen Datenqualität zu erreichen (d. h. eine, die der intuitiven Einschätzung der Datennutzer am nächsten kommt) und möglichst alle Qualitätsdefizite möglichst früh aufzudecken.

- *Rücksetzen und Wiederholen*
  Treten Fehler in Verarbeitungsprozessen (z. B. Netzwerkfehler bei der Übermittlung von Quelldaten an ein DWS) auf, so können diese Prozesse neu aufgesetzt werden. Eine möglichst feingranulare Spezifikation von Aufsetzpunkten und eine Verwendung von Transaktionskonzepten stellen sicher, dass die Ausführung nur ab demjenigen Teilprozess wiederholt werden muss, in dem der Fehler aufgetreten ist.

- *Feedback an Datenquellen*
  In Informationssystemen, die Daten aus externen Quellen beziehen, ist der Einfluss auf die Prozesse innerhalb der angeschlossenen Quellsysteme oft begrenzt. Dennoch sollten Quellsysteme

(bzw. deren Sachbearbeiter oder Administratoren – je nach Problemtyp) bei Bedarf möglichst detailliert auf entdeckte Datenqualitätsmängel hingewiesen werden, damit deren Ursachen an ihren Entstehungsorten behoben werden können, insbesondere durch Verbesserung der dortigen (Daten-)Verarbeitungsprozesse.

Alle durchgeführten Maßnahmen sollten in geeigneter Weise aufgezeichnet werden, um eine Rückverfolgbarkeit zu gewährleisten. Einzelne Reaktionen können durchaus in verschiedenen Phasen wiederholt angewandt oder kombiniert werden, bis keine (signifikante) Verbesserung der Daten- bzw. Prozessqualität mehr erreicht wird oder andere der aufgeführten Maßnahmen ergriffen werden müssen. Ein geeignetes Abbruchkriterium ist vorzusehen. Tabelle 4.1 fasst die Kategorien von Reaktionen und die mit ihnen assoziierten phasenorientierten Mängelursachen (vgl. Abschnitt 3.3.1) noch einmal zusammen.

Phase / **Reaktion**	**Entwicklung**	**Datenerfassung**	**Datenübertragung**	**Datenintegration**	**Datenhaltung/-analyse**
Aussortieren und Neuanfordern	✓	✓			
Einschränken der Nutzung		✓			
Verbessern der Datenqualität		✓			
Verbessern der (Daten-)Verarbeitungsprozesse	✓	✓	✓	✓	✓
Verbessern der Prüfprozesse	✓	✓		✓	
Rücksetzen und Wiederholen			✓	✓	✓
Feedback an Datenquellen	✓	✓			

Tabelle 4.1: Reaktionen auf Datenqualitätsmängel in Abhängigkeit ihrer Fehlerklasse

4.5.4 Kosten von Maßnahmen zur Datenqualitätsverbesserung

Kosten von Maßnahmen zur Datenqualitätsverbesserung können über Abschätzungen hinsichtlich des Aufwands der jeweiligen Maßnahme berechnet werden. Folgende (groben) Kostenarten können im DQM unterschieden werden:

- *Kosten der ursachenorientierten Datenqualitätsverbesserung*
  Kosten eines Reengineering von Prozessen (Anforderungsanalyse, Entwurf, Implementierung) inkl. Kosten personeller und technischer Ressourcen sowie ggf. Kosten der Beeinträchtigung des Systembetriebs (zeitliche Ressourcen).

- *Kosten der symptomorientierten Datenqualitätsverbesserung*
  Kosten der Ausführung von Verbesserungsmaßnahmen (Laufzeit, personelle und technische Ressourcen), inkl. Mehraufwände für parallele Datenbereinigung im Rahmen von Datenanalysen.

- *Kosten nicht entdeckter bzw. nicht behobener Datenqualitätsmängel*
  Kosten von Auswirkungen mangelhafter Datenqualität (siehe auch Kapitel 1), z. T. schwierig zu quantifizieren (z. B. Frustration der Datennutzer, Folgekosten falscher Entscheidungen).

Da sowohl mangelhafte Datenqualität (bzw. deren Folgen) als auch Maßnahmen zur Datenqualitätsverbesserung Kosten verursachen, stellt sich die Frage nach dem „Break Even Point“, insbesondere:

- Wieviel darf eine Verbesserungsmaßnahme maximal kosten, um noch rentabel zu sein?
- Welchen Nutzen muss eine Verbesserungsmaßnahme mindestens bringen, um ihre Durchführung zu rechtfertigen?

Während die Kosten von Daten als Summe von Beschaffungs-, Aufbereitungs- und Nutzungskosten noch relativ einfach beziffert werden können, gestaltet sich die Quantifizierung des Nutzens von Daten i. Allg. schwierig, da die Auswirkungen von auf Daten basierenden Entscheidungsprozessen oft nicht monetär ausgedrückt werden können. Und selbst wenn dies möglich ist (z. B. bei Weitervermarktung der aufbereiteten Daten), bleibt die Bestimmung des isolierten Beitrags eines Datenausschnitts an einer Entscheidung problematisch. Frei nach [Mün95] gilt: „Die Beschaffung und Aufbereitung von Daten lohnt sich so lange, wie die Kosten den zu erwartenden Nutzen nicht übersteigen.“

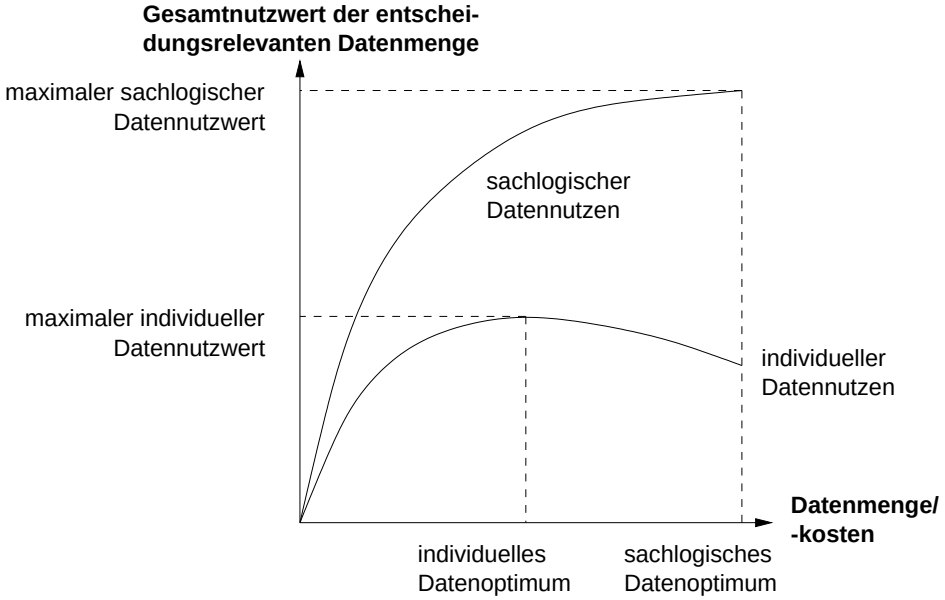


Abbildung 4.8: Gesamtnutzwert von Daten nach [Hau84]

Aus oben genannten Gründen ist das objektive Treffen solcher Aussagen schwierig. Die Bewertung des Datennutzens erfolgt daher in der Praxis oft subjektiv. Abbildung 4.8 zeigt den sog. *Gesamtnutzwert* von Daten nach [Hau84] in Abhängigkeit von einer (objektiven) Entscheidung und einem (subjektiv geprägten) Entscheidungsträger. Letzterer hat naturgemäß begrenzte kapazitive und kognitive Fähigkeiten zur Aufnahme und Verarbeitung von Daten [Pfe74], wodurch sich die voneinander abweichenden Optima sowie der nicht-monotone Verlauf der Kurve des individuellen Datennutzens ergeben. Der von [Hau84] verfolgte Ansatz weist einen retrospektiven Charakter auf. In der Praxis wären darüber hinaus prospektive Ansätze wünschenswert, die Aussagen darüber treffen, wann sich die Beschaffung zusätzlicher Daten lohnt [Gro76].

[Mün95] definiert in diesem Zusammenhang eine sog. *Bedürfnispyramide* (siehe Abb. 4.9). Demnach wird die Artikulierung des Informationsbedarfs durch einen Entscheidungsträger durch unterschiedliche Bedürfnisse beeinflusst. Im Allgemeinen stellt Wirtschaftlichkeit das primäre Bedürfnis dar, das

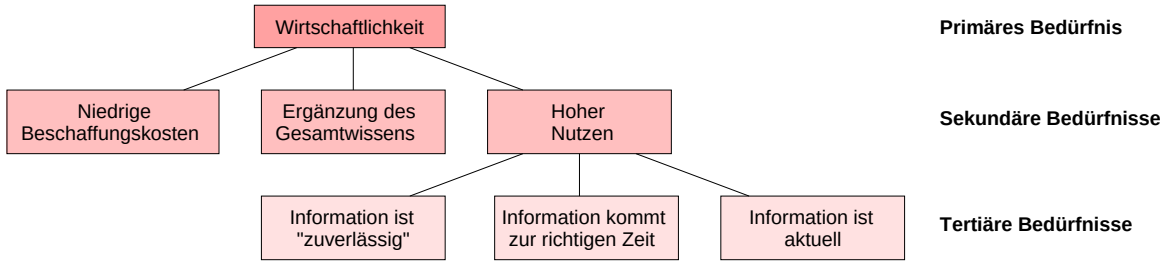


Abbildung 4.9: Bedürfnispyramide nach [Mün95]

sich in sekundäre Bedürfnisse wie „niedrige Beschaffungskosten“ und „hoher Nutzen“ von Daten zergliedern lässt. Hoher Datennutzen wiederum ergibt sich aus der Erfüllung von tertiären Bedürfnissen, die Forderungen an Datenqualitätsmerkmale entsprechen.

4.6 Datenqualitätssicherung

Datenqualitätssicherung umfasst in erster Linie organisatorische Maßnahmen in einer Organisation, die dokumentiert und dann – z. B. über eine Zertifizierung – nach außen getragen werden, um bei Kunden, Lieferanten etc. Vertrauen in die Erfüllung von Datenqualitätsanforderungen zu schaffen (vgl. [Deu00a]). Zu solchen Maßnahmen gehört z. B. die Festlegung und Dokumentation von Produktionsprozessen (inkl. Arbeitsanweisungen etc.), Prüfprozessen und standardisierten Kommunikationsstrukturen. Ferner sollten z. B. die Durchführung von Qualifizierungsmaßnahmen für das Personal, verwendete technische Ressourcen sowie ermittelte Datenqualitätstrends regelmäßig nach außen dokumentiert werden.

4.7 Zusammenfassung

Moderne QM-Konzepte, beispielsweise die der Normenfamilie ISO 9000:2000 ff., beziehen sämtliche Bereiche einer Organisation ein und fordern eine ständige Qualitätsverbesserung im Sinne eines Regelkreises. Im vorliegenden Kapitel wurde an Hand von Ansätzen aus der Literatur (z. B. TDQM) gezeigt, dass sich diese Konzepte grundsätzlich auf den Bereich der Datenverarbeitung anwenden lassen. Die einzelnen QM-Funktionen (Qualitätsplanung, Qualitätslenkung, Qualitätssicherung und Qualitätsverbesserung) wurden im betrachteten Kontext untersucht und mittels Beispielen illustriert. In Teil II der Arbeit werden diese DQM-Konzepte im Hinblick auf ihren Einsatz in DWS konkretisiert.

Kapitel 5

Verwandte Ansätze

In der wissenschaftlichen Literatur finden sich nur relativ wenige Arbeiten, die sich mit dem Thema DQM auseinandersetzen. Zwei vielversprechende Ansätze, die sich – wie sich später zeigen wird – gut mit dem in dieser Arbeit verfolgten Ansatz ergänzen, werden nun im Detail vorgestellt. Dabei handelt es sich zum einen um Arbeiten am französischen Institut INRIA (Abschnitt 5.1), zum anderen um einen Ansatz einer Forschergruppe der DaimlerChrysler AG (Abschnitt 5.2).

Auch im kommerziellen Bereich spielte DQM bis vor Kurzem praktisch keine Rolle. Die auf dem Markt angebotenen ETL-Werkzeuge (siehe Anhang A) bieten auch heute meist lediglich Funktionen zur syntaktischen Vereinheitlichung (Datenqualitätsmerkmal „Einheitlichkeit“), unterstützen aber weder eine Datenintegration auf semantischer Ebene (z. B. „intelligente“ Zusammenführung redundanter Daten) noch ein methodisch fundiertes DQM (Qualitätsplanung, -lenkung, -verbesserung und -sicherung). Eine Ausnahme stellt das in Abschnitt 5.3 näher beschriebene Werkzeug Ascential Quality Manager dar.

Weitere im Bereich des DQM angesiedelte Arbeiten werden in Abschnitt 5.4 kurz skizziert. In Abschnitt 5.5 wird abschließend eine bewertende Gegenüberstellung vorgenommen und der in der vorliegenden Arbeit verfolgte Lösungsansatz in das Szenario der bereits existierenden Ansätze eingeordnet.

5.1 AJAX: Deklaratives Data Cleaning (INRIA)

Im Rahmen dieses Projekts wurde ein Framework zum Data Cleaning entwickelt, welches in Analogie zum Datenbankentwurf die logische Sicht auf den Datenbereinigungsprozess von der physischen Sicht zu trennen versucht [GFS+01]. Auf der logischen Ebene wird ein Datenflussgraph spezifiziert, der Datentransformationen deklarativ beschreibt, ohne auf deren Implementierungsdetails einzugehen. Letztere werden, insbesondere im Hinblick auf Optimierungsmöglichkeiten, erst auf der physischen Ebene behandelt. Das Framework enthält folgende Komponenten:

- Eine deklarative, auf SQL99 [EM00] aufsetzende Sprache, basierend auf fünf logischen, komponierbaren Transformationsoperatoren.
- Einen Mechanismus zur Ausnahmebehandlung, der explizite Anwendereingriffe und eine schrittweise Verfeinerung des Datenbereinigungsprozesses erlaubt (sog. *Data Lineage*).
- Eine Notation zur Auswahl und Parametrierung von optimierten Record Linkage-Algorithmen (siehe auch Abschnitt 6.2.5).

Die Komponenten wurden im Datenbereinigungssystem AJAX [GFSS00] implementiert und an Hand von Experimenten evaluiert.

5.1.1 Logische Ebene: Die Spezifikationssprache

Jeder der fünf logischen Transformationsoperatoren erlaubt den Aufruf extern definierter Funktionen, um so domänenspezifische Prüfungen (z. B. Abstandsberechnung) oder Umformungen (z. B. Normalisierung von Zeichenketten) zu ermöglichen. Die Semantik jedes Operators beinhaltet die automatisierte Generierung einer Reihe von Ausnahmen (Exceptions), die Tupel kennzeichnen, die nicht automatisiert behandelt werden konnten. Die betroffenen Tupel werden zusammen mit einer textuellen Beschreibung der Ausnahme gesondert gespeichert und können interaktiv bearbeitet werden, ggf. unter Anpassung des Datenflussgraphen. Die fünf von AJAX unterstützten Operatoren werden im Folgenden beispielhaft vorgestellt.

Mapping-Operator

Der Mapping-Operator spezifiziert 1:*-Abbildungen von einer (logischen) Relation auf eine oder mehrere (logische) Relationen, d. h. jedes Tupel der Eingaberelation kann auf beliebig viele Tupel der Ausgaberelationen abgebildet werden.

Im folgenden Beispiel wird eine Namenszerlegung vorgenommen. Dazu wird ein Mapping mit dem Namen `NameExtraction` definiert, das aus den (von *NULL* abweichenden) Werten des Attributs `name` der Eingaberelation `InputData` mit Hilfe der extern definierten Funktionen `ExtractFirstName()` und `ExtractLastName()` die Vor- bzw. Nachnamen extrahiert und diese in die Attribute `first_name` bzw. `last_name` der Ausgaberelation `OutputData` einträgt. Dabei werden die extrahierten Namen in den (logischen) Relationen `FirstName` und `LastName` zwischengespeichert.

```
CREATE MAPPING NameExtraction
FROM InputData
LET FirstName = ExtractFirstName(InputData.name),
    LastName = ExtractLastName(InputData.name)
WHERE InputData.name IS NOT NULL
{SELECT FirstName.name AS first_name,
 LastName.name AS last_name INTO OutputData}
```

`SELECT ... INTO ...`-Klauseln können zusätzlich mit Constraints (*not null, unique, foreign key* und *check*) belegt werden, die allerdings eine von SQL abweichende Semantik aufweisen: Bei Verletzung eines Constraints wird das Mapping nicht abgebrochen, sondern eine Ausnahme generiert, die das betroffene Tupel in eine Hilfsrelation (im obigen Beispiel $\texttt{OutputData}^{Exc}$) schreibt.

View-Operator

Der View-Operator entspricht im Wesentlichen einer herkömmlichen SQL-Anfrage, auf deren Ergebnis zusätzliche Integritätsprüfungen durchgeführt werden können, die ihrerseits wiederum eine Ausnahmebehandlung anstoßen. Mit dem View-Operator lassen sich 1..*:1-Abbildungen formulieren.

Im folgenden Beispiel werden Angaben zur Diagnose und zum Wohnort von Patienten aus zwei Eingaberelationen extrahiert und in eine gemeinsame Ausgaberelation eingefügt, wobei die Zuordnung über den eindeutigen Schlüssel `patient_key` erfolgt. Eine Ausnahme wird generiert, falls die Wohnortangabe zu einem Patienten fehlt.

```
CREATE VIEW CombineDiagnosisAndCity
FROM MedicalHistories M, Addresses A
WHERE M.patient_key = A.patient_key
{SELECT M.patient_key AS patient_key,
 M.diagnosis AS diagnosis,
 A.city AS city INTO OutputData
 CONSTRAINT NOT NULL city}
```

Matching-Operator

Der Matching-Operator berechnet einen „approximativen“ Join zwischen zwei Relationen. Mit Hilfe dieses Operators kann abweichungstolerant nach Duplikaten (redundanten Tupeln) gesucht werden. Aus logischer Sicht wird für jedes Tupelpaar des kartesischen Produkts der beiden Eingaberelationen ein Abstandswert berechnet – auch wenn auf der physischen Ebene aus Effizienzgründen z. B. bestimmte Tupelpaare vorselektiert werden (siehe Abschnitt 5.1.2).

Die im nachfolgenden Beispiel spezifizierte Matching-Operation berechnet den Editierabstand aller Tupelpaare der Relation `Patients` mit Hilfe der extern definierten Funktion `EditDistance()`, filtert diejenigen Paare heraus, deren Abstand kleiner ist als 15% der Länge des längeren der beiden jeweils beteiligten Namen (Funktion `MaxDistance()`), und fügt diese in die Relation `MatchedPatients` ein.

```
CREATE MATCHING MatchPatients
FROM Patients P1, Patients P2
LET distance = EditDistance(P1.last_name, P2.last_name)
WHERE distance < MaxDistance(P1.last_name, P2.last_name, 15)
INTO MatchedPatients
```

Clustering-Operator

Der Clustering-Operator bildet Gruppen von Elementen, die durch eine Eingaberelation gegeben sind, und legt diese Gruppen in einer Ausgaberelation ab. Aus logischer Sicht ist die Ausgaberelation geschachtelt, wobei jede innere Relation einer Gruppe entspricht. Die Menge der zu gruppierenden Elemente kann auf zweierlei Weise in der Eingaberelation spezifiziert sein:

1. Jedes Tupel der Eingaberelation stellt ein Element dar. In diesem Fall entspricht das Clustering einem klassischen `GROUP BY` in SQL, d. h. alle Tupel, die in bestimmten Attributwerten übereinstimmen, werden derselben Gruppe zugeordnet.

2. Jedes Tupel der Eingaberelation definiert einen Abstand zwischen zwei Elementen, die jeweils durch einen Schlüssel identifizierbar sind. Eine solche Eingaberelation könnte insbesondere aus einer zuvor durchgeführten Matching-Operation hervorgegangen sein.

Im folgenden Beispiel wird der zweite Fall betrachtet. Als Schlüsselattribute dienen hier `patient_key1` und `patient_key2` aus der Eingaberelation. Die `BY METHOD`-Klausel legt das Verfahren zur Cluster-Bildung fest. In diesem Fall soll der transitive Abschluss aller Matches, an denen die beiden als Parameter übergebenen Schlüssel beteiligt sind, berechnet werden. Andere Clustering-Verfahren wie z. B. `nearest neighbor` erfordern ggf. weitere Parameter wie den maximalen Abstand vom Zentroiden („Mittelpunkt“) eines Clusters.

```
CREATE CLUSTERING ClusterPatients
FROM MatchedPatients
ON patient_key1, patient_key2
BY METHOD transitive closure
WITH PARAMETERS patient_key1, patient_key2
INTO ClusteredPatients
```

Die Umsetzung des (ohne die Generierung von Ausnahmen auskommenden) Clustering-Operators zeigt, dass die Trennung von logischer und physischer Ebene in AJAX nicht durchgängig eingehalten wurde bzw. werden konnte: Da letztendlich der Anwender für die Auswahl des Clustering-Verfahrens zuständig ist, mischen sich physische Aspekte in die logische Spezifikation (`BY METHOD...WITH PARAMETERS...`-Klausel).

Merging-Operator

Der Merging-Operator verdichtet jedes Cluster einer Eingaberelation (typischerweise erzeugt durch eine zuvor ausgeführte Clustering-Operation) nach gegebenen Kriterien zu einem Tupel und schreibt dieses in eine Ausgaberelation.

In untenstehendem Beispiel werden die in der Relation `ClusteredPatients` (vgl. voriges Beispiel) gespeicherten Cluster als Grundlage zum Merging verwendet. Die Variable `c` läuft dabei über die Ausprägungen des Attributs `cluster_id`. Der Ausdruck `Patients(c).name` beschreibt dann die Menge von Werten des Attributs `name` in denjenigen Tupeln der Relation `Patients`, die sich auf das Cluster `c` beziehen. Aus dieser Menge wird mittels einer externen Funktion der längste Name selektiert (Länge ist hier das Auswahlkriterium bei voneinander abweichenden Werten) und – zusammen mit einem neu generierten Schlüssel – in die Ausgaberelation eingetragen. Gibt es mehr als einen Namen mit maximaler Länge in einem Cluster, wird eine Ausnahmebehandlung angestoßen.

```
CREATE MERGING MergePatients
USING ClusteredPatients (cluster_id) c
LET name = GetLongestName(Patients(c).name)
    key = GenerateKey()
{SELECT key AS patient_key, name AS name INTO MergedPatients}
```

5.1.2 Physische Ebene: Implementierung und Optimierung

Bei der Umsetzung logischer Operationen auf die physische Ebene (konkret: in ein JAVA-Programm) können Optimierungsmaßnahmen ergriffen werden, z. B. die Auswahl eines effizienten Algorithmus aus einer Menge von alternativen Algorithmen und dessen angemessene Parametrierung.

Als Beispiel nennt [GFS+01] den Matching-Operator, zu dem unterschiedliche Implementierungen bzw. Optimierungen existieren, um nicht alle Tupelpaare des kartesischen Produkts zweier Eingaberelationen betrachten zu müssen. Es werden zwei Optimierungsansätze betrachtet (abstandsfilternd und approximativ), auf die an dieser Stelle jedoch nicht genauer eingegangen wird (siehe auch Abschnitt 6.2).

Allerdings gilt auch hier – wie schon beim Clustering – dass die Optimierungsmaßnahmen in AJAX nicht angemessen vor dem Anwender gekapselt werden. Stattdessen wird ihre Spezifizierung in die deklarative Sprache einbezogen, so dass die Trennung zwischen logischer und physischer Ebene, die ja ein herausragendes Merkmal von AJAX sein soll, wieder „verwischt“.

5.2 Data Quality Mining (DaimlerChrysler AG)

Im Rahmen erster Schritte zur Einführung eines unternehmensweiten DQM werden bei DaimlerChrysler Forschungsarbeiten zum Einsatz von Data Mining-Techniken zur Messung und Verbesserung von Datenqualität durchgeführt (vgl. Abschnitt 4.5.1) . Diese Arbeiten werden in [GH01b] an Hand der Datenbank QUIS (Qualitätsinformationssystem) evaluiert. QUIS verwaltet technische und kaufmännische Daten zu Fahrzeugen sowie Forderungen zu Gewährleistungsanspruch und Kulanz (siehe Abb. 5.1).

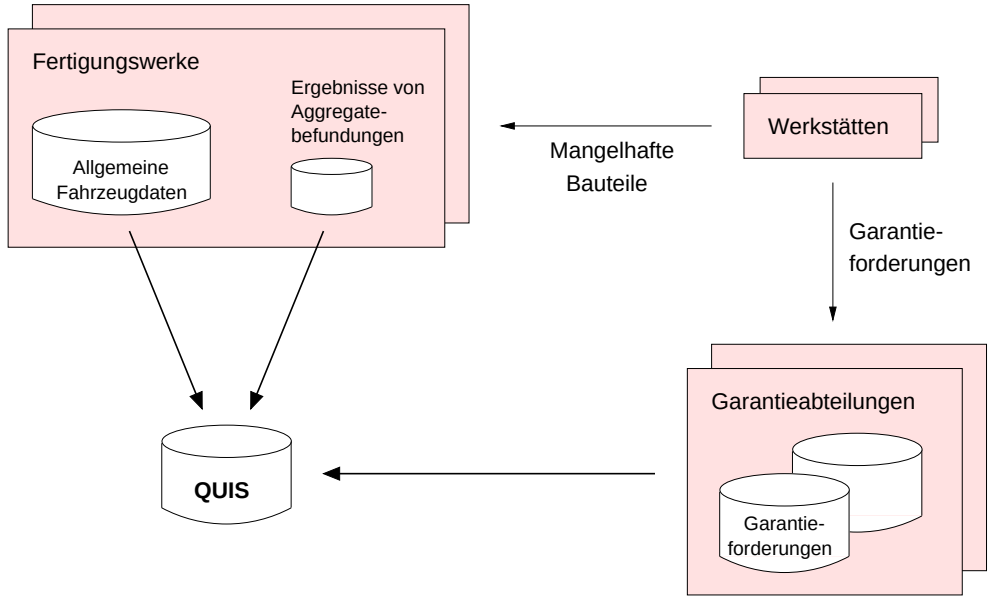


Abbildung 5.1: Überblick über die QUIS-Umgebung

Die Grundidee des Ansatzes besteht darin, dass sich der Einsatz eines Data Mining-Algorithmus zur Entdeckung neuer Muster (Regelmäßigkeiten) von dem Einsatz desselben Algorithmus zur Entdeckung von Datenanomalien nur in der Anwendung der gefundenen Muster und der Interpretation der Ergebnisse unterscheidet. In [HGG01] wird der zweite Fall zur Abgrenzung als *Data Quality Mining* bezeichnet – definiert als die „bewusste Anwendung von Data Mining-Techniken zum Zwecke der Datenqualitätsmessung und -verbesserung“. Ziel von Data Quality Mining ist es, Datenqualitätsmängel in großen Datenbeständen aufzuspüren, zu quantifizieren, zu erklären sowie zu korrigieren. Im Kontext des Data Warehousing ist Data Quality Mining zwischen der Datenextraktion aus den Quellen und dem Laden in das Data Warehouse anzusetzen. In den folgenden Abschnitten wird näher auf einzelne Forschungsschwerpunkte eingegangen.

5.2.1 Beschreibung von Datenbankinhalten mittels deskriptiver Statistik

In diesem Teilprojekt geht es insbesondere darum, das Verständnis von Daten und Prozessen zu verbessern, z. B. im Hinblick auf Prozessoptimierungen oder als Vorbereitung von Data Mining-Prozessen zur Datenanalyse. Letzteres ist vor dem Hintergrund zu sehen, dass die Qualität von Analyseergebnissen wesentlich stärker von einer gründlichen Vorbereitungsphase als von der späteren Optimierung von Data Mining-Parametern abhängt [CRI00].

Zur Dokumentation von Datenbankstrukturen und -inhalten wurde ein Prototyp entwickelt, der automatisiert zu jedem Attribut folgende Angaben ermittelt und in einem Gesamtdokument (PDF oder HTML) aufbereitet (vgl. Data Profiling, Abschnitt 4.5.1):

- Bezeichner und Datentyp eines Attributs
- *NULL* erlaubt? (ja/nein)
- Anzahl unterschiedlicher Werte
- Durchschnittliche Häufigkeit eines Attributwerts
- Standardabweichung
- Häufigste und seltenste Werte.

Zusätzlich können weitere Attributbeschreibungen, die z. B. aus Modellierungswerkzeugen stammen (Fachbegriffe etc.), in das Dokument integriert werden.

5.2.2 Ausreißererkennung mit Hilfe statistischer Verfahren

Dieses Teilprojekt beschäftigt sich mit der Ableitung statistischer Modelle aus Datenbeständen, der persistenten Speicherung dieser Modelle sowie deren Anwendung auf neu eingehende Daten im Sinne eines permanenten Prozessmonitoring mit dem Ziel einer frühzeitigen, automatisierten Fehlererkennung (siehe Abbildung 5.2). Tupel, deren Werte signifikante Abweichungen („Ausreißer“) vom jeweiligen Modell aufweisen, werden als verdächtig gekennzeichnet. Modelle werden in frei wählbaren Zeitabständen oder auf Nutzeranforderung aktualisiert.

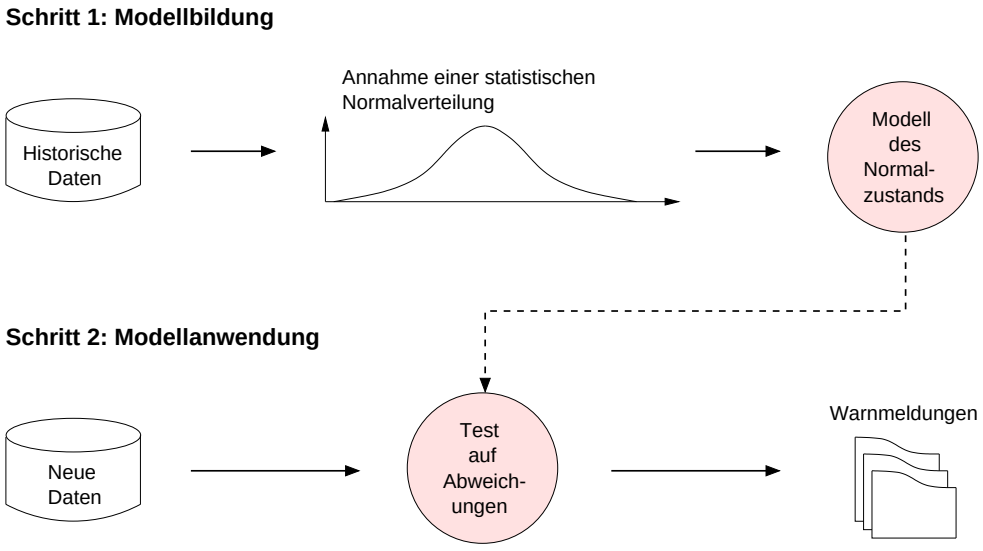


Abbildung 5.2: Abweichungsanalyse: Modellbildung und -anwendung

Mit Hilfe der Anwendung des in C implementierten Prototypen auf die QUIS-Datenbank wurden z. B. für einzelne Reparaturbetriebe stark vom erwarteten Wert abweichende Schadensforderungen innerhalb einzelner Wochen festgestellt [GH01b].

5.2.3 Entdeckung potentieller Datenanomalien mittels Entscheidungsregeln

Zur Entdeckung von Inkonsistenzen in QUIS wird bei DaimlerChrysler das kommerzielle Werkzeug GritBot [Qui00] eingesetzt. Dabei werden sowohl einzelne Relationen als auch Joins über mehrere Relationen betrachtet. Diese müssen zunächst in „flache“ Dateien geschrieben werden, da GritBot ausschließlich dateibasiert arbeitet. GritBot analysiert eine Datei in mehreren Durchläufen, wobei in

jedem Durchlauf ein anderes Attribut einer Menge von n Attributen als (potentiell) abhängiges Attribut angesehen wird. Numerische Werte werden dabei automatisch diskretisiert. In jedem Durchlauf erzeugt GritBot Entscheidungsregeln, z. B.:

```
GritBot [Release 1.02]       Mon Jul 3 13:30:52 2001
Read 5219164 cases (14 attributes) from ARTIKEL.data
1995 possible anomalies identified

case 3773780: [0.000]
     mpos = 2 (118966 cases, 100.00% '1')
     brv  = 510
...
```

Die mit Fall 3.773.780 assoziierte Regel besagt, dass in allen 118.966 Tupeln, in denen das Attribut `brv` den Wert 510 aufweist, das Attribut `mpos` den Wert 1 hat, mit Ausnahme des dargestellten Tupels (Wert 2). Damit wurde der Wert von `mpos` in diesem Tupel als potentiell falsch identifiziert – es ist allerdings ebenso möglich, dass der Fehler im Wert von `brv` liegt. Eine interaktive Nachbearbeitung dieses Tupels ist daher sinnvoll.

Die Evaluierung von GritBot ergab, dass in vielen Fällen Geschäftsregeln, die Fachexperten bereits bekannt, aber noch nicht explizit spezifiziert waren, bestätigt wurden. Darüber hinaus konnten weitere zufällige (d. h. nicht prozessbedingte) Datenanomalien aufgespürt werden. Als Hauptdefizit von GritBot nennt [GH01b] die fehlende Datenbankschnittstelle.

In zukünftigen Forschungsarbeiten ist eine Nutzung von Assoziationsregeln zum DQM geplant, vergleichbar mit dem in [MM00a] verfolgten Ansatz (siehe Abschnitt 5.4).

5.3 Ascential Quality Manager

Die Funktionalität des Ascential Quality Manager [Asc01a] umfasst vor allem die Datenqualitätsmessung und -analyse. Dazu wird eine Art „Baukasten“ von Funktionen bereitgestellt, die z. T. aufeinander aufbauen und miteinander kombiniert werden können (siehe Abschnitt 5.3.1). Zusätzlich wird von Ascential eine von TQM-Konzepten abgeleitete Methodik zum DQM in Organisationen propagiert (siehe Abschnitt 5.3.2), welches sich die einzelnen Elemente des Baukastens zunutze macht. Zur softwaretechnischen Unterstützung der Methodik wird ein Werkzeug zur Datenqualitätsmessung und -analyse angeboten (siehe Abschnitt 5.3.3).

5.3.1 Bausteine zur Bestimmung von Datenqualität

Ascential unterscheidet fünf Ebenen von Bausteinen zunehmender Komplexität. Die angebotenen Funktionen sollen dazu dienen, Eigenschaften der betrachteten Daten zu verstehen und Ansatzpunkte für eine Qualitätverbesserung bzgl. bestimmter Merkmale – soweit möglich – aufzuzeigen. Es wird dabei zwischen sechs Klassen von Daten (Datenklasse $\neq$ Datentyp) unterschieden:

- *Kennung (Identifier):* ein Wert ohne domänenbezogene Semantik, der auf eine Menge von Attributwerten zeigt, die wiederum eine Entität beschreiben (Primärschlüssel)
- *Indikator:* ein Wert aus einer Menge von genau zwei Elementen, der beschreibt, ob ein Merkmal zutrifft oder nicht (boolescher Wert)

- *Code:* ein Wert aus einer gegebenen endlichen Menge von mehr als zwei Elementen, die ein Merkmal annehmen kann (List of Values)
- *Datum/Zeit:* ein Wert, der einen Zeitpunkt beschreibt
- *Quantität:* ein Wert, der eine Quantität beschreibt (Umsatz, Alter, Gewicht etc.)
- *Text:* eine beliebige Zeichenkette.

Ein Indikator kann als Speziallfall eines Codes angesehen werden und wird daher im Folgenden darunter subsumiert.

Ebene 0: Domänenanalyse

Mittels einer Domänenanalyse kann ermittelt werden, welche Werte in einem bestimmten Attribut wie oft auftreten. Damit kann – im Sinne eines Metadaten-Reengineering – Wissen über Eigenschaften eines Datenbestandes gewonnen werden (vgl. Data Profiling, Abschnitt 4.5.1). Bezogen auf die einzelnen Datenklassen (siehe oben) werden auf Ebene 0 folgende Analysen durchgeführt:

- *Kennung:* Bestimmung von minimalen und maximalen Werten (falls sinnvoll), Bestimmung des Datenformats (falls sinnvoll)
- *Code:* Bestimmung von gültigen Werten und Defaultwerten
- *Datum/Zeit:* Bestimmung von zulässigen Zeiträumen und Defaultwerten, Bestimmung des Datenformats
- *Quantität:* Bestimmung von minimalen und maximalen Werten sowie Defaultwerten, Bestimmung des Datenformats
- *Text:* Bestimmung eines ggf. vorhandenen Formats, Bestimmung von Defaultwerten.

Ebene 1: Vollständigkeits- und Gültigkeitsanalyse

Auf Ebene 1 wird geprüft, wie viele Tupel in bestimmten Attributen einen Nullwert bzw. einen mit *NULL* äquivalenten Wert (Blank, 0, 999999 und sonstige Defaultwerte) aufweisen. Darüber hinaus wird nach ungültigen Werten, also Werten außerhalb des jeweils zulässigen Wertebereichs (Ergebnis aus Ebene 0), gesucht. Dabei sind NULL- bzw. dazu äquivalente Werte nur dann unzulässig, wenn es sich um ein Pflichtfeld handelt. Wiederum wird nach Datenklassen unterschieden:

- *Kennung, Datum/Zeit, Quantität:* Aufdeckung von ungültigen Formaten sowie Werten außerhalb des zulässigen Bereichs (falls sinnvoll).
- *Code:* Aufdeckung von Werten außerhalb der zulässigen Wertemenge.
- *Text:* Aufdeckung von ungültigen Formaten (falls sinnvoll), bei Namen/Adressen: Abgleich mit Registern (falls vorhanden), Aufdeckung von Sonderzeichen.

Ebene 2: Analyse der strukturellen Integrität

Die Analyse auf Ebene 2 erstreckt sich auf folgende Aspekte:

- *Primärschlüsselattribute und deren definierende Attribute:* Aufdeckung von nicht-eindeutigen Primärschlüsseln; Aufdeckung von übereinstimmenden definierenden Attributwerten, zu denen mehr als ein Primärschlüssel existiert (nicht zwangsläufig Duplikate).
- *Fremdschlüsselattribute und ihre referentielle Integrität:* Aufdeckung von Fremdschlüsseln in einer Kindrelation, zu denen kein Primärschlüssel in der Elternrelation existiert; Aufdeckung von Primärschlüsseln in einer Elternrelation, zu denen kein Fremdschlüssel in einer der Kindrelationen existiert (kein Datenqualitätsmangel, falls 1:0..1- oder 1:*-Beziehung).

Ebene 3: Analyse der Einhaltung von Geschäftsregeln

Während auf Ebene 1 lediglich die Gültigkeit einzelner Werte bestimmt wurde, wird auf Ebene 3 die Einhaltung von Geschäftsregeln geprüft, die sich auf mehrere Attribute eines Tupels oder auch mehrerer, inhaltlich zusammenhängender Tupel erstrecken. Es werden vier Typen von Geschäftsregeln unterschieden:

- *Regeln zur Prüfung gültiger Attributwertkombinationen,* z. B. Ortsangabe und Postleitzahl.
- *Regeln zur Prüfung der Einhaltung von Berechnungsvorschriften,* z. B.
```
Bestellung.Positionenanzahl = SELECT COUNT(*)
FROM Positionen WHERE Bestellung_Ref = ...
```
- *Regeln zur Prüfung des gültigen Zeitbezugs,* z. B. „Zeitpunkt einer Projektaktivität zwischen Projektstart und Projektende“ (Bereichsprüfung) oder „Lieferzeitpunkt nach Bestellzeitpunkt“ (Prüfung auf Reihenfolge).
- *If-Then-Else–Regeln,* z. B. „Existenz offener Posten nur bei aktiven Konten“.

Ebene 4: Analyse von Transformationsregeln

Ebene 4 bezieht sich auf den Transformationsprozess im Rahmen einer Datenintegration aus heterogenen Quellen. Die hier durchgeführten Analysen dienen folgenden Zwecken:

- Simulation und Test von Transformationen *vor* deren Inbetriebnahme
- Suche nach alternativen Transformationen mit höherer Produktivität
- Sicherstellung der korrekten Durchführung von Transformationen.

Im Gegensatz zu den Ebenen 0 bis 3 wird in Ebene 4 also *nicht* Datenqualität, sondern die Qualität der Spezifikation von Transformationsprozessen analysiert. Diese hat natürlich ihrerseits wiederum Auswirkungen auf die Datenqualität der Transformationsergebnisse.

Die bisher beschriebenen Bausteine der Ebenen 0 bis 4 sollen die Spezifizierung wiederholbarer, robuster Prozesse zur Datenqualitätsmessung und -analyse ermöglichen. Im folgenden Abschnitt wird die auf diesen Bausteinen aufsetzende DQM-Methodik von Ascential vorgestellt.

5.3.2 Methodik zum DQM

Die von Ascential propagierte DQM-Methodik [Asc01a] basiert auf Konzepten des Total Quality Management (siehe Abschnitt 4.1). Durch die systematische Anwendung der Methodik soll eine hinreichend hohe Datenqualität erreicht und langfristig gesichert werden. Die Methodik besteht aus drei Phasen (jeweils mit mehreren Subphasen), die in den nachfolgenden Abschnitten kurz skizziert werden.

Phase 1: Initiale Datenqualitätsbestimmung

Ziel dieser Phase ist es, den Ist-Zustand eines gegebenen Datenbestands bzgl. Datenqualität einzuschätzen. Dazu werden nach einer initialen Planung (Prozessspezifizierung, Ressourcenzuweisung, Festlegung von Anwendungsbereich, Dokumentationsanforderungen und Datenqualitätszielen) und Datenvorbereitung (Synchronisierung von Daten auf denselben Untersuchungszeitpunkt) die Datenqualitätsanalysefunktionen der Ebenen 0 bis 3 (siehe Abschnitt 5.3.1) sequentiell auf den Datenbestand angewandt. Aus den Analyseergebnissen werden priorisierte Empfehlungen in Form von Aktionsplänen (sowohl symptom- als auch ursachenorientiert) abgeleitet und dem Management zur Entscheidung vorgelegt.

Phase 2: Überwachung der Datenqualitätsverbesserung

Nach Abschluss der initialen Datenqualitätsbestimmung, der Feststellung eines Verbesserungsbedarfs und der Zusage des Managements, Verbesserungsmaßnahmen zu unterstützen, kann in die zweite Hauptphase der Methodik übergegangen werden. Dabei wird ein Monitoring-System entwickelt und in Betrieb genommen, welches den Datenbestand gezielt periodisch (z. B. monatlich) daraufhin untersucht, inwieweit sich dessen Qualität seit der letzten Messung verändert hat, um so Trends über die Wirksamkeit der in der Zwischenzeit ergriffenen Verbesserungsmaßnahmen ermitteln und ggf. korrigierende Eingriffe in den Prozess vornehmen zu können. Dabei kommen die Analysemethoden der Ebenen 1 bis 3 zum Einsatz. Die Entscheidung, ob ein Datenqualitätsmangel beseitigt werden sollte oder nicht, muss durch ein Abwägen des Aufwands zur Beseitigung eines Mangels gegen den negativen Einfluss des Mangels getroffen werden – eine Beseitigung *aller* vorhandenen Mängel ist oft nicht praktikabel. Wenn eine hinreichend hohe Datenqualität erreicht ist, kann die Überwachung der Datenqualitätsverbesserung durch eine auf statistisch validen Datenstichproben beruhende (und damit weniger aufwändige) Datenqualitätsüberwachung (Phase 3) ersetzt werden.

Phase 3: Datenqualitätsüberwachung

Ziel dieser Phase ist es zu überwachen, dass ein in Phase 2 erreichter Datenqualitätslevel über den gesamten „Lebenszyklus“ eines Datenbestands gehalten oder sogar verbessert werden kann. Dazu reicht es aus, statistisch valide Datenstichproben zu betrachten, auf diese die Analysemethoden der Ebenen 0 bis 3 anzuwenden und von den Analyseergebnissen auf den Gesamtdatenbestand zu extrapolieren.

Die Ergebnisse der periodisch (z. B. monatlich) durchzuführenden Datenqualitätsüberwachung sollten in Trend-Charts (ein Chart pro untersuchter Einheit, z. B. einer Relation) aufbereitet werden, die dann dem Management, den Datennutzern, den Datenproduzenten und/oder anderen Interessengruppen vorzulegen sind.

Zusätzlich zu der stichprobenorientierten Datenqualitätsüberwachung sollte regelmäßig – allerdings mit längerer Periodendauer (z. B. jährlich) – ein vollständiges Audit des Datenbestandes

erfolgen. Dieses besteht in der Wiederholung der Ausführung von Phase 1 (initiale Datenqualitätsbestimmung). Ziel des Audits ist es zum einen, die Wirksamkeit der stichprobenorientierten Datenqualitätsüberwachung zu belegen, und zum anderen, signifikante Änderungen in den Eigenschaften des Datenbestands (z. B. die Erweiterung des gültigen Wertebereichs eines Attributs) zu erkennen und in nachfolgenden Datenqualitätsüberwachungsprozessen zu berücksichtigen.

Zur softwaretechnischen Unterstützung der vorgestellten DQM-Methodik stellt Ascential mit dem Quality Manager ein Werkzeug für die Windows-Plattform bereit, das die in Abschnitt 5.3.1 präsentierten funktionalen Bausteine (zumindest teilweise) implementiert und in eine Ablaufumgebung zum DQM einbettet. Der folgende Abschnitt geht näher auf dieses Werkzeug ein.

5.3.3 Werkzeug

Die Ausführungen dieses Abschnitts beruhen auf [Asc01b], das Werkzeug selbst wurde im Rahmen der Arbeit nicht evaluiert. Laut Dokumentation stellt der Ascential Quality Manager eine Menge von miteinander in Beziehung stehenden Objekttypen bereit, deren Instanzen entweder ausführbare Prozessspezifikationen oder Ergebnisse von ausgeführten Prozessen sind. Die einzelnen Objekttypen werden im Folgenden kurz charakterisiert (siehe Abbildung 5.3).

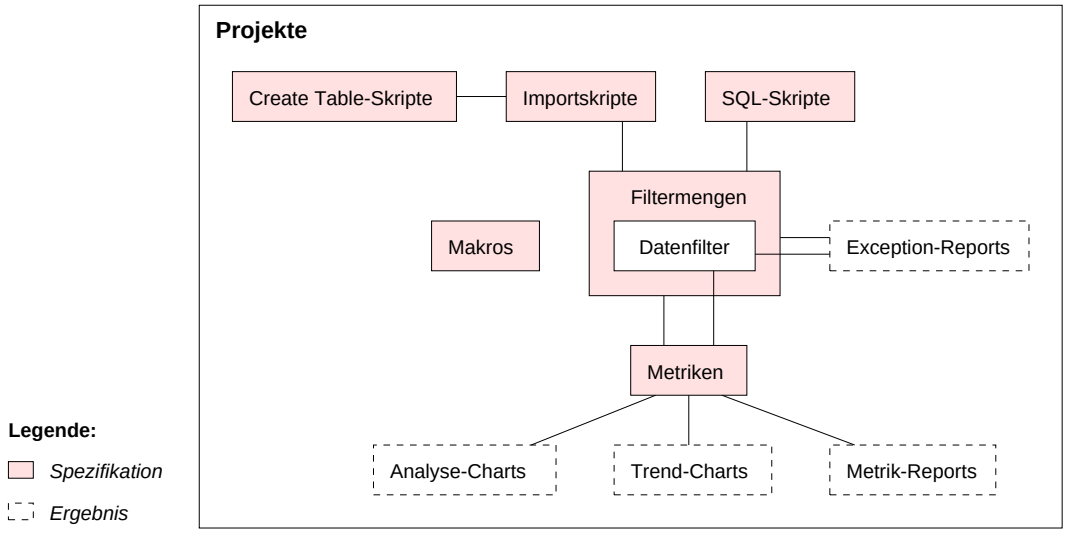


Abbildung 5.3: Objekttypen im Ascential Quality Manager (frei nach [Asc01b])

- *Datenfilter* spezifizieren Eigenschaften von Daten und separieren Tupel, die diese Eigenschaften aufweisen, von Tupeln, die dies nicht tun. Eigenschaften in diesem Kontext sind Verletzungen elementarer Datenqualitätsprinzipien und/oder nutzerdefinierter Geschäftsregeln. Ein Datenfilter kann wiederholt auf sich ändernden Datenbeständen ausgeführt werden. Datenfilter basieren auf dem relationalen Datenmodell und werden deklarativ spezifiziert. Es wird zwischen einfachen und komplexen Filtern unterschieden: Einfache Filter beziehen sich auf ein einzelnes Attribut einer Relation. Komplexe Filter verknüpfen einfache und/oder komplexe Filter mittels logischer Operatoren und können so relationen- und attributübergreifend nach unterschiedlichen Eigenschaften suchen.

- *Filtermengen* fassen inhaltlich verwandte Datenfilter zu Kategorien zusammen.

- *Exception-Reports* enthalten historisierte Ergebnisse, die aus der wiederholten Ausführung von Datenfiltern und Filtermengen hervorgegangen sind, auf Tupelebene in Tabellenform.

- *Metriken* berechnen numerische Werte, die bestimmte Datenqualitätsaspekte des betrachteten Datenbestands beschreiben. Eine Metrik baut dazu auf den Ergebnissen von Datenfiltern und/oder Filtermengen auf und weist diesen Gewichte zu, die domänenspezifische Kosten repräsentieren. Aus den gewichteten Ergebnissen wird dann durch Anwendung arithmetischer Operatoren ein einzelner numerischer Wert aggregiert. Ergebnisse der Metrikanwendung werden wiederum historisiert gespeichert und können in Metric Reports, Analysis Charts und Trend Charts aufbereitet werden.
- *Analyse-Charts* sind Balken- oder Tortendiagramme zur Präsentation zusammengehöriger Dateneigenschaften (z. B. Gegenüberstellung der gravierendsten Datenqualitätsprobleme in einem Datenbestand).
- *Trend-Charts* sind Linien- oder Balkendiagramme zur Präsentation von Dateneigenschaften über die Zeit.
- *Metrik-Reports* enthalten historisierte Ergebnisse, die aus der wiederholten Ausführung von Metriken hervorgegangen sind, in Tabellenform.
- *Create Table-Skripte* erzeugen relationale Strukturen in der internen Datenbank des Werkzeugs.
- *Importskripte* übertragen Daten aus externen Quellen in die interne Datenbank des Werkzeugs.
- *SQL-Skripte* dienen der Spezifizierung nutzerdefinierter Datenbankzugriffe.
- *Makros* kombinieren mehrere Objekte zu einem automatisierten Ausführungsstrom.
- *Projekte* enthalten eine Menge von (ausführbaren) Spezifikationen und (aus der Ausführung gewonnenen) Messergebnissen. Sämtliche Analysen werden projektorientiert durchgeführt.

Datenfilter können laut [Asc01b] für Analysen der Ebenen 0 bis 4 (siehe Abschnitt 5.3.1) genutzt werden. Tatsächlich werden aber lediglich die Ebenen 0 bis 2 explizit unterstützt. Komplexe Geschäftsregeln (Ebene 3), insbesondere If-Then-Else–Regeln, sind in der vorliegenden Version 4.8.1 nicht spezifizierbar. Eine Simulation, Optimierung und Kontrolle von Transformationsprozessen (Ebene 4) ist ebenfalls nicht möglich.

Trotz dieser Einschränkungen stellt der Ascential Quality Manager ein relativ flexibles, einfach strukturiertes und praxisnahes Werkzeug zur Datenqualitätsmessung und -analyse dar. Außergewöhnlich ist die zusätzliche Bereitstellung einer Methodik zur effektiven Anwendung des Werkzeugs.

5.4 Weitere Ansätze

Neben den in den vorangegangenen Abschnitten vorgestellten Lösungen finden sich in der Literatur noch weitere Ansätze, die nun kurz skizziert werden:

- *Total Data Quality Management (MIT)* [Wan98, BWPT98]
  Ziel des seit 1992 an der MIT Sloan School of Management laufenden Forschungsprogramms Total Data Quality Management (TDQM) ist es, eine theoretische Basis im Forschungsgebiet Datenqualität zu schaffen. Basierend auf der Unternehmensphilosophie des Total Quality Management [Dem00] wurde ein sog. *TDQM-Zyklus* von Demings klassischem PDCA-Zyklus abgeleitet, der die Phasen Datenqualitätsplanung, -messung, -analyse and -verbesserung umfasst (vgl. Abschnitt 4.3). Weiterhin wurde im Rahmen des TDQM-Programms eine Menge von Datenqualitätsmerkmalen (Datenqualitätsdimensionen genannt) identifiziert, empirisch untersucht und mit einigen einfachen Metriken zur Datenqualitätsmessung assoziiert. Schließlich

wurde ein Ansatz zur Anreicherung des relationalen Datenmodells mit Metainformationen zur Datenqualität entwickelt. Insgesamt bleiben die meisten TDQM-Ansätze oberflächlich und wenig IT-orientiert.

- *Foundations of Data Warehouse Quality (Esprit)* [JJQV99, JQJ98, GJJ+98, JV97]
  In dem Esprit-Projekt Foundations of Data Warehouse Quality (DWQ) wurde eine Entwurfsmethode entwickelt, die Qualitätsanforderungen möglichst aller DWS-Beteiligten (Datennutzer, Datenlieferanten, DWS-Administratoren etc.) berücksichtigen soll [JJQ99]. Dabei ist Datenqualität neben der Qualität der Systemarchitektur (Skalierbarkeit etc.), Softwarequalität sowie Schemaqualität nur ein Einzelaspekt. Es wird klar differenziert zwischen konzeptuellen, logischen und physischen Schemata, deren Semantik explizit in einer Metadatenbank (basierend auf ConceptBase [JJNS98]) beschrieben ist. Die Integration von Quellschemata wird durch sog. *Interschema Knowledge Networks* unterstützt, die Beziehungen zwischen Schemata spezifizieren [CGL+99]. Zur Qualitätsbewertung wird auf den aus dem Software Engineering stammenden GQM-Ansatz zurückgegriffen (siehe Abschnitt 4.5). DWQ deckt ein breites Spektrum von Qualitätsaspekten im Data Warehousing ab, bleibt aber bzgl. einzelner Punkte – insbesondere bzgl. Datenqualitätsaspekten – zu oberflächlich.

- *Potter's Wheel A-B-C (Uni Berkeley)* [RH00]
  In dem Projekt wird ein interaktives Framework zum Data Cleansing entwickelt, welches Datentransformation und Fehlerentdeckung eng miteinander koppelt. Mit Hilfe einer Menge einfacher Transformationen lassen sich Beispieltupel manipulieren, wobei Transformationsergebnisse „on the fly" angezeigt werden. Während der Transformation wird im Hintergrund automatisiert nach Inkonsistenzen gesucht.

- *IntelliClean (Uni Singapore)* [LLL00, LLLK99]
  Das Projekt verfolgt einen regelbasierten Ansatz zum Data Cleansing. Domänenspezifisches Wissen (Geschäftsregeln, Merging-Regeln etc.) wird von einem regelbasierten System verwaltet und von einer Rule Engine ausgewertet (unter Verwendung der Java Expert System Shell). Dieser Ansatz ist jedoch für ein umfassendes DQM bei weitem nicht ausreichend.

- *DataSphere (AT&T Labs Research)* [DJK00]
  In diesem Projekt werden Data Mining-Techniken zur automatisierten Entdeckung von Fehlern in multivariaten Datenmengen eingesetzt (vgl. Abschnitt 5.2). Dazu wurde ein als DataSphere bezeichnetes Vergleichsverfahren entwickelt, das Anomalien aufspürt.

- *Automatisierte Fehlersuche (Uni Memphis)* [MM00a, MM00b]
  In diesem Projekt soll ein Framework zum automatisierten Data Cleansing definiert werden. In großen Datenbeständen sollen automatisiert (syntaktische und semantische) Fehler gefunden und korrigiert werden. Dabei werden die zu Grunde liegenden theoretischen Anteile aus der Forschung zum Thema Datenqualität mit Problemlösungsmethoden aus den Bereichen Software Testing, Data Mining, Statistik, Wissensbasierte Systeme, Clustering und Maschinelles Lernen kombiniert (vgl. Abschnitt 5.2).

- *Semiotikbasiertes Datenqualitätsframework (Uni Melbourne)* [SC99]
  Das Projekt zielt auf die Entwicklung eines sog. *semiotikbasierten Datenqualitätsframeworks* (vgl. Abschnitt 3.2.1), das aus den fünf Kernkonzepten „Interessengruppen", „Datenqualitätsziele", „Datenqualitätsmerkmale", „Verbesserungsstrategien" (pro Ziel) und „Metriken" (pro Merkmal) besteht. Die einzelnen Kernkonzepte werden auf den semiotischen Ebenen „syntaktisch", „semantisch", „pragmatisch" und „sozial" untersucht mit dem Ziel, ein besseres Verständnis für Datenqualitätsaspekte und deren Beziehungen und Wirkungen zu gewinnen.

- *HiQiQ (Humboldt-Universität Berlin)* [NLF99]
  Hier geht es vorwiegend um die Einbeziehung von Qualitätsinformationen in die Anfrageverarbeitung in verteilten Datenbanken.

Weitere, hier nicht näher betrachtete Ansätze werden in [Epp01, Nee01, JE01, BT99, TB98, KKPP98, Orr98, BFG98, DJ98, SLW97a, Pij94] beschrieben. Einen Überblick über DQM-Ansätze aus unterschiedlichen Anwendungsbereichen geben [EW00] und [WSF95]. Grundlegende Überlegungen zum DQM finden sich weiterhin in [Hin99b, Hin00a, Hin00b].

5.5 Zusammenfassung und Bewertung

Tabelle 5.1[1] stellt die in den vorangegangenen Abschnitten vorgestellten Ansätze einander gegenüber. Die dabei zur Bewertung herangezogenen Kriterien wurden subjektiv – vor dem Hintergrund der Ziele der hier vorliegenden Arbeit – ausgewählt:

- *Domänenunabhängigkeit:* Anpassungsfähigkeit an verschiedene Anwendungsgebiete
- *Automatisierbarkeit:* Anteil automatisierbarer DQM-Maßnahmen
- *Systemunabhängigkeit:* Einsetzbarkeit in unterschiedlichen IT-Umgebungen
- *Ganzheitlichkeit:* Berücksichtigung aller Teilbereiche des DQM
- *Metriken:* Entwicklung/Verwendung von Datenqualitätsmetriken
- *DQM-Methodik:* Entwicklung/Verwendung einer Methodik zur Durchführung des DQM
- *Werkzeugunterstützung:* Implementierung der entwickelten Konzepte
- *Metadatenstandard:* Berücksichtigung eines Metadatenstandards
- *Evaluierung:* Durchführung einer Evaluierung der Konzepte bzw. der Implementierung.

Projekt	**Domänenunabhängigkeit**	**Automatisierbarkeit**	**Systemunabhängigkeit**	**Ganzheitlichkeit**	**Datenqualitätsmetriken**	**DQM-Methodik**	**Werkzeugunterstützung**	**Metadatenstandard**	**Evaluierung**
AJAX (INRIA)	+	+	+	–	–	–	+	–	+
DQ Mining (DaimlerChrysler AG)	+	+	+	–	○	–	+	–	+
Ascential Quality Manager	+	+	+	○	○	+	+	–	
TDQM (MIT)	+	–	+	–	○	○	–	–	+
DWQ (Esprit)	+	○	+	+	–	–	○	–	○
Potter's Wheel A-B-C (Berkeley)	+	+		–	–	–	+	–	
IntelliClean (Singapore)	+	+		–	–	–	+	–	
DataSphere (AT&T Labs Research)	+	+		–	–	–	+	–	+
Autom. Fehlersuche (Uni Memphis)	+	+	+	–	–	○	+	–	+
Semiot. DQ-Framework (Uni Melbourne)	+	–	+	–	–	–	–	–	+
HiQiQ (Humboldt Uni Berlin)	+	○	+	○	○	–	+	–	+

Tabelle 5.1: Forschungsarbeiten zum DQM

Kriterien wie Portierbarkeit, Laufzeiteffizienz, Robustheit, Benutzungsfreundlichkeit, Wartbarkeit und Erweiterbarkeit wurden ausgeklammert, da sie Details der softwaretechnischen Implementierung der einzelnen Ansätze (falls vorhanden) betreffen und diese aus der verfügbaren Literatur nicht zu entnehmen waren.

[1] Legende: + = erfüllt, ○ = teils/teils, – = nicht erfüllt, kein Eintrag = keine Angabe.

Alle betrachteten Arbeiten sind domänen- und systemunabhängig – soweit feststellbar. Dies ist allerdings zum Teil dadurch bedingt, dass die Ansätze *sehr* allgemein gehalten bzw. nicht implementiert sind.

Bis auf wenige Ausnahmen wird eine (oft prototypische) Werkzeugunterstützung geboten, mit der DQM-Maßnahmen automatisiert werden können. Allerdings beschränken sich die meisten Ansätze auf bestimmte Teilaspekte des DQM (z. B. Konsistenzprüfungen, Record Linkage etc.). Diese z. T. sehr unterschiedliche Ausrichtung erschwert ihren direkten Vergleich. Insgesamt lässt sich jedoch feststellen, dass es noch erheblichen Forschungsbedarf in den Bereichen Datenqualitätsmetriken und DQM-Methodiken gibt. Des weiteren existieren bisher keine auf einem Metadatenstandard basierenden Lösungen.

Im Folgenden werden die in den Abschnitten 5.1 bis 5.3 näher betrachteten Ansätze in Beziehung zu den Zielen dieser Arbeit gesetzt:

- *AJAX: Deklaratives Data Cleaning (INRIA)*
  Bei AJAX stehen die Beschreibung von Datenintegrationsprozessen auf logischer Ebene und die Transformation derartiger Spezifikationen auf die physische Ebene (Programmcode) im Vordergrund. Damit deckt AJAX Aspekte ab, die von der vorliegenden Arbeit nicht oder nur am Rande berührt werden. Hier werden Integrationsprozesse nicht – wie in AJAX – über eine deklarative Sprache, sondern über Eingabemasken spezifiziert (siehe Kapitel 11), woraus sich ggf. Einschränkungen im Hinblick auf die Mächtigkeit und Flexibilität der Spezifizierung ergeben. AJAX wiederum mangelt es an einer Methodik zur systematischen Anwendung der angebotenen Operatoren sowie an Funktionen zur Qualitätsplanung, -messung und -lenkung. Eine Kombination beider Ansätze wäre daher denkbar und sinnvoll.

- *Data Quality Mining (DaimlerChrysler AG)*
  Der Data Quality Mining-Ansatz setzt Data Mining-Verfahren zur Aufdeckung von Datenanomalien sowie zur Abschätzung der Ist-Qualität von Datenbeständen im Sinne von Datenaudits ein. Ein vergleichbarer, allerdings weniger weitgehender Ansatz wurde in einer Diplomarbeit im Kontext der vorliegenden Arbeit verfolgt (siehe [Wil00] sowie Abschnitt 9.7). Es bietet sich daher an, den Data Quality Mining-Ansatz mit den im Rahmen dieser Arbeit entwickelten Lösungen zusammenzuführen und so in eine umfassende DQM-Methodik einzubetten. Ein entsprechender Forschungskontakt zur DaimlerChrysler AG wurde bereits geknüpft (siehe [GH01b, GH01a] sowie Abschnitt 12.3).

- *Ascential Quality Manager*
  Die Zielsetzung dieser Arbeit wird durch den Ascential Quality Manager nur gestreift: Es werden weder Metriken (im klassischen Sinne) für Datenqualitätsmerkmale noch Verfahren zur Datenqualitätsverbesserung oder -lenkung angeboten. Einzelne Datenqualitätsmerkmale wie z. B. Redundanzfreiheit werden vernachlässigt. Metadaten werden proprietär gespeichert. Parallelen zu dieser Arbeit sind in der Nutzung von TQM-Konzepten sowie in der angebotenen Funktionalität zur Datenqualitätsmessung und -analyse zu sehen. Die DQM-Methodik von Ascential ist hingegen eher orthogonal zu den Zielen dieser Arbeit ausgerichtet, da nicht das DQM im Datenintegrationsprozess behandelt wird (die Analogie zum Fertigungsprozess fehlt völlig), sondern die Durchführung regelmäßiger Datenaudits im Mittelpunkt steht, unabhängig von konkreten Datenverarbeitungsprozessen.

TDQM und DWQ sind der Grundlagenforschung zuzurechnen. Die übrigen in Abschnitt 5.4 skizzierten Ansätze stehen in keinem unmittelbaren Zusammenhang zu den Zielen dieser Arbeit.

Zur Homepage der Dissertation

Teil II

Datenqualitätsmanagement in Data Warehouse-Systemen

Kapitel 6

Datenqualitätsmetriken, Mess- und Verbesserungsverfahren

Wie in Abschnitt 4.5 erläutert, existieren bisher keine etablierten Metriken für Datenqualitätsmerkmale. Solche Metriken sind jedoch unabdingbar zur Einschätzung der Qualität eines gegebenen Datenbestandes sowie zur Durchführung eines DQM („You cannot control what you cannot measure.“ [DeM82]). In diesem Kapitel wird daher versucht, das Defizit an Datenqualitätsmetriken zumindest für ausgewählte Merkmale zu beheben.

Aufbauend auf der in Abb. 3.5 (siehe Abschnitt 3.2.3) dargestellten Taxonomie von Datenqualitätsmerkmalen (konzeptuelle Sicht) werden in Abschnitt 6.1 systematisch objektive, merkmalsbezogene Metriken auf unterschiedlichen Granularitätsebenen definiert (logische Sicht). Weiterhin werden Messverfahren entwickelt bzw. der Literatur entnommen, die die jeweiligen Metriken operationalisieren (physische Sicht), indem sie für ein gegebenes Datenprodukt (vgl. Abschnitt 3.1) den Erfüllungsgrad einer Anforderung an ein Merkmal berechnen.

Der Einsatz der Metriken und Messverfahren setzt voraus, dass alle für eine Qualitätsbewertung relevanten Metadaten (z. B. Angaben zu Update-Häufigkeit oder Befunddatum) vorliegen und zugreifbar sind. In der gängigen Praxis ist dies oft (noch) nicht gewährleistet, da das zur Erreichung dieses Ziels durchzuführende Metadaten-Reengineering kostenintensiv ist und sich nur langfristig rentiert, so dass viele Organisationen nur zögerlich in diesen Bereich investieren. Für eine effektive Datenqualitätsbewertung ist die Bereitstellung qualitativ hochwertiger Metadaten jedoch unabdingbar.

In Abschnitt 6.2 werden – wiederum differenziert nach Datenqualitätsmerkmalen – Verfahren zur (symptomorientierten) Verbesserung von Datenqualität diskutiert. Zusätzlich zu den allgemein einsetzbaren merkmalsbezogenen Metriken stellt Abschnitt 6.3 exemplarisch einige domänenspezifische Datenqualitätsindikatoren vor.

6.1 Metriken und Messverfahren nach Datenqualitätsmerkmalen

Die in der vorliegenden Arbeit entwickelten Metriken setzen auf einem relationalen Datenbankschema auf; eine Anpassung an objektorientierte oder objektrelationale Datenmodelle ist ebenfalls denkbar. Zur besseren Skalierbarkeit werden die Metriken bottom-up für unterschiedliche Granularitätsebenen (Attributwert, Tupel, Relation, Datenbank) entwickelt, d. h. eine Metrik auf Ebene $n + 1$ (z. B. Vollständigkeit auf Tupelebene) wird auf Basis einer zuvor definierten Metrik auf Ebene n (hier: Vollständigkeit auf Attributwertebene) definiert.

Alle Metriken sind normiert auf ein Intervall $]0, 1]$ (in einigen Fällen auch $[0, 1]$). Dies vereinfacht die Aggregierung von Metriken zu gröberen Granularitätsebenen sowie den Austausch von Metriken

(Kapselung). Weiterhin weisen alle Metriken mindestens das Skalenniveau „ordinal“ auf, d. h. es existiert eine (Rang-)Ordnung der Werte (siehe auch [FP97]). Diese verhalten sich monoton steigend bei steigender Qualität im betrachteten Merkmal. Damit sind Vergleichsoperatoren wie „ist größer als“ anwendbar. Auf die Einführung eines Abstandsmaßes (Skalenniveau „Intervall“ nach [FP97]) wird hingegen verzichtet. Damit werden zwar Operatoren wie „liegt näher bei ... als bei ...“ ausgeschlossen, da jedoch im DQM eher Operationen typisch sind, die ermitteln, ob eine gegebene Mindestqualität erreicht wird oder nicht, sind ordinale Metriken für die Ziele dieser Arbeit ausreichend.

Bei der Entwicklung von Metriken wird lediglich eine Teilmenge der in Abb. 3.5 dargestellten Datenqualitätsmerkmale betrachtet, nämlich (im weiteren Verlauf verwendete Kürzel in Klammern):

- Korrektheit *(Korr)*
- Konsistenz *(Kon)*
- Vollständigkeit *(Voll)*
- Redundanzfreiheit *(Red)*
- Genauigkeit *(Gen)*
- Zeitnähe *(Zeit)*
- Eindeutigkeit *(Eind)*
- Schlüsseleindeutigkeit *(Schl)*
- Referentielle Integrität *(Ref)*.

Die Auswahl der genannten Merkmale erfolgte nicht willkürlich, sondern basiert auf folgenden Argumenten:

- Alle betrachteten Merkmale sind *objektiv* messbar, wie im Folgenden noch gezeigt wird. Das schließt nicht aus, dass das Ergebnis einer Qualitätsmessung nach kontextsensitiven oder subjektiven Kriterien bewertet wird.

- Die Bewertung der Relevanz von Daten ist nur interaktiv – unter Einbeziehung von umfangreichem Domänen- und Kontextwissen – möglich. In dieser Arbeit wird davon ausgegangen, dass bereits bei der Datenextraktion sichergestellt wird, dass ausschließlich relevante Daten in das DWS gelangen, d. h. es muss mindestens einen Kunden geben, der auf die extrahierten Daten angewiesen ist. Konzepte aus dem Bereich des Information Retrieval [Rij79] können bei der Bewertung hilfreich sein (Deskribierung von Datenbeständen, Stichwortsuche, Nutzerprofile etc.).

- Auch beim Merkmal „Verständlichkeit“ spielen domänenspezifische und subjektive Einflüsse eine große Rolle, so dass eine automatisierte Messung nicht sinnvoll erscheint. Arbeiten aus den Bereichen Ergonomie/User Satisfaction [CDN88] und Modellierungsqualität [BRS95, MS94, RG94] könnten im Rahmen einer interaktiven Bewertung der Verständlichkeit von Daten genutzt werden.

- Die Einheitlichkeit von Daten könnte mit Hilfe der Anzahl und Komplexität der im Vereinheitlichungsprozess durchzuführenden Transformationsschritte bestimmt werden. Da in dem in Kapitel 7 entwickelten Vorgehensmodell zur Datenintegration jedoch die Vereinheitlichung der Repräsentation von Daten bereits in einer frühen Phase durchgeführt und die Einheitlichkeit der Daten von den nachfolgenden Phasen als gegeben vorausgesetzt wird, wird auf die detaillierte Spezifizierung einer Metrik für das Merkmal „Einheitlichkeit“ verzichtet. In Abschnitt 6.2 werden jedoch Maßnahmen zur Verbesserung der Qualität von Daten bzgl. dieses Merkmals diskutiert.

Tabelle 6.1 zeigt, dass für die Mehrzahl der betrachteten Datenqualitätsmerkmale auf allen vier Granularitätsebenen Metriken definiert werden konnten. Da es bei den Merkmalen „Redundanzfreiheit“, „Schlüsseleindeutigkeit“ und „referentielle Integrität“ um tupelübergreifende Aspekte (z. B. mehrfach in einer Relation repräsentierte Entitäten) geht, ist die Definition von Metriken auf den feineren Granularitätsebenen „Attributwert“ und „Tupel“ für diese Merkmale nicht sinnvoll.

Datenqualitätsmerkmal	**Attributwert**	**Tupel**	**Relation**	**Datenbank**
Korrektheit	•	•	•	•
Konsistenz	•	•	•	•
Vollständigkeit	•	•	•	•
Redundanzfreiheit			•	•
Genauigkeit	•	•	•	•
Zeitnähe	•	•	•	•
Eindeutigkeit	•	•	•	•
Schlüsseleindeutigkeit			•	•
Referentielle Integrität			•	•

Tabelle 6.1: Zuordnung von Granularitätsebenen zu Datenqualitätsmerkmalen

Die in der vorliegenden Arbeit entwickelten Metriken genügen den zu Beginn dieses Abschnitts festgelegten Anforderungen bzgl. Normierung und Skalenniveau. Um die „Sensitivität“ der Metriken zu erhöhen (z. B. Annäherung an lineares oder logarithmisches Verhalten), könnten diese um Faktoren, Exponenten etc. erweitert werden. Auf dieses „Feintuning“ wird bewusst verzichtet, da dies Erfahrungswissen im praktischen Umgang mit den Metriken in einer konkreten Anwendungsdomäne voraussetzt. Hauptanliegen des hier verfolgten Ansatzes ist stattdessen zu zeigen, dass es prinzipiell möglich ist, die Qualität von Daten bzgl. verschiedener Merkmale objektiv, domänenunabhängig und (weitgehend) automatisiert zu messen.

In den folgenden Abschnitten werden die einzelnen Metriken vorgestellt, beginnend mit der jeweils feinsten Ebene pro Merkmal. Eine Metrik auf Ebene $n + 1$ greift dabei i. Allg. auf die zuvor definierte Metrik auf Ebene n zurück, indem die jeweils dort berechneten Kennzahlen – ggf. gewichtet – arithmetisch gemittelt werden. Diese einfache Art der Aggregierung ist nicht robust gegenüber Ausreißern (extrem große bzw. extrem kleine Werte) in den Daten. Alternativ wäre eine Berechnung des Medians denkbar. Dieser gibt allerdings u. U. ein sehr verzerrtes Abbild wieder, z. B. würde bei 101 Messwerten, von denen 50 bei 0.1, 50 bei 0.9 und 1 bei 0.85 liegen, der Median 0.85 betragen. Der Einsatz einer geometrischen Mittelung ist nicht sinnvoll, da sich diese ausschließlich für die Analyse multiplikativer numerischer Relationen (z. B. durchschnittliche prozentuale Veränderungen) eignet [HEK99].

Einer Metrik der jeweils feinsten Granularitätsebene (bezogen auf ein Qualitätsmerkmal) wird schließlich ein Messverfahren zugeordnet, das die Metrik operationalisiert. Auf den gröberen Ebenen werden die so ermittelten Messwerte dann gewichtet und gemäß der Metrikdefinitionen gemittelt.

6.1.1 Korrektheit

Die Korrektheit von Daten ist nur durch einen (i. Allg. nicht automatisierbaren) Vergleich des Diskursweltzustandes (Soll-Zustand) mit dem Zustand des Informationssystems (Ist-Zustand) zu beurteilen. Ein entsprechendes Beispiel aus der Geschäftswelt ist die Durchführung einer Inventur. Hier wird der tatsächliche Artikelbestand im Lager ermittelt und mit den Bestandsangaben im Informationssystem verglichen.

Im Kontext dieser Arbeit kann der Grad der Ähnlichkeit zwischen Ist- und Sollwerten mit Hilfe eines Abstandsmaßes (attributbezogenes Metadatum) bestimmt werden. Die Korrektheit wird dann über die Ähnlichkeit von Attributwerten eines Datenprodukts mit denen der jeweils repräsentierten Entitäten der Diskurswelt bewertet.

Attributwertebene

Sei w_I ein Attributwert im Informationssystem, w_R der mit w_I korrespondierende Attributwert einer modellierten Entität in der Diskurswelt.

Sei d ein (domänenspezifisches) Abstandsmaß.

Beispiel 1: $d(w_1, w_2) := \begin{cases} 0 & \text{falls } w_1 = w_2 \\ \infty & \text{sonst.} \end{cases}$ Beispiel 2: $d(w_1, w_2) := |w_1 - w_2|$

Andere in Frage kommende Abstandsmaße sind z. B. Editierabstand, Euklidischer Abstand oder Hamming-Abstand.

Der Qualitätswert Q_{Korr} von w_I bzgl. w_R wird nun definiert als:

$$Q_{Korr}(w_I, w_R) := \frac{1}{d(w_I, w_R) + 1} \tag{6.1}$$

Ob diese Metrik den zu Beginn von Abschnitt 6.1 spezifizierten Anforderungen bzgl. Normierung und Skalenniveau genügt, hängt von der verwendeten Metrik ab. Allgemein gilt: Bei einem Abstand von 0 (Identität) ist der Qualitätswert 1. Je größer der Abstand ist, desto näher liegt der Qualitätswert bei 0 (Nenner geht gegen unendlich).

Tupelebene (einelementiger View)

Sei t ein Tupel, das (per SQL) durch eine Selektion (ggf. inkl. Joins) einer einelementigen Ergebnismenge sowie eine Projektion auf eine beliebige, nichtleere Teilmenge von Attributen $A_1, ..., A_n$ definiert ist.

Sei $t.A_j$ der Wert von t im Attribut $A_j \quad (j = 1, ..., n)$.

Sei e eine Entität der Diskurswelt und $e.A_j$ der Wert von e im Attribut A_j.

Sei g_j die relative Wichtigkeit bzgl. der Korrektheit von $t.A_j$ mit $g_j \in]0, 1]$. Mittels dieser, von einem Domänenexperten zu bestimmenden Gewichtung lässt sich modellieren, dass bestimmte Attribute (z. B. Geburtsjahr) eine größere Bedeutung für die Korrektheit eines Tupels haben als andere (z. B. Körpergröße).

Der Qualitätswert Q_{Korr} des Tupels t bzgl. der Entität e wird definiert als das gewichtete arithmetische Mittel der Qualitätswerte auf Attributebene:

$$Q_{Korr}(t, e) := \frac{\sum_{j=1}^{n} Q_{Korr}(t.A_j, e.A_j) g_j}{\sum_{j=1}^{n} g_j} \tag{6.2}$$

Relationenebene (mehrelementiger View)

Sei T eine (nicht-leere) Relation bzw. ein (mehrelementiger) View, der (per SQL) über einer Menge von Relationen definiert ist (inkl. ggf. Selektion, Projektion, Joins, Vereinigung, Schnitt und Differenz).

Sei E eine Menge von Entitäten der Diskurswelt, die von T repräsentiert wird.

Der Korrektheitswert von T bzgl. E wird definiert durch das arithmetische Mittel der Korrektheitswerte der Tupel $t_i \in T$ bzgl. den Entitäten $e_i \in E \quad (i = 1, ..., |T|)$:

$$Q_{Korr}(T, E) := \frac{\sum_{i=1}^{|T|} Q_{Korr}(t_i, e_i)}{|T|} \tag{6.3}$$

Datenbankebene

Sei D eine Datenbank, die durch eine disjunkte Überdeckung einer Menge von Relationen (bzw. mehrelementigen Views über Relationen) $T_k \in D \quad (k = 1, ..., p)$ definiert ist. Die Forderung nach Disjunktheit verhindert dabei, dass Relationeninhalte über verschiedene Sichten mehrfach in die Bewertung eingehen:

$$D := T_1 \cup T_2 \cup ... \cup T_p \quad \text{mit} \quad T_1 \cap T_2 \cap ... \cap T_p = \varnothing \tag{6.4}$$

Sei R der durch D modellierte Ausschnitt der Diskurswelt.

Der Korrektheitswert von D bzgl. R wird wiederum durch das arithmetische Mittel der Korrektheitswerte der Relationen (bzw. mehrelementigen Views über Relationen) in D bzgl. ihren korrespondierenden Entitätsmengen in R definiert:

$$Q_{Korr}(D, R) := \frac{\sum_{k=1}^{p} Q_{Korr}(T_k, E_k)}{p} \tag{6.5}$$

Messverfahren

Korrektheit kann i. Allg. nur stichprobenartig gemessen werden, da hierzu ein (üblicherweise nicht automatisierbarer) Abgleich des Informationssystemzustands mit dem Diskursweltzustand vorgenommen werden muss und dieser bei großen Datenvolumina nicht praktikabel ist. Bei hinreichend großem, repräsentativem Stichprobenumfang (entspricht $|T|$ in der o. a. Metrik auf Relationenebene) können von der Korrektheit von Stichprobenelementen Rückschlüsse auf die Korrektheit eines Gesamtdatenbestandes gezogen werden. Der Qualitätswert Q_{Korr} ist in diesem Fall also nur eine Schätzung.

Ein weiterer Ansatz zur Erkennung von Mängeln bzgl. Korrektheit besteht in der Anwendung von Konzepten der *Statistischen Prozesskontrolle (SPC)* [She31], die in der industriellen Fertigung seit mehreren Jahrzehnten etabliert sind. Die Kernidee dabei ist, zu neu erfassten oder neu angelieferten Daten attributspezifische statistische Kennzahlen (Mittelwert, Varianz, Attributwerthäufigkeiten etc.) zu berechnen und über die Zeit zu protokollieren. Die jeweils neu berechneten Kennzahlen werden dann mit Erfahrungswerten aus dem Gesamtdatenbestand (z. B. aus einem Data Warehouse) verglichen, um z. B. Übertragungsfehler oder quellenspezifische Fehler frühzeitig, also vor weiteren Verarbeitungsschritten, erkennen zu können. Bei Bedarf, d. h. wenn die Abweichung zwischen Istwert und

erwartetem Wert so groß ist, dass a priori festgelegte attributspezifische Schwellenwerte (sog. *Kontrollgrenzen*) über- bzw. unterschritten werden, kann der Administrator bzw. Datenverantwortliche zu einer Intervention aufgerufen werden, z. B. zur Neuanforderung eines offensichtlich mangelhaften Datenpakets von der zuständigen Quelle. Alternativ bzw. ergänzend kann SPC auch direkt auf den Datenquellen aufsetzen, um den Datenerfassungsprozess zu überwachen. Weitere Details zu SPC im DQM-Kontext finden sich in [HZ95, Hin00c, Hin01b].

6.1.2 Konsistenz

Zur Bewertung der Konsistenz von Daten ist i. Allg. domänenspezifisches Wissen erforderlich. Als Repräsentationsformen dieses Domänenwissens kommen insbesondere in Frage:

- *(Geschäfts-)Regeln*, z. B.
  „WENN lieferdatum $<$ bestelldatum, DANN Mangel mit Schweregrad 0.9“
- *Lookup-Relationen*, z. B. Bankleitzahlregister
- *Reguläre Ausdrücke*, z. B. für Artikelbezeichner
- *Kausale Netze* (zum probabilistischen Schließen, siehe z. B. [Nea90])
- Beliebige domänenspezifische Funktionen.

Im Folgenden wird vereinfachend angenommen, dass Domänenwissen regelbasiert repräsentiert ist. Konsistenz wird bewertet über die Anzahl und den jeweiligen Schweregrad von Inkonsistenzen, angezeigt durch Regelverletzungen (vgl. Integritätsbedingungen in DBMS). Problematisch sind dabei widersprüchliche Aussagen in der Wissensbasis und unsichere, d. h. mit Wahrscheinlichkeiten behaftete Zusammenhänge (vgl. z. B. MYCIN [BS84]). Derartige Aspekte werden im Folgenden ausgeklammert.

Attributwertebene

Sei w ein Attributwert.

Sei R eine n-elementige Menge von Konsistenzregeln, wobei $\forall r_j \in R \quad (j = 1, ..., n)$ gelte:

$$r_j(w) := \begin{cases} 0 & \text{falls } w \text{ genügt } r_j \\ 1 & \text{sonst.} \end{cases} \tag{6.6}$$

Sei g_j die relative Wichtigkeit von r_j mit $g_j \in]0, 1]$ (im Aktionsteil der r_j implementiert). g_j entspricht damit dem Schweregrad einer Verletzung von Regel r_j.

Der Qualitätswert Q_{Kon} von w wird definiert als:

$$Q_{Kon}(w) := \frac{1}{\sum_{j=1}^{n} r_j(w) g_j + 1} \tag{6.7}$$

An dieser Stelle werden keine Einschränkungen bzgl. der Herkunft der Konsistenzregeln gemacht. Es kann sich dabei sowohl um explizit definierte Geschäftsregeln als auch um zur Laufzeit im Rahmen eines Data Mining-Prozesses aufgestellte Regeln handeln. Die Regeln können neben klassischen „Hard Constraints“ auch „Soft Constraints“ umfassen, die nicht unbedingt erfüllt sein müssen.

Diese Unterscheidung lässt sich mit Hilfe der o. a. Schweregrade modellieren, indem für Hard Constraints Gewichte von 1 und für Soft Constraints Gewichte kleiner 1 vergeben werden. Die Constraint-Problematik ist in Wirklichkeit komplexer, als dieser einfache Ansatz vermuten lässt. Für eine eingehende Diskussion siehe z. B. [BM86, PT87, Hor88, Man90, GL95].

Tupelebene (einelementiger View)

Sei t ein Tupel (bzw. ein einelementiger View). Durch die Einbeziehung von Joins können auch datensatzübergreifende Konsistenzregeln berücksichtigt werden.

Seien $R, r_j \in R$ und $g_j \quad (j = 1, ..., n)$ analog zu oben definiert.

Der Qualitätswert Q_{Kon} von t wird analog zu oben definiert als:

$$Q_{Kon}(t) := \frac{1}{\sum_{j=1}^{n} r_j(t) g_j + 1} \tag{6.8}$$

Es wird nicht unterschieden, ob eine Regel nur *einmal* von einem Tupel verletzt wird oder ggf. *mehrfach*. Die Auswirkung dieser Vereinfachung dürfte in der Praxis vernachlässigbar sein, da jede Regel i. Allg. genau einen Typ von Inkonsistenz – typischerweise bezogen auf ein einzelnes Attribut oder eine einzelne Attributkombination – beschreibt. Folglich werden bei zwei unterschiedlichen Inkonsistenzen typischerweise auch zwei unterschiedliche Regeln „feuern“ und damit in den Qualitätswert eingehen.

Relationenebene (mehrelementiger View)

Sei T eine (nicht-leere) Relation (oder ein View über mehrere Relationen).

Der Konsistenzwert von T wird definiert durch das arithmetische Mittel der Konsistenzwerte der Tupel $t_i \in T \quad (i = 1, ..., |T|)$:

$$Q_{Kon}(T) := \frac{\sum_{i=1}^{|T|} Q_{Kon}(t_i)}{|T|} \tag{6.9}$$

Datenbankebene

Sei D eine Datenbank, die wie in Abschnitt 6.1.1 definiert ist.

Der Konsistenzwert von D wird wiederum durch das arithmetische Mittel der Konsistenzwerte der Relationen (bzw. mehrelementigen Views über Relationen) definiert:

$$Q_{Kon}(D) := \frac{\sum_{k=1}^{p} Q_{Kon}(T_k)}{p} \tag{6.10}$$

Messverfahren

Konsistenz auf Attribut- oder Tupelebene kann gemessen werden, indem die jeweiligen Datenwerte daraufhin geprüft werden, inwieweit sie einem Satz domänenspezifischer Aussagen genügen. Diese sollten so (formal) repräsentiert sein, dass sie softwaretechnisch verarbeitet werden können. Bei einer regelbasierten Repräsentation sollte im Aktionsteil einer Regel der Schweregrad einer Regelverletzung quantifiziert und ggf. ein Hinweis auf die jeweils angemessene Reaktion (korrigieren, ignorieren, aussondern, neu einstufen etc.) gegeben werden. Über die Schweregrade und die Anzahl von Regelverletzungen ist eine ausgewogene Einschätzung des Konsistenzgrades eines Datenprodukts möglich.

Bei der Berechnung der $r_j(w)$ für einen gegebenen Wert w durch ein regelverarbeitendes System wird überprüft, welche Regeln w erfüllt und welche nicht. Mit den Ergebnissen dieser Prüfung kann dann $Q_{Kon}(w)$ auf einfache Weise berechnet werden. Die Gewichte g_j müssen von einem Domänenexperten einmalig für jede Regel spezifiziert und als Metadaten gespeichert bzw. in die Aktionsteile der Regeln eingetragen werden.

6.1.3 Vollständigkeit

Vollständigkeit wird bewertet über das (Nicht-)Vorhandensein eines semantisch von „unbekannt“ (*NULL*) abweichenden Wertes pro Tupelattribut.

Attributwertebene

Sei w ein Attributwert.

Sei $NotNull$ eine Funktion mit:

$$NotNull(w) := \begin{cases} 0 & \text{falls } w = \textit{NULL} \text{ oder } w \text{ zu } \textit{NULL} \text{ äquivalent} \\ 1 & \text{sonst.} \end{cases} \tag{6.11}$$

Dann gelte:

$$Q_{Voll}(w) := NotNull(w) \tag{6.12}$$

Bisher unberücksichtigt ist das Problem, dass ein Wert w mit $w = \textit{NULL}$ auch in der Diskurswelt nicht existiert (z. B. Name des Ehepartners bei Ledigen) bzw. dass nicht bekannt ist, ob ein solcher Wert existiert (vgl. Diskussion zu Nullwertsemantiken in Abschnitt 2.2.3 und in [Atz00]). Im erstgenannten Fall liegt kein Qualitätsdefizit bzgl. Vollständigkeit vor. Im zweitgenannten Fall kann keine Qualitätsaussage getroffen werden.

Tupelebene (einelementiger View)

Sei t ein Tupel (bzw. ein einelementiger View) mit Attributwerten $t.A_1, ..., t.A_n$ für Attribute $A_1, ..., A_n$.

Sei g_j die (von einem Domänenexperten zu bestimmende) relative Wichtigkeit von A_j im Hinblick auf Vollständigkeit mit $g_j \in]0, 1]$. Relative Wichtigkeiten sind als Parameter der Metrik zu verstehen, die in Abhängigkeit eines konkreten Anwendungsfalls zu spezifizieren sind.

Dann gelte:

$$Q_{Voll}(t) := \frac{\sum_{j=1}^{n} Q_{Voll}(t.A_j) g_j}{\sum_{j=1}^{n} g_j} \tag{6.13}$$

Relationenebene (mehrelementiger View)

$Q_{Voll}(T)$ wird analog zu $Q_{Kon}(T)$ definiert (siehe Abschnitt 6.1.2).

Datenbankebene

$Q_{Voll}(D)$ wird analog zu $Q_{Kon}(D)$ definiert (siehe Abschnitt 6.1.2).

Messverfahren

Die Umsetzung der Messung zum Merkmal „Vollständigkeit“ auf Attributwertebene ergibt sich unmittelbar aus der entsprechenden Metrik (unter Verwendung von SQL-Abfragen).

6.1.4 Redundanzfreiheit

Zur Bewertung der Redundanzfreiheit wird das Verhältnis der Anzahl der in einem Datenprodukt (auf Relationenebene) repräsentierten Entitäten der Diskurswelt zur Gesamtzahl der Tupel in dem Datenprodukt herangezogen.

Relationenebene

Sei E die Extension eines Entitätstyps der Diskurswelt und T eine Relation oder eine Sicht auf mehrere Relationen ($|T| > 0$), die E in einem Informationssystem beschreibt. Die Einbeziehung des Sichtenkonzeptes erlaubt es, auch solche Entitäten zu behandeln, deren Repräsentation auf mehrere Relationen verteilt ist.

Sei $Rep(T, E)$ die Anzahl derjenigen Elemente in E, für die (mindestens) ein korrespondierendes Tupel in T existiert.

Sei $Dup(T, E)$ die Anzahl der Duplikate in T bzgl. E, also:

$$Dup(T, E) := |T| - Rep(T, E) \tag{6.14}$$

Der Qualitätswert $Q_{Red}(T, E)$ für die Redundanzfreiheit von T bzgl. E wird nun wie folgt definiert:

$$Q_{Red}(T, E) := \frac{Rep(T, E)}{|T|} = 1 - \frac{Dup(T, E)}{|T|} \tag{6.15}$$

Die rechte Seite der Gleichung ergibt sich durch Einsetzen von Formel 6.14. Aus $|T| > Dup(T, E) \geq 0$ folgt $0 < Q_{Red}(T, E) \leq 1$. Der Qualitätswert $Q_{Red}(T, E)$ ist streng monoton steigend bei sinkender Anzahl Duplikate.

$|T|$ kann einfach per SQL-Anfrage berechnet werden, $Dup(T, E)$ hingegen ist i. Allg. eine unbekannte Größe. Sie kann jedoch mit Hilfe von Record Linkage-Verfahren geschätzt werden (siehe unten).

Datenbankebene

Sei D eine Datenbank, die wie in Abschnitt 6.1.1 definiert ist.

Sei g_k die relative Gewichtung der Relation $T_k \in D \quad (k = 1, ..., p)$ bzgl. Redundanzfreiheit mit $g_k \in]0, 1]$ (anwendungsspezifischer Parameter der Metrik). Mittels der g_k lässt sich modellieren, dass bestimmte Relationen einen größeren Einfluss auf die Bewertung der Redundanzfreiheit einer Datenbank haben als andere. Beispielsweise könnte die Redundanzfreiheit einer Relation *ARTIKEL* stärker gewichtet werden (Gewicht 0.9) als die einer Relation *KUNDEN* (Gewicht 0.7), um auszudrücken, dass in *KUNDEN* eher Duplikate geduldet werden als in *ARTIKEL*. Zur Vergabe von Gewichten ist grundsätzlich ein konkreter Anwendungskontext erforderlich.

Der Qualitätswert $Q_{Red}(D)$ der Redundanzfreiheit von D wird definiert durch das gewichtete arithmetische Mittel der Qualitätswerte $Q_{Red}(T_k, E_k)$ der in D enthaltenen Relationen (bzw. mehrelementigen Sichten auf Relationen) T_k bzgl. ihrer Extensionen E_k:

$$Q_{Red}(D) := \frac{\sum_{k=1}^{p} Q_{Red}(T_k, E_k) g_k}{\sum_{k=1}^{p} g_k} \tag{6.16}$$

Auch $Q_{Red}(D)$ ist normiert auf $]0, 1]$. Dies folgt aus $g_k > 0$ und der Normierung von $Q_{Red}(T_k, E_k)$. Bei steigender Qualität auf Relationenebene verhält sich $Q_{Red}(D)$ ebenfalls streng monoton steigend.

Messverfahren

Redundanzfreiheit (auf Relationenebene) kann nur gemessen werden, wenn die Anzahl der in einer Relation repräsentierten Entitäten der Diskurswelt und damit die Anzahl der Duplikate ($Dup(T, E)$ im obigen Abschnitt) in der Relation abschätzbar ist. Im Gegensatz zum Merkmal „Korrektheit“ wird hierzu üblicherweise nicht direkt auf die Diskurswelt zurückgegriffen. Stattdessen wird die Abschätzung durch Attributwertvergleiche im Informationssystem vorgenommen, und zwar mit Hilfe sog. *Record Linkage-Verfahren*. Es sind verschiedene Klassen von Record Linkage-Verfahren zu unterscheiden: Im Idealfall kann eine Zuordnung zweier Tupel an Hand ihres übereinstimmenden (globalen) Primärschlüssels erfolgen (sog. *schlüsselbasiertes Linkage*). In der Praxis ist dies jedoch nur in Ausnahmefällen möglich, da häufig – insbesondere, wenn die Daten aus heterogenen Quellen stammen – keine globalen Primärschlüssel vorliegen. Eine Tupelzuordnung kann dann nur noch auf der Basis einer Analyse der Ähnlichkeit von Nicht-Schlüssel-Attributwerten erfolgen. In der Literatur finden sich folgende Klassen von ähnlichkeitsbasierten Record Linkage-Verfahren:

- Probabilistisches Record Linkage [FS69, Jar89, Win99] (siehe auch Abschnitt 6.2)
- Duplicate Elimination Sorted-Neighbourhood Method [HS95]
- Neighbourhood Hash Joins [GFS+01].

Von diesen Verfahren hat das probabilistische Record Linkage (für Details siehe Abschnitt 6.2) in den letzten Jahren die größte Verbreitung erlangt. Es handelt sich dabei um ein abweichungstolerantes Verfahren, bei dem in Abhängigkeit von der Ähnlichkeit der Attributwerte zweier Tupel Gewichte vergeben werden. Die Summe der Attributgewichte pro Tupelpaar korreliert dann mit der Wahrscheinlichkeit, dass beide Tupel dieselbe Diskursweltentität beschreiben und damit einander zuzuordnen sind. Die Anwendung eines Record Linkage-Verfahrens resultiert also in einer Menge von Zuordnungen zwischen Tupeln. An Hand dieser Menge kann der Anteil der Duplikate am Gesamtdatenbestand einer Relation und damit der Qualitätsgrad bzgl. Redundanzfreiheit geschätzt werden.

6.1.5 Genauigkeit

Das Qualitätsmerkmal „Genauigkeit“ wird entscheidend durch die sog. *Stelligkeit* von Attributwerten beeinflusst. Mit Stelligkeit ist hier nicht nur die Anzahl der Nachkommastellen (bei numerischen Attributen) gemeint, sondern auch die Position eines Wertes in einer gegebenen Klassifikationshierarchie (bei symbolischen Attributen): Je weiter ein Wert von der Wurzel der Hierarchie entfernt ist, desto höher ist seine Stelligkeit (siehe Beispiel in Abb. 6.1). Die Problematik von unbalancierten („schiefen“) Baumstrukturen wird an dieser Stelle nicht betrachtet. Attributen, bei denen man nicht nach Genauigkeit differenzieren kann (z. B. Fließtextattribute), ist die Default-Stelligkeit 1 zugeordnet. Praxisnahe Beispiele zur Stelligkeit werden in Abschnitt 11.2 gegeben.

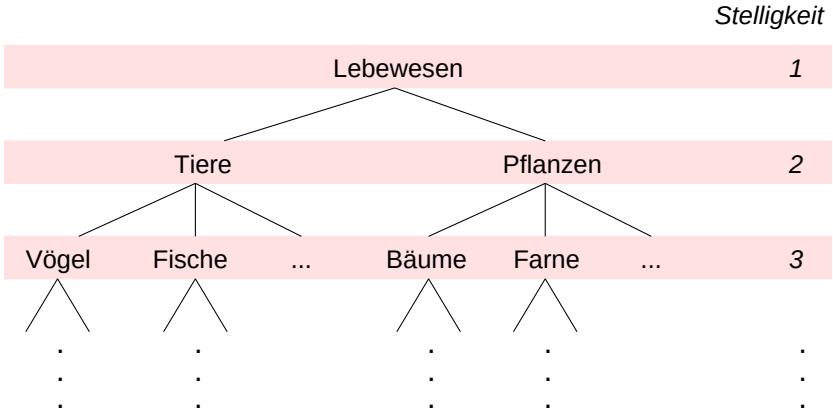


Abbildung 6.1: Stelligkeiten in einer Klassifikationshierarchie

Attributwertebene

Die Genauigkeit eines Attributwerts wird nun bewertet über das Verhältnis seiner Stelligkeit zur jeweils „optimalen“ Stelligkeit. Letztere ist durch die Anforderungen des jeweiligen Anwendungskontextes gegeben und als anwendungsspezifischer Parameter der Metrik vorzuhalten (siehe z. B. Abschnitt 11.2). Da „optimal“ nicht unbedingt „maximal“ bedeutet, kann der Quotient größer als 1 sein und muss daher auf das Intervall $]0, 1]$ normiert werden.

Sei A ein numerisches Attribut (symbolische Attribute werden im nachfolgenden Absatz behandelt), $s_{opt}(A)$ die „optimale“ Anzahl von (Nachkomma-)Stellen für A und w ein Attributwert von A. Sei $s : Dom(A) \mapsto \mathbb{N}$ eine Funktion, die die Stelligkeit von Werten $w \in Dom(A)$ zurückliefert. Dann gelte für die Genauigkeit von w im Attribut A:

$$Q_{Gen}(w, A) := min\left(\frac{s(w)}{s_{opt}(A)}, 1\right) \tag{6.17}$$

Falls A symbolisch ist (gilt z. B. auch für numerische Codierungen wie 0/1 als Geschlechtsangabe) und sich w in Ebene i einer Klassifikation K (attributbezogenes Metadatum) mit n Ebenen $(K_1, ..., K_n)$ einordnen lässt (also auf Ebene K_i), gelte $s_{opt}(A) \leq n$ und $s(w) = i$ (vgl. Abb. 6.1).

Existiert für ein symbolisches Attribut A keine solche Klassifikationshierarchie (z. B. bei Freitextfeldern), gelte $s_{opt}(A) = 1$ und $s(w) = 1$ (nur eine Hierarchieebene und damit mangels Alternativen immer „optimale“ Genauigkeit).

Tupelebene (einelementiger View)

Sei t ein Tupel (bzw. ein einelementiger View) mit Attributwerten $t.A_1, ..., t.A_n$ für Attribute $A_1, ..., A_n$.

Sei g_j die (von einem Domänenexperten zu bestimmende) relative Wichtigkeit von A_j im Hinblick auf Genauigkeit mit $g_j \in]0, 1]$ (anwendungsspezifischer Parameter der Metrik).

Dann gelte:

$$Q_{Gen}(t) := \frac{\sum_{j=1}^{n} Q_{Gen}(t.A_j, A_j) g_j}{\sum_{j=1}^{n} g_j} \tag{6.18}$$

Relationenebene (mehrelementiger View)

$Q_{Gen}(T)$ wird analog zu $Q_{Kon}(T)$ definiert (siehe Abschnitt 6.1.2).

Datenbankebene

$Q_{Gen}(D)$ wird analog zu $Q_{Kon}(D)$ definiert (siehe Abschnitt 6.1.2).

Messverfahren

Die Umsetzung der Messung zum Merkmal „Genauigkeit“ auf Attributwertebene ergibt sich unmittelbar aus der entsprechenden Metrik (unter Verwendung von SQL-Abfragen).

6.1.6 Zeitnähe

Zeitnähe wird über das sog. *Befunddatum* von Attributwerten bzw. Tupeln im Verhältnis zur Update-Häufigkeit der beteiligten Attribute bewertet. Als Befunddatum wird hierbei der Zeitpunkt der Ermittlung einer Eigenschaft der Diskurswelt bezeichnet. Dieser muss nicht zwangsläufig mit dem Zeitpunkt der Aufnahme in eine Datenbank (Transaktionszeit) übereinstimmen. An dieser Stelle wird vereinfachend davon ausgegangen, dass das Befunddatum bekannt ist und explizit als Metadatum mitgeliefert wird. Sollte diese Angabe in der Praxis dennoch für einzelne Attributwerte oder Tupel fehlen, sind die betroffenen Daten von der Messung der Zeitnähe auszuklammern (Abfangen von Nullwerten im Metaattribut erforderlich). Durch die Einbeziehung der Update-Häufigkeit in die Beurteilung der Zeitnähe kann die Dynamik von Wertänderungen angemessen berücksichtigt werden. Für weitere Ausführungen zu diesem Aspekt siehe auch [TB99].

Attributwertebene

Sei A ein Attribut und w ein Attributwert in A.

Sei $Age(w)$ das Alter von w (Differenz von aktuellem Zeitpunkt und Befunddatum, bei multinationalen Organisationen ggf. unter Berücksichtigung von Zeitzonen) und $Upd(A)$ die Update-Häufigkeit von Werten in A.

Sei u die Zeiteinheit, in der das Alter und die Update-Häufigkeit angegeben werden (z. B. ms, min, Tage, Monate, je nach Domäne).

Dann gelte für die Zeitnähe von w im Attribut A:

$$Q_{Zeit}(w, A) := \frac{1}{\frac{Upd(A)}{u} Age(w) u + 1} = \frac{1}{Upd(A) Age(w) + 1} \tag{6.19}$$

Bei Attributen, deren Werte sich in der betrachteten Diskurswelt niemals ändern (z. B. Vorname, Geburtsdatum etc.), ist $Upd(A) = 0$. Damit gilt unabhängig vom Alter eines Attributwerts w immer $Q_{Zeit}(w, A) = 1$. Eine Aktualisierung eines Attributs mit $Upd(A) = 0$ wird grundsätzlich als Korrektur einer vormals mangelhaften Angabe (z. B. Zahlendreher im Geburtsdatum) gewertet. Je höher die Update-Häufigkeit ist, desto schneller altern Attributwerte (z. B. Börsenkurse).

Tupelebene (einelementiger View)

Sei t ein Tupel (bzw. ein einelementiger View) mit Attributwerten $t.A_1, ..., t.A_n$ für Attribute $A_1, ..., A_n$.

Sei g_j die relative Wichtigkeit von A_j im Hinblick auf die Zeitnähe mit $g_j \in]0, 1]$ (anwendungsspezifischer Parameter der Metrik).

Dann gelte:

$$Q_{Zeit}(t) := \frac{\sum_{j=1}^{n} Q_{Zeit}(t.A_j, A_j) g_j}{\sum_{j=1}^{n} g_j} \tag{6.20}$$

Für den Fall, dass das Befunddatum nur tupelbezogen als Metadatum mitgeliefert wird und folglich das Alter nur auf Tupelebene berechnet werden kann (also $Age(t)$ für ein Tupel t), wird die Update-Häufigkeit von t (mit Attributen $A_1, ..., A_n$) als maximale Update-Häufigkeit der beteiligten Attribute definiert:

$$Upd(t) := max_{j=1}^{n} \left(Upd(A_j) \right) \tag{6.21}$$

Für die Zeitnähe von t gilt dann:

$$Q_{Zeit}(t) := \frac{1}{Upd(t) Age(t) + 1} \tag{6.22}$$

Im Falle von asynchronen Updates wird bei dieser Lösung die tatsächliche Update-Häufigkeit unterschätzt. Diese Problematik wird hier nicht näher betrachtet.

Relationenebene (mehrelementiger View)

$Q_{Zeit}(T)$ wird analog zu $Q_{Kon}(T)$ definiert (siehe Abschnitt 6.1.2).

Datenbankebene

$Q_{Zeit}(D)$ wird analog zu $Q_{Kon}(D)$ definiert (siehe Abschnitt 6.1.2).

Messverfahren

Die Umsetzung der Messung zum Merkmal „Zeitnähe“ auf Attributwertebene ergibt sich unmittelbar aus der entsprechenden Metrik (unter Verwendung von SQL-Abfragen).

6.1.7 Eindeutigkeit

Die Eindeutigkeit von Daten bzgl. ihrer Semantik bzw. Interpretierbarkeit wurde in Abschnitt 3.2.3 über die Qualität der Metadaten, die diese Daten beschreiben, definiert. Dabei sind sowohl attributbezogene (z. B. Maßeinheit) als auch datenbezogene (z. B. Befunddatum) Metadaten zu berücksichtigen. Es werden keine Qualitätsgrade außer 1 und 0 unterschieden, d. h. entweder ist ein Datum eindeutig (Qualitätswert 1) oder nicht (Qualitätswert 0). In einer verfeinerten Lösung könnten ggf. auch Abstufungen mittels Certainty Factors in Betracht gezogen werden.

Es ist nun zu entscheiden, welche Datenqualitätsmerkmale auf Metadatenebene die Eindeutigkeit von Daten beeinflussen. Folgende Feststellungen können getroffen werden:

- Die Korrektheit, Zuverlässigkeit und Zeitnähe von Metadaten haben – laut der in Abschnitt 3.2.3 gegebenen Definition – keinen Einfluss auf die Eindeutigkeit der beschriebenen Daten.
- Defizite in Metadaten bzgl. Redundanz, Relevanz, Einheitlichkeit und Verständlichkeit werden in dem hier verfolgten Ansatz a priori ausgeschlossen.
- Eine Metrik zur Bewertung der Eindeutigkeit von Daten an Hand der Eindeutigkeit von Metadaten würde eine zyklische Abhängigkeit implizieren und wird daher ausgeklammert.

Es verbleiben folgende Datenqualitätsmerkmale, die im Folgenden – auf Metadaten angewandt – zur Bewertung der Eindeutigkeit von Daten herangezogen werden sollen:

- Konsistenz
- Vollständigkeit
- Genauigkeit.

Attributwertebene

Sei A ein Attribut und w ein Attributwert in A. Metadaten zu A seien in einem Tupel $t_M(A)$, Metadaten zu w in einem Tupel $t_M(w)$ zusammengefasst.

Seien $Q_{Kon}(t_M(A))$, $Q_{Voll}(t_M(A))$ und $Q_{Gen}(t_M(A))$ (siehe Abschnitte 6.1.2, 6.1.3 und 6.1.5) die Qualitätswerte von $t_M(A)$ in den die Eindeutigkeit beeinflussenden Merkmalen (siehe oben). Analog ergeben sich die Qualitätswerte zu $t_M(w)$.

Da – wie oben beschrieben – keine graduelle Eindeutigkeit zugelassen ist, brauchen keine relativen Wichtigkeiten der Qualitätswerte von Metadaten unterschieden zu werden. Die Metadatenqualität $Q_{M,Eind}(w, A)$ eines Wertes w im Attribut A bzgl. Eindeutigkeit wird daher als einfaches arithmetisches Mittel – ohne Gewichtung der einzelnen Qualitätswerte – definiert. Der Nenner ergibt sich dabei aus der Anzahl der Summanden (2 Metadatentupel werden auf je 3 Qualitätsmerkmale untersucht):

$$Q_{M,Eind}(w, A) := \frac{\sum_{dim=Kon,Voll,Gen}^{arg=t_M(A),t_M(w)} Q_{dim}(arg)}{2 \cdot 3} \tag{6.23}$$

Daraus ergibt sich für die Eindeutigkeit eines Werts w im Attribut A:

$$Q_{Eind}(w, A) := \begin{cases} 0 & \text{falls } Q_{M,Eind}(w, A) < 1 \\ 1 & \text{sonst.} \end{cases} \tag{6.24}$$

Tupelebene (einelementiger View)

Sei t ein Tupel (bzw. ein einelementiger View) mit Attributwerten $t.A_1, ..., t.A_n$ für Attribute $A_1, ..., A_n$.

Dann gelte:

$$Q_{Eind}(t) := \begin{cases} 0 & \text{falls } \frac{\sum_{j=1}^{n} Q_{Eind}(t.A_j, A_j)}{n} < 1 \\ 1 & \text{sonst.} \end{cases} \tag{6.25}$$

Tupelbezogene Metadaten (z. B. $Age(t)$) werden bisher nicht explizit unterstützt.

Relationenebene (mehrelementiger View)

Sei T eine Relation (oder ein View über mehrere Relationen) mit Tupeln $t_i \in T \quad (i = 1, ..., |T|)$.

Dann gelte:

$$Q_{Eind}(T) := \begin{cases} 0 & \text{falls } \frac{\sum_{i=1}^{|T|} Q_{Eind}(t_i)}{|T|} < 1 \\ 1 & \text{sonst.} \end{cases} \tag{6.26}$$

Datenbankebene

Sei D eine Datenbank mit Relationen $T_k \in D \quad (k = 1, ..., p)$.

Dann gelte:

$$Q_{Eind}(D) := \begin{cases} 0 & \text{falls } \frac{\sum_{k=1}^{p} Q_{Eind}(T_k)}{p} < 1 \\ 1 & \text{sonst.} \end{cases} \tag{6.27}$$

Messverfahren

Zur Messung der Eindeutigkeit kann auf die Messverfahren zu den beteiligten Datenqualitätsmerkmalen Konsistenz, Vollständigkeit und Genauigkeit zurückgegriffen werden.

6.1.8 Schlüsseleindeutigkeit

Schlüsseleindeutigkeit wird bewertet über das Verhältnis der Anzahl der in einem Datenprodukt (auf Relationenebene) vorkommenden unterschiedlichen Primärschlüssel zur Gesamtzahl der Tupel in dem Datenprodukt.

Relationenebene

Sei T eine Relation (oder eine Sicht auf mehrere Relationen) mit $|T| > 0$. O. B. d. A. sei P das Primärschlüsselattribut von T. Zusammengesetzte Primärschlüssel können analog behandelt werden.

Sei $M_{distinct}(T, P)$ die Menge der in T vorkommenden unterschiedlichen Ausprägungen in P.

Der Qualitätswert $Q_{Schl}(T, P)$ für die Schlüsseleindeutigkeit von T bzgl. P sei nun wie folgt definiert:

$$Q_{Schl}(T, P) := \frac{|M_{distinct}(T, P)|}{|T|} \tag{6.28}$$

Aus $|T| \geq |M_{distinct}(T, P)| > 0$ folgt $0 < Q_{Schl}(T, P) \leq 1$. $Q_{Schl}(T, P)$ ist streng monoton steigend bei sinkender Anzahl von Duplikaten im Primärschlüsselattribut.

Datenbankebene

Sei D eine Datenbank, die wie in Abschnitt 6.1.1 definiert ist.

Sei g_k die relative Gewichtung der Relation $T_k \in D \quad (k = 1, ..., p)$ bzgl. Schlüsseleindeutigkeit mit $g_k \in]0, 1]$ (anwendungsspezifischer Parameter der Metrik).

Der Qualitätswert $Q_{Schl}(D)$ der Schlüsseleindeutigkeit von D wird definiert durch das gewichtete arithmetische Mittel der Qualitätswerte $Q_{Schl}(T_k, P_k)$ der in D enthaltenen Relationen (bzw. mehrelementigen Sichten auf Relationen) T_k bzgl. ihrer Primärschlüsselattribute P_k. Wiederum wird von zusammengesetzten Primärschlüsseln abstrahiert.

$$Q_{Schl}(D) := \frac{\sum_{k=1}^{p} Q_{Schl}(T_k, P_k) g_k}{\sum_{k=1}^{p} g_k} \tag{6.29}$$

Die Normierung von $Q_{Schl}(D)$ auf $]0, 1]$ folgt aus $g_k > 0$ und der Normierung von $Q_{Schl}(T_k, P_k)$. Bei steigender Qualität bzgl. Schlüsseleindeutigkeit auf Relationenebene verhält sich $Q_{Schl}(D)$ ebenfalls streng monoton steigend.

Messverfahren

Schlüsseleindeutigkeit auf Relationenebene kann mittels zweier SQL-Abfragen (`SELECT COUNT(*) FROM ...` und `SELECT COUNT(DISTINCT(PK)) FROM ...` mit Primärschlüsselattribut `PK`) und anschließender Quotientenbildung der Abfrageergebnisse gemessen werden.

6.1.9 Referentielle Integrität

Referentielle Integrität wird bewertet über die Anzahl der Verstöße gegen die im Schema spezifizierten Beziehungen zwischen Relationen.

Relationenebene

Sei T eine Relation (oder eine Sicht auf mehrere Relationen) mit $|T| > 0$. O. B. d. A. seien $F_1, ..., F_m$ die Fremdschlüsselattribute in T und $T(F_i)$ die durch F_i $(i = 1, ..., m)$ referenzierte Relation mit Primärschlüsselattribut P. Zusammengesetzte Fremd- bzw. Primärschlüssel können analog behandelt werden.

Sei $num_{dangling}(T)$ die Anzahl derjenigen Fremdschlüssel in T, zu denen kein Primärschlüssel in der referenzierten Relation existiert, formal:

$$num_{dangling}(T) := \sum_{i=1}^{m} \left| \{t \in T | \nexists t_{ref} \in T(F_i) : t_{ref}.P = t.F_i\} \right| \tag{6.30}$$

Sei weiterhin $mult(T, F_i)$ die Multiplizität der Beziehung zwischen T und $T(F_i)$ mit $mult(T, F_i) \in$ {0..1:1, 1:1, 1..*:1, *:1} (UML-Notation, siehe [Obj01]).

Sei nun $num_{mult_viol}(T, F_i)$ die Anzahl der Multiplizitätsverletzungen in T bzgl. F_i, genauer:

$$num_{mult_viol}(T, F_i) :=$$

$$\begin{cases} 1 & \text{falls} \quad (mult(T, F_i) = 0..1{:}1 \ \wedge \ \exists t_{ref} \in T(F_i) : |\{t \in T | t.F_i = t_{ref}.P\}| > 1) \\ & \quad \vee \quad (mult(T, F_i) = 1{:}1 \quad \wedge \ \exists t_{ref} \in T(F_i) : |\{t \in T | t.F_i = t_{ref}.P\}| \neq 1) \\ & \quad \vee \quad (mult(T, F_i) = 1..*{:}1 \ \wedge \ \exists t_{ref} \in T(F_i) : |\{t \in T | t.F_i = t_{ref}.P\}| < 1) \\ 0 & \text{sonst.} \end{cases} \tag{6.31}$$

Es werden keine unterschiedlichen „Verletzungsgrade“ unterschieden, d. h. entweder liegt eine Verletzung bzgl. F_i vor ($num_{mult_viol}(T, F_i) = 1$) oder nicht ($num_{mult_viol}(T, F_i) = 0$). Im Falle von $mult(T, F_i) =$ *:1 können keine Multiplizitätsverletzungen auftreten.

Der Qualitätswert $Q_{Ref}(T)$ für die referentielle Integrität von T wird nun wie folgt definiert:

$$Q_{Ref}(T) := \frac{1}{num_{dangling}(T) + \sum_{i=1}^{m} num_{mult_viol}(T, F_i) + 1} \tag{6.32}$$

Diese Metrik bildet auf das Intervall $]0, 1]$ ab und stellt sicher, dass der Qualitätswert umso kleiner ist, je größer die Anzahl der Verstöße ist. Eine Verfeinerung der Metrik könnte die Anzahl der Verstöße in Bezug zur Anzahl der Tupel in T setzen.

Datenbankebene

$Q_{Ref}(D)$ wird analog zu $Q_{Schl}(D)$ definiert (siehe Abschnitt 6.1.8).

Messverfahren

Die Umsetzung der Messung zum Merkmal „Referentielle Integrität“ auf Relationenebene ergibt sich unmittelbar aus der entsprechenden Metrik (unter Verwendung von SQL-Abfragen).

6.2 Verbesserungsmaßnahmen nach Datenqualitätsmerkmalen

Im Folgenden werden die in Abschnitt 4.5.3 bereits kurz erwähnten Maßnahmen zur (symptomorientierten) Verbesserung von Datenqualität, klassifiziert nach Datenqualitätsmerkmalen, im Detail betrachtet.

Symptomorientierte Verbesserungsmaßnahmen sind auf wenige Datenqualitätsmerkmale beschränkt. Es ist z. B. offensichtlich, dass Daten nicht nachträglich relevanter oder aktueller gemacht werden können als sie sind. Derartige Qualitätsmängel können nur ursachenorientiert behoben werden (z. B. durch eine Prozessoptimierung). Für die folgenden Merkmale sind nachträgliche Verbesserungen möglich:

- Einheitlichkeit
- Korrektheit
- Konsistenz
- Vollständigkeit
- Redundanzfreiheit.

In den nachfolgenden Abschnitten werden mögliche Verbesserungsmaßnahmen bzgl. dieser Datenqualitätsmerkmale diskutiert.

6.2.1 Einheitlichkeit

Daten, die aus heterogenen Quellen stammen, unterscheiden sich üblicherweise in ihrer Repräsentation. Um eine Vergleichbarkeit solcher Daten zu gewährleisten, ist daher eine Vereinheitlichung der Repräsentationen erforderlich. Dazu gehören u. a. folgende Teilaufgaben (vgl. auch Abschnitt 2.2.3):

- Erzeugung von globalen Primärschlüsseln und ihre Verknüpfung mit lokalen Schlüsseln
- Syntaktische Vereinheitlichung von Zeichenketten (bzgl. Umlauten, White Spaces etc.)
- Semantische Vereinheitlichung von Zeichenketten (z. B. bei Synonymen)
- Dekomposition komplexer in atomare Attributwerte (z. B. Adressangaben)
- Aggregierung atomarer zu komplexen Attributwerten (z. B. Datumsangaben)
- Konvertierung von Codierungen (z. B. für Geschlecht oder Nationalität)
- Konvertierung von Maßeinheiten und Skalierungen (z. B. inch in cm).

Letztlich handelt es sich bei den genannten Aufgaben um Abbildungen von Quellattributwerten auf Zielattributwerte. Zur Spezifizierung solcher Abbildungen und zur eigentlichen Durchführung der Vereinheitlichung eignen sich sog. *ETL-Werkzeuge* (siehe Anhang A). Zur Erledigung ihrer Aufgabe nutzen diese Werkzeuge i. Allg. kein Domänenwissen, d. h. der Anwender (i. Allg. ein DWS-Administrator oder Datenverantwortlicher) ist dafür verantwortlich, die Abbildungen domänengerecht zu spezifizieren.

6.2.2 Korrektheit

Zur Verbesserung der Korrektheit von Daten muss auf die Diskurswelt zurückgegriffen werden, um den tatsächlichen Zustand bzw. Attributwert zu bestimmen, durch den ein als fehlerhaft erkannter Wert im Informationssystem zu ersetzen ist. Da dieser Schritt aus dem Informationssystem heraus i. Allg. nicht automatisierbar ist (die Anbindung von Messgeräten ausgenommen) und gleichzeitig

die Datenvolumina in modernen Informationssystemen sehr groß sein können, folgt, dass Verbesserungen der Korrektheit von Datenprodukten nur stichprobenartig erfolgen können. Wird per Stichprobe eine Inkorrektheit festgestellt (Abweichung zwischen Zustand in der Diskurswelt und Zustand im Informationssystem), so kann der abweichende Wert der Diskurswelt direkt ins Informationssystem übernommen werden, um den Korrektheitsgrad zumindest punktuell zu erhöhen. Gelingt es, von Qualitätsmängeln in einer Stichprobe auf Qualitätsmängel in anderen, assoziierten Teilen eines Datenbestands zu schließen (z. B. in anderen Tupeln aus derselben Datenquelle bzw. Datenlieferung), ist eine weitergehende Qualitätsverbesserung möglich. Gleiches gilt, wenn der Suchraum durch eine Klassifizierung von Stichproben, die Qualitätsmängel aufweisen, auf „verdächtige“ Teilmengen eingeschränkt werden kann.

Zur Qualitätsverbesserung von verrauschten Daten wurden im Rahmen der Arbeit Experimente mit *Kalmanfiltern* [Bro83] durchgeführt, auf die jedoch nicht näher eingegangen wird.

6.2.3 Konsistenz

Inkonsistenzen können zwar – angemessen repräsentiertes Domänenwissen vorausgesetzt – automatisiert erkannt werden, eine automatisierte Beseitigung von Inkonsistenzen ist jedoch nur in wenigen Ausnahmefällen möglich, z. B. bei Verwendung von Wörterbüchern, Namensregistern etc. zur Korrektur offensichtlicher Tippfehler. In derartigen Fällen kann die korrigierende Maßnahme (üblicherweise Ersetzung eines Attributwerts) in den Aktionsteil der jeweils verletzten Konsistenzregel (vgl. Abschnitt 6.1.2) aufgenommen werden. Dies entspricht den klassischen Stored Procedures-/Trigger-Konzepten herkömmlicher DBMS.

Nicht-triviale Inkonsistenzen können i. Allg. nur behoben werden, indem (ggf. automatisiert erzeugte) Rückfragen an die betroffenen Datenquellen geschickt werden. Diese sollten intern den Inkonsistenzen nachgehen und anschließend entweder eine Neulieferung der korrigierten Daten vornehmen oder Korrekturanweisungen zurückschicken, an Hand derer eine Nachbearbeitung der inkonsistenten Datenprodukte vorgenommen werden kann. Je nach Repräsentationsform und Übertragungsmedium kann die Nachbearbeitung manuell oder automatisiert erfolgen. Insbesondere bei externen Quellen dürfte sich die Abwicklung von Rückfragen auf Grund der nicht etablierten Kommunikationswege schwierig gestalten.

Ein innovativer Ansatz zur teilautomatisierten Behebung von Inkonsistenzen besteht in der Anwendung von Data Mining-Verfahren (siehe Abschnitte 4.5.1 und 5.2).

6.2.4 Vollständigkeit

Zur Verbesserung der Vollständigkeit können – analog zur Konsistenz – wiederum Geschäftsregeln genutzt werden. Legt z. B. eine Regel fest, dass bei Wert x im Attribut A zwangsläufig Wert y im Attribut B vorliegen muss, so kann bei einer Unvollständigkeit in B automatisiert y nachgetragen werden, falls A den Wert x hat. Solche Regeln können entweder von einem Domänenexperten vorgegeben sein oder zur Laufzeit per Data Mining (siehe Abschnitt 4.5.1) explizit gemacht werden. Fehlende Angaben, die nicht regelbasiert ermittelt werden können, müssen – wie in Abschnitt 6.2.3 beschrieben – beim Datenlieferanten nachgefragt werden.

6.2.5 Redundanzfreiheit

Auch bzgl. dieses Merkmals sind Qualitätsmessung und -verbesserung eng miteinander verknüpft: Die vom Record Linkage-Verfahren (siehe Abschnitt 6.1 und untenstehenden Exkurs) als Ergebnis

gelieferte Menge von Zuordnungen zwischen Datensätzen kann nicht nur zur Schätzung des Qualitätsgrads bzgl. Redundanzfreiheit genutzt werden, sondern stellt gleichzeitig den Ausgangspunkt zur Qualitätsverbesserung dar. Dazu werden gefundene Zuordnungen konsolidiert, d. h. die einander zugeordneten Datensätze miteinander verschmolzen und damit Duplikate eliminiert. Beim Verschmelzen werden nach bestimmten Kriterien (Informationsgehalt, Quellenpriorität, Zeitnähe etc.) die jeweils „besten“ Informationen aus den beteiligten Datensätzen extrahiert und in einen Zieldatensatz eingefügt. Die Ausgangsdatensätze sind anschließend zu löschen.

Der folgende Exkurs beschäftigt sich näher mit dem Verfahren des probabilistischen Record Linkage nach [Jar89].

Exkurs: Probabilistisches Record Linkage

Formal lässt sich probabilistisches Record Linkage wie folgt beschreiben: Zu zwei Mengen S_1 und S_2 von Tupeln sind eine Menge $M \subset S_1 \times S_2$ aller Matches und eine (zu M komplementäre) Menge $U \subset S_1 \times S_2$ aller Non-Matches zu ermitteln. Dabei gilt $M \cup U = S_1 \times S_2$, $M \cap U = \varnothing$ und i. Allg. auch $|U| \gg |M|$. An Hand des Grades der Übereinstimmung zweier Tupel $t_1 \in S_1$ und $t_2 \in S_2$ wird (t_1, t_2) entweder als Element von M oder U klassifiziert.

Der quadratische Berechnungsaufwand, den man erhält, wenn jedes Tupel aus S_1 mit jedem Tupel aus S_2 verglichen werden muss, ist schon bei relativ kleinen Kardinalitäten von S_1 und S_2 unpraktikabel. Hier setzt das Konzept des sog. *Blocking* an: An Hand einer Teilmenge aller Attribute (sog. *Blockvariablen*) werden S_1 und S_2 in Blöcke partitioniert, wobei ein Block genau diejenigen Tupel enthält, deren Ausprägungen der Blockvariablen exakt übereinstimmen. Ein Blocking nach dem Attribut „Geschlecht“ liefert z. B. einen Block, der alle Tupel mit der Ausprägung „weiblich“ enthält, und einen Block mit allen Tupeln mit Ausprägung „männlich“. Es werden nun nur noch Tupel innerhalb eines Blocks miteinander verglichen, nämlich daraufhin, inwieweit sie auch in den Ausprägungen weiterer Attribute (sog. *Matchvariablen*) übereinstimmen. Dabei können – je nach Attribut – verschiedene Vergleichsoperatoren zum Einsatz kommen, z. B. zur Bemessung der Ähnlichkeit zweier Zeichenketten oder der prozentualen Abweichung zweier numerischer Werte.

Geht man bei obigem Beispiel von einer Gleichverteilung beider Ausprägungen des Attributs „Geschlecht“ aus, reduziert sich die Anzahl der Vergleiche durch das Blocking auf 50%. Um Mängel und Datenänderungen in den Ausprägungen der Blockvariablen zu kompensieren, werden i. Allg. pro zu verknüpfendem Datensatz mehrere Umläufe mit wechselnden Block- und Matchvariablen durchgeführt. Der Berechnungsaufwand pro Block und Umlauf verhält sich zwar quadratisch zur Blockgröße, allerdings ist diese relativ klein im Vergleich zur Gesamtzahl von Tupeln. In der Praxis sollten Attribute oder Attributkombinationen mit relativ großen, gleichverteilten Wertebereichen zur Blockbildung herangezogen werden, um kleine Blockgrößen zu erreichen und damit die Einschränkung auf potentielle Matchkandidaten zu optimieren. Da Wertebereiche über die Zeit variieren können, sind Blockkonfigurationen regelmäßig zu überprüfen und ggf. anzupassen.

Innerhalb eines Blocks wird pro Tupelpaar der Grad der Übereinstimmung in den Matchvariablen gemessen und an Hand eines Gewichtes repräsentiert, das der Wahrscheinlichkeit einer Zusammengehörigkeit der Tupel entspricht, d. h. je höher die Übereinstimmung, desto höher das Gewicht. Zur Berechnung der Gewichte werden verschiedene Parameter hinzugezogen, die a priori unbekannt sind und per Heuristik geschätzt werden müssen. Dies betrifft vor allem die Wahrscheinlichkeit der Korrektheit von Werten sowie die Wahrscheinlichkeit einer zufälligen Übereinstimmung zweier Werte. Details zur Berechnung von Gewichten finden sich in Abschnitt 7.7.2.

Mit Hilfe der Gewichte wird – unter Hinzunahme zweier Schwellenwerte (*Cut-Offs*) – die Klassifizierung eines (mit Gewicht und Umlauf attributierten) Tupelpaares in eine der Mengen M und U vorgenommen (siehe Abb. 6.2): Liegt ein Gewicht unterhalb des unteren Schwellenwerts, wird das

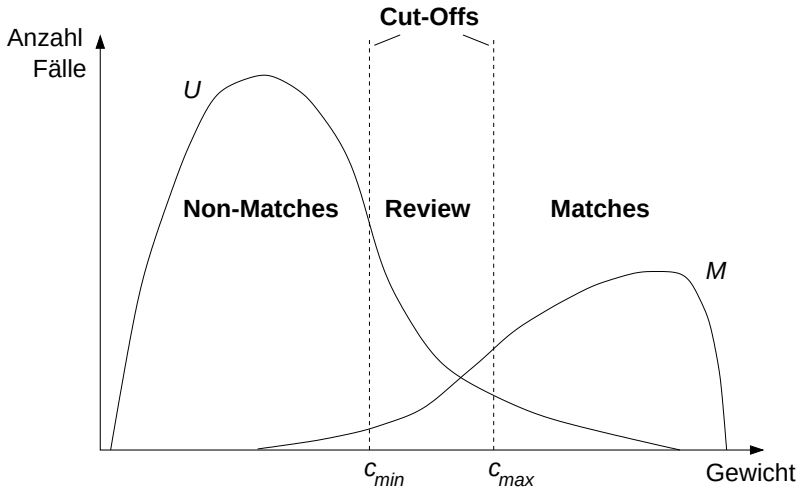


Abbildung 6.2: Verteilung von Gewichten im probabilistischen Record Linkage

Paar der Menge U (Non-Match) zugeordnet; liegt es oberhalb des oberen Schwellenwerts, wird das Paar als Element von M (Match) interpretiert. Befindet sich das Gewicht hingegen zwischen beiden Schwellenwerten, ist i. d. R. eine interaktive Nachbearbeitung erforderlich. Für den Sonderfall, dass beide Schwellenwerte identisch sind, läuft der Abgleich entweder vollkommen automatisch ab – die Nachbearbeitung entfällt dann unter Inkaufnahme höherer Homonym- und Synonymfehlerraten (vgl. Abschnitt 2.2.3) – oder alle Fälle oberhalb oder unterhalb des Schwellenwerts, je nachdem, ob der Schwellenwert eher niedrig oder hoch angesetzt wurde, sind nachzubearbeiten. Die Entscheidung, welche Homonym- und Synonymfehlerraten noch akzeptabel sind, ist in der jeweiligen Anwendungsdomäne zu treffen. Für eine Diskussion dieser Problematik siehe auch [VEE+00]. Mögliche Auswirkungen von Zuordnungsfehlern am Beispiel von Krankheitsregistern werden in [BS97b] und [SB97] behandelt.

Die Höhe der vergebenen Gewichte wird beeinflusst von dem Ergebnis einer sog. *Frequenzanalyse*, die optional vor dem Start des eigentlichen Record Linkage-Prozesses angestoßen werden kann. Dabei werden Werteverteilungen der am Record Linkage beteiligten Merkmale berechnet. Eine Übereinstimmung in einer seltenen Ausprägung resultiert dann später in einer höheren Gewichtung als eine Übereinstimmung in einer häufigen Ausprägung.

Die Anzahl der Umläufe sowie deren Parametrierung sind variabel. Neben der Angemessenheit einer solchen Konfiguration können Werteverteilungen, statistische Abhängigkeiten, temporale Einflüsse, die Zuverlässigkeit der beteiligten Datenquellen und andere Faktoren die Qualität der Record Linkage-Ergebnisse beeinflussen.

6.3 Domänenspezifische Datenqualitätsindikatoren

Als domänenspezifische Datenqualitätsindikatoren werden in dieser Arbeit Kennzahlen bezeichnet, die – unter Einbeziehung von Domänenwissen – bestimmte, für die Datennutzung relevante (Datenqualitäts-)Eigenschaften einer betrachteten Tupelmenge beschreiben. Im Anwendungsbereich der epidemiologischen Krebsregistrierung (siehe Abschnitt 11.1) sind z. B. folgende Datenqualitätsindikatoren verbreitet:

- *DCO-Rate:* Diese bezeichnet das Verhältnis derjenigen in einem Krebsregister gespeicherten Inzidenzfälle, zu denen ausschließlich eine Todesbescheinigung (DCO = Death Certificate Only) vorliegt, zur Gesamtzahl der gespeicherten Inzidenzfälle in Prozent.

- *DCN-Rate:* Diese bezeichnet den Anteil derjenigen Fälle, die dem Krebsregister erst über die Todesbescheinigung bekannt werden (DCN = Death Certificate Notification). Ein DCN-Fall gilt als DCO-Fall, solange keine weiteren Meldungen zu diesem Fall eingegangen sind.
- *M/I-Index:* Dieser gibt das Verhältnis (den Quotienten) von Mortalität (M) zu Inzidenz (I) in der Registerdatenbank an.
- *HV-Anteil:* Dieser bezeichnet den Anteil derjenigen Fälle, die eine histologisch, d. h. durch eine Gewebeuntersuchung abgesicherte Tumordiagnose aufweisen (HV = histologisch verifiziert).

Der HV-Anteil gibt Aufschluss über die „Verlässlichkeit“ tumorbezogener Angaben. DCO-Rate, DCN-Rate und M/I-Index werden hingegen zur Abschätzung der Vollständigkeit einer Registerdatenbank verwendet (Verhältnis von im Krebsregister erfassten Fällen zur Gesamtzahl der aufgetretenen Fälle). In [BSZ94] wird ein alternativer Ansatz zur Abschätzung der Vollständigkeit vorgestellt. Weitere Untersuchungen zur Datenqualität in Anwendungen des Gesundheitswesens finden sich in [GD00a] und [GD00b].

Datenqualitätsindikatoren ergänzen die in Abschnitt 6.1 definierten Metriken für (domänenunabhängige) Qualitätsmerkmale um Kennzahlen, die zwar nur in einem eng umrissenden Diskursbereich anwendbar sind, dort aber ggf. eine sehr bedeutende Rolle spielen.

Ein Ansatz zur Operationalisierung von Datenqualitätsindikatoren wird in Abschnitt 7.9.2 vorgestellt.

6.4 Zusammenfassung

Im Mittelpunkt dieses Kapitels stand die Entwicklung von Metriken für verschiedene Datenqualitätsmerkmale. Dabei wurde zwischen den Granularitätsebenen Attributwert, Tupel, Relation und Datenbank unterschieden. Des weiteren wurden Ansätze zur Operationalisierung der Metriken in Form von Messverfahren diskutiert. Für diejenigen Datenqualitätsmerkmale, die eine nachträgliche Beseitigung von Qualitätsmängeln erlauben, wurden Verbesserungsmaßnahmen skizziert. Abschließend erfolgte eine exemplarische Vorstellung domänenspezifischer Qualitätsindikatoren als Ergänzung der merkmalsbezogenen Metriken.

Kapitel 7

Das CLIQ-Vorgehensmodell zur inkrementellen Datenintegration in DWS

In Kapitel 1 wurde festgestellt, dass es der gängigen Praxis bei der Integration heterogener Daten insbesondere an Konzepten für ein methodisches Vorgehen fehlt. Dies hat zur Folge, dass DQM-Maßnahmen in der Praxis – wenn überhaupt – oft ineffektiv eingesetzt, d. h. in der falschen Reihenfolge angestoßen, auf die falschen Daten angewandt oder komplett vernachlässigt werden. Als Voraussetzung für die Etablierung eines wirksamen, auf die langfristige Zufriedenheit von Kunden (Datennutzern) ausgerichteten DQM ist es daher wünschenswert, methodische Grundsätze festzuschreiben, an denen sich eine Organisation bei der Erstellung von „Datenprodukten“ (wie in Abschnitt 3.1 definiert) orientieren kann.

Da in dieser Arbeit das DQM in DWS – in Abgrenzung zum DQM in OLTP-Systemen – im Mittelpunkt steht, konzentriert sich dieses Kapitel auf den (bedingt durch die Heterogenität von Datenquellen hochkomplexen) Prozess der inkrementellen Datenintegration in einem DWS. Viele der in diesem Kapitel beschriebenen Maßnahmen lassen sich aber auch auf OLTP-Systeme übertragen (siehe dazu Abschnitt 7.16).

Für den Datenintegrationsprozess im DWS wird in diesem Kapitel ein Vorgehensmodell entwickelt, das genau festlegt, welche Integrationsphasen in welcher Reihenfolge auszuführen sind. In Anlehnung an die Norm ISO 9001:2000 [Deu00b] werden DQM-Maßnahmen eingebunden, die den Integrationsprozess lenken und auf diese Weise sicherstellen, dass die resultierenden Datenprodukte den gestellten Anforderungen genügen. Die entwickelten Konzepte werden unter der Bezeichnung *CLIQ* (Data Cleansing mit intelligentem Qualitätsmanagement) zusammengefasst.

Das Kapitel gliedert sich wie folgt: Nach einem kurzen Überblick über das CLIQ-Vorgehensmodell (Abschnitt 7.1) und einigen grundlegenden Definitionen (Abschnitt 7.2) werden die einzelnen Phasen des Vorgehensmodells weitgehend formal beschrieben (Abschnitte 7.3 bis 7.12). Die Abschnitte 7.13 bis 7.16 diskutieren ergänzende Aspekte des Vorgehensmodells wie Parallelisierbarkeit, Laufzeitkomplexität und Einsatz von DQM-Maßnahmen in OLTP-Systemen.

7.1 Überblick

Operative DQM-Aspekte sind in der Integrationsschicht eines DWS anzusiedeln, genauer zwischen der Extraktion von Daten aus den Quellen und dem Laden konsolidierter Daten in das Data Warehouse. Damit ersetzt das DQM quasi das „T“ in „ETL“. Da in der Integrationsschicht von DWS das relationale Datenmodell vorherrscht, richtet sich auch das CLIQ-Vorgehensmodell an diesem Datenmodell aus. Einen Überblick über die Phasen des Vorgehensmodells gibt Abb. 7.1. Den Ausgangs-

punkt der Datenintegration stellen sog. *Quelldatenpuffer (QDP)* dar. Dabei handelt es sich um Datenhaltungskomponenten, die die Ergebnisse einer Extraktion (neue, geänderte oder gelöschte Tupel) aus einer angebundenen Quelle temporär speichern. Die jeweils verwendeten Extraktionskomponenten haben dabei Abhängigkeiten zwischen Tupeln angemessen zu berücksichtigen. Jeder QDP (genau einer pro Quelle) implementiere ein relationales Datenbankschema, auf das die in das DWS zu übertragenden Quelldaten (per Wrapper) abgebildet werden. Quelldatenpuffer dienen dazu, das in den jeweiligen Quellen verwendete Datenmodell (relational, objektorientiert, dateibasiert etc.) zu kapseln und somit eine einheitlich relationale Sicht auf die Quelldaten anzubieten.

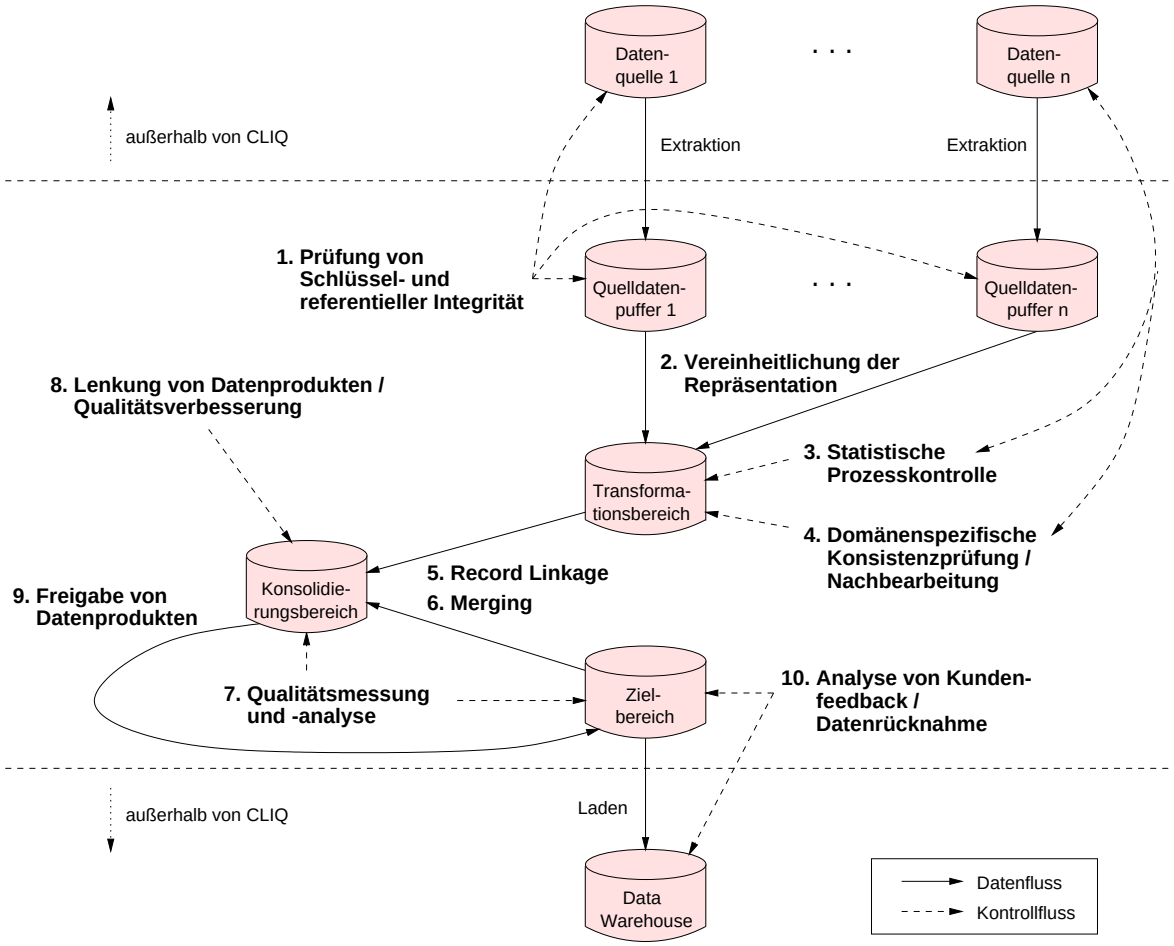


Abbildung 7.1: CLIQ-Vorgehensmodell zur inkrementellen Datenintegration in DWS

Als zentrale Datensenke des Integrationsprozesses dient der sog. *Zielbereich (ZB)*. Dieser entspricht dem in Abschnitt 2.1.2 eingeführten ODS. Die Bezeichnung „Zielbereich“ wurde gewählt, um zu verdeutlichen, dass in dieser Datenhaltungskomponente ausschließlich qualitätsgesicherte integrierte Datenprodukte gespeichert werden. Der Zielbereich implementiere ebenfalls ein relationales Schema, das alle für das DWS relevanten Relationen und Beziehungen enthält, allerdings – im Gegensatz zum Data Warehouse – nicht analyseorientiert sein muss. Zwischen Zielbereich und Quelldatenpuffern werden zwei weitere Datenhaltungskomponenten geschaltet, die Zwischenergebnisse von Integrationsprozessen temporär speichern: zum einen ein *Transformationsbereich* (*TB*, siehe Abschnitt 2.1.3), der auch – gelegentlich unter der Bezeichnung *Staging Area* – in klassischen DWS-Architekturen existiert, zum anderen ein sog. *Konsolidierungsbereich (KB)*, der in CLIQ neu eingeführt wird, um integrierte, aber noch nicht abschließend qualitätsgesicherte Daten puffern zu können. Die Schemata von Transformations-, Konsolidierungs- und Zielbereich seien identisch.

Da in den Quelldatenpuffern sowie in Transformations- und Konsolidierungsbereich auch inkonsistente Daten zwischengespeichert werden müssen, wird angenommen, dass keine DBMS-seitigen Integrationsprüfungen auf diesen Datenhaltungskomponenten stattfinden. Dies betrifft im Transformations- und Konsolidierungsbereich insbesondere verschiedene Typen von Constraints (Eindeutigkeit, Wertebereiche, *NOT NULL*), in den Quelldatenpuffern zusätzlich Primärschlüssel- und Fremdschlüssel-Constraints [Tha94, VA96, BMP01]. Im Zielbereich, in den ausschließlich konsolidierte und qualitätsgeprüfte Daten eingefügt werden, sind hingegen DBMS-seitige Integrationsprüfungen erlaubt, wenn auch aufgrund der nutzerseitig ausschließlich lesenden Zugriffe nicht unbedingt erforderlich.

In den nachfolgenden Abschnitten werden die operativen Phasen des CLIQ-Vorgehensmodells weitgehend formal spezifiziert. Dies erweckt ggf. den Eindruck eines sehr strikten Ansatzes. In der Praxis wird es in der Tat an gewissen Stellen nötig sein, die Vorgaben des Vorgehensmodells auf Grund von anwendungsbezogenen Rahmenbedingungen „aufzuweichen“. Ist es beispielsweise einer Datenquelle – aus Mangel an Ressourcen oder anderen Gründen – nicht möglich, bestimmten Qualitätsmängeln nachzugehen, so müssen ggf. (falls keine alternativen Quellen existieren) auch qualitativ mangelhafte Daten in den Zielbereich übernommen werden, da die Alternative, auf die Daten ganz zu verzichten, ebenfalls nicht praktikabel wäre. In einem solchen Fall kann das DQM zumindest dazu genutzt werden, die Kunden (Nutzer) der Daten auf die enthaltenen Mängel hinzuweisen, so dass diese bei Auswertungen entsprechend berücksichtigt werden können.

Das CLIQ-Vorgehensmodell ist zwar stark von der ISO 9001:2000 geprägt, deckt jedoch nicht alle Anforderungen der Norm ab. Dies liegt daran, dass ein Vorgehensmodell nur operative Anforderungen umsetzen kann. Beim praktischen Einsatz des Vorgehensmodells in einem DQMS sind jedoch auch nicht-operative DQM-Aspekte einzubeziehen. Diese sind von der organisatorischen Umgebung, in die das Vorgehensmodell eingebettet wird, zu berücksichtigen und umfassen insbesondere:

- Verantwortungsübernahme und Engagement der Organisationsleitung
- Festlegung von Zuständigkeiten
- Bereitstellung und Management von Ressourcen
- Schulung und Motivierung beteiligter Personen
- Erfassung und Bewertung von Anforderungen
- Auswahl von Datenquellen an Hand von Kriterien wie Relevanz, Zeitnähe und Kosten von Daten sowie Reputation und verwendete Erfassungsmethoden von Quellen
- Beschaffung von Rohdaten
- Reengineering und Pflege von Metadaten
- Dokumentation von Integrations- und Prüfprozessen.

Die Beziehungen zwischen CLIQ-Vorgehensmodell und ISO 9001:2000 werden in [HA01] und [Ade01] im Detail erläutert.

7.2 Grundlegende Definitionen

In diesem Abschnitt werden einige formale Konstrukte definiert, die im Rahmen der Prozessbeschreibung Verwendung finden werden.

Sei $Dom(A)$ die Domäne (der Wertebereich) eines Attributs A.

Sei $Nullable(A) := \begin{cases} \text{wahr, falls } \textit{NULL} \in Dom(A) \\ \text{falsch sonst.} \end{cases}$

Sei $R\{A_1, ..., A_n\}$ eine Relation über den Attributen $A_1, ..., A_n$. Als Kurzform wird auch nur das Prädikatensymbol (Relationenbezeichner) R als Relation bezeichnet.

Sei $I(R)$ die Instanz von R, also die Menge der Tupel in R.

Sei $R\{a_1, ..., a_n\}$ ein Tupel in $I(R)$, wobei $a_i \in Dom(A_i)$ für $i = 1, ..., n$.

Sei $t.A_i$ der Wert eines Attributs A_i $(i \in \{1, ..., n\})$ im Tupel t.

O. B. d. A. wird festgelegt, dass alle Relationen im Transformations-, Konsolidierungs- und Zielbereich jeweils genau ein als glb_id bezeichnetes Primärschlüsselattribut aufweisen. Primärschlüssel seien numerische Werte (Surrogate) ohne domänenbezogene Semantik.

Des weiteren seien folgende Metaattribute gegeben:

1. Sei $hist_state$ ein Metaattribut über der Domäne $\{new, delete, historize, overwrite\}$, das den sog. *Historisierungsstatus* eines Tupels enthält. Dieser sagt aus, ob ein Tupel in der Quelle neu erzeugt (new), gelöscht ($delete$) oder geändert wurde. Bei Änderungen wird unterschieden, ob aktualisierte Werte im DWS zu historisieren ($historize$) oder zu überschreiben ($overwrite$) sind. Die Stati $historize$ und $overwrite$ werden in Abschnitt 7.11, der Status new in Abschnitt 7.7.1 und der Status $delete$ in Abschnitt 7.13 behandelt.

2. Sei $is_current$ ein boolesches Metaattribut, welches angibt, ob ein Tupel (im Zielbereich) historisiert ist (Wert $false$) oder nicht (siehe Abschnitt 7.11).

3. Sei old_id ein Metaattribut, das auf die chronologische Vorgängerversion, d. h. auf die zuletzt durchgeführte Historisierung eines Tupels verweist (siehe Abschnitt 7.11). Falls diese nicht existiert, gelte $old_id = \textit{NULL}$.

4. Seien schließlich $validity_period_begin$ und $validity_period_end$ Metaattribute, die den Gültigkeitszeitraum eines Tupels beschreiben (siehe Abschnitt 7.11). Für $validity_period_end$ gelte:
$$(is_current = true) \Leftrightarrow (validity_period_end = \textit{NULL}) \tag{7.1}$$
Zu Grundlagen temporaler Metaattribute siehe z. B. [Sno90, Sno95].

Jedes Tupel in Transformations-, Konsolidierungs- und Zielbereich weise einen derartigen Satz von Metainformationen auf. Diese werden in den einzelnen Phasen des CLIQ-Vorgehensmodells bei Bedarf ausgelesen bzw. manipuliert. Es wird angenommen, dass $hist_state$ und $validity_period_begin$ quellenseitig bzw. im Rahmen der Datenextraktion gesetzt werden. Folglich müssen Tupel in den Quelldatenpuffern diese Metainformationen bereits mitführen.

7.3 Phase 1: Prüfung von Schlüssel- und referentieller Integrität

Zwei grundlegende Voraussetzungen für eine Integrierbarkeit von Daten sind die lokale (innerhalb einer Quellrelation) Eindeutigkeit von Primärschlüsseln (Schlüsselintegrität) sowie die referentielle Integrität. Letztere setzt sich aus zwei Teilaspekten zusammen:

- Zu jedem Fremdschlüssel muss ein korrespondierender Primärschlüssel existieren.

- Die Anzahl der Fremdschlüssel zu einem Primärschlüssel muss der auf logischer Ebene spezifizierten Multiplizität der Beziehung entsprechen, d. h. bei einer Multiplizität von 1:1 muss genau ein Fremdschlüssel existieren, bei 1:1..* mindestens einer etc.

Ist eine dieser Bedingungen nicht erfüllt, können die nachfolgenden Phasen des CLIQ-Vorgehensmodells nicht ordnungsgemäß ausgeführt werden. Vor einem Übergang in Phase 2 ist daher für jede Relation eines Quelldatenpuffers sicherzustellen, dass beide obigen Bedingungen gelten. Gegebenenfalls ist dazu eine Neuanforderung von Daten oder eine Nachbearbeitung von Daten im QDP erforderlich.

Im Folgenden werden die Prüfungen der Phase 1 formal beschrieben (vgl. Abschnitte 6.1.8 und 6.1.9):

Sei $R_{QDP}\{local_id, A_1, ..., A_m\}$ eine zu prüfende Relation eines QDP ($local_id \notin \{A_1, ..., A_m\}$).

O. B. d. A. bezeichne $local_id$ das (lokale) Primärschlüsselattribut von R_{QDP}. Es wird also von zusammengesetzten Primärschlüsselattributen abstrahiert; die Prüfungen würden hier analog ablaufen.

7.3.1 Schlüsselintegrität

Alle Tupel der Instanz von R_{QDP} werden daraufhin überprüft, ob ihr Primärschlüssel lokal (d. h. innerhalb von R_{QDP}) eindeutig ist. Gilt dies nicht, wird eine Fehlermeldung generiert:

$\forall t_1 \in I(R_{QDP})$ `do`

  `if` $\exists t_2 \in I(R_{QDP}) : (t_2 \neq t_1 \wedge t_2.local_id = t_1.local_id)$

    $CreateMessage(t_1,$ 'Nicht-eindeutiger Primärschlüssel'$)$

  `fi`

`od`

Dabei sei $CreateMessage$ eine Funktion, die zu einem Tupel eine Fehlermeldung erzeugt und diese an einer zentralen Stelle ablegt.

7.3.2 Referentielle Integrität

Sei $R_{QDP,child}\{local_ref, B_1, ..., B_n\}$ eine Relation, zu der R_{QDP} eine 1:0..1-, 1:1-, 1:1..*- oder 1:*-Beziehung (UML-Notation) aufweist. Der Beziehungstyp sei in den Metadaten abgelegt und über eine Funktion rel mit Relationen R_1 und R_2 als Argumenten abrufbar. Sei $local_ref$ das mit $local_id$ korrespondierende Fremdschlüsselattribut (wiederum wird von zusammengesetzten Schlüsseln abstrahiert). Die Prüfung auf Integrität läuft dann in zwei Schritten ab: einmal von einem Kindtupel zum Elterntupel und dann von einem Elterntupel zu den Kindtupeln.

$\forall t_{child} \in I(R_{QDP,child})$ `do` `//Schritt 1`

  `if` $\nexists t \in I(R_{QDP}) : t.local_id = t_{child}.local_ref$

    $CreateMessage(t_{child},$ 'Kein Elterntupel'$)$

  `fi`

`od`

$\forall t \in I(R_{QDP})$ `do` `//Schritt 2`

$num_children := \left| \{t_{child} \in I(R_{QDP,child}) | t_{child}.local_ref = t.local_id\} \right|$

`switch` $(rel(R_{QDP}, R_{QDP,child}))$

`case 1:0..1: if` $num_children > 1$

$CreateMessage(t,$ 'Zu viele Kindtupel') `od`

`case 1:1: if` $num_children \neq 1$

$CreateMessage(t,$ 'Zu viele oder zu wenig Kindtupel') `od`

`case 1:1..*: if` $num_children < 1$

$CreateMessage(t,$ 'Zu wenig Kindtupel') `od`

`hctiws`

`od`

Im dritten Fall entspricht „*“ einem Platzhalter für einen Wert ≥ 1. Wurde dieser bei der logischen Schemamodellierung mit einem konkreten Wert w instantiiert (unter bestimmten, domänenspezifischen Umständen sinnvoll), so ist eine Fehlermeldung zu erzeugen, falls $num_children \neq w$ gilt.

Nach Abschluss der Prüfungen für einen QDP werden alle zu Daten aus diesem QDP erzeugten Fehlermeldungen – falls vorhanden – an die zugehörige Datenquelle übertragen. Die Datenquelle hat die Fehlermeldungen zu bearbeiten und anschließend das korrigierte Datenpaket erneut an das DWS zu liefern. Vor dem Einspielen des korrigierten Datenpakets ist das mangelhafte Paket aus dem jeweiligen QDP zu entfernen. Alternativ ist eine Korrektur „vor Ort“ (ohne Rückgriff auf Datenquellen) denkbar, vorausgesetzt, es liegt hinreichendes Wissen über die korrekten Tupelbeziehungen vor. Schließlich ist Phase 1 auf den korrigierten Daten zu wiederholen (zu Terminierungsfragen siehe Abschnitt 7.17).

Ist sowohl die Schlüssel- als auch die referentielle Integrität gewährleistet, kann zu Phase 2 des Vorgehensmodells übergegangen werden.

7.4 Phase 2: Vereinheitlichung der Repräsentation

Ziel von Phase 2 des CLIQ-Vorgehensmodells ist es, die heterogenen, bereits auf Schlüssel- und referentielle Integrität geprüften Daten aus den Quelldatenpuffern in eine einheitliche Repräsentation zu überführen, um zu gewährleisten, dass zum einen Daten vergleichbar sind und zum anderen nachfolgende Maßnahmen auf einem einheitlichen, quellenunabhängigen Schema durchgeführt werden können.

Bevor ein Vereinheitlichungsprozess (vgl. Abschnitt 6.2.1) angestoßen werden kann, muss dieser zunächst über eine Menge von Abbildungsvorschriften von Attributen aus Relationen der Quelldatenpuffer auf die Attribute einer Relation des Transformationsbereichs spezifiziert werden. Spezifikationen sollten persistent gespeichert werden, damit sie wiederverwendet werden können.

7.4.1 Spezifizierung

Sei $R_{TB}\{glb_id, A_1, ..., A_m\}$ eine Relation des Transformationsbereichs. R_{TB} kann Daten aus verschiedenen Relationen verschiedener Quelldatenpuffer beziehen. Alle Abbildungsvorschriften für

R_{TB} werden in einem sog. *Transformationspaket* $TP(R_{TB})$ zusammengefasst, welches pro beteiligter QDP-Relation R_{QDP} genau ein sog. *Task Set* $TS(R_{TB}, R_{QDP})$ beinhaltet (siehe Abb. 7.2). Ein solches Task Set stellt eine Menge von Abbildungsvorschriften von Attributen aus R_{QDP} auf Attribute aus R_{TB} dar.

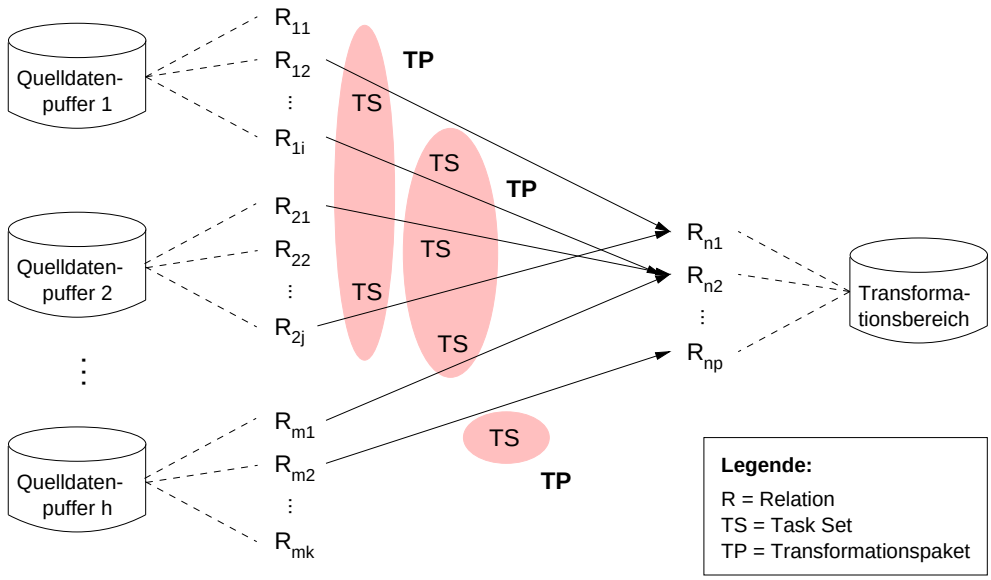


Abbildung 7.2: Task Sets und Transformationspakete

Eine Abbildungsvorschrift f kann – je nach durchzuführender Transformation – unterschiedlichen Typs sein:

- *Erzeugen eines global eindeutigen Primärschlüssels*
  Lokale Schlüssel, also Primärschlüssel aus den Datenquellen, werden im DWS lediglich zur Zuordnung inkrementeller Updates herangezogen (siehe Abschnitt 7.7.1). Sowohl die Datenanalyse im DWS als auch weite Teile des Integrationsprozesses verwenden stattdessen global (bezogen auf eine Zielrelation R) eindeutige Schlüssel. Diese werden künstlich erzeugt (Surrogate), wie im Folgenden dargestellt.

  Sei $I_R := I(R_{TB}) \cup I(R_{KB}) \cup I(R_{ZB})$ die Vereinigungsmenge der Instanzen der Relation R in Transformations-, Konsolidierungs- und Zielbereich.

  Sei $K_R := \{w \in Dom(glb_id) | \exists t \in I_R : t.glb_id = w\}$ die Menge aller bereits vergebenen (globalen) Primärschlüssel in R.

  Die Funktion $f_{pkey} : P(Dom(glb_id)) \mapsto Dom(glb_id)$ erzeuge nun einen neuen Schlüssel $pk := f_{pkey}(K_R) \notin K_R$. $P(X)$ bezeichne dabei die Potenzmenge einer Menge X. Bei der Erzeugung eines neuen Tupels t_{new} in R_{TB} wird dann $t_{new}.glb_id := pk$ gesetzt und pk in die Menge K_R aufgenommen. Die lokalen Schlüssel bleiben (in einer Lookup-Tabelle) erhalten, so dass inkrementelle Updates korrekt zugeordnet werden können.

  Im Gegensatz zu den übrigen Transformationen soll die Schlüsselerzeugung intern vorgenommen werden, d. h. in jedem Task Set gebe es eine (vor dem Systemnutzer versteckte) Funktion f_{pkey}.

- *Kopieren von Attributwerten*
  Die einfachste Transformation stellt das Kopieren von Attributwerten aus einem Attribut B_i von R_{QDP} in ein Attribut A_j von R_{TB} dar. Dazu muss gelten: $Dom(B_i) \subseteq Dom(A_j)$. Die Funktion $f_{copy} : Dom(B_i) \mapsto Dom(B_i)$ realisiere den Kopiervorgang als identische Abbildung.

- *Abbilden von Attributwerten zwischen Domänen*
  Sei A_j ein Attribut von R_{TB} und B_i ein Attribut von R_{QDP}. Falls es zu jedem Wert $w \in Dom(B_i)$ einen Wert $v \in Dom(A_j)$ gibt, so dass v semantisch äquivalent zu w ist (d. h. von einem Domänenexperten gleich interpretiert wird) oder aus w folgt (Berechnung abgeleiteter Werte, siehe unten), so kann eine Funktion $f_{dom} : Dom(B_i) \mapsto Dom(A_j)$ definiert werden, so dass $v = f_{dom}(w)$ gilt.

  Falls $Dom(A_j) = Dom(B_i)$ gilt und f_{dom} die identische Abbildung ist, entspricht dies einem Kopieren von Attributwerten (siehe oben).

  Darüber hinaus kann f_{dom} zu folgenden Zwecken eingesetzt werden (vgl. Abschnitt 2.2.3):

  – Behandlung von Nullwerten
  – Anpassung von Datentypen
  – Konvertierung von Codierungen
  – Vereinheitlichung von Zeichenketten
  – Vereinheitlichung von Datumsangaben
  – Umrechnung von Maßeinheiten und Skalierungen
  – Berechnung abgeleiteter Werte (aus jeweils einem Quellattribut).

- *Kombination von Attributwerten*
  Sei A_j ein Attribut von R_{TB}, seien $B_i, ..., B_k$ Attribute von R_{QDP} und sei $Corresp(t, R_{TB}) \in I(R_{TB})$ ein zu $t \in I(R_{QDP})$ korrespondierendes, d. h. dieselbe Entität des Diskursbereichs beschreibendes Tupel in R_{TB}. Mit Hilfe einer Funktion $f_{comb} : Dom(B_i) \times ... \times Dom(B_k) \mapsto Dom(A_j)$ kann dann eine (tupelinterne) Kombination von Attributwerten erfolgen (z. B. Umwandlung von Einzelwerten zu Tag, Monat und Jahr in einen Datumswert).

  $$Corresp(t, R_{TB}).A_j := f_{comb}(t.B_i, ..., t.B_k)$$

  Die Berechnung abgeleiteter Werte aus mehr als einem Quellattribut kann als Sonderfall der Kombination von Attributwerten angesehen werden.

- *Separierung von Attributwerten*
  Seien $A_j, ..., A_k$ Attribute von R_{TB}, sei B_i ein Attribut von R_{QDP} und sei $Corresp$ wie oben definiert. Mit Hilfe von Funktionen $f_{sep,j} : Dom(B_i) \mapsto Dom(A_j), ..., f_{sep,k} : Dom(B_i) \mapsto Dom(A_k)$ kann dann eine (tupelinterne) Separierung von Attributwerten erfolgen (z. B. Extraktion von Postleitzahl und Ort aus einer Adressangabe):

  $$Corresp(t, R_{TB}).A_j := f_{sep,j}(t.B_i)$$

  ...

  $$Corresp(t, R_{TB}).A_k := f_{sep,k}(t.B_i)$$

Sowohl f_{dom} als auch f_{comb} und f_{sep} erfordern i. Allg. domänenspezifisches Wissen, z. B. Codierungsschemata, Namensregister etc., welches wiederum unterschiedlich repräsentiert werden kann, z. B. als Lookup-Tabellen, als Regelmengen oder als reguläre Ausdrücke.

7.4.2 Ausführung

Bevor ein Transformationspaket $TP(R_{TB})$ ausgeführt werden kann, muss sichergestellt werden, dass $I(R_{TB}) = \varnothing$ gilt. Gegebenenfalls muss dazu eine andere, noch laufende Integrationsprozessinstanz zu R_{TB} zumindest bis einschließlich Phase 6, in der die jeweilige Relation des Transformationsbereichs „geleert“ wird, fortgeführt werden (siehe auch Abschnitt 7.14).

Eine Ausführung von $TP(R_{TB})$ ist durch den untenstehenden Algorithmus beschrieben. Dabei sei K_R wiederum die Menge der bereits vergebenen Primärschlüssel zur Relation R in Transformations-, Konsolidierungs- und Zielbereich. Die Funktion $ftype(f)$ liefere den Typ der Funktion f gemäß der o. a. Klassifikation zurück, hier genutzt zum Ausschluss von f_{pkey}. Es wird davon ausgegangen, dass das Transformationspaket korrekt spezifiziert wurde, d. h. dass die vorgenommenen Transformationen keine logischen Widersprüche einführen – dies ist durch einen Domänenexperten sicherzustellen – und dass ein Attribut A von R_{TB}, für das $Nullable(A) = false$ gilt, nach Ausführung des Transformationspakets in allen Tupeln einen Wert ungleich *NULL* aufweist.

$\forall TS(R_{TB}, R_{QDP}) \in TP(R_{TB}) : I(R_{QDP}) \neq \varnothing$ do

  $\forall t \in I(R_{QDP})$ do

    $t_{new} := CreateNewTuple(R_{TB})$

    $t_{new}.glb_id := TS(R_{TB}, R_{QDP}).f_{pkey}(K_R)$

    $\forall f \in TS(R_{TB}, R_{QDP}) : ftype(f) \in \{copy, dom, comb, sep\}$ do

      $t_{new}.A_j := f(t.B_i)$ bzw. $t_{new}.A_j := f(t.B_i, ..., t.B_k)$

    od

  od

od

Alle Task Sets des Transformationspakets zu R_{TB} werden also auf alle nicht-leeren, an R_{TB} angebundenen Quelldatenpuffer angewandt. Dabei wird für jedes QDP-Tupel ein neues Tupel t_{new} in R_{TB} erzeugt und mit einem global eindeutigen Schlüssel versehen. Weiterhin werden mit Hilfe der in dem jeweiligen Task Set spezifizierten Abbildungsvorschriften die einzelnen Attributwerte von t_{new} gesetzt.

Konnte ein Transformationspaket $TP(R_{TB})$ erfolgreich ausgeführt werden, sind die Tupel aus den beteiligten Quelldatenpuffern als „gelöscht“ zu markieren, es sei denn, es existiert noch ein anderes Transformationspaket $TP(R'_{TB})$ mit $R'_{TB} \neq R_{TB}$, das sich ebenfalls auf einen oder mehrere der an $TP(R_{TB})$ beteiligten Quelldatenpuffer bezieht und das noch nicht (erfolgreich) auf den aktuell in den betroffenen Quelldatenpuffern gespeicherten Tupeln ausgeführt wurde. Die physische Löschung der QDP-Tupel sollte erst nach erfolgreichem Abschluss der Integration in den Zielbereich (also nach Phase 9) erfolgen, um im Rahmen der Lenkung mangelhafter Daten (Phase 8) in früheren Phasen des CLIQ-Vorgehensmodells wiederaufsetzen zu können, ohne die Daten von den Quellen neu anfordern zu müssen.

Lassen sich Tupel aufgrund von Datenqualitätsmängeln (z. B. ungültige Werte, die nicht im Definitionsbereich der Abbildungen enthalten sind) nicht transformieren, ist analog zum in Abschnitt 7.3 beschriebenen Vorgehen eine Löschung des mangelhaften Datenpakets und eine Neuanforderung einer korrigierten Fassung vorzunehmen (Wiederaufsetzen bei Phase 1) – es sei denn, es handelt sich um manuell korrigierbare Einzelfälle.

7.5 Phase 3: Statistische Prozesskontrolle

Nach Abschluss von Phase 2 liegen vereinheitlichte Tupel im Transformationsbereich vor. Um grobe Erfassungs-, Verarbeitungs- und Übertragungsfehler (z. B. Altersangabe aufgrund eines Übertragungsfehlers in allen Sätzen eines Datenpakets gleich 0) im Sinne einer Eingangskontrolle frühzeitig – d. h. *vor* der Durchführung weiterer Integrationsschritte – zu erkennen, können in Phase 3 Statistische Prozesskontrollen (vgl. Abschnitt 6.1.1) zwischengeschaltet werden. Die Kernidee dabei ist,

zu den Daten im Transformationsbereich – differenziert nach Datenquellen – attributspezifische statistische Kennzahlen (Mittelwert, Varianz etc.) zu berechnen und über die Zeit zu protokollieren. Die jeweils in einem Integrationslauf neu berechneten Kennzahlen werden dann mit Erfahrungswerten aus früheren Integrationsprozessen oder mit explizit vorgegebenen Werten verglichen. Bei signifikanten Abweichungen werden entsprechende Maßnahmen eingeleitet.

Statistische Prozesskontrollen werden in sog. *SPC-Plänen* spezifiziert. Im Folgenden werden Spezifizierung und Ausführung eines SPC-Plans (inkl. Analyse von SPC-Messergebnissen) formal beschrieben.

7.5.1 Spezifizierung

Sei $R_{TB}\{glb_id, A_1, ..., A_m\}$ eine Relation des Transformationsbereichs und $A \in \{A_1, ..., A_m\}$ ein numerisches Attribut von R_{TB} (zur Prüfung nicht-numerischer Attributwerte siehe Abschnitt 7.5.3).

Ein SPC-Plan $SP(A)$ zum Attribut A ist ein 7-Tupel $(cond, \mu, \sigma, lcl_\mu, ucl_\mu, lcl_\sigma, ucl_\sigma)$, wobei für die einzelnen Bestandteile von $SP(A)$ gilt:

1. $cond$ ist ein boolescher Ausdruck über $\{A_1, ..., A_m\}$ zur Selektion einer Teilmenge von Tupeln aus $I(R_{TB})$ (entspricht einer WHERE-Klausel in SQL).

2. μ ist der erwartete Mittelwert und σ die erwartete Standardabweichung im Attribut A. Diese erwarteten Werte werden entweder von Domänenexperten vorgegeben oder zur Laufzeit an Hand der bereits integrierten Daten aus der mit R_{TB} korrespondierenden Relation R_{ZB} im Zielbereich wie folgt berechnet:

$$\mu := \frac{1}{|I(R_{ZB})|} \sum_{t \in I(R_{ZB})} t.A \tag{7.2}$$

$$\sigma := \sqrt{\frac{1}{|I(R_{ZB})|} \sum_{t \in I(R_{ZB})} (t.A - \mu)^2} \tag{7.3}$$

3. Zu diesen erwarteten Werten werden untere und obere Kontrollgrenzen lcl_μ, ucl_μ, lcl_σ und ucl_σ spezifiziert, bei deren Unter- bzw. Überschreitung ein „manueller“ Eingriff in den Prozess vorgenommen werden soll. Per Default werden dazu die Formeln der klassischen SPC-Theorie [She31, Red96] herangezogen, im Falle des Mittelwerts also die sog. 3σ-*Grenzen*:

$$lcl_\mu := \mu - 3\sigma \quad \text{(Lower Control Limit)} \tag{7.4}$$

$$ucl_\mu := \mu + 3\sigma \quad \text{(Upper Control Limit)} \tag{7.5}$$

   Auf die Spezifizierung von Defaultwerten für Kontrollgrenzen zur Standardabweichung wird hier verzichtet.

7.5.2 Ausführung

Sei d die auf A projizierte und mittels $cond$ auf eine Teilmenge von $I(R_{TB})$ eingeschränkte Tupelmenge, im Folgenden als „Datenobjekt“ bezeichnet.

Bei der Ausführung eines SPC-Plans werden die spezifizierten Prüfungen auf das Datenobjekt d angewandt, aufgeschlüsselt nach der Herkunft der Tupel (quellenbezogene Messung). Sei dazu $Q(R_{TB})$

die Menge aller R_{TB} mit Daten versorgenden Quellen und $d(q) := \{t \in d | t \text{ stammt aus Quelle } q\}$ die Menge der aus Quelle q stammenden Tupel in d.

$\forall q \in Q(R_{TB})$ do

$\mu_d(q) := \frac{1}{|d(q)|} \sum_{t \in d(q)} t.A$

$\sigma_d(q) := \sqrt{\frac{1}{|d(q)|} \sum_{t \in d(q)} (t.A - \mu_d(q))^2}$

od

Die berechneten Werte werden nun – separiert nach Datenquellen – in sog. *Kontrollkarten* unterschiedlichen Typs eingetragen. Im Falle eines sog. *X-Chart* werden die Messwerte über die Zeit aufgetragen, während in sog. *Pareto-Charts* eine Kumulierung der Abweichungen von den erwarteten Werten über alle bisher durchgeführten Messungen erfolgt [Red96].

Ein X-Chart zu einem SPC-Plan $SP(A)$ bzgl. einer Quelle q ist definiert als Menge $XC(SP(A), q)$ von Tripeln (ts, μ_X, σ_X), wobei μ_X und σ_X die im Attribut A zum Zeitpunkt ts gemessenen Werte zum Mittel bzw. zur Standardabweichung darstellen.

Ein Pareto-Chart zu einem SPC-Plan $SP(A)$ bzgl. einer Quelle q ist definiert als Menge $PC(SP(A), q)$ von Tripeln $(num, sum_{\Delta\mu}, sum_{\Delta\sigma})$, wobei num die Anzahl der bereits vorgenommenen Messungen in $PC(SP(A), q)$ und $sum_{\Delta\mu}$ bzw. $sum_{\Delta\sigma}$ die über diese Messungen kumulierten Abweichungen von den erwarteten Werten bzgl. Mittel bzw. Standardabweichung darstellen.

Die Eintragung der oben berechneten Messwerte in die Kontrollkarten geschieht wie folgt:

$\forall q \in Q(R_{TB})$ do

$XC(SP(A), q) := XC(SP(A), q) \cup \{(ts_{meas}, \mu_d(q), \sigma_d(q))\}$

$\forall (num_{max,old}, sum_{\Delta\mu,old}, sum_{\Delta\sigma,old}) \in PC(SP(A), q) :$
$num_{max,old} = max(\{num \in \mathbb{N} | (num, sum_{\Delta\mu}, sum_{\Delta\sigma}) \in PC(SP(A), q)\})$ do

$PC(SP(A), q) := PC(SP(A), q) \cup$
$\{(num_{max,old} + 1, sum_{\Delta\mu,old} + |\mu_d(q) - \mu|, sum_{\Delta\sigma,old} + |\sigma_d(q) - \sigma|)\}$

od

od

Dabei ist ts_{meas} der Zeitpunkt, an dem die Messung durchgeführt wurde. Vor dem Einfügen des neuen Pareto-Chart–Eintrags wird zunächst derjenige Eintrag ausgewählt, der aus der zuletzt vorgenommenen Messung hervorgegangen ist (maximale Anzahl der eingeflossenen Messungen). Im neuen Eintrag wird die Anzahl der Messungen um 1 inkrementiert; weiterhin werden die kumulierten Abweichungen von den erwarteten Werten für Mittel und Standardabweichung mit den neu gemessenen Werten aktualisiert.

Im Anschluss an das Eintragen in die Kontrollkarten müssen die Messergebnisse auf Grenzüberschreitungen hin analysiert werden. Falls diese vorliegen, sind entsprechende, interaktiv festzulegende Maßnahmen durchzuführen, z. B. Löschen der aus einer betroffenen Quelle stammenden Daten aus R_{TB}, Neuanfordern der Daten nach Beseitigung der Problemursache – i. Allg. quellenseitig – und Fortfahren mit Phase 1 des Vorgehensmodells, notfalls Aussondern von nicht hinreichend „korrigierbaren“ Tupeln. Über derartige Maßnahmen ist im Einzelfall zu entscheiden; eine Automatisierung der Entscheidungsfindung ist i. Allg. nicht möglich.

$\forall q \in Q(R_{TB}) : (\mu_d(q) < lcl_\mu \vee \mu_d(q) > ucl_\mu)$ do

Veranlasse interaktive Maßnahmen

od

$\forall q \in Q(R_{TB}) : (\sigma_d(q) < lcl_\sigma \vee \sigma_d(q) > ucl_\sigma)$ do

Veranlasse interaktive Maßnahmen

od

Nachdem – ggf. nach mehreren Iterationen der Phasen 1 bis 3 – alle für R_{TB} spezifizierten SPC-Pläne auf der aktuellen Instanz $I(R_{TB})$ erfolgreich ausgeführt worden sind (d. h. alle Messwerte liegen innerhalb der erwarteten Bereiche oder wurden interaktiv als tolerierbar bzw. korrekt bestätigt), kann mit Phase 4 des Vorgehensmodells fortgefahren werden.

7.5.3 Weitere statistische Prüfungen

Analog zu den Ausführungen in den vorangegangenen zwei Abschnitten können weitere statistische Prüfungen auf Attributen spezifiziert und ausgeführt werden. Auf eine detaillierte Beschreibung wird an dieser Stelle verzichtet. Im Einzelnen können z. B. folgende Fragestellungen betrachtet werden:

- *Welche Attributausprägungen treten wie oft auf?*
  Diese Fragestellung ist inbesondere bei Codes sinnvoll (ggf. bei Datumsangaben, selten bei Quantitäten), um z. B. als Defaultwerte verwendete Ausprägungen ($\neq$ *NULL*) zu identifizieren. Prozentuale Angaben sind absoluten Angaben vorzuziehen. Die Ausgabe sollte in einem nach Häufigkeit sortierten Report erfolgen. Dieser ist interaktiv auf Auffälligkeiten hin zu untersuchen.

- *Wie hoch ist die durchschnittliche Häufigkeit von Attributwerten?*
  Messergebnisse können hier – analog zu Mittelwert und Standardabweichung – in Kontrollkarten eingetragen werden. Bei Über- oder Unterschreiten vordefinierter Grenzwerte (wiederum prozentual statt absolut) sind entsprechende Maßnahmen zu ergreifen. Eine Anwendung ist wiederum bei Codes und ggf. Datumsangaben sinnvoll (Beispiel: erwartete Gleichverteilung von Monatsangaben über das Jahr $\Rightarrow$ Häufigkeit jeder Monatsausprägung $\approx 8.3\%$).

- *Wie lauten die minimalen und maximalen Attributausprägungen?*
  Diese Prüfung eignet sich für Datums-/Zeitangaben und Quantitäten. Eine Analyse über Kontrollkarten ist analog zum vorigen Fall denkbar (Beispiele: kein Buchungsdatum *vor* dem Zeitpunkt der Aufnahme des Geschäftsbetriebs, kein Gehalt *über* dem Maximalgehalt).

7.6 Phase 4: Domänenspezifische Konsistenzprüfung/Nachbearbeitung

An die statistischen Prüfungen aus Phase 3 schließen sich nun – unter Hinzunahme domänenspezifischen Wissens – Prüfungen auf Tupelebene an, die Inkonsistenzen im Transformationsbereich aufspüren (vgl. Abschnitt 6.1.2). Gegenstand einer solchen Prüfung ist wiederum eine Teilmenge T der Instanz einer Relation R_{TB}, also $T \subset I(R_{TB})$.

7.6.1 Spezifizierung

O. B. d. A. wird angenommen, dass das domänenspezifische Wissen regelorientiert repräsentiert ist. Andere Repräsentationsformen wie Lookup-Tabellen können auf eine regelorientierte Repräsentation abgebildet werden. Jede Regel weise die folgende Form auf:

```
WENN <Bedingung>, DANN <Aktion> (Schweregrad <x>)
```

Dabei spezifiziert `<Bedingung>` einen inkonsistenten Sachverhalt, auf den ein Tupel $t \in T$ geprüft werden soll. Tupelübergreifende Inkonsistenzen können durch Nutzung von Sichtenkonzepten (z. B. Join über mehrere Relationen) ebenfalls einbezogen werden. Wird `<Bedingung>` bzgl. t zu $true$ ausgewertet, wird die in `<Aktion>` spezifizierte Anweisungssequenz ausgeführt, z. B. zur Erzeugung einer interaktiv zu bearbeitenden Fehlermeldung. Des weiteren kann über einen Wert $0 < x \leq 1$ der Schweregrad der Inkonsistenz festgelegt werden. Wie in Abschnitt 6.1.2 gezeigt wurde, kann x zu einer Bewertung von Datenprodukten bzgl. des Datenqualitätsmerkmals „Konsistenz“ herangezogen werden.

Letztendlich ist es dabei unerheblich, ob Regeln explizit von einem Domänenexperten definiert worden sind oder zur Laufzeit – z. B. mittels Data Mining-Techniken – an Hand des aktuellen Datenbestandes im Zielbereich ermittelt werden.

Umgesetzt auf eine konkrete Regelsprache (hier exemplarisch OPS-5 [BFKM85]) könnte eine Beispielregel (hier aus der Domäne der Krebsregistrierung) wie folgt aussehen:

```
1:  ( defrule Dignitaetspruefung default
2:    ?tupel:( Meldungen
3:             dignitaet = 3
4:             t_val Eq_Str "0" )
5:  -> {
6:    ?tupel->IncWgt( 0.7 );
7:    ?tupel->AddLog( "FEHLER: Die Dignitaet ist 3, dann darf t_val
8:              nicht den Wert ’0’ aufweisen. (Schweregrad 0.7)" ); }
9:  )
```

Der Bedingungsteil dieser Regel erstreckt sich von Zeile 2 bis 4. Der dadurch beschriebene inkonsistente Sachverhalt lautet: *„Es existiert ein Tupel in der Relation Meldungen mit Dignität 3 und T-Wert ’0’.“* Im Aktionsteil (Zeile 6 bis 8) wird ein zugehöriger Schweregrad festgelegt und eine Fehlermeldung erzeugt. Ein Vergleich verschiedener Regel-/Triggersprachen war nicht Gegenstand der Arbeit. OPS-5 wurde lediglich gewählt, da bereits praktische Erfahrungen mit dieser Sprache vorlagen.

Sei $Rul(R_{TB})$ die Menge aller auf R_{TB} definierten (Geschäfts-)Regeln, und bezeichne $rul(t)$ die Anwendung einer Regel $rul \in Rul(R_{TB})$ auf ein Tupel $t \in T$, wobei gelte:

$$rul : Dom(glb_id) \times Dom(A_1) \times ... \times Dom(A_m) \mapsto]0, 1] \tag{7.6}$$

$$rul(t) = \begin{cases} \text{Schweregrad } x \text{ des Verstoßes} & \text{falls } t \text{ die in } rul \text{ angegebene Bedingung erfüllt} \\ 0 & \text{sonst.} \end{cases} \tag{7.7}$$

Da das Primärschlüsselattribut glb_id keine Semantik trägt, sollten sich keine Regeln darauf beziehen.

7.6.2 Ausführung

Die Strategie der Anwendung von $Rul(R_{TB})$ auf T ergibt sich aus dem jeweils verwendeten Inferenzmechanismus (für eine Übersicht zu Inferenzmechanismen siehe [Tap99]). Grundsätzlich sind alle Tupel gegen alle Regeln zu prüfen, allerdings kann durch eine geeignete Implementierung (z. B. effizientes Pattern Matching zwischen Tupel und Regelbedingung) u. U. eine signifikante Reduzierung der Laufzeitkomplexität erreicht werden. Allgemein gilt:

$\forall t \in T$ `do`

  $\forall rul \in Rul(R_{TB})$ `do`

    `if` $rul(t) > 0$

      $CreateMessage(t, rul.action, rul(t))$

    `fi`

  `od`

`od`

Dabei sei $CreateMessage$ eine Funktion, die zu einem Tupel t eine Warn- bzw. Fehlermeldung erzeugt, die sich aus dem Aktionsteil der jeweiligen Regel ergibt ($rul.action$), und diese zusammen mit dem Schweregrad an einer zentralen Stelle ablegt. Nach Abschluss der Konsistenzprüfung sind diejenigen Tupel, zu denen eine Warn- bzw. Fehlermeldung erzeugt wurde, interaktiv nachzubearbeiten (ggf. parallel durch mehrere Nutzer). Eine automatisierte Beseitigung von Inkonsistenzen ist problematisch, da oft nicht ohne umfassendes Domänen- und Kontextwissen festzustellen ist, in welchem Attribut der Fehler liegt. In der Beispielregel aus Abschnitt 7.6.1 kommen sowohl `dignitaet` als auch `t_val` in Frage.

Im Rahmen der Nachbearbeitung müssen ggf. Rückfragen an die Datenquellen gestellt werden, aus denen die inkonsistenten Tupel stammen. Diese Rückfragen sollten möglichst ohne Medienbruch, d. h. elektronisch erfolgen. Die adressierten Datenquellen haben den Rückfragen nachzugehen, ggf. Korrekturen an den Quelldaten vorzunehmen und schließlich die Rückfragen zu beantworten. Dies kann auf unterschiedliche Arten vollzogen werden:

- *Neulieferung von korrigierten Tupeln*
  Dies ist i. Allg. nur sinnvoll, wenn zu einer Datenquelle eine hinreichend große Anzahl von korrigierten Tupeln vorliegt. Diese Anzahl ist ausschließlich im konkreten Anwendungskontext zu quantifizieren, da Einflussfaktoren wie z. B. Zeitaufwand und Kosten für eine Neulieferung berücksichtigt werden müssen.

  Die korrigierten Tupel werden gemeinsam im jeweiligen Quelldatenpuffer abgelegt und in einem neuen Durchlauf des Integrationsprozesses (beginnend bei Phase 1) einem wiederholten Integrationsversuch unterzogen. Die ursprünglichen Tupel sind rechtzeitig aus R_{TB} zu löschen, gleiches gilt für die zwischengespeicherten Warn- bzw. Fehlermeldungen zu diesen Tupeln. Der „alte“ Integrationsprozess wird ohne die rückgefragten Tupel fortgesetzt; diese werden im nachfolgenden Integrationsprozess „nachgeliefert“. Problematisch ist dabei die Behandlung von Tupelabhängigkeiten, da das Löschen eines Tupels auch das Löschen der davon abhängigen Tupel erfordert und andererseits die Neuanlieferung eines Tupels auch diejenigen Tupel einschließen muss, von denen es abhängt.

- *Lieferung von Update-Anweisungen*
  Bei einer geringen Anzahl von korrigierten Tupeln pro Datenquelle empfiehlt sich eine Strategie, bei der eine Datenquelle nicht das komplette, korrigierte Tupel (inkl. Tupeln, zu denen

Abhängigkeiten existieren) liefert, sondern lediglich eine detaillierte Anweisung, die festlegt, wie das bereits in R_{TB} vorliegende Tupel zu manipulieren ist, um die Inkonsistenz zu beseitigen. Dabei ist zu beachten, dass sich von einer Quelle gelieferte Update-Anweisungen i. Allg. auf die Datenstrukturen der Quelle beziehen werden und daher zunächst an die Datenstrukturen des Transformationsbereichs angepasst werden müssen. Liegen Anweisungen als SQL-UPDATE-Statements vor, kann die Datenmanipulation auch automatisiert (im Batchbetrieb) erfolgen. Ansonsten sind die Update-Anweisungen interaktiv umzusetzen. Anschließend sind sowohl die Update-Anweisungen als auch die zugehörigen Warn- bzw. Fehlermeldungen zu löschen. Eine erneute Durchführung der Phasen 1 bis 4 des CLIQ-Vorgehensmodells, wie sie bei der Neulieferung von korrigierten Tupeln erforderlich ist, entfällt bei diesem Ansatz. Durch die nachträglichen Änderungen können allerdings SPC-Ergebnisse obsolet werden.

- *Bestätigung der (vermeintlichen) Inkonsistenz*
  Stellt eine Datenquelle fest, dass es sich bei einer fraglichen Attributwertkombination um keine Inkonsistenz, sondern um ein Phänomen des Diskursbereichs handelt, so hat sie den Fall gegenüber der rückfragenden Stelle zu bestätigen. Geschieht dies in angemessener (elektronischer) Form, so können die Bestätigungen automatisiert bearbeitet werden, indem die referenzierten Warn- bzw. Fehlermeldungen gelöscht werden und die Inkonsistenz damit als bearbeitet gilt.

- *Keine Klärung möglich*
  Es kann vorkommen (z. B. aufgrund von Ressourcenmangel), dass auch die zuständige Datenquelle nicht entscheiden kann, ob ein Tupel tatsächlich inkonsistent ist, welche Attributwerte fehlerhaft sind bzw. wie die korrekten Werte lauten. Die Datenquelle hat dann der rückfragenden Stelle mitzuteilen, dass der fragliche Fall nicht geklärt werden konnte. Letztere muss nun interaktiv entscheiden, wie mit dem betroffenen Tupel verfahren werden soll. Folgende Alternativen sind pro Einzelfall gegeneinander abzuwägen:
  - Tolerieren der Inkonsistenz
  - Schätzen von Werten und anschließendes Manipulieren des Tupels
  - Löschen des Tupels.

  Zugehörige Warn- bzw. Fehlermeldungen sind jeweils anschließend zu löschen. Alle drei genannten Fälle sind unbefriedigend, da i. Allg. Qualitätsmängel zurückbleiben (Inkonsistenz, ggf. falsch geschätzte Werte, fehlende Tupel). Spätere Nutzer der Daten sind über diese Mängel zu informieren.

Sind alle aktuell zu R_{TB} vorliegenden Warn- bzw. Fehlermeldungen bearbeitet, kann mit Phase 5 fortgefahren werden. Beispiele für die regelbasierte Umsetzung domänenspezifischer Konsistenzprüfungen im Anwendungsbereich der epidemiologischen Krebsregistrierung finden sich in [HP99].

7.7 Phase 5: Record Linkage

Sei $R_{ZB}\{glb_id, A_1, ..., A_m\}$ eine Relation des Zielbereichs und $R_{TB}\{glb_id, A_1, ..., A_m\}$ die zugehörige Relation des Transformationsbereichs. In dieser Phase des Vorgehensmodells geht es darum, Duplikate (redundante Repräsentanten derselben Entität des Diskursbereichs) zwischen $I(R_{TB})$ und $I(R_{ZB})$ sowie innerhalb von $I(R_{TB})$ aufzuspüren. Diese Aufgabe wird aus Effizienzgründen in zwei Teilschritten durchgeführt: In einem ersten Schritt (schlüsselbasiertes Linkage) werden Tupel im Transformationsbereich, die inkrementelle Updates darstellen, an Hand ihrer lokalen Schlüssel mit den korrespondierenden Tupeln des Zielbereichs assoziiert. Im zweiten Schritt (probabilistisches Linkage, vgl. Abschnitt 6.2.5) findet eine abweichungstolerante Verknüpfung an Hand von Nicht-Schlüsselattributen statt, um auch (ähnliche) Tupel aus unterschiedlichen Quellen zusammenzuführen. Durch das Vorschalten des (sehr effizienten) schlüsselbasierten Linkage wird die Anzahl der im

(vergleichsweise ineffizienten) probabilistischen Linkage zu betrachtenden Tupel minimiert. Beide Schritte werden nun im Detail erläutert.

7.7.1 Schlüsselbasiertes Linkage

Sei $loc_{R_{TB}} : Dom(glb_id) \mapsto Dom(local_id)$ eine Funktion, die zu einem globalen Schlüssel (Attribut glb_id) eines Tupels aus $I(R_{TB})$ den zugehörigen lokalen Schlüssel (Attribut $local_id$) zurückliefert. An dieser Stelle wird von der Tatsache abstrahiert, dass ein lokaler Schlüssel sich aus mehreren Attributwerten zusammensetzen kann und dass er außerdem mit einem bestimmten QDP sowie einer QDP-Relation assoziiert ist.

Sei $loc_{R_{ZB}} : Dom(glb_id) \mapsto P(Dom(local_id))$ eine Funktion, die zu einem globalen Schlüssel eines Tupels $t \in I(R_{ZB})$ die Menge der lokalen Schlüssel derjenigen Tupel zurückliefert, aus denen t (durch Merging, siehe Phase 6) hervorgegangen ist.

Das schlüsselbasierte Linkage läuft nun wie folgt ab:

$\forall t_{TB} \in I(R_{TB}) : \exists t_{ZB} \in I(R_{ZB}) : loc_{R_{TB}}(t_{TB}) \in loc_{R_{ZB}}(t_{ZB})$ `do`

`if` $|loc_{R_{ZB}}(t_{ZB})| = 1$

Verschiebe t_{TB} in den Konsolidierungsbereich

`else`

Markiere t_{TB} und t_{ZB} als „zu mergen“[1]

`fi`

`od`

Gegenstand des schlüsselbasierten Linkage sind alle Tupel in $I(R_{TB})$, die ein inkrementelles Update eines bereits in $I(R_{ZB})$ enthaltenen Tupels darstellen (erkennbar an Hand der übereinstimmenden lokalen Schlüssel). Falls das ZB-Tupel aus mehreren Quelltupeln entstanden ist (erkennbar an der Anzahl der beteiligten lokalen Schlüssel), ist ein interaktives Merging von TB- und ZB-Tupel erforderlich (siehe Phase 6). Ansonsten wird das TB-Tupel in den KB verschoben, so dass es in Phase 9 das bisherige ZB-Tupel ersetzen kann (konkret: durch Überschreiben oder Historisieren des ZB-Tupels).

Bei der Implementierung des schlüsselbasierten Linkage sollte die Tupelauswahl aus Effizienzgründen durch Ausklammerung derjenigen Tupel, die im Metaattribut $hist_state$ den Wert new aufweisen (siehe Abschnitt 7.2), realisiert werden. Die Behandlung von Tupeln, die quellenseitig gelöscht wurden ($hist_state = delete$, Spezialfall eines inkrementellen Update), wird separat in Abschnitt 7.13 diskutiert.

7.7.2 Probabilistisches Linkage

Bevor ein probabilistisches Linkage (zum theoretischen Hintergrund siehe Abschnitt 6.2.5) stattfinden kann, ist eine Konfiguration des Verfahrens zu spezifizieren. In dieser ist definiert, welche Attribute mittels welcher Vergleichsoperatoren in den Linkage-Prozess einfließen.

[1] Konkret werden t_{TB} und t_{ZB} als „Matchgruppe“ in die Menge GM aufgenommen (siehe Abschnitt 7.7.2) und so in Phase 6 dem interaktiven Merging zugeführt.

Spezifizierung

Eine Konfiguration $Conf_{PLink}(R_{ZB})$ zu einer Relation R_{ZB} ist definiert als eine Menge von k Umläufen. Die Anzahl k ist dabei vom DWS-Administrator bzw. Datenverantwortlichen je nach Anwendungsbereich festzulegen (typischerweise gilt: $2 \leq k \leq 10$). Ein Umlauf $pass_j \in Conf_{PLink}(R_{ZB})$ $(1 \leq j \leq k)$ ist definiert als ein Tripel $(BVSet_j, MVSet_j, co_j)$, wobei $co_j > 0$ einen Schwellenwert (Cut-Off), $BVSet_j \subset \{A_1, ..., A_m\}$ eine Menge von Blockvariablen und $MVSet_j$ eine Menge von Matchvariablen mit assoziierten Vergleichsoperatoren darstellt.

Formal ist $MVSet_j$ definiert als Menge von Paaren (MV, op), wobei $MV \in \{A_1, ..., A_m\}$ und $op : Dom(MV) \times Dom(MV) \mapsto [0, 1]$ ein Operator ist, der die Ähnlichkeit zweier Attributwerte $t_1.MV$ und $t_2.MV$ berechnet (z. B. Editierabstand zweier Zeichenketten, Differenz zweier numerischer Werte etc.). Es gelte $t.MV\ op\ t.MV = 1$.

Ausführung

Im Rahmen der Durchführung des probabilistischen Linkage spielen zwei attributbezogene Wahrscheinlichkeiten, die m- und die u-Wahrscheinlichkeit, eine bedeutende Rolle (zur zu Grunde liegenden Theorie siehe [Jar89]):

$$m(A) = P\{t_1.A = t_2.A \mid t_1 \text{ und } t_2 \text{ beschreiben dieselbe Entität}\} \tag{7.8}$$

$$u(A) = P\{t_1.A = t_2.A \mid t_1 \text{ und } t_2 \text{ beschreiben verschiedene Entitäten}\} \tag{7.9}$$

Es gilt:

$$m(A) = 1 - \text{Fehlerwahrscheinlichkeit im Attribut } A \tag{7.10}$$

$$u(A) \approx \text{Wahrscheinlichkeit einer zufälligen Übereinstimmung zweier Tupel im Attribut } A \tag{7.11}$$

Beide Wahrscheinlichkeiten sind vor der Durchführung eines Record Linkage-Prozesses zu schätzen. Die u-Wahrscheinlichkeit kann relativ einfach durch eine Analyse der Attributwertverteilung von A geschätzt werden. Eine Schätzung der m-Wahrscheinlichkeit ist mit Hilfe des in [Jar89] vorgestellten *EM-Algorithmus* möglich.

Ziel des Linkage ist es, automatisiert eine Menge M von Matches (d. h. Zuordnungen zwischen Tupeln) zu berechnen. Jeder Match ist beschrieben durch ein 4-Tupel (t_1, t_2, j, g_{comp}). Dabei sind t_1 und t_2 die einander zugeordneten Tupel, j spezifiziert den Umlauf, in dem die Zuordnung erfolgt ist, und g_{comp} ist ein zusammengesetztes Gewicht („composite weight“), das den Grad der Ähnlichkeit von t_1 und t_2 beschreibt. Das Linkage selbst läuft wie folgt ab (siehe auch [Jar89]):

$M := \varnothing$

$\forall t_{TB} \in I(R_{TB})$ do

  $\forall j \in \{1, ..., k\}$ do

    $\forall t \in I(R_{TB}) \cup I(R_{KB}) \cup$
    $\{t_{ZB} \in I(R_{ZB}) \mid \nexists t_{KB} \in I(R_{KB}) : t_{KB}$ ist Update von $t_{ZB}\}$:
    $\forall BV \in BVSet_j : t_{TB}.BV = t.BV$ do

      $\forall (MV, op) \in MVSet_j$ do

        $similarity := t_{TB}.MV\ op\ t.MV$

        if $IsSufficient(similarity)$

$g_{full}(MV) := log_2 \frac{m(MV)}{u(MV)}$

$g(MV) := Decrease(g_{full}(MV), similarity)$

else

$g_{full}(MV) := log_2 \frac{1-m(MV)}{1-u(MV)}$

$g(MV) := Increase(g_{full}(MV), similarity)$

fi

od

$g_{comp} := \sum_{MV \in MVSet_j} g(MV)$

if $g_{comp} \geq co_j$

$M := M \cup (t_{TB}, t, j, g_{comp})$

fi

od

od

od

Für jedes Tupel t_{TB} der TB-Relation werden k Umläufe vorgenommen. In Umlauf j werden alle Tupel der TB-, und KB-Relation untersucht, die in allen für Umlauf j spezifizierten Blockvariablen exakt mit t_{TB} übereinstimmen. Zusätzlich werden aus der ZB-Relation diejenigen Tupel betrachtet, zu denen kein Update im KB existiert. Für jedes übereinstimmende Tupel wird an Hand jeder in Umlauf j spezifizierten Matchvariable MV ein Ähnlichkeitswert zu t_{TB} berechnet. Falls zwei Attribute als eher ähnlich als unähnlich eingestuft werden (Funktion $IsSufficient$), wird mit Hilfe der m- und u-Wahrscheinlichkeiten ein Gewicht $g_{full}(MV)$ berechnet:

$$g_{full}(MV) := log_2 \frac{m(MV)}{u(MV)} \tag{7.12}$$

Da i. Allg. die Wahrscheinlichkeit, dass zwei Tupel keinen Fehler im Attribut MV aufweisen, größer ist als die Wahrscheinlichkeit einer zufälligen Übereinstimmung zweier Werte in MV und damit $m(MV) > u(MV)$ gilt, folgt hier $\frac{m(MV)}{u(MV)} > 1$ und damit $g_{full}(MV) > 0$.

Im Falle der Unähnlichkeit zweier Attributwerte erfolgt die Gewichtsberechnung wie folgt:

$$g_{full}(MV) := log_2 \frac{1 - m(MV)}{1 - u(MV)} \tag{7.13}$$

Wegen $m(MV) > u(MV)$ gilt hier $\frac{1-m(MV)}{1-u(MV)} < 1$ und damit $g_{full}(MV) < 0$.

Durch die Logarithmierung des jeweiligen Quotienten wird also erreicht, dass eine Ähnlichkeit zweier Attributwerte in einem positiven und eine Unähnlichkeit in einem negativen Gewicht resultiert.

Da der Übergang von Ähnlichkeit zu Unähnlichkeit oft kontinuierlich ist (z. B. bei Zeichenketten), wird das errechnete Gewicht relativiert, d. h. ein positives Gewicht wird bei nicht vollständiger Übereinstimmung verringert (Funktion $Decrease$), ein negatives Gewicht bei nicht vollständiger Abweichung erhöht (Funktion $Increase$), ohne dass sich dabei jedoch das Vorzeichen ändern darf. Das

Ausmaß der Verringerung bzw. Erhöhung hängt von dem errechneten Ähnlichkeitsgrad ($similarity$) ab.

Aus den sich ergebenden attributbezogenen Gewichten wird dann per Summierung ein zusammengesetztes Gewicht g_{comp} für das betrachtete Tupelpaar berechnet. Dies liegt offensichtlich umso höher im positiven Bereich, je größer die attributbezogenen Ähnlichkeiten ausfallen. Bei zwei sehr unähnlichen Tupeln liegt g_{comp} weit im negativen Bereich.

Falls g_{comp} den in Umlauf j spezifizierten Schwellenwert co_j erreicht oder übersteigt, gilt das betrachtete (mit Gewicht und Umlauf attributierte) Tupelpaar als Match und wird in die Menge M aufgenommen. Ansonsten wird die Assoziation zwischen den Tupeln verworfen.

Um auch transitive Zuordnungen explizit zu machen, wird in einem Nachbearbeitungsschritt – ausgehend von der Menge M von Matches (4-Tupel) – der transitive Abschluss von Zuordnungen in Form einer Menge von sog. *Matchgruppen* berechnet, wobei jede Matchgruppe G eine Menge von potentiell zusammengehörigen Matches darstellt. Alle Matchgruppen werden in einer Menge GM zusammengefasst:

$GM := \varnothing$

$\forall m_1 = (t_{1,1}, t_{2,1}, j_1, g_{comp,1}) \in M$ `do`

`if` $\exists G \in GM : \exists m_2 = (t_{1,2}, t_{2,2}, j_2, g_{comp,2}) \in G :$
$(t_{1,2} = t_{1,1} \vee t_{1,2} = t_{2,1} \vee t_{2,2} = t_{1,1} \vee t_{2,2} = t_{2,1})$

$G := G \cup \{m_1\}$

`else`

$G := \{m_1\}$

$GM := GM \cup \{G\}$

`fi`

`od`

Zu jedem Match $(t_{1,1}, t_{2,1}, j_1, g_{comp,1})$ in M wird geprüft, ob bereits eine Matchgruppe G existiert, in der ein Match $(t_{1,2}, t_{2,2}, j_2, g_{comp,2})$ enthalten ist, bei dem mindestens eines der beiden Tupel $t_{1,2}$ und $t_{2,2}$ mit $t_{1,1}$ oder $t_{2,1}$ übereinstimmt. Ist dies der Fall, wird der erstgenannte Match in G aufgenommen. Ansonsten wird eine neue Matchgruppe erzeugt, der Match dort eingefügt, und GM um die neue Matchgruppe erweitert.

Die Menge GM stellt nun die Eingabe für die nachfolgende Phase 6 des CLIQ-Vorgehensmodells dar. Eine exemplarische Umsetzung eines probabilistischen Record Linkage im Anwendungsbereich der epidemiologischen Krebsregistrierung wird in [Hin99a] behandelt.

7.8 Phase 6: Merging

Die in Phase 5 einander automatisiert zugeordneten Tupel werden nun nach bestimmten Kriterien miteinander verschmolzen, um einer unbeabsichtigten Redundanz im Zieldatenbestand entgegen zu wirken. Dabei ist darauf zu achten, dass keine Homonym- oder Synonymfehler (vgl. Abschnitt 2.2.3) begangen werden. Ausgangspunkt für das Verschmelzen von Tupeln ist die in Phase 5 berechnete Menge GM von Matchgruppen. Deren Elemente stammen teils aus dem schlüsselbasierten, teils aus dem probabilistischen Linkage.

Zur Vereinfachung der Darstellung wird angenommen, dass gefundene Zuordnungen immer bestätigt werden, d. h. dass alle an einer Matchgruppe beteiligten Tupel zu einem einzigen Tupel verschmolzen werden sollen. In der Praxis ist hingegen zumindest in Anwendungsbereichen, in denen es auf eine hohe Redundanzfreiheit ankommt, ein interaktives Review von Matchgruppen erforderlich, wobei eine Matchgruppe ggf. manuell in mehrere Teilgruppen aufgeteilt wird. Die exakte Quantifizierung des geforderten Grades der Redundanzfreiheit muss im konkreten Anwendungskontext vorgenommen werden.

Pro Attribut ist nun mit Hilfe eines (domänenspezifischen) Regelwerks zu bestimmen, welcher Attributwert der Ausgangstupel in das Zieltupel übernommen werden soll bzw. wie sich der Zielwert aus den Einzelwerten ergibt (z. B. Mittelung). Das aus der Verschmelzung hervorgegangene Tupel t_{merged} wird schließlich in den Konsolidierungsbereich geschrieben. Dabei erhält t_{merged} den Primärschlüssel eines der beteiligten Tupel aus dem Transformationsbereich. Falls ein Tupel aus dem Zielbereich an der Verschmelzung beteiligt war, wird dessen Primärschlüssel im Metaattribut old_id (siehe Abschnitt 7.2) zu t_{merged} zwischengespeichert, um eine spätere Zuordnung zwecks Überschreiben bzw. Historisieren zu ermöglichen.

Sei MR nun eine Menge von Merging-Regeln, wobei jede Regel einer Funktion mr_i entspricht, die zu einer Matchgruppe einen (Ziel-)Wert im Attribut A_i einer Relation $R\{glb_id, A_1, ..., A_m\}$ zurückliefert:

$$mr_i : GM \mapsto Dom(A_i) \tag{7.14}$$

Die Merging-Regeln sollten alle Attribute von R abdecken (mit Ausnahme des Primärschlüsselattributs). Zur Berechnung von $mr_i(G)$ mit $G \in GM$ können verschiedene Informationen herangezogen werden:

- Domänenunabhängige Informationen
  - das Gewicht g_{comp}, das mit der Zuordnung zweier Tupel t_1 und t_2 assoziiert ist (ggf. unter Hinzunahme des Umlaufs j, in dem eine Zuordnung zwischen t_1 und t_2 hergestellt wurde)
  - die Zeitnähe von $t.A_i$ (z. B. immer den aktuellsten Wert übernehmen)
  - die Häufigkeit des Auftretens eines Attributwerts in den Tupeln aus G (z. B. immer den häufigsten Wert übernehmen).
- Domänenabhängige Informationen
  - die Datenquelle $Q(t)$, aus der ein Tupel t stammt (attributspezifische Quellenpriorität)
  - der Informationsgehalt des Attributwerts $t.A_i$ relativ zum Informationsgehalt von Werten desselben Attributs in anderen Tupeln aus G
  - die Wahrscheinlichkeit des Auftretens von $t.A_i$ (z. B. immer den wahrscheinlichsten Wert übernehmen)
  - Domänenspezifische Berechnungen, die ggf. Kombinationen von Attributwerten einbeziehen oder aus Einzelwerten einen verdichteten Wert aggregieren.

Der Merging-Vorgang läuft wie folgt ab: Für jede Matchgruppe aus GM wird ein neues Tupel im Konsolidierungsbereich (genauer: in R_{KB}) erzeugt (Funktion $CreateNewTuple$), das die aus der Verschmelzung der einzelnen beteiligten Tupel hervorgegangenen Werte aufnimmt. Diese werden mit Hilfe der in MR enthaltenen Merging-Regeln bestimmt. Nach Abarbeitung einer Matchgruppe werden die beteiligten Tupel im Transformationsbereich als „gelöscht“ markiert (Funktion $DeleteTuple$).

Die nach Abarbeitung aller Matchgruppen in R_{TB} verbliebenen Tupel, also diejenigen, die im Rahmen des Record Linkage nicht mit anderen Tupeln assoziiert wurden, werden schließlich 1:1 in den Konsolidierungsbereich verschoben (Funktionen $InsertTuple$ und $DeleteTuple$). Die physische Löschung der TB-Tupel sollte – analog zu den Ausführungen in Abschnitt 7.4.2 – erst nach Abschluss von Phase 9 erfolgen, um im Rahmen der Lenkung mangelhafter Daten (Phase 8) ein einfaches Wiederaufsetzen zu ermöglichen.

Formal ist das Merging folgendermaßen spezifiziert:

$\forall G \in GM$ do

$t_{new} := CreateNewTuple(R_{KB})$

$\forall i \in \{1, ..., m\}$ do

$t_{new}.A_i := mr_i(G)$

od

$\forall (t_1, t_2, j, g_{comp}) \in G$ do

$DeleteTuple(R_{TB}, t_1)$

$DeleteTuple(R_{TB}, t_2)$

od

od

$\forall t \in I(R_{TB})$ do

$InsertTuple(R_{KB}, t)$

$DeleteTuple(R_{TB}, t)$

od

Falls ein Tupel t bereits gelöscht wurde, habe der Aufruf $DeleteTuple(\ldots, t)$ keine Wirkung.

Nach erfolgreichem Abschluss der Merging-Phase liegen alle in die Relation R_{ZB} zu integrierenden Tupel im Konsolidierungsbereich vor. Bevor sie in den Zielbereich übertragen werden, muss zunächst sichergestellt werden, dass alle im Vorfeld spezifizierten Qualitätsanforderungen an $I(R_{ZB})$ auch nach Durchführung der Übertragung noch erfüllt sein werden (im Sinne einer Invariante).

7.9 Phase 7: Qualitätsmessung und -analyse

Gegenstand jeder Qualitätsmessung zu einer Relation R – streng genommen zu einem Relationenpaar $R_{KB}\{glb_id, A_1, ..., A_m\}$ und $R_{ZB}\{glb_id, A_1, ..., A_m\}$ – ist folgende Tupelmenge TM_R:

$$TM_R = I(R_{KB}) \cup \{t_{ZB} \in I(R_{ZB}) |\ \nexists t_{KB} \in I(R_{KB}) : t_{KB} \text{ ist Update von } t_{ZB}\} \tag{7.15}$$

Es werden also alle Tupel der KB-Relation betrachtet sowie diejenigen Tupel der ZB-Relation, zu denen kein Update in der KB-Relation existiert. Dies entspricht einer Sicht auf den Zielbereich, wie er sich nach einer Freigabe der Tupel der KB-Relation und deren Übertragung in den Zielbereich darstellen würde.

7.9.1 Spezifizierung von Qualitätsplänen

Sei QCM die Menge der betrachteten Datenqualitätsmerkmale (siehe Abschnitt 3.2):

$$QCM = \{\text{Konsistenz, Redundanzfreiheit, Vollständigkeit, Genauigkeit, } \dots\} \tag{7.16}$$

TM_R soll gegen eine Menge QPM von a priori spezifizierten Qualitätsplänen, die Anforderungen an Daten festschreiben, geprüft werden. Jeder Qualitätsplan qp ist ein 7-Tupel $(cust, cond, n, AV, g, qc, mdv)$, wobei für die einzelnen Bestandteile von qp gilt:

1. $cust$ ist eine Kundenrolle (allg.: eine Rolle für eine Interessengruppe), die die durch qp spezifizierte Anforderung stellt.
2. $cond$ ist ein boolescher Ausdruck über $\{A_1, ..., A_m\}$ zur Selektion einer Teilmenge von Tupeln aus TM_R (entspricht einer WHERE-Klausel in SQL).
3. AV ist ein n-dimensionaler Vektor ($n \leq m$) von Attributen aus R_{ZB} (bzw. R_{KB}) – ausgenommen glb_id (Projektion).
4. g ist ein n-dimensionaler Vektor von (attributbezogenen) Gewichten $g_i \in]0, 1]$ $(i \in \{1, ..., n\})$. Mit Hilfe dieser Gewichte ist es möglich, den Einfluss von Attributen auf die Qualitätsbewertung zu steuern.
5. qc ist ein Datenqualitätsmerkmal mit $qc \in QCM$.
6. $mdv \in]0, 1]$ ist ein Sollwert, der auf der durch $cond$ und AV spezifizierten und mittels g gewichteten Tupelmenge bzgl. einer zu Grunde liegenden Metrik (siehe Abschnitt 6.1) für Datenqualitätsmerkmal qc mindestens erreicht werden muss, damit die Anforderung des Plans qp als erfüllt gilt (mdv = „minimal desired value“).

7.9.2 Spezifizierung domänenspezifischer Datenqualitätsindikatoren

Zusätzlich zu der im vorangegangenen Abschnitt behandelten Messung bzgl. vorgegebener Datenqualitätsmerkmale sollte es möglich sein, domänenspezifische Datenqualitätsindikatoren (vgl. Abschnitt 6.3) zu spezifizieren und auszuwerten. Ein Datenqualitätsindikator wird über einen neuen Typ eines Qualitätsplans (siehe Abschnitt 7.9.1), im Folgenden als *Indikatorplan* bezeichnet, spezifiziert. Jeder Indikatorplan ip ist ein 8-Tupel $(cust, cond, n, AV, s, g, f, mdv)$, wobei $cust$, $cond$ und AV analog zu einem Qualitätsplan definiert sind. Für die übrigen Komponenten von ip gelte:

1. s ist ein n-dimensionaler Vektor von SQL-SELECT-Statements, wobei jedes Statement (Vektorelement) s_i $(i \in \{1, ..., n\})$ sich auf die durch $cond$ und AV spezifizierte Tupelmenge beziehen und als Ergebnis genau einen numerischen Wert $v_i \in \mathbb{R}$ zurückliefern muss (z. B. mittels Operatoren wie `COUNT`, `SUM`, `AVG` etc.).
2. g ist ein n-dimensionaler Vektor von Gewichten $g_i \in \mathbb{R}$ $(i \in \{1, ..., n\})$.
3. $f : \mathbb{R}^n \mapsto \mathbb{R}_0^+$ ist eine (von einem Domänenexperten zu spezifizierende) Funktion, die zu gegebenen gewichteten Anfrageergebnissen eine Zahl liefert, die einem Messwert bzgl. der Indikatorspezifikation entspricht. Dabei sollte die Funktion f so gewählt werden, dass sie sich streng monoton steigend bei steigendem Erfüllungsgrad der untersuchten Qualitätseigenschaft verhält, d. h. eine *hohe* Datenqualität mit einem *hohen* Indikatorwert assoziiert.

4. $mdv \in \mathbb{R}_0^+$ ist ein Sollwert, der auf der durch $cond$ und AV spezifizierten Tupelmenge bzgl. der Funktion f, angewandt auf die mit den Elementen von g gewichteten Ergebnisse der Anfragen in s, mindestens erreicht werden muss, damit die Anforderung des Indikatorplans ip als erfüllt gilt.

Wenn in den nachfolgenden Ausführungen von Qualitätsplänen die Rede ist, sind dabei Indikatorpläne eingeschlossen.

7.9.3 Ausführung

Die durch einen Qualitätsplan spezifizierte Tupelmenge wird im Folgenden wiederum als „Datenobjekt“ bezeichnet. Ein Datenobjekt stellt also den Gegenstand einer Qualitätsmessung dar. Sei D die Menge der mittels Projektion (über AV) und Selektion (über $cond$) auf TM_R spezifizierbaren Datenobjekte.

Sei $m : QCM \times D \times]0,1]^n \mapsto [0,1]$ eine Funktion, die pro Datenqualitätsmerkmal $qc \in QCM$ durch ein merkmalsspezifisches Messverfahren – als Operationalisierung einer Metrik für qc – zu implementieren ist (zu Metriken und Messverfahren siehe Kapitel 6). m bildet ein gegebenes Qualitätsmerkmal qc, ein Datenobjekt $d \in D$ sowie einen n-dimensionalen Gewichtsvektor g auf einen Wert $m(qc, d, g)$ ab, der als Indikator für die Qualität von d bzgl. qc dient. Dabei entspricht ein Wert von 1 dem „best case“ und ein Wert von 0 dem „worst case“.

Sei $passed(qp, d) \in \{true, false, unknown\}$ ein Flag, das angibt, ob das Datenobjekt d den Qualitätsplan qp erfüllt oder nicht. Ist die entsprechende Messung noch nicht durchgeführt worden, gilt $passed(qp, d) = unknown$.

Sequentiell werden nun für alle auf R (konkret: R_{KB} und R_{ZB}) definierten Qualitätspläne (inkl. Indikatorplänen) Erfüllungsgrade berechnet. Sei dazu IPM die Menge aller Indikatorpläne und $s_i(d)$ das zu einer Anfrage s_i (i-tes Element eines Vektors s eines Indikatorplans) auf Datenobjekt d gelieferte Ergebnis.

$\forall qp \in QPM \cup IPM$ `do`

$d := GetDataObject(TM_R, qp.cond, qp.AV)$

`if` $qp \in QPM$

$qv := m(qp.qc, d, qp.g)$

`else`

$qv := qp.f(qp.s_1(d)qp.g_1, qp.s_2(d)qp.g_2, ..., qp.s_n(d)qp.g_n)$

`fi`

`if` $qv \geq qp.mdv$

$passed(qp, d) := true$

`else`

$passed(qp, d) := false$

`fi`

`od`

Zunächst wird mittels einer Funktion $GetDataObject$ das Datenobjekt bestimmt, auf das sich ein Qualitätsplan bezieht. Anschließend wird zu diesem Datenobjekt bzgl. des jeweils betrachteten Qualitätsmerkmals ein Qualitätswert bzw. ein Indikatorwert (mit Hilfe der Anfragen im Vektor s, der Gewichte im Vektor g sowie der Funktion f) gemessen und mit dem im Plan spezifizierten Sollwert verglichen. Wurde der Sollwert erreicht oder überschritten, gilt der Qualitätsplan – und damit die darin festgeschriebene Anforderung – als erfüllt, ansonsten als nicht erfüllt. Diese Ergebnisse dienen als Grundlage für die Lenkungs- und Verbesserungsmaßnahmen in Phase 8 des Vorgehensmodells.

Um die Anzahl der durchzuführenden Messungen zu minimieren, können sog. *Dominanzbeziehungen* zwischen Qualitätsplänen ausgenutzt werden.

Definition 7.1 (Dominanzbeziehung): Ein Qualitätsplan $qp_a = (cust_a, cond_a, n_a, AV_a, g_a, qc_a, mdv_a)$ *dominiert* einen Qualitätsplan $qp_b = (cust_b, cond_b, n_b, AV_b, g_b, qc_b, mdv_b)$, kurz $qp_a \rightarrow_{dom} qp_b$, gdw. alle folgenden Bedingungen gelten:

1. qp_a und qp_b beziehen sich auf dieselbe Relation
2. $cond_a = cond_b \wedge AV_a = AV_b$ (d. h. es wird dasselbe Datenobjekt gemessen)
3. $g_a = g_b$ (d. h. die betrachteten Attribute werden gleich gewichtet)
4. $qc_a = qc_b$ (d. h. es wird dasselbe Qualitätsmerkmal gemessen, bei Indikatorplänen muss stattdessen $s_a = s_b \wedge g_a = g_b \wedge f_a = f_b$ gelten)
5. $mdv_a \geq mdv_b$ (d. h. die Anforderung aus qp_a ist mindestens so streng wie die Anforderung aus qp_b).

□

Es gilt:

$$(qp_a \rightarrow_{dom} qp_b \ \wedge \ passed(qp_a, d) = true) \ \Rightarrow \ passed(qp_b, d) = true \tag{7.17}$$

wobei d das zu messende Datenobjekt bezeichnet. In diesem Fall kann also auf eine Messung bzgl. qp_b verzichtet werden. Analog gilt:

$$(qp_a \rightarrow_{dom} qp_b \ \wedge \ passed(qp_b, d) = false) \ \Rightarrow \ passed(qp_a, d) = false \tag{7.18}$$

Diejenigen Tupel aus $I(R_{KB})$, die zu Datenobjekten gehören, welche alle auf ihnen spezifizierten Qualitätspläne erfüllen, können unmittelbar für eine Freigabe (in Phase 9) vorbereitet werden. Dazu werden sie in eine Menge $ForRelease_R$ aufgenommen, die in Phase 9 als Eingabe für den Freigabeprozess dient.

$\forall t \in I(R_{KB})$ `do`

`if` $\nexists qp \in QPM \cup IPM : (t \in GetDataObject(TM_R, qp.cond, qp.AV) \wedge passed(qp, GetDataObject(TM_R, qp.cond, qp.AV)) = false)$

$ForRelease_R := ForRelease_R \cup \{t\}$

`fi`

`od`

Tupel in $I(R_{KB})$, die Datenobjekten angehören, zu denen nicht erfüllte Qualitätspläne existieren, werden in Phase 8 behandelt.

7.9.4 Stichprobenorientierte Qualitätsmessung und -analyse

Bei sehr großen Datenvolumina im Zielbereich können Qualitätsmessungen u. U. sehr ineffizient werden. Ist dies der Fall, empfiehlt sich ein Umschalten auf die Untersuchung von Zufallsstichproben aus dem Zielbereich. Dabei wird von der gemessenen Qualität einer Datenstichprobe auf die Qualität des Gesamtdatenbestandes geschlossen (sog. *statistische Inferenz*). Dies bringt eine erhebliche Reduzierung des erforderlichen Berechnungsaufwands mit sich, allerdings unter Inkaufnahme einer Konfidenz von weniger als 100%. Hinweise zur Durchführung einer statistischen Inferenz finden sich in [CB01].

Zusätzlich zu einer stichprobenorientierten Qualitätsmessung und -analyse sollte der Zielbereich regelmäßig einem Audit unterzogen werden, bei dem der Erfüllungsgrad der Qualitätspläne auf dem Gesamtdatenbestand bestimmt wird. Ziele eines solchen Audits sind:

- Überwachung der Wirksamkeit der stichprobenorientierten Messungen
- Rechtzeitige Erkennung signifikanter Änderungen in den Eigenschaften des Datenbestandes.

Detaillierte Hinweise zur Durchführung von Audits in Informationssystemen gibt [Web99]. Da i. Allg. $|I(R_{KB})| \ll |I(R_{ZB})|$ gilt, können die Daten des Konsolidierungsbereichs ohne Effizienzprobleme vollständig untersucht werden.

7.9.5 Initiale Qualitätsmessung und -analyse

Wird das CLIQ-Vorgehensmodell erst nachträglich in ein bereits bestehendes DWS integriert, so ist *vor* dem ersten, durch das Vorgehensmodell gesteuerten inkrementellen Update zu ermitteln, inwieweit der aktuelle Datenbestand im Zielbereich des DWS die gestellten Anforderungen erfüllt. Dazu können die in den vorangegangenen Abschnitten eingeführten Qualitätspläne herangezogen werden. Gegenstand der Messungen ist in diesem Fall die Menge $I(R_{ZB})$ (der Konsolidierungsbereich ist leer).

Sind die Anforderungen initial erfüllt (für den Sonderfall $I(R_{ZB}) = \varnothing$ gilt dies implizit), so stellt die konsequente Anwendung des CLIQ-Vorgehensmodells sicher, dass diese Eigenschaft im Sinne einer Invariante auch nach der Durchführung inkrementeller Updates weiterhin gilt.

Existieren jedoch Anforderungen, die aufgrund von Datenqualitätsmängeln in $I(R_{ZB})$ nicht erfüllt sind, so müssen vor der Inbetriebnahme des CLIQ-Vorgehensmodells zunächst die in Abschnitt 7.10 beschriebenen Lenkungs- und Verbesserungsmaßnahmen auf den Zielbereich angewandt werden.

7.10 Phase 8: Lenkung von Datenprodukten/Qualitätsverbesserung

Diejenigen KB-Tupel, die zu Datenobjekten gehören, die den Prüfungen in Phase 7 nicht standgehalten haben, sind nun entsprechend zu lenken. Dabei kann es sinnvoll sein, auch ZB-Tupel von mangelhaften Datenobjekten in die Lenkungsmaßnahmen einzubeziehen. In der Praxis ist dies jedoch oft nur eingeschränkt möglich. So scheidet z. B. eine Aussonderung und Neuanforderung häufig aus, da die entsprechenden Daten in den Quellen nicht mehr verfügbar sind. Lediglich eine nachträgliche symptomorientierte Qualitätsverbesserung der ZB-Daten kommt dann noch in Frage.

Im Folgenden sollen verschiedene, miteinander kombinierbare Optionen zur Lenkung von Tupeln eines Datenobjekts d, zu dem mindestens ein Plan qp mit $passed(qp, d) = false$ existiert, diskutiert werden.

1. *Symptomorientierte Datenqualitätsverbesserung*
   Zusätzlich zu der in den vorangegangenen Phasen vorgenommenen Beseitigung von Datenqualitätsmängeln (z. B. Phase 2 bzgl. Einheitlichkeit, Phase 4 bzgl. Konsistenz, Phase 6 bzgl. Redundanzfreiheit) kann versucht werden, weitere Qualitätsmängel aufzuspüren und zu beseitigen. Hierzu können Verfahren des Data Mining verwendet werden, die – wie in [DJK00], [HW00] und [HGG01] dargestellt – automatisiert Regelmäßigkeiten im Datenbestand aufdecken und Tupel des zu verbessernden Datenobjekts d, die diesen Regelmäßigkeiten nicht gehorchen, als auffällig kennzeichnen und ggf. mit Vorschlägen zur Beseitigung des Mangels (i. Allg. Updates von Attributwerten) versehen. Die Suche nach Regelmäßigkeiten erstreckt sich auf die zu Beginn von Abschnitt 7.9 definierte Tupelmenge TM_R.

   Auffällige Tupel in d sind im Anschluss interaktiv zu untersuchen und ggf. zu ändern. Die Wirksamkeit und Effizienz solcher Maßnahmen hängt stark von der Konfiguration des zu Grunde liegenden Data Mining-Prozesses ab. Weiterhin ist zu beachten, dass Maßnahmen dieser Art auf die Datenqualitätsmerkmale Konsistenz (und damit indirekt auch Korrektheit) sowie Vollständigkeit (Ergänzung fehlender Werte an Hand gefundener Regelmäßigkeiten) beschränkt sind. Eine detaillierte Diskussion zu diesem Thema findet sich in [HW00].

   Nach Durchführung von Maßnahmen zur nachträglichen Beseitigung von Datenqualitätsmängeln sind diejenigen Qualitätspläne, die Tupel betreffen, welche durch die Maßnahme geändert worden sein könnten, erneut auf ihre Erfüllung zu prüfen, d. h. Phase 7 ist erneut anzustoßen.

   Alternativ wäre es natürlich auch möglich, Verbesserungsmaßnahmen grundsätzlich *vor* Phase 7 anzusetzen. Da diese jedoch sehr zeitaufwändig sein können (z. B. manuelle Aufarbeitung von Data Mining-Ergebnissen), wird im CLIQ-Vorgehensmodell darauf verzichtet. Stattdessen wird die Strategie verfolgt, zunächst durch die Messungen in Phase 7 abzuschätzen, ob überhaupt Bedarf für derartige Maßnahmen besteht und wieviel Verbesserungspotential jeweils vorliegt.

2. *Vergabe von Nutzungsauflagen*
   Sind Anforderungen eines Kunden an ein Datenobjekt d teilweise, aber nicht vollständig erfüllt, kann versucht werden, mit dem Kunden über eine Lockerung der nicht erfüllten Anforderungen zu verhandeln, um schließlich doch noch eine Freigabe der in d enthaltenen Tupel (in Phase 9) durchzusetzen. Diese ist dann ggf. mit Auflagen für die Nutzung von d verbunden: Die Durchführung eines Analyseverfahrens A auf d („Analyseverfahren“ subsumiere alle Arten der Nutzung von d, z. B. auch die Generierung von Serienbriefen aus einer Kundenrelation) ist zu untersagen, wenn eine hinreichende Qualität des Analyseergebnisses $A(d)$ aufgrund des vorliegenden Datenqualitätsmangels in d nicht garantiert werden kann. Dazu wird eine Menge $Restraints$ von Tripeln $(d, cust, ExclMeth)$ verwaltet, wobei $cust$ einen Kunden (bzw. eine Kundenrolle), an den das Datenobjekt d ausgeliefert werden soll, und $ExclMeth$ eine Menge von Analyseverfahren bezeichnet, die explizit von der Nutzung von d ausgeschlossen werden. Die Elemente von $ExclMeth$ sind von einem Domänenexperten festzulegen. In der Praxis ist es u. U. nicht möglich, die Menge $ExclMeth$ exakt zu spezifizieren. Für OLAP-Anwendungen (vgl. Kapitel 1) sind z. B. Ad Hoc-Anfragen typisch, die sich nicht a priori umreißen lassen. Dies erschwert die Vergabe von Nutzungsauflagen erheblich.

   Für alle Datenobjekte d, für die mit einem Kunden $cust$ eine Freigabe unter Auflagen ausgehandelt werden konnte, ist ein neues Element in die Menge $Restraints$ aufzunehmen:

   $$Restraints := Restraints \cup \{(d, cust, ExclMeth)\}$$

   Die Menge $ExclMeth$ enthält hierbei diejenigen Analyseverfahren, deren Anwendung auf d untersagt ist (z. B. Analyse des Kundenverhaltens im Rahmen einer CRM-Anwendung bei unzureichender Zeitnähe oder Vollständigkeit der Daten). Im laufenden (Analyse-)Betrieb des DWS ist vor der Durchführung einer Analyse auf einem Datenobjekt jeweils an Hand der Menge $Restraints$ zu prüfen, ob die Anwendung des jeweiligen Analyseverfahrens zugelassen ist.

Derjenige Qualitätsplan qp, der Gegenstand der „Nachverhandlung“ war, ist nun bzgl. d als „erfüllt“ zu kennzeichnen:

$$passed(qp, d) := true$$

Nach Abschluss der Nachverhandlungen ist der am Ende von Abschnitt 7.9.3 beschriebene Algorithmus erneut auf die in $I(R_{KB})$ verbliebenen Tupel anzuwenden, um diese – sofern sie nun die gestellten Bedingungen erfüllen – in die Menge $ForRelease_R$ aufzunehmen und somit für die Freigabe vorzubereiten.

Nachverhandlungen sind im DWS-Kontext essentiell, da ein Kunde oft keine Wahl zwischen Datenprodukten hat: Zu verwendende Quelldaten sind oft vorgegeben, und selbst eine „vorbildliche“ Datenintegration (wie durch das CLIQ-Vorgehensmodell beschrieben) kann i. Allg. nicht alle Qualitätsmängel beheben. Der Kunde muss also mit den Qualitätsmängeln „leben“, insbesondere muss er sie kennen und bei der Datennutzung berücksichtigen.

3. *Ursachenorientierte Datenqualitätsverbesserung*
   Um das Auftreten von Datenqualitätsproblemen in der Zukunft zu reduzieren, können ursachenorientierte Maßnahmen in Betracht gezogen werden (siehe dazu auch Abschnitt 4.5.3). Diese umfassen sowohl den Integrationsprozess selbst (z. B. Tuning von Prozessparametern) als auch beteiligte Qualitätsplanungs- und Prüfprozesse (z. B. bessere Erfassung von Nutzeranforderungen, Einführung neuer Messverfahren etc.) sowie die umgebenden Organisationsstrukturen (z. B. optimierte Ressourcenzuweisung). Die Wirksamkeit von Prozessen kann dabei mit Hilfe von Referenzdaten bestimmt werden, für die sowohl das angestrebte Datenprodukt als auch die enthaltenen Unzulänglichkeiten a priori bekannt sind.

   Ursachenorientierte Maßnahmen greifen grundsätzlich erst bei einer erneuten Ausführung des betroffenen Prozesses. Ein Wiederaufsetzen des Integrationsprozesses sollte daher in der frühesten von den Verbesserungsmaßnahmen beeinflussten Phase des CLIQ-Vorgehensmodells erfolgen – die Verfügbarkeit der als Eingabe für diese Phase erforderlichen Zwischenergebnisse vorausgesetzt. Um eine Rückverfolgbarkeit von Maßnahmen zu gewährleisten und eine Analyse der Wirksamkeit von Verbesserungsmaßnahmen zu ermöglichen, ist zu protokollieren, welche Integrations- und Messergebnisse mit welchen Prozesskonfigurationen erzielt wurden.

4. *Aussonderung und Neuanforderung*
   Konnte im Falle einer nicht erfüllten Anforderung weder eine Maßnahme zur Beseitigung von Qualitätsmängeln noch eine Verhandlung mit dem Kunden zum Erfolg geführt werden, so sind die betroffenen Tupel auszusondern, d. h. aus dem Konsolidierungsbereich zu löschen. Da dies wiederum Auswirkungen auf die Erfüllung anderer Qualitätspläne haben kann, sind diejenigen Qualitätspläne, an deren Datenobjekt ausgesonderte Tupel beteiligt sind, erneut einer Messung zu unterziehen (d. h. Übergang zu Phase 7).

   Liegt die Ursache der Qualitätsmängel bei der bzw. den Datenquelle(n), so sollte dort – nach detailliertem Feedback – eine ursachenorientierte Datenqualitätsverbesserung (siehe oben) vorgenommen werden (in der Praxis oft schwierig), gefolgt von einer Neulieferung der an dem ausgesonderten Datenprodukt beteiligten Quelldaten und einem Ansetzen des Datenintegrationsprozesses bei Phase 1.

5. *Auflösung konkurrierender Anforderungen*
   Konflikte zwischen konkurrierenden Anforderungen unterschiedlicher Kunden(-rollen) bzw. Interessengruppen sind in geeigneter Weise aufzulösen. Es wird definiert:

   Definition 7.2 (Konkurrenzbeziehung): Ein Qualitätsplan $qp_a = (cust_a, cond_a, n_a, AV_a, g_a, qc_a, mdv_a)$ *konkurriert* mit einem Qualitätsplan $qp_b = (cust_b, cond_b, n_a, AV_b, g_b, qc_b, mdv_b)$ bzgl. eines Datenobjekts d, gdw. gilt:

   $$cust_a \neq cust_b \wedge passed(qp_a, d) \neq passed(qp_b, d)$$ □

Eine Konkurrenzbeziehung zwischen zwei von unterschiedlichen Kundenrollen $cust_1$ und $cust_2$ spezifizierten Plänen bzgl. eines Datenobjekts d drückt aus, dass d zwar einen der beiden Pläne erfüllt, den anderen jedoch nicht.

Folgendes Beispielszenario skizziert Problem und Lösungsansatz: Stellt ein Kunde $cust_1$ hohe Anforderungen an die Zeitnähe eines Datenprodukts d, während ein Kunde $cust_2$ insbesondere an einer hohen Vollständigkeit von d interessiert ist, so können u. U. nicht beide Anforderungen gleichzeitig erfüllt werden (d. h. die zugehörigen Qualitätspläne konkurrieren). Dies liegt daran, dass eine notwendige Maßnahme zur Erhöhung der Vollständigkeit (z. B. Data Mining-basiert, siehe oben) so zeitaufwändig ist, dass die gleichzeitig geforderte Zeitnähe nicht eingehalten werden kann.

Dieser Konflikt kann über eine Datenreplikation aufgelöst werden. Dazu werden die Tupel aus d zwar zur Freigabe vorbereitet, allerdings (über $Restraints$) vor Zugriffen durch $cust_2$ gesperrt. Bei der Freigabe werden die Tupel aus dem Konsolidierungsbereich dann *nicht* (wie in Abschnitt 7.11 beschrieben) in den Zielbereich verschoben, sondern dorthin *kopiert*. $cust_1$ kann die Daten dann bereits nutzen, während die im Konsolidierungsbereich verbliebene Instanz für $cust_2$ nachbearbeitet werden kann. Im Rahmen der Datenproduktfreigabe für $cust_2$ können dann die bereits freigegebenen Tupel aus d mit den vervollständigten aus dem Konsolidierungsbereich überschrieben werden. $cust_1$ ist über die nachträgliche Datenmanipulation zu informieren.

Die Auflösung konkurrierender Anforderungen ist praktisch nicht automatisierbar, sondern erfordert gezielte Diskussionen zwischen den beteiligten Parteien.

7.11 Phase 9: Freigabe von Datenprodukten

Diejenigen Tupel im Konsolidierungsbereich, deren zugehörige Datenobjekte die Maßnahmen der Phasen 7 und 8 erfolgreich passiert haben, d. h. die gestellten Anforderungen (ggf. unter Auflagen) erfüllen, werden nun explizit zur Nutzung freigegeben. Im Sinne einer Endkontrolle kann die Datenfreigabe interaktiv durch eine verantwortliche Person vorgenommen werden, die an Hand der im Laufe des Integrationsprozesses erzeugten Protokollinformationen sicherstellt, dass alle notwendigen Verarbeitungs- und Prüfschritte auch tatsächlich ordnungsgemäß durchgeführt wurden.

Ausgangspunkt der Freigabe ist die in Phase 7 gebildete (siehe Abschnitt 7.9.3) und in Phase 8 im Rahmen der Vergabe von Nutzungsauflagen ggf. erweiterte Menge $ForRelease_R$. Die in dieser Menge enthaltenen Tupel sind nun vom Konsolidierungsbereich in den Zielbereich zu verschieben. Dies entspricht einem inkrementellen Update des Zielbereichs. ZB-Tupel, zu denen ein Update in $ForRelease_R$ existiert, müssen – je nach Art der vorgenommenen Aktualisierung – unterschiedlich behandelt werden:

Wurde mindestens ein Wert eines Attributs verändert, das in den Metadaten als „zu historisieren“ gekennzeichnet ist (siehe unten), so ist das ZB-Tupel „einzufrieren“, d. h. als veraltete Version zu markieren und so von zukünftigen Updates auszuschließen. Das entsprechende KB-Tupel nimmt nun im Zielbereich die Position des „alten“ ZB-Tupels ein. Beziehen sich die Änderungen jedoch ausschließlich auf (beschreibende) Attribute, die nicht zu versionieren sind, so ist das „alte“ ZB-Tupel zu löschen, d. h. durch das KB-Tupel zu überschreiben.

Im Rahmen des Freigabeprozesses kommen die in Abschnitt 7.2 eingeführten tupelbezogenen Metaattribute $hist_state$, $is_current$, old_id, $validity_period_begin$ und $validity_period_end$ zum Einsatz. Sei weiterhin $\rightarrow_{upd}$ eine Beziehung zwischen einem KB-Tupel t_{KB} und einem ZB-Tupel t_{ZB}, wobei $t_{KB} \rightarrow_{upd} t_{ZB}$ als „t_{KB} aktualisiert t_{ZB}“ zu lesen ist. Der Freigabeprozess gestaltet sich wie folgt:

$\forall t_{KB} \in ForRelease_R$ do

if $\exists t_{ZB} \in I(R_{ZB}) : t_{KB} \rightarrow_{upd} t_{ZB}$

if $t_{KB}.hist_state = historize$

$t_{ZB}.is_current := false$

$t_{ZB}.validity_period_end := t_{KB}.validity_period_begin$

$t_{KB}.old_id := t_{ZB}.glb_id$

$t_{KB}.is_current := true$

$InsertTuple(R_{ZB}, t_{KB})$

else if $t_{KB}.hist_state = overwrite$

$\forall i \in \{1, ..., m\}$ do

$t_{ZB}.A_i := t_{KB}.A_i$

od

fi

else

$t_{KB}.is_current := true$

$InsertTuple(R_{ZB}, t_{KB})$

fi

$DeleteTuple(R_{KB}, t_{KB})$

od

Für jedes Tupel t_{KB} in $ForRelease_R$ wird geprüft, ob es bereits ein korrespondierendes Tupel t_{ZB} im Zielbereich gibt. Ist dies der Fall, müssen – je nach Historisierungsstatus von t_{KB} – unterschiedliche Metaattribute neu gesetzt werden. Bei einer Historisierung werden beide Tupel über old_id miteinander „verzeigert“. Anschließend wird t_{KB} als aktuelle Version gekennzeichnet und in den Zielbereich eingefügt. Letzteres geschieht auch dann, wenn es noch kein Pendant zu t_{KB} im Zielbereich gibt. Im Falle eines Überschreibens sind lediglich die Werte von t_{KB} nach t_{ZB} zu übernehmen. In allen Fällen wird t_{KB} schließlich aus dem Konsolidierungsbereich entfernt. Die bisher nur als „gelöscht“ markierten Tupel in den Quelldatenpuffern sowie im Transformationsbereich können nun ebenfalls physisch gelöscht werden (im Algorithmus nicht dargestellt).

Nach Abschluss von Phase 9 können Kunden auf den Daten des Zielbereichs – unter Berücksichtigung ggf. verhängter Auflagen ($Restraints$) – Analysen durchführen. Darüber hinaus können freigegebene Daten in die analyseorientierten Strukturen eines Data Warehouse geladen und dort ausgewertet werden.

7.12 Phase 10: Auswertung von Kundenfeedback/Datenrücknahme

Nach ISO 9001:2000 [Deu00b] muss eine Organisation nach der Auslieferung von (Daten-)Produkten ständig das Feedback von Kunden protokollieren, analysieren und als Maß für die Bewertung der Wirksamkeit des (D)QMS einsetzen.

Wird ein signifikanter Qualitätsmangel erst während der Nutzung (Auswertung) eines Datenprodukts durch den Kunden entdeckt, so hat die Organisation das Produkt zurückzunehmen (d. h. vorübergehend aus Data Warehouse und Zielbereich zu entfernen) und vor der erneuten Freigabe zu „reparieren“ (falls möglich). Wiederum sollten – soweit erforderlich – auch ursachenorientierte Maßnahmen einbezogen werden. Allgemein gilt, dass die Maßnahmen der Phase 8 (siehe Abschnitt 7.10) prinzipiell auch auf zurückgenommene Datenprodukte anwendbar sind. Dies setzt voraus, dass die betroffenen Daten in den Konsolidierungsbereich eingefügt und die zugehörigen Metadaten entsprechend gesetzt worden sind. Der Konsolidierungsbereich entspricht hier einer „Werkstatt“ für reklamierte Datenprodukte.

Einschränkend wird angenommen, dass zum Zeitpunkt der Rücknahme alle Tupel des reklamierten Datenprodukts d aktuell sind ($is_current = true$), d. h. dass noch keine inkrementellen Updates auf den Tupeln stattgefunden haben. Im Zielbereich muss nun – falls vorhanden und soweit möglich – die vorangegangene Version des Datenprodukts reaktiviert werden, es sei denn, diese weist ebenfalls den bei der aktuellen Version festgestellten Mangel auf. Überschriebene Tupel (Updates mit $hist_state = overwrite$ sowie Tupel, die in ein Merging eingegangen sind) können allerdings nicht wiederhergestellt werden. Zur Reaktivierung von historisierten Tupeln ist Phase 9 „umgekehrt“ zu durchlaufen, wie im folgenden Algorithmus dargestellt. Sei dabei $GetTuple : Dom(glb_id) \mapsto I(R_{ZB})$ eine Funktion, die zu einem gegebenen Primärschlüssel das zugehörige Tupel liefert.

$\forall t_{ZB} \in d$ do

  if $t_{ZB}.old_id \neq NULL$ //Vorgängerversion existiert

    $t_{ZB,old} := GetTuple(t_{ZB}.old_id)$

    $t_{ZB,old}.is_current := true$

    $t_{ZB,old}.validity_period_end := NULL$

    $t_{ZB}.old_id := NULL$

  fi

  $t_{ZB}.is_current := false$

  $InsertTuple(R_{KB}, t_{ZB})$

  $DeleteTuple(R_{ZB}, t_{ZB})$

od

Falls zu einem Tupel t_{ZB} eines reklamierten Datenobjekts eine Vorgängerversion im Zielbereich existiert, werden für diese die Metaattribute $is_current$, $validity_period_end$ und old_id auf die Werte *vor* der Historisierung zurückgesetzt. Unabhängig von der Existenz einer Vorgängerversion wird t_{ZB} zunächst deaktiviert und anschließend in den Konsolidierungsbereich verschoben. Nach Abschluss dieser Rücknahmemaßnahmen ist mit Phase 8 fortzufahren.

Wie auch die anderen in diesem Kapitel vorgestellten Algorithmen ist der obige Rücknahmealgorithmus transaktionsorientiert auszuführen. Inkonsistente Zwischenzustände ergeben sich in diesem Fall z. B. daraus, dass für mehr als eine Version eines Tupels temporär $is_current = true$ gilt und dass ein Datenobjekt vorübergehend teilweise (de-)aktiviert ist.

7.13 Sonderfall: Behandlung von quellenseitig gelöschten Tupeln

Bisher wurden lediglich inkrementelle Updates zur Integration neuer oder geänderter Tupel in den Zielbereich betrachtet. Einen Sonderfall stellen Updates dar, die die Löschung von Tupeln in einer Datenquelle betreffen (z. B. Schließung einer Filiale, Auslaufen eines Artikels etc.).

Grundsätzlich werden aus dem Zielbereich keine Daten gelöscht. Stattdessen wird lediglich eine Tupelhistorie abgeschlossen, d. h. das zuletzt aktuelle Tupel ($is_current = true$) wird deaktiviert, ohne dass eine Nachfolgeversion des Tupels in den Zielbereich eingefügt wird. Auswertungen auf der Tupelhistorie sind somit auch nach einer quellenseitigen Löschung noch möglich. In diesem Abschnitt wird beschrieben, wie im CLIQ-Vorgehensmodell mit quellenseitigen Löschungen umgegangen wird.

Die Löschung eines Tupels in einer Datenquelle führt bei der nächsten Extraktion dazu, dass ein Tupel in den zugehörigen Quelldatenpuffer eingefügt wird, das denselben lokalen Schlüssel wie das gelöschte Tupel aufweist. Die übrigen Attributwerte des Tupels spielen keine Rolle und sollten daher leer sein. Dies ist aufgrund der fehlenden DBMS-seitigen Integritätsprüfungen in den Quelldatenpuffern sowie in Transformations- und Konsolidierungsbereich zulässig (vgl. Abschnitt 7.1). Tupel des oben genannten Typs werden im Folgenden als *Löschanweisungen* bezeichnet. Eine Löschanweisung t_d zeichnet sich weiterhin dadurch aus, dass im Metaattribut $hist_state$ der Wert $delete$ eingetragen ist. Darüber hinaus muss im Metaattribut $validity_period_end$ der Zeitpunkt vermerkt sein, an dem das zu löschende Tupel ungültig geworden ist.

Nach den Prüfungen auf Schlüsselintegrität und referentielle Integrität in Phase 1 wird t_d in Phase 2 ein global eindeutiger Primärschlüssel zugewiesen. Von den Transformationen zur Vereinheitlichung der Repräsentation (Phase 2), den statistischen Prozesskontrollen (Phase 3) sowie den domänenspezifischen Konsistenzprüfungen (Phase 4) kann t_d ausgeschlossen werden, da keine relevanten Attributwerte enthalten sind. Im schlüsselbasierten Linkage (Phase 5, Teil 1) wird durch Abgleich der lokalen Schlüssel (inkl. Angaben zu Datenquelle und QDP-Relation) dasjenige Tupel t_{ZB} im Zielbereich bestimmt, in welches das zu löschende Tupel eingeflossen ist. Wenn kein solches Tupel existiert, bleibt die Löschanweisung wirkungslos.

Es ist nun zu bestimmen, ob noch Tupel aus anderen Quellen in t_{ZB} eingeflossen sind, die ihrerseits noch nicht quellenseitig gelöscht wurden. Ist dies der Fall, bleibt t_{ZB} zunächst aktiviert ($is_current = true$); es wird lediglich in einem Metaattribut vermerkt, dass das von t_d referenzierte Tupel quellenseitig gelöscht wurde ($deleted := true$). Nur wenn alle anderen in t_{ZB} eingeflossenen Tupel bereits quellenseitig gelöscht sind, wird auch t_{ZB} selbst als „gelöscht“ markiert. Ist t_{ZB} *nicht* aus einer Verschmelzung mehrerer Tupel hervorgegangen, ist dies beim Eintreffen der Löschanweisung implizit der Fall. Die übrigen Metaattribute von t_{ZB} bleiben zunächst unverändert, insbesondere bleibt t_{ZB} aktiviert.

Es ist unbedingt darauf zu achten, dass die Behandlung von Löschanweisungen und die Zuordnung von neuen Tupeln zu Tupeln aus dem Zielbereich *nicht kommutativ* ist: Wird t_{ZB} als „gelöscht“ markiert, können andere Tupel diesem nicht mehr zugeordnet werden, sondern müssen in ein neues Tupel im Zielbereich einfließen. Erfolgt hingegen zunächst die Zuordnung eines neuen Tupels zu t_{ZB}, wird t_{ZB} anschließend nicht als „gelöscht“ markiert, da nun ein weiteres Tupel in t_{ZB} eingeflossen ist, das quellenseitig noch existiert.

Zum probabilistischen Record Linkage (Phase 5, Teil 2), zum Merging (Phase 6), zur Qualitätsmessung und -analyse (Phase 7) sowie zur Qualitätslenkung und -verbesserung (Phase 8) dürfen im Folgenden ausschließlich Tupel des Zielbereichs herangezogen werden, die *nicht* als „gelöscht“ markiert sind. Die Löschanweisung selbst kann im Transformationsbereich verbleiben; eine Übertragung in den Konsolidierungsbereich ist nicht erforderlich.

Im Rahmen der Aktualisierung des Zielbereichs in Phase 9 des Vorgehensmodells werden die als „gelöscht“ markierten Tupel des Zielbereichs endgültig deaktiviert und historisiert. Die in der Löschanweisung t_d vermerkte Angabe zum Ende des Gültigkeitszeitraums des Tupels wird dabei nach t_{ZB} übernommen.

$$t_{ZB}.is_current := false$$

$$t_{ZB}.validity_period_end := t_d.validity_period_end$$

Anschließend kann die Löschanweisung (physisch) gelöscht werden, da sie ihren Zweck nun erfüllt hat. Die Behandlung von quellenseitig gelöschten Tupeln ist damit abgeschlossen.

7.14 Parallelisierung der Phasenausführung

Zur Effizienzsteigerung der technischen Umsetzung des CLIQ-Vorgehensmodells können die einzelnen Phasen in drei sog. *Integrationsabschnitte* IA_{QDP}, IA_{TB} und IA_{KB} und zwei sog. *Transferabschnitte* $TA_{QDP \rightarrow TB}$ und $TA_{TB \rightarrow KB}$ eingeteilt werden:

- Integrationsabschnitte
  - IA_{QDP} umfasse das Füllen der Quelldatenpuffer (streng genommen nicht Teil des Vorgehensmodells) sowie die auf den Daten der Quelldatenpuffer arbeitende Phase 1 des Vorgehensmodells.
  - IA_{TB} umfasse die auf den Daten des Transformationsbereichs arbeitenden Phasen 3 bis 5 des Vorgehensmodells.
  - IA_{KB} umfasse die auf den Daten des Konsolidierungsbereichs arbeitenden Phasen 7 bis 9 des Vorgehensmodells.
- Transferabschnitte
  - $TA_{QDP \rightarrow TB}$ umfasse Phase 2 des Vorgehensmodells.
  - $TA_{TB \rightarrow KB}$ umfasse Phase 6 des Vorgehensmodells.

Die Parallelisierbarkeit der Phasen des CLIQ-Vorgehensmodells ist nun durch die drei beteiligten Datenhaltungskomponenten bzw. durch die auf ihnen definierten Integrations- und Transferabschnitte gegeben. Phase 10 nimmt eine Sonderstellung ein und wird hier ausgeklammert.

Bezeichne $X||Y$ die Parallelisierbarkeit zweier Prozesse X und Y. Dann gilt:

$$IA_{QDP} \,||\, IA_{TB} \;\wedge\; IA_{QDP} \,||\, IA_{KB} \;\wedge\; IA_{TB} \,||\, IA_{KB} \tag{7.19}$$

$$IA_{QDP} \,||\, TA_{TB \rightarrow KB} \;\wedge\; TA_{QDP \rightarrow TB} \,||\, IA_{KB} \tag{7.20}$$

Die Gültigkeit dieser Regeln ergibt sich aus der Tatsache, dass durch sie jeweils sichergestellt ist, dass parallel aktive Abschnitte auf disjunkten Daten arbeiten. Zugriffskonflikte sind damit ausgeschlossen. Nicht parallelisierbar sind hingegen ein Integrationsabschnitt und sein(e) angrenzender/n Transferabschnitt(e) sowie die Transferabschnitte untereinander. Jede Ausführung eines Integrations- bzw. Transferabschnitts entspricht einer (langen) Transaktion.

Alle Abschnitte mit Ausnahme von IA_{QDP} sind relationenorientiert, d. h. ein in einem dieser Abschnitte ablaufender Prozess bezieht sich auf genau eine Relation aus Transformations-, Konsolidierungs- bzw. Zielbereich. Prozesse zu unterschiedlichen Relationen sind daher unabhängig voneinander und können folglich ebenfalls parallelisiert werden (z. B. paralleles Record Linkage auf den Relationen „Kunden“ und „Artikel“).

7.15 Komplexitätsabschätzungen

In diesem Abschnitt werden Abschätzungen der Laufzeitkomplexität (genauer: Größenordnung der asymptotischen Komplexität) von Datenintegrationsprozessen, die dem CLIQ-Vorgehensmodell folgen, vorgenommen. Dabei wird nach den einzelnen Phasen des Vorgehensmodells differenziert. Der Abschnitt wird abgerundet durch einige Anmerkungen zum Speicherbedarf der Datenintegration.

7.15.1 Phase 1: Prüfung von Schlüssel- und referentieller Integrität

Sei n die Anzahl der Quelltupel, die im Rahmen der Durchführung einer Integrationsaufgabe in eine Zielrelation R_{ZB} überführt werden sollen. Die Quelltupel können auf mehrere Quelldatenpuffer verteilt sein. Von Struktur und Umfang der Tupel (von Quelle zu Quelle i. Allg. unterschiedlich) wird an dieser Stelle abstrahiert, da hierüber keine allgemeinen Aussagen getroffen werden können. Typischerweise bleibt die Größenordnung von n im Laufe des DWS-Betriebs konstant, es sei denn, die Anzahl der angebundenen Quellen oder die Extraktionsfrequenz ändern sich erheblich (bei seltener Extraktion werden die Extrakte größer).

Zur Prüfung der Schlüsselintegrität muss für jedes Quelltupel ermittelt werden, ob ein anderes Quelltupel mit identischem Primärschlüssel existiert. Die Berechnungskomplexität dieser Subphase ist bei einer indexsequentiellen Organisation des Primärschlüssels $O(n \log n)$ (binäre Suche). Bei einer Indizierung durch Hashing kann keine allgemeine Aussage zur Komplexität getroffen werden.

Die im Rahmen der Prüfung der referentiellen Integrität durchzuführende Suche nach „ins Leere zeigenden“ Fremdschlüsseln und unzulässigen Multiplizitäten von in Beziehung stehenden Tupeln hat bei binärer Suche ebenfalls einen Aufwand von $O(n \log n)$.

7.15.2 Phase 2: Vereinheitlichung der Repräsentation

Für jedes Quelltupel sind a priori spezifizierte Abbildungsvorschriften auszuführen, um die Quellattributwerte auf die Strukturen des Transformationsbereichs abzubilden. Es ist leicht einzusehen, dass sich der Berechnungsaufwand für diese Phase linear zur Anzahl der zu transformierenden Tupel verhält, also $O(n)$.

7.15.3 Phase 3: Statistische Prozesskontrolle

Da die Quelltupel nach Abschluss von Phase 2 in der Relation R_{TB} des Transformationsbereichs vorliegen, gilt im Folgenden $n = |I(R_{TB})|$.

Die Berechnung statistischer Kennzahlen wie Mittelwert, Standardabweichung und Attributwerthäufigkeiten hat eine Komplexität von $O(n)$, wie sich direkt aus den in Abschnitt 7.5 angegebenen Formeln ergibt. Die Bestimmung minimaler und maximaler Attributwerte erfordert bei binärer Suche einen Aufwand von $O(\log n)$, bei linearer Suche $O(n)$.

Werden die SPC-Kontrollgrenzen erst zur Laufzeit an Hand der Daten im Zielbereich bestimmt, muss ein zusätzlicher Aufwand einkalkuliert werden, der allerdings vernachlässigbar sein sollte.

Die Komplexität interaktiver Maßnahmen im Falle von Über- oder Unterschreitungen von Kontrollgrenzen kann nicht allgemein abgeschätzt werden.

7.15.4 Phase 4: Domänenspezifische Konsistenzprüfung/Nachbearbeitung

Analog zu Phase 2 sind alle in Phase 4 durchzuführenden Arbeitsschritte unmittelbar auf einzelne Tupel bezogen, wobei jedes Tupel genau einmal betrachtet wird, so dass sich ein Aufwand von $O(n)$ ergibt.

7.15.5 Phase 5: Record Linkage

In der Subphase zum schlüsselbasierten Linkage (vgl. Abschnitt 7.7.1) werden diejenigen Tupel in R_{TB}, die inkrementelle Updates darstellen, korrespondierenden Tupeln in R_{ZB} zugeordnet. Der Aufwand steigt also linear mit der Anzahl der zu integrierenden Tupel, d. h. $O(n)$.

Sei nun $m = |I(R_{ZB})|$ die Anzahl der Tupel in R_{ZB}. Die Berechnungskomplexität des probabilistischen Linkage lässt sich durch das in den Abschnitten 6.2.5 und 7.7.2 skizzierte Blocking-Verfahren erheblich reduzieren. Ohne Blocking müsste jedes Tupel in R_{TB} mit allen anderen Tupeln in R_{TB} sowie mit allen Tupeln in R_{ZB} verglichen werden, was einer in der Praxis inakzeptablen Anzahl von $n(n+m)$ Tupelvergleichen entspricht. Bei einem Blocking mit k Umläufen (i. Allg. gilt $k < 10$), wobei die Blockvariablen im j-ten Umlauf i_j unterschiedliche Ausprägungen annehmen können, verringert sich die Anzahl der Tupelvergleiche unter der vereinfachenden Annahme einer Gleichverteilung von Ausprägungen (best case) auf:

$$n \sum_{j=1}^{k} \frac{n+m}{i_j} \approx nk \frac{n+m}{i_{avg}}, \tag{7.21}$$

da im j-ten Umlauf nur $\frac{1}{i_j}$-tel der Tupel in den Blockvariablen übereinstimmt und damit in die Betrachtung eingeht. Die Näherung ergibt sich aus der Annahme, dass die Wertebereiche der Blockvariablen ungefähr gleich groß sind und eine durchschnittliche Kardinalität von i_{avg} aufweisen. Bei geeigneter Spezifizierung von Umläufen und Auswahl von Blockvariablen (vgl. Abschnitt 6.2.5) sollte $k \ll i_{avg}$ und $n + m \sim i_{avg}$ gelten, was den Einfluss von $n + m$ erheblich reduziert und eine signifikante Effizienzsteigerung gegenüber $n(n+m)$ Vergleichen bedeutet. Bei wachsendem m im Laufe des DWS-Betriebs sind die Blockvariablen dynamisch so anzupassen, dass i_{avg} ebenfalls entsprechend größer wird. Eine allgemeine Aussage zur Laufzeitkomplexität ist aufgrund der zahlreichen Einflussgrößen schwierig. Erfahrungen im Epidemiologischen Krebsregister Niedersachsen (siehe Kapitel 11, aktuell gilt dort $m \approx 120.000$) deuten jedoch darauf hin, dass sich der Berechnungsaufwand für das probabilistische Linkage bei Verwendung einer geeigneten Prozesskonfiguration näherungsweise linear zu n verhält.

Die abschließende Berechnung von Matchgruppen (transitiver Abschluss) erfordert einen Berechnungsaufwand von $O(p^2)$, wobei p die Anzahl der im Record Linkage gefundenen Matches ist. Näherungsweise gilt $p \sim n$, da sich Tupel mit mehreren Matchpartnern und Tupel ohne Matchpartner typischerweise ungefähr egalisieren. Wird die Berechnung dahingehend optimiert, dass eine binäre Suche verwendet wird, reduziert sich der Aufwand auf $O(p \log p)$. Eine weitere Effizienzsteigerung wird erreicht, wenn der Suchraum zur Laufzeit durch Entfernen der bereits einer Matchgruppe zugeordneten Matches verkleinert wird.

7.15.6 Phase 6: Merging

Sei q die Anzahl der im vorigen Schritt berechneten Matchgruppen. Jede Matchgruppe wird im Merging-Schritt zu einem Zieltupel verschmolzen, was einem Aufwand von $O(q)$ entspricht. Dabei gilt $q \leq n$, da in jeder Matchgruppe mindestens ein neu zu integrierendes Tupel enthalten ist.

7.15.7 Phase 7: Qualitätsmessung und -analyse

Im Hinblick auf die einzelnen in Abschnitt 6.1 vorgestellten Messverfahren, die in Phase 7 des Vorgehensmodells zum Einsatz kommen, wird davon ausgegangen, dass Messwerte zu neu integrierten Tupeln persistent gespeichert werden und mit vernachlässigbar geringem Laufzeitaufwand ausgelesen

werden können. Damit müssen in Phase 7 ausschließlich neue und geänderte Tupel Qualitätsmessungen unterzogen werden, woraus eine Komplexität von $O(n)$ folgt. Der Aufwand zur Aggregierung (i. Allg. Mittelwertbildung) von Qualitätswerten einer Granularitätsebene i (z. B. Tupel) zu einem Qualitätswert der Granularitätsebene $i + 1$ (hier: Relation) sei ebenfalls vernachlässigbar.

Die obigen Aussagen gelten z. B. für die Datenqualitätsmerkmale Konsistenz, Genauigkeit und Vollständigkeit, nicht jedoch für die Merkmale Zeitnähe und Redundanzfreiheit. Da Qualitätswerte zur Zeitnähe kontinuierlich abnehmen (Ausnahmen: invariante Angaben und historisierte Daten), sind hier die Tupel des Zielbereichs erneut in die Messung einzubeziehen (also $O(m)$), was allerdings aufgrund des minimalen Aufwands pro Tupel (Auslesen des Befunddatums) akzeptabel sein sollte. Da Redundanzfreiheit erst ab Relationenebene gemessen werden kann (vgl. Abschnitt 6.1.4), ist hier kein Zugriff auf tupelbezogene Qualitätswerte aus früheren Messungen möglich. Eine vollständige Neuberechnung des Qualitätswertes zur Redundanzfreiheit auf dem gesamten Zielbereich wäre jedoch selbst bei linearem Aufwand (vgl. Abschnitt 7.15.5), also $O(m)$, inakzeptabel.

Als Lösung dieses Problems wäre ein Messverfahren denkbar, das direkt die Ergebnisse von Phase 6 (Merging, siehe Abschnitt 7.8) eines Datenintegrationsprozesses gemäß CLIQ-Vorgehensmodell einbezieht. Konkret muss zu einer Relation R des Zielbereichs eine Menge $Dup(R)$ von Tupeln (bzw. von deren Primärschlüsseln) verwaltet werden, die im Rahmen von Phase 6 wie folgt zu pflegen ist: Ist ein Tupel $t \notin Dup(R)$ mit keinem anderen Tupel der jeweiligen Matchgruppe verschmolzen worden, wird es als Duplikat aufgefasst und in die Menge $Dup(R)$ aufgenommen. (Ausnahme: Dasjenige Tupel der Matchgruppe mit dem am längsten zurückliegenden Befunddatum wird als „Original“ interpretiert und nicht in $Dup(R)$ eingefügt, um eine Überschätzung der Redundanz zu vermeiden.) Geht t hingegen in einem anderen Tupel auf und gilt $t \in Dup(R)$, so ist t aus der Menge $Dup(R)$ zu entfernen. Gleiches gilt für den Fall, dass für t eine Löschanweisung (vgl. Abschnitt 7.13) auszuführen ist. Wird t aufgrund eines inkrementellen Update in Form eines Tupels t_{upd} historisiert, so ist t in $Dup(R)$ durch t_{upd} zu ersetzen. Mit Hilfe der Menge $Dup(R)$ kann der Qualitätswert von R bzgl. Redundanz nun inkrementell, d. h. unter Verwendung der Ergebnisse von Phase 6, analog zu Formel 6.15 in Abschnitt 6.1.4 definiert werden als:

$$Q_{Red}(R) := 1 - \frac{|Dup(R)|}{|I(R)|}, \tag{7.22}$$

wobei $I(R)$ wiederum die (nicht-leere) Instanz von R bezeichne. $Q_{Red}(R)$ hat den Wert 1, wenn keine nicht zusammengeführten Tupel vorliegen ($Dup(R) = \varnothing$), und geht gegen 0, je mehr Tupel in $Dup(R)$ enthalten sind ($Dup(R) \subset I(R)$). Der Laufzeitaufwand zur Verwaltung der Menge $Dup(R)$ sowie zur Berechnung von $Q_{Red}(R)$ ist unabhängig von m und sehr gering im Vergleich zum Aufwand zur Durchführung eines neuen Record Linkage-Prozesses auf dem gesamten Zielbereich. Die letztgenannte Maßnahme empfiehlt sich allerdings im Falle einer Änderung der zu Grunde liegenden Konfiguration des Messverfahrens (z. B. Auswahl anderer Block- und Matchvariablen). Analoges gilt für das Qualitätsmerkmal „Konsistenz“ (z. B. Änderungen im Regelwerk).

Bezüglich des Laufzeitaufwandes zur Berechnung domänenspezifischer Qualitätsindikatoren gemäß Abschnitt 7.9.2 kann keine allgemeingültige Aussage getroffen werden.

7.15.8 Phase 8: Lenkung von Datenprodukten/Qualitätsverbesserung

Der Aufwand von Lenkungsmaßnahmen (Vergabe von Nutzungsauflagen, ursachenorientierte Datenqualitätsverbesserung etc.) ist nicht a priori quantifizierbar und steht i. Allg. nicht in direktem Zusammenhang mit den zu integrierenden Datenvolumina. Eine Ausnahme bildet die nachträgliche Beseitigung von Datenqualitätsmängeln. Deren Berechnungskomplexität hängt unmittelbar von dem verwendeten Verfahren (z. B. Data Mining-basiert) ab.

7.15.9 Phase 9: Freigabe von Datenprodukten

In dieser Phase erfolgt die Übertragung von Tupeln aus dem Konsolidierungs- in den Zielbereich. Mit $s = |I(R_{KB})|$ ergibt sich ein Berechnungsaufwand von $O(s)$.

7.15.10 Phase 10: Auswertung von Kundenfeedback/Datenrücknahme

Analog zu Phase 9 hängt der Aufwand zur Rücknahme von Tupeln aus dem Ziel- in den Konsolidierungsbereich linear von der Tupelanzahl ab. Der Aufwand für die Auswertung von Kundenfeedback sowie für die aus der Auswertung abgeleiteten Maßnahmen kann nicht allgemeingültig abgeschätzt werden.

7.15.11 Speicherbedarf der Datenintegration

Grundsätzlich ist zu unterscheiden zwischen dem Hauptspeicherbedarf zur Laufzeit eines Datenintegrationsprozesses einerseits und Mengengerüsten für (dauerhaft zu speichernde) Metadaten andererseits. Der Hauptspeicherbedarf verhält sich bei einer blockweisen Verarbeitung i. Allg. höchstens linear zur Anzahl der im Rahmen eines Integrationsprozesses neu zu integrierenden Tupel. Beim probabilistischen Record Linkage ist eine geeignete Blocking-Strategie erforderlich (vgl. Abschnitt 6.2.5), da hier zusätzlich die Anzahl der Tupel im Zielbereich eine Rolle spielt. Im Hinblick auf Metadaten kann differenziert werden zwischen Attributdeskriptoren, deren Speicherbedarf konstant ist (unabhängig von den Volumina der zu integrierenden Daten), Datendeskriptoren, deren Speicherbedarf linear mit der Anzahl der zu integrierenden Tupel wächst, und Messergebnissen, deren Speicherbedarf sich linear zur Anzahl der Integrationsläufe verhält. In letzterem Fall wird die „Steigung“ des Wachstums durch die Anzahl der zu prüfenden Qualitätspläne (inkl. SPC- und Indikatorplänen) bestimmt. Auf detaillierte Untersuchungen zum Speicherbedarf wird verzichtet.

7.16 DQM in OLTP-Systemen

Zahlreiche der in diesem Kapitel beschriebenen DQM-Maßnahmen sind nicht auf die Anwendung in DWS beschränkt, sondern lassen sich auch in OLTP-Systemen sinnvoll einsetzen. Im Folgenden werden einige Beispiele dafür gegeben:

- *Prüfung von Schlüssel- und referentieller Integrität:* Derartige Prüfungen werden i. Allg. DBMS-seitig bereits vor dem Einfügen eines neuen Tupels in die Datenbank durchgeführt. In Legacy-Systemen, bei denen dies nicht der Fall ist, sollten Prüfungen dieses Typs im Rahmen regelmäßiger Datenaudits vorgenommen werden (siehe dazu [Web99]).
- *Statistische Prozesskontrolle:* Die Anwendung von SPC-Maßnahmen in einem OLTP-System kann als Bestandteil von Datenaudits sinnvoll sein, um z. B. typische Eingabe- oder Softwarefehler aufzudecken.
- *Domänenspezifische Konsistenzprüfung/Nachbearbeitung:* Die Sicherung der Datenkonsistenz spielt in OLTP-Systemen eine elementare Rolle. Gängige DBMS erlauben die Formulierung von Integritätsbedingungen, deren Einhaltung vor dem Einfügen, Ändern oder Löschen eines Tupels geprüft wird (siehe z. B. [BMP01]). In der Praxis ist es jedoch oft vorteilhaft, alternativ oder zusätzlich zu diesen Integritätsbedingungen applikationsseitige Konsistenzprüfungen zu implementieren, um Domänenwissen nicht in einer proprietären DBMS-Sprache formulieren

zu müssen und auf Konsistenzverletzungen gezielter reagieren zu können. Ein Nachteil applikationsseitiger Konsistenzprüfungen ist allerdings, dass sie bei direktem Datenbankzugriff („an der Applikation vorbei“) umgangen werden können.

- *Record Linkage/Merging:* Um versehentlich mehrfach erfasste Entitäten (Kunden, Artikel etc.) im Datenbestand eines OLTP-Systems aufzuspüren und Duplikate zu beseitigen, bietet sich der Einsatz von Record Linkage- und Merging-Verfahren an, allerdings anders als in einem DWS nicht im Routinebetrieb, sondern im Rahmen regelmäßiger Datenaudits.
- *Qualitätsmessung, -analyse und -verbesserung:* Derartige Maßnahmen sollten den Kern von Datenaudits bilden. Mit Hilfe der in Kapitel 6 eingeführten Metriken, Messverfahren und domänenspezifischen Qualitätsindikatoren kann eine Abschätzung der Ist-Qualität eines Datenbestands erfolgen. Eine Analyse der Ursachen entdeckter Datenqualitätsmängel kann Hinweise für erforderliche Maßnahmen zur Qualitätsverbesserung im OLTP-System liefern (z. B. Optimierung von operativen Geschäftsprozessen).

7.17 Zusammenfassung

Die Integration von Daten in ein DWS läuft in der gängigen Praxis oft unsystematisch ab, wodurch Wartbarkeit, Reproduzierbarkeit und Ergebnisqualität von Integrationsprozessen beeinträchtigt werden. Vor diesem Hintergrund wurde in dem vorliegenden Kapitel ein formal fundiertes, auf den Grundsätzen der Norm ISO 9001:2000 beruhendes Vorgehensmodell entwickelt, welches aus zehn Phasen (mit Rückkopplungen) besteht. Jede dieser Phasen definiert einen einzelnen Integrations- oder Prüfschritt. Eine Ausführung dieser Schritte in der durch das CLIQ-Vorgehensmodell vorgegebenen Reihenfolge gewährleistet eine qualitätsgelenkte Datenintegration mit nachvollziehbaren Ergebnissen. Mehrere der DQM-Maßnahmen des Vorgehensmodells lassen sich auch außerhalb des DWS-Kontexts sinnvoll einsetzen, insbesondere in klassischen OLTP-Systemen.

Das in Abb. 7.3 (siehe folgende Seite) dargestellte UML-Aktivitätsdiagramm fasst noch einmal den Kontrollfluss inkl. Rückkopplungen zusammen. Der eigentliche Integrationsprozess terminiert nach Abschluss von Phase 9. Es schließt sich die Phase der Datennutzung an. Diese wird im Idealfall nicht verlassen, solange das DWS genutzt wird. Nur für den Fall, dass im Rahmen der Nutzungsphase ein signifikanter Qualitätsmangel entdeckt wird, ist eine Wiederaufnahme des Integrationsprozesses erforderlich.

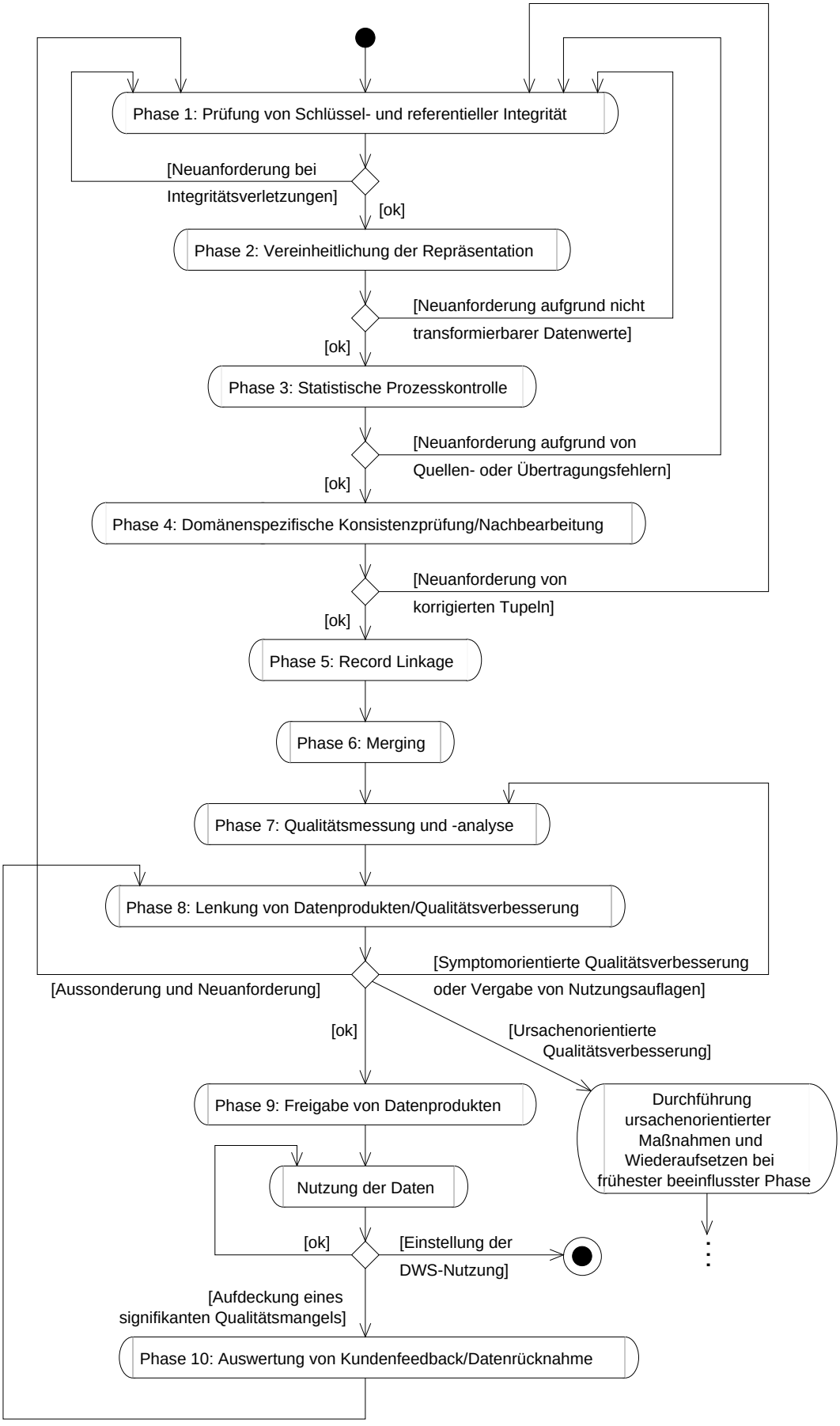


Abbildung 7.3: UML-Aktivitätsdiagramm zum CLIQ-Vorgehensmodell

Kapitel 8

Anforderungen an die CLIQWorkbench

Das in dieser Arbeit zu entwickelnde, im Folgenden als *CLIQWorkbench* bezeichnete Softwaresystem zum DQM soll eine durchgängige, weitgehend automatisierte Unterstützung des in Kapitel 7 spezifizierten qualitätsgelenkten Datenintegrationsprozesses bieten. In dem vorliegenden Kapitel werden zentrale Anforderungen an die CLIQWorkbench diskutiert.

In Abschnitt 8.1 wird zunächst ein grober Überblick über die gestellten Anforderungen gegeben. Anschließend wird in Abschnitt 8.2 auf der Basis grundlegender Konzepte aus [PS94], [Bal95] und [Som00] eine objektorientierte Analyse (OOA) nach [Oes97] durchgeführt.

8.1 Überblick über die Anforderungen

Gemäß [SR98] wird im Folgenden zwischen operativen (d. h. funktionalen) und technischen Anforderungen sowie Qualitäts-, Wartungs- und Realisierungsanforderungen unterschieden.

8.1.1 Operative Anforderungen

Folgende operative Anforderungen werden an die CLIQWorkbench gestellt:

- *Planung* von Datenintegrationsprozessen
  Zur Planung der „Erstellung“ eines Datenprodukts müssen Prozesse konfiguriert werden. Dazu gehört neben der Festlegung der beteiligten Datenquellen und Zieldatenstrukturen auch die Parametrierung von Integrations- und Prüfprozessen sowie die Erfassung von Qualitätsanforderungen an das Datenprodukt. Operationalisierte und damit messbar gemachte Qualitätsanforderungen sollen in Qualitäts- bzw. Indikatorplänen (vgl. Abschnitt 7.9.1) festgeschrieben werden.
- *Durchführung* von Datenintegrationsprozessen
  Das Anstoßen eines Integrationsprozesses soll – eine abgeschlossene Planungsphase vorausgesetzt – zu einer weitgehend automatisierten Datenprodukterstellung führen. Treten Abweichungen vom geplanten Prozess auf (verursacht durch mangelhafte Daten, technische Fehler oder Bedienungsfehler), soll die CLIQWorkbench den Kontrollfluss dahingehend lenken, dass ein als Reaktion auf das Problem geeigneter Prozess angestoßen wird.
- *Analyse* von Datenintegrationsprozessen
  Zusätzlich zur Analyse von Prozessresultaten als Grundlage für die Lenkung des Kontrollflusses im Datenprodukterstellungsprozess sollen Prozessergebnisse historisiert gespeichert und

über die Zeit ausgewertet werden. Derartige Auswertungen können dann z. B. Aufschluss darüber geben, ob Korrekturmaßnahmen an Datenintegrationsprozessen vorgenommen werden müssen. Ferner lassen sich ggf. Trends über die mittel- bis langfristige Entwicklung der Qualität von Quelldaten aufzeigen.

Zusammen mit nicht-operativen DQM-Maßnahmen wie der Übernahme von Verantwortung durch die Organisationsleitung oder der Bereitstellung von Ressourcen soll die CLIQWorkbench ein DQMS bilden, das den Prozess der Datenintegration in DWS unterstützt.

Operative Anforderungen an die einzelnen Module der CLIQWorkbench werden nicht in diesem Kapitel, sondern jeweils zu Beginn der entsprechenden Entwurfsabschnitte in Kapitel 9 diskutiert.

8.1.2 Technische Anforderungen

Die CLIQWorkbench soll aus Gründen der Laufzeiteffizienz in der Programmiersprache C++ implementiert werden. Das Softwaresystem ist so zu modularisieren, dass es zu großen Teilen einfach auf andere Plattformen portiert werden kann und lediglich die Benutzungsoberfläche (Dialogklassen) und einzelne plattformspezifische Teile (z. B. COM, ActiveX) ausgetauscht werden müssen.

Des weiteren soll die CLIQWorkbench über eine Schnittstelle zu allen gängigen relationalen DBMS verfügen (z. B. über ODBC). In der Integrationsschicht von DWS werden typischerweise relationale DBMS eingesetzt, so dass hierdurch keine maßgebliche Einschränkung der Anwendbarkeit der CLIQWorkbench gegeben ist. Metadaten sollten in einem dedizierten Metadaten-Repository verwaltet werden, um sie „sauber“ von den Fachdaten zu trennen und sie gemeinsam in mehreren Applikationen nutzen zu können. Da sich zum Zeitpunkt des Entwurfsbeginns noch kein Metadatenstandard endgültig durchgesetzt hat, sind Zugriffe auf ein konkretes Metadaten-Repository geeignet zu kapseln, um den Aufwand für einen eventuellen Austausch des Repository zu minimieren.

8.1.3 Qualitätsanforderungen

Zu den an die CLIQWorkbench gestellten Qualitätsanforderungen gehören neben allgemeinen Anforderungen wie Robustheit, Benutzungsfreundlichkeit und Laufzeiteffizienz insbesondere Anforderungen hinsichtlich der Erweiterbarkeit: Es muss möglich sein, neue oder geänderte Integrations- oder Prüfprozesse in einfacher Weise zu integrieren.

Aufgrund der großen Datenvolumina in DWS ist ferner ein hoher Automatisierungsgrad des DQM unerlässlich. Dies betrifft sowohl den Datenintegrationsprozess selbst als auch die beteiligten Messprozesse.

Der Kern der CLIQWorkbench soll domänenunabhängig sein, um das Anwendungsfeld nicht a priori einschränken zu müssen. Allerdings soll domänenspezifisches Wissen einbezogen werden können, sei es über Zusatzmodule, die an die CLIQWorkbench angebunden sind, oder über angemessen repräsentierte Metadaten, auf die zur Laufzeit zugegriffen werden kann.

Schließlich soll die CLIQWorkbench so in eine gegebene DWS-Umgebung integrierbar sein, dass keine wesentlichen Umstellungen der Aufbauorganisation des DWS erforderlich sind – die Ablauforganisation ändert sich durch die DQM-Einführung zwangsläufig – und der Routinebetrieb des DWS nicht maßgeblich nachteilig beeinflusst wird.

8.1.4 Wartungsanforderungen

Wartungsanforderungen wie Garantiebedingungen und Wartungskonditionen spielen im Rahmen dieser Arbeit keine Rolle.

8.1.5 Realisierungsanforderungen

Hinsichtlich der Realisierungsanforderungen wird festgelegt, dass der softwaretechnische Entwurf der CLIQWorkbench gemäß den Prinzipien der *objektorientierten Analyse* (OOA) und des *objektorientierten Entwurfs* (engl.: Object Oriented Design, OOD) in der Variante nach [Oes97] erfolgen soll. Nach Coad und Yourdon weist die OOA/OOD-Methodik gegenüber anderen Ansätzen insbesondere folgende Vorzüge auf [CY91a, CY91b]:

- Reduzierung der Problemkomplexität durch Anwendung objektorientierter Prinzipien
- Verbesserung der Interaktion zwischen Domänenexperten und Entwicklern
- Erhöhung der internen Konsistenz zwischen Analyse, Entwurf und Implementierung.

Charakterisierende Eigenschaften der OOA/OOD-Methodik sind:

- Keine wesentlichen Unterschiede zwischen Analyse- und Entwurfsnotationen
- Fließender Übergang von der Analyse zum Entwurf
- Wasserfallmodell, Spiralmodell oder inkrementelles Vorgehensmodell anwendbar.

Als Modellierungsnotation soll die *Unified Modeling Language* (UML) [BJRR98, RJB98, Alh98, Obj01] verwendet werden, da sie auf die objektorientierte Modellierung zugeschnitten ist und sich in diesem Bereich als Standard etabliert hat.

8.2 Objektorientierte Analyse des Problembereichs

Als erster Schritt der objektorientierten Analyse nach [Oes97] werden in Abschnitt 8.2.1 sog. *Anwendungsfälle* identifiziert. Ein Anwendungsfall beschreibt einen Ausschnitt eines Geschäftsprozesses, spezifiziert also operative Anforderungen an das System. Aus Anwendungsfällen lassen sich sog. *Business-Klassen* ableiten (Abschnitt 8.2.2). Diese beschreiben Objekte des betrachteten Diskursbereichs (Organisationseinheiten, Produkte etc.). Sie repräsentieren in dieser frühen Phase des Softwareentwicklungsprozesses bereits wichtige Klassen des späteren Klassenmodells. Eine Business-Klasse kann durch Attribute charakterisiert sein und Beziehungen zu anderen Business-Klassen aufweisen. Mittels sog. *Aktivitätsdiagramme* (Diagrammtyp der UML) können schließlich einzelne Schritte (Aktivitäten) eines Arbeitsablaufs analysiert werden. Für diejenigen Arbeitsabläufe, die schon durch das CLIQ-Vorgehensmodell zur Datenintegration (siehe Kapitel 7) spezifiziert sind, kann eine Erstellung von Aktivitätsdiagrammen entfallen.

8.2.1 Identifizierung von Rollen und Anwendungsfällen

Für den Diskursbereich des DQM werden, ausgehend von den in Abschnitt 8.1.1 formulierten operativen Anforderungen, die in Abb. 8.1 dargestellten (groben) Anwendungsfälle unterschieden.

Folgende vier miteinander kommunizierende Rollen sind darin involviert:

- *Organisationsleitung*
  Die Organisationsleitung setzt sich aus Personen der Managementebene zusammen. Sie trifft – auf der Basis von operationalisierten Kundenanforderungen – Entscheidungen über die Bereitstellung von Ressourcen und die Auswahl von Datenquellen.

- *Kunde*
  Ein Kunde stellt Anforderungen an ein zu erstellendes Datenprodukt, das er beziehen möchte. Nach Abschluss des Integrationsprozesses wird das Datenprodukt an den Kunden ausgeliefert. Der Kunde kann der Organisationsleitung Informationen über seine Zufriedenheit mit dem Datenprodukt zukommen lassen.

- *Datenquelle (Lieferant)*
  Eine Datenquelle liefert Rohdaten, die durch den Integrationsprozess zu einem Datenprodukt aufbereitet werden. Die Rohdaten müssen Anforderungen erfüllen, die vor der Lieferung zwischen Datenquelle und Organisationsleitung auszuhandeln sind. Bei einer Datenquelle kann es sich sowohl um eine externe Organisation als auch um ein operatives System derjenigen Organisation handeln, die die Datenintegration betreibt.

- *Systemnutzer*
  Ein Systemnutzer ist für den Betrieb der CLIQWorkbench zuständig. Insbesondere konfiguriert er die einzelnen Phasen des Datenprodukterstellungsprozesses gemäß den Vorgaben der Organisationsleitung und der Kunden, startet den Prozess der Datenintegration, arbeitet mit den durch das Softwaresystem zur Verfügung gestellten interaktiven Softwaremodulen, führt Nachverhandlungen mit Kunden, falls Anforderungen nicht erfüllt werden, und gibt ggf. Datenprodukte zur Auslieferung an Kunden frei. Systemnutzer können weiter nach den von ihnen durchzuführenden Aufgaben differenziert werden. Dazu werden von der Organisationsleitung jedem Systemnutzer Verantwortlichkeiten und Befugnisse übertragen.

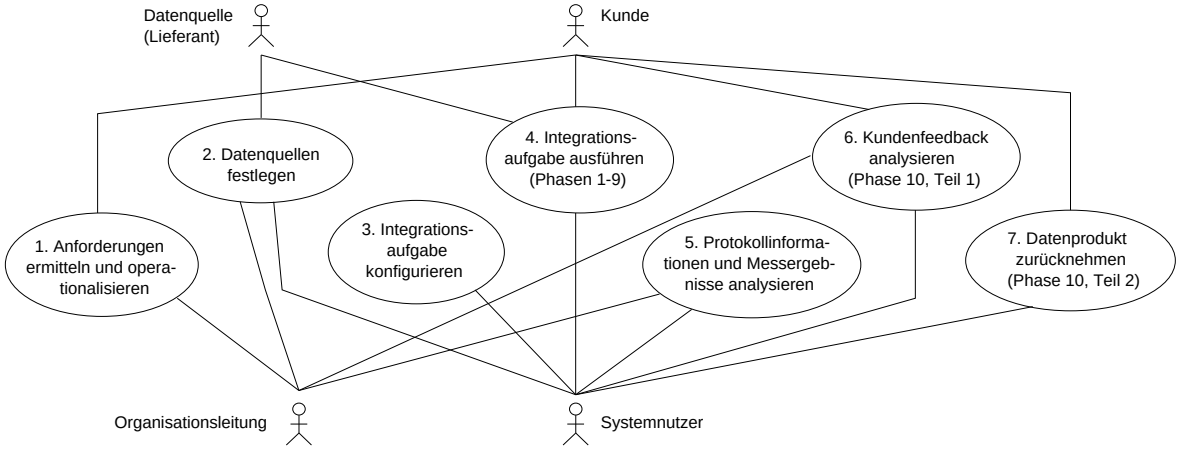


Abbildung 8.1: UML-Anwendungsfalldiagramm zur CLIQWorkbench (aus [Ade01], revidiert)

Im Folgenden wird ein Überblick über die einzelnen, in Abb. 8.1 skizzierten Anwendungsfälle gegeben:

1. *Anforderungen ermitteln und operationalisieren*
   Die Organisationsleitung muss in Verhandlungen mit den Kunden zunächst deren Anforderungen an das zu erstellende Datenprodukt (insbesondere hinsichtlich der in Abschnitt 3.2.3 genannten Datenqualitätsmerkmale) ermitteln. Weiterhin sind Anforderungen der Organisation selbst (z. B. bzgl. Wirtschaftlichkeit des Integrationsprozesses) sowie ggf. gesetzliche Anforderungen (z. B. bzgl. Datenschutz) zu berücksichtigen. Die ermittelten (zumeist subjektiven) Anforderungen werden nun miteinander abgeglichen und ggf. in Verhandlungen überarbeitet; insbesondere sind Widersprüche zwischen Anforderungen aufzulösen. Die überarbeiteten Anforderungen sind zu operationalisieren, d. h. messbar zu machen, z. B. mit Hilfe des in Abschnitt 4.4.1 beschriebenen QFD-Verfahrens.

2. *Datenquellen festlegen*
   Aufbauend auf Anwendungsfall 1 müssen jetzt Datenquellen identifiziert werden, die in der Lage sind, Anforderungen an grundlegende Merkmale wie Relevanz, Zeitnähe und Genauigkeit von Rohdaten zu erfüllen. Potentielle Datenquellen müssen sich dazu äußern, inwieweit ihre Rohdaten den gestellten Anforderungen genügen. Die Organisationsleitung wählt dann die am besten geeigneten Quellen aus, wobei ggf. Kompromisse einzugehen sind, da oft nicht die Möglichkeit besteht, zwischen alternativen Quellen zu wählen. Die Feinanbindung einer Quelle (Festlegung der für die Extraktion relevanten Teilmenge des verfügbaren Datenbestands und technische Spezifizierung der Datenübertragung) wird in Zusammenarbeit von Datenquelle und Systemnutzer durchgeführt.

3. *Integrationsaufgabe konfigurieren*
   Eine Integrationsaufgabe beschreibt einen Integrationsprozess von den Datenquellen bis hin zu einem fertig integrierten Datenprodukt. Dabei bezieht sich jede Integrationsaufgabe auf genau eine Relation der Zieldatenbank. Einer neu initiierten Integrationsaufgabe werden Datenquellen bzw. darin enthaltene Quellrelationen sowie eine Zielrelation zugewiesen. Dabei kann jede Quellrelation in beliebig viele Zielrelationen eingehen. Die Verantwortlichkeiten und Befugnisse der Systemnutzer bzgl. einer Integrationsaufgabe werden von der Organisationsleitung vergeben. Nach Abschluss dieser initialen Planung kann die Integrationsaufgabe im zweiten Schritt hinsichtlich einzelner Phasen des CLIQ-Vorgehensmodells konfiguriert werden (vgl. Kapitel 7).

4. *Integrationsaufgabe ausführen (Phasen 1-9)*
   Veranlasst durch den verantwortlichen Systemnutzer, wird eine Integrationsaufgabe weitgehend automatisiert ausgeführt. Dabei ist sicherzustellen, dass die für die jeweiligen Phasen des CLIQ-Vorgehensmodells zuständigen Softwaremodule zum richtigen Zeitpunkt mit den richtigen Parametern aktiviert werden. Die im Rahmen der einzelnen Phasen auftretenden, feingranulareren Anwendungsfälle werden in den Abschnitten 9.3 bis 9.8 diskutiert.

   Die im Zuge der Ausführung aufgezeichneten Protokollinformationen werden dem zuständigen Systemnutzer zugänglich gemacht, der bei Bedarf korrigierend in den Integrationsprozess einzugreifen hat. Die Durchführung einer Integrationsaufgabe gilt als erfolgreich abgeschlossen, wenn die Phasen 1 bis 8 des Vorgehensmodells durchlaufen worden sind und das erstellte Datenprodukt den spezifizierten Anforderungen genügt. Falls einzelne Anforderungen nicht erfüllt werden konnten, sind ggf. Nachverhandlungen mit dem Kunden notwendig. Nach einer „Endkontrolle“ wird das Datenprodukt durch den zuständigen Systemnutzer freigegeben, d. h. an den Kunden ausgeliefert (Phase 9).

5. *Protokollinformationen und Messergebnisse analysieren*
   Zur Überwachung von Integrationsprozessen sowie zur Identifizierung von Ansatzpunkten zur ständigen Verbesserung des DQMS (sowohl im operativen als auch im organisatorischen Bereich) sind die im Prozessverlauf erstellten Protokollinformationen und Messergebnisse regelmäßig durch den Systemnutzer zu analysieren. Optimierungsmaßnahmen, die über ein Finetuning von Prozesskonfigurationen hinausgehen (z. B. Ersetzen einer unzuverlässigen Datenquelle), sind mit der Organisationsleitung abzustimmen.

6. *Kundenfeedback analysieren (Phase 10, Teil 1)*
   Die von Kunden eingehenden Rückmeldungen bzgl. ausgelieferter Datenprodukte sind zu analysieren, um die Kundenzufriedenheit abzuschätzen und Verbesserungspotentiale im DQMS aufzudecken. Dazu sind Rückmeldungen zu erfassen und durch Systemnutzer und Organisationsleitung im Einzelfall interaktiv, zusätzlich ggf. auch empirisch auszuwerten. Aus Analyseergebnissen sollen – wie im vorigen Anwendungsfall beschrieben – Optimierungsmaßnahmen für das DQMS abgeleitet werden können.

7. *Datenprodukt zurücknehmen (Phase 10, Teil 2)*
   Stellt ein Kunde nach der Auslieferung signifikante Mängel an einem Datenprodukt fest, so muss eine Rücknahme erfolgen, wie in Abschnitt 7.12 beschrieben. Damit Mängel gleichen Typs bei zukünftigen Ausführungsläufen einer Integrationsaufgabe vermieden oder zumindest rechtzeitig erkannt werden können, sollte das DQMS entsprechend optimiert werden, z. B. durch Modifizierung von Prozesskonfigurationen. Zur Überarbeitung eines zurückgenommenen Datenprodukts ist in Phase 8 (Lenkung von Datenprodukten/Qualitätsverbesserung) des CLIQ-Vorgehensmodells anzusetzen (siehe Anwendungsfall 4).

Aus softwaretechnischer Sicht sind insbesondere die Anwendungsfälle 3, 4 und 5 interessant, da sie den Kern der Funktionalität der CLIQWorkbench ausmachen. Die übrigen Anwendungsfälle werden im weiteren Verlauf der Analyse und des Entwurfs nicht näher betrachtet, da die durch sie beschriebene Funktionalität entweder nur rudimentär softwaretechnisch zu unterstützen ist (z. B. Ermittlung von Anforderungen, Auswahl geeigneter Datenquellen) oder aber bereits durch die näher betrachteten Anwendungsfälle abgedeckt wird (z. B. Folgemaßnahmen nach der Analyse von Kundenfeedback, Überarbeitung eines zurückgenommenen Datenprodukts).

8.2.2 Identifizierung von Business-Klassen

Aus den identifizierten Anwendungsfällen wird im nächsten Schritt ein Modell von Business-Klassen abgeleitet (siehe Abb. 8.2). Im weiteren Verlauf der Analyse und des anschließenden Entwurfs der CLIQWorkbench wird dieses Modell weiter verfeinert. Folgende Business-Klassen sind identifiziert worden:

- *Customer* (entspricht der Rolle „Kunde“ aus Abschnitt 8.2.1)
- *SystemUser* (entspricht der Rolle „Systemnutzer“ aus Abschnitt 8.2.1)
- *DataSource* (entspricht der Rolle „Datenquelle“ aus Abschnitt 8.2.1)
- *DataObject*
  Ein *DataObject* repräsentiert eine Menge von Datensätzen einer relationalen Datenbank. Es können die in Abschnitt 6.1 eingeführten Granularitätsebenen „Attributwert“, „Tupel“, „Relation“ und „Datenbank“ unterschieden werden. Somit können sowohl die zu erstellenden Datenprodukte als auch die von Datenquellen (Klasse *DataSource*) gelieferten Daten als Instanzen der Klasse *DataObject* definiert werden.
- *Requirement* (entspricht dem Konzept der „Anforderung“ aus Abschnitt 8.2.1)
- *Job* (entspricht dem Konzept der „Integrationsaufgabe“ aus Abschnitt 8.2.1)
  Ein *Job* ist unveränderbar auf eine bestimmte Zielrelation (*DataObject*) festgelegt. Diese Entwurfsentscheidung ist nicht als Einschränkung zu verstehen, sondern als Maßnahme zur Vereinfachung der Konfigurierung und Durchführung von Integrationsprozessen.
- *JobConfiguration*
  Jedem *Job* ist eine Menge von *JobConfigurations* zugeordnet, von denen zu jedem Zeitpunkt genau eine aktiviert ist, d. h. bei der Ausführung eines *Jobs* verwendet wird. Eine *JobConfiguration* besteht aus einer Menge von *ProcessConfigurations* (siehe unten).
- *Process*
  Ein *Process* repräsentiert einen Prozess (eine Phase) des CLIQ-Vorgehensmodells, z. B. einen Record Linkage-Prozess oder einen Messprozess.

- *ProcessConfiguration*
  Eine *ProcessConfiguration* beschreibt die Konfiguration eines Prozesses, z. B. eine Record Linkage-Konfiguration oder einen Qualitätsplan.

- *ProcessResult*
  Diese Klasse repräsentiert Ergebnisse von Prozessen, z. B. SPC-Messergebnisse oder Tupelzuordnungen aus dem Record Linkage. Ein erstelltes Datenprodukt ist streng genommen natürlich auch ein Prozessergebnis, wird jedoch nicht als *ProcessResult,* sondern als *DataObject* modelliert (siehe oben).

- *ProtocolInformation*
  Die Klasse *ProtocolInformation* dient zur Dokumentation der Prozessausführung und damit zum Nachweis der Konformität eines Datenprodukts mit den Anforderungen.

- *Feedback*
  Mit der Klasse *Feedback* werden Rückmeldungen von der Organisation an Datenquellen modelliert. Diese beziehen sich auf im Datenintegrationsprozess festgestellte Auffälligkeiten in Quelldaten.

- *AnalysisProcess*
  Die Klasse *AnalysisProcess* fasst alle Verfahren zur Analyse der Ergebnisse von Datenprodukterstellungsprozessen zusammen (siehe auch Abschnitt 9.3.1).

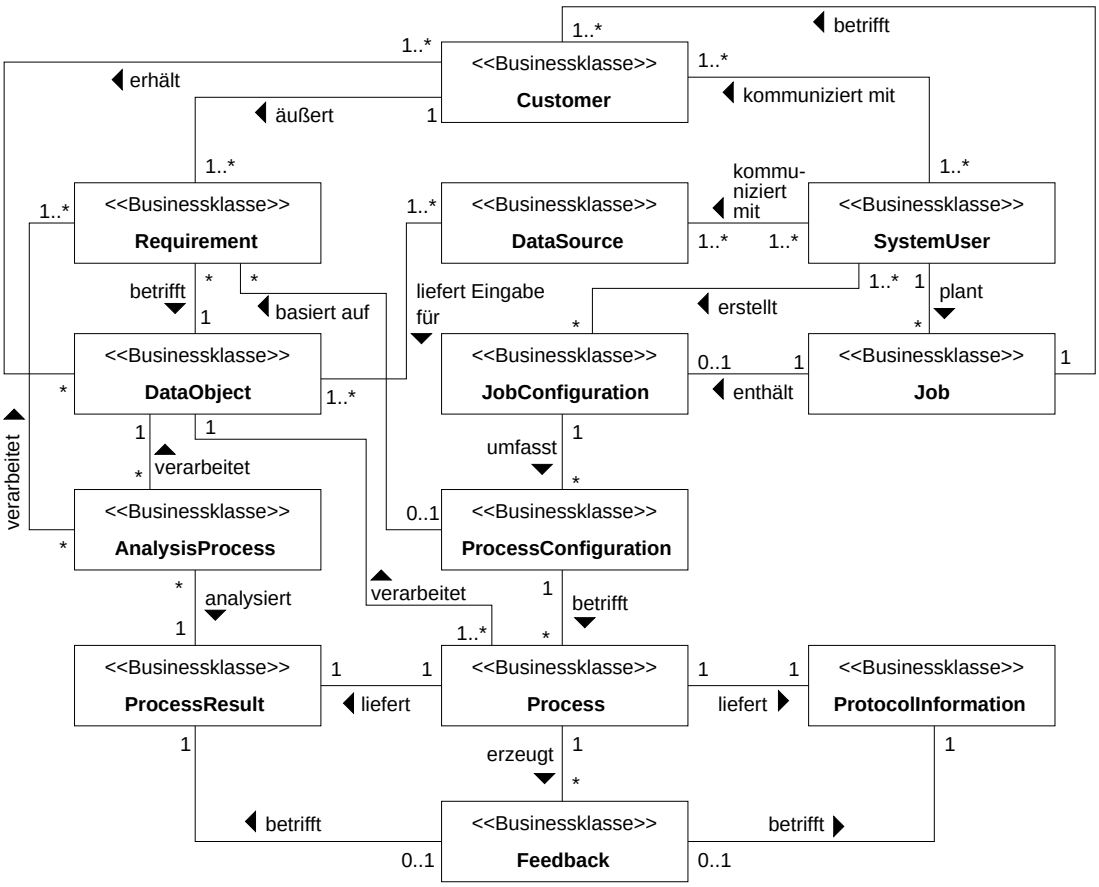


Abbildung 8.2: UML-Klassendiagramm der Business-Klassen (aus [Ade01], revidiert)

Die Organisationsleitung interagiert ausschließlich indirekt (über den Systemnutzer) mit der CLIQ-Workbench und wird daher nicht als Business-Klasse modelliert.

8.3 Zusammenfassung

In diesem Kapitel wurden Anforderungen an das im folgenden Kapitel zu entwerfende Softwaresystem spezifiziert, differenziert nach operativen und technischen Anforderungen sowie Qualitäts-, Wartungs- und Realisierungsanforderungen. Der OOA-Variante nach [Oes97] folgend, wurden anschließend – auf grober Ebene – Rollen, Anwendungsfälle und Business-Klassen identifiziert. Eine Verfeinerung dieser Aspekte im Hinblick auf die einzelnen Komponenten des Softwaresystems wird im nächsten Kapitel vorgenommen.

Kapitel 9

Softwaretechnischer Entwurf der CLIQWorkbench

Dieses Kapitel beschäftigt sich mit dem objektorientierten Entwurf der einzelnen Komponenten der CLIQWorkbench. Dabei wird auf die im vorigen Kapitel identifizierten Business-Klassen zurückgegriffen. Im Rahmen des Entwurfs werden inhaltlich zusammengehörige Klassen in sog. *Paketen* gruppiert, einem Strukturierungsmittel der UML auf logischer Ebene. Pakete können wiederum andere Pakete enthalten oder von ihnen abhängig sein. Die zu einem Paket gehörenden Klassen werden jeweils hinsichtlich derjenigen Attribute und Methoden beschrieben, die zum Verständnis des Systems erforderlich sind. Auf die Darstellung und Beschreibung der für die privaten und geschützten Attribute notwendigen Zugriffsmethoden wird hingegen verzichtet. Gleiches gilt für Hilfsattribute und -methoden, die nicht wesentlich zum Verständnis beitragen.

In Abschnitt 9.1 wird zunächst kurz auf die wesentlichen Annahmen, auf denen der Entwurf basiert, eingegangen. Anschließend werden in Abschnitt 9.2 zentrale Systemkomponenten identifiziert. Die Komponenten werden dann in den nachfolgenden Abschnitten 9.3 bis 9.10 im Detail erläutert. Abschnitt 9.11 geht schließlich auf die für das Verständnis der CLIQWorkbench wichtigen Metarelationen zur Handhabung inkrementeller Updates ein.

9.1 Dem Entwurf zu Grunde liegende Annahmen

Bezüglich des zu entwickelnden Softwaresystems werden folgende Annahmen getroffen:

- *Abgeschlossene Schemaintegration*
  Es wird davon ausgegangen, dass bereits ein globales Schema (in Transformations-, Konsolidierungs- und Zielbereich des DWS) existiert. Dieses Schema sei auf den Anwendungszweck des DWS ausgerichtet. Daten, die aus heterogenen Datenquellen in das DWS integriert werden sollen, müssen auf die Strukturen des globalen Schemas abgebildet werden. Dies entspricht einem Top Down-Ansatz zur Schemaintegration, wie z. B. in [BKL00] beschrieben. Bei der Entwicklung einheitlicher Schemata sollte laut [Has00b] auf domänenspezifische Standards zurückgegriffen werden (Beispiele: Dublin Core [Dub02] u. A. für Digitale Bibliotheken, HL7 [Hea02] für Krankenhausinformationssysteme oder OAGIS [Ope02] im Electronic Commerce).
- *Abgeschlossenes Metadaten-Reengineering*
  Es wird angenommen, dass zur Durchführung des DQM erforderliche Metadaten in hoher Qualität (Vollständigkeit, Konsistenz etc.) vorliegen. Dies betrifft insbesondere Schemabeschreibungen zu Datenquellen und Datenbanken der Integrationsschicht des DWS, ohne die keine

(teilautomatisierte) Datenintegration möglich ist. Ein Reengineering dieser Metadaten sollte in der Praxis problemlos sein (z. B. Auslesen aus dem Data Dictionary des jeweiligen DBMS). Ebenfalls unerlässlich sind Metadaten zu proprietären Nullwerten und Maßeinheiten von Attributen, da ohne sie keine sinnvolle Interpretation von Daten denkbar ist. Wünschenswert, aber nicht unbedingt notwendig sind Metadaten zu Update-Häufigkeiten von Attributen sowie instanzbezogene Metadaten wie Angaben zum „Befunddatum“ (vgl. Abschnitt 6.1.6). Liegen diese nicht vor, können bestimmte Qualitätsmessungen (hier bzgl. des Merkmals Zeitnähe) nicht durchgeführt werden. Zum Reengineering von Metadaten eignen sich z. B. Data Mining-Techniken.

9.2 Objektorientierter Entwurf

Wie in Abschnitt 8.1.5 festgelegt, wird der Entwurf der CLIQWorkbench mit der objektorientierten Entwurfsmethode OOD nach [Oes97] vorgenommen. Diese setzt auf den Ergebnissen einer objektorientierten Analyse auf (siehe Abschnitt 8.2) und resultiert in Paketen, die gemäß [CY91b] jeweils einer der folgenden vier Systemkomponenten zugeordnet werden können:

- *Human Interaction Component:* (Graphische) Benutzungsoberfläche
- *Problem Domain Component:* Problemlösung im engeren Sinne, im Kontext dieser Arbeit z. B. die Realisierung einzelner Schritte des CLIQ-Vorgehensmodells zur Datenintegration
- *Task Management Component:* Prozessplanung, -beobachtung und -steuerung
- *Data Management Component:* (Persistente) Datenverwaltung, z. B. in Datenbanken oder Dateien.

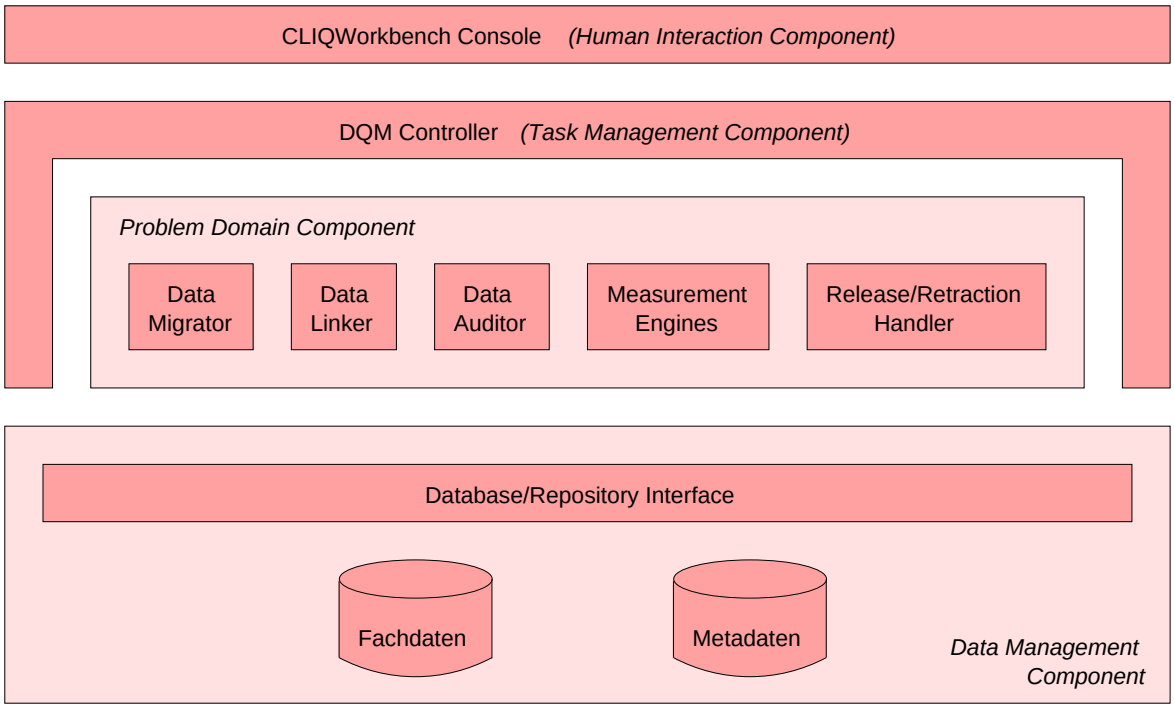


Abbildung 9.1: Grobarchitektur der CLIQWorkbench

Abbildung 9.1 gibt einen Überblick über die identifizierten Pakete und deren Zuordnung zu den Systemkomponenten gemäß OOD. Gesteuert durch einen zentralen DQM Controller (Task Management Component), unterstützen der Data Migrator, der Data Linker, der Data Auditor, die Measurement Engines sowie der Release/Retraction Handler (ergeben zusammen die Problem Domain Component) jeweils bestimmte Schritte des in Kapitel 7 vorgestellten CLIQ-Vorgehensmodells. Fach- und Metadaten werden in relationalen DBMS verwaltet und über entsprechende Zugriffsschnittstellen ausgelesen bzw. geschrieben (Database/Repository Interface, Data Management Component). Die Interaktion mit dem Systemnutzer erfolgt über eine graphische Benutzungsoberfläche (CLIQWorkbench Console, Human Interaction Component).

9.3 DQM Controller

Der DQM Controller stellt das zentrale Paket der CLIQWorkbench dar. Er ist für die Planung und Ausführung der im Zuge einer Datenintegration ablaufenden Prozesse sowie für die Analyse der gesammelten Ablaufinformationen zuständig. Wesentliche Konzepte des DQM Controllers wurden von Thomas Aden im Rahmen einer Diplomarbeit [Ade01] entwickelt.

9.3.1 Operative Anforderungen

Die operativen Anforderungen an den DQM Controller ergeben sich aus den in Abschnitt 8.2.1 beschriebenen Anwendungsfällen 3 (Integrationsaufgabe konfigurieren), 4 (Integrationsaufgabe ausführen) und 5 (Protokollinformationen und Messergebnisse analysieren). Diese Anwendungsfälle werden in den folgenden Unterabschnitten mit Hilfe von Aktivitätsdiagrammen konkretisiert. Aktivitäten werden dabei über Transitionen miteinander verknüpft. Durch gerichtete Kanten von (zustandsbehafteten) Objekten (Instanzen von Business-Klassen) zu Aktivitäten können Objektzustände als Bedingungen modelliert werden. Transitionen können von einer Aktivität zu mehreren Aktivitäten verzweigen, jedoch nur, wenn durch assoziierte Bedingungen eine Fallunterscheidung möglich ist.

Anwendungsfall 3: Integrationsaufgabe konfigurieren

Zur Planung eines Jobs bzw. einer Job-Konfiguration muss zunächst ein Job ausgewählt oder neu angelegt werden (siehe Abb. 9.2). Ist der Job bereits definiert, so kann lediglich die getroffene Auswahl an Datenquellen, die Bezeichnung des Jobs sowie ein beschreibender Text nachträglich verändert werden. Die für einen Job festgelegte Zielrelation ist hingegen fix.

Als nächstes wird die zu planende bzw. zu ändernde Job-Konfiguration ausgewählt. Soll eine bestehende Konfiguration geändert werden, so ist zunächst zu prüfen, ob mit der Konfiguration bereits einmal eine Integrationsaufgabe durchgeführt wurde (Aktivität „Job-Konfigurationsstatus prüfen“). Ist dies der Fall, so wird – verborgen für den Nutzer – eine Kopie der gesamten Konfiguration angelegt (Aktivität „Job-Konfiguration duplizieren“). Auf dieser Kopie können nun Änderungen vorgenommen werden. Erfolgte bisher kein Durchlauf des Jobs mit der zu ändernden Konfiguration, so können die Änderungen direkt auf dem „Original“ erfolgen. Durch dieses Historisieren von bereits verwendeten Konfigurationen wird die Rückverfolgbarkeit von Integrationsprozessen sichergestellt.

Die Spezifizierung einer neuen Job-Konfiguration beschränkt sich zunächst auf die Eingabe eines Bezeichners sowie einer textuellen Beschreibung. Bevor eine Job-Konfiguration für Integrationsprozesse genutzt werden kann, müssen darüber hinaus die einzelnen, im CLIQ-Vorgehensmodell festgeschriebenen Prozesse konfiguriert werden. Dazu ruft der Nutzer in beliebiger Reihenfolge Dialoge zur Prozesskonfigurierung auf (siehe Abschnitt 9.9), spezifiziert die erforderlichen Parameter (Transforma-

tionsregeln, Qualitätspläne etc.) und vervollständigt damit die Gesamtkonfiguration. Zum Abschluss werden die geänderten bzw. neu geplanten, vollständigen Konfigurationen des Jobs gespeichert.

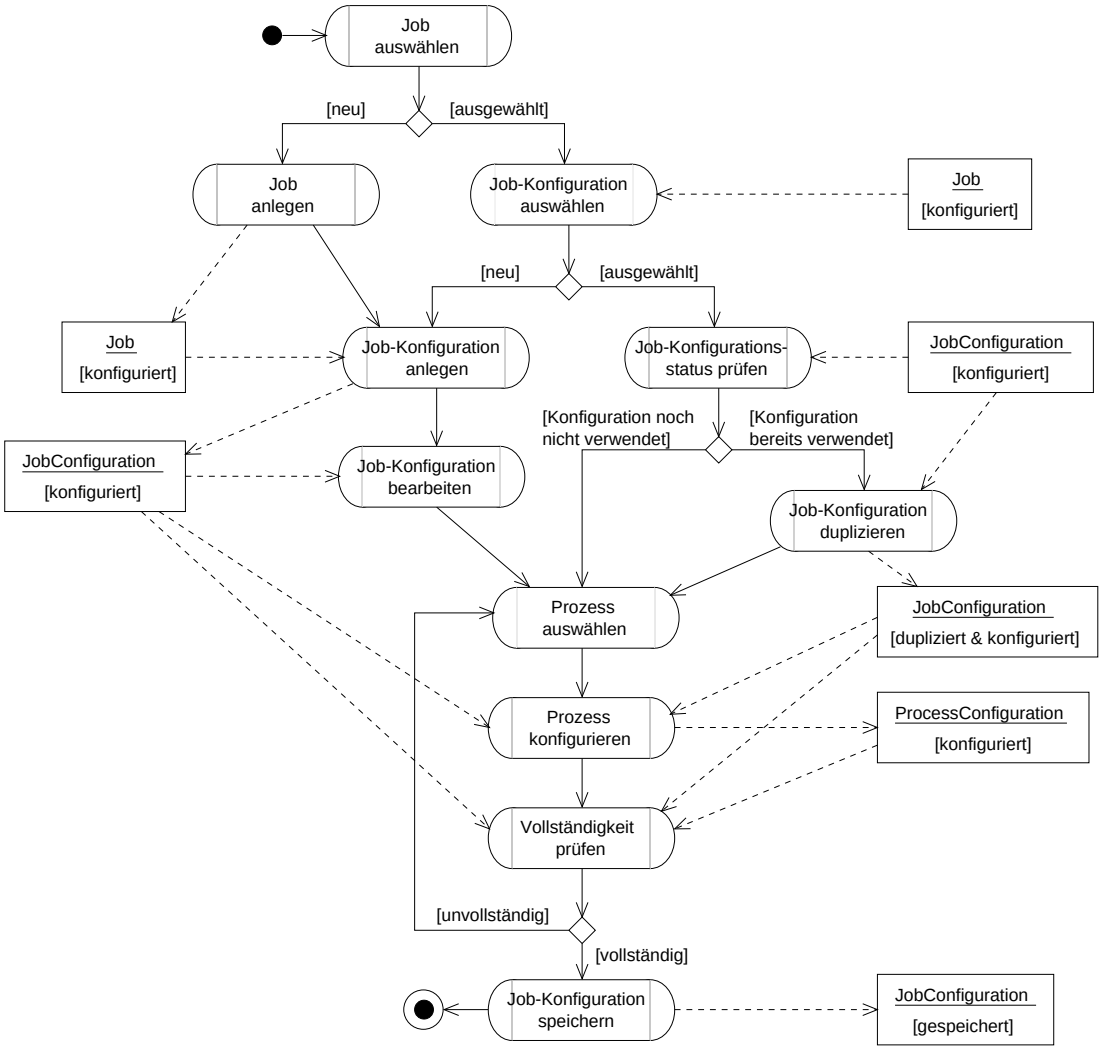


Abbildung 9.2: UML-Aktivitätsdiagramm zu Anwendungsfall 3 (aus [Ade01], revidiert)

Anwendungsfall 4: Integrationsaufgabe ausführen (Phasen 1-9)

Zur Durchführung einer Integrationsaufgabe (siehe Abb. 9.3) wählt der Systemnutzer zunächst einen Job aus. Anschließend muss geprüft werden, ob eine Durchführung der Aufgabe möglich ist. Diese Prüfung besteht aus folgenden Teilschritten:

1. Sicherstellen, dass die aktivierte Konfiguration des ausgewählten Jobs vollständig ist, d. h. dass zu allen konfigurierbaren Prozessen des CLIQ-Vorgehensmodells eine Prozesskonfiguration vorliegt.
2. Sicherstellen, dass neu zu integrierende Rohdaten in den mit dem ausgewählten Job assoziierten Quelldatenpuffern vorliegen.
3. Sicherstellen, dass eine vorherige Ausführung des ausgewählten Jobs – falls vorhanden – zumindest so weit fortgeschritten ist, dass die für den Job zu nutzende Relation des Transformationsbereichs für die neue Ausführung verfügbar ist (vgl. Abschnitt 7.14 zur Parallelisierung von Integrationsprozessen).

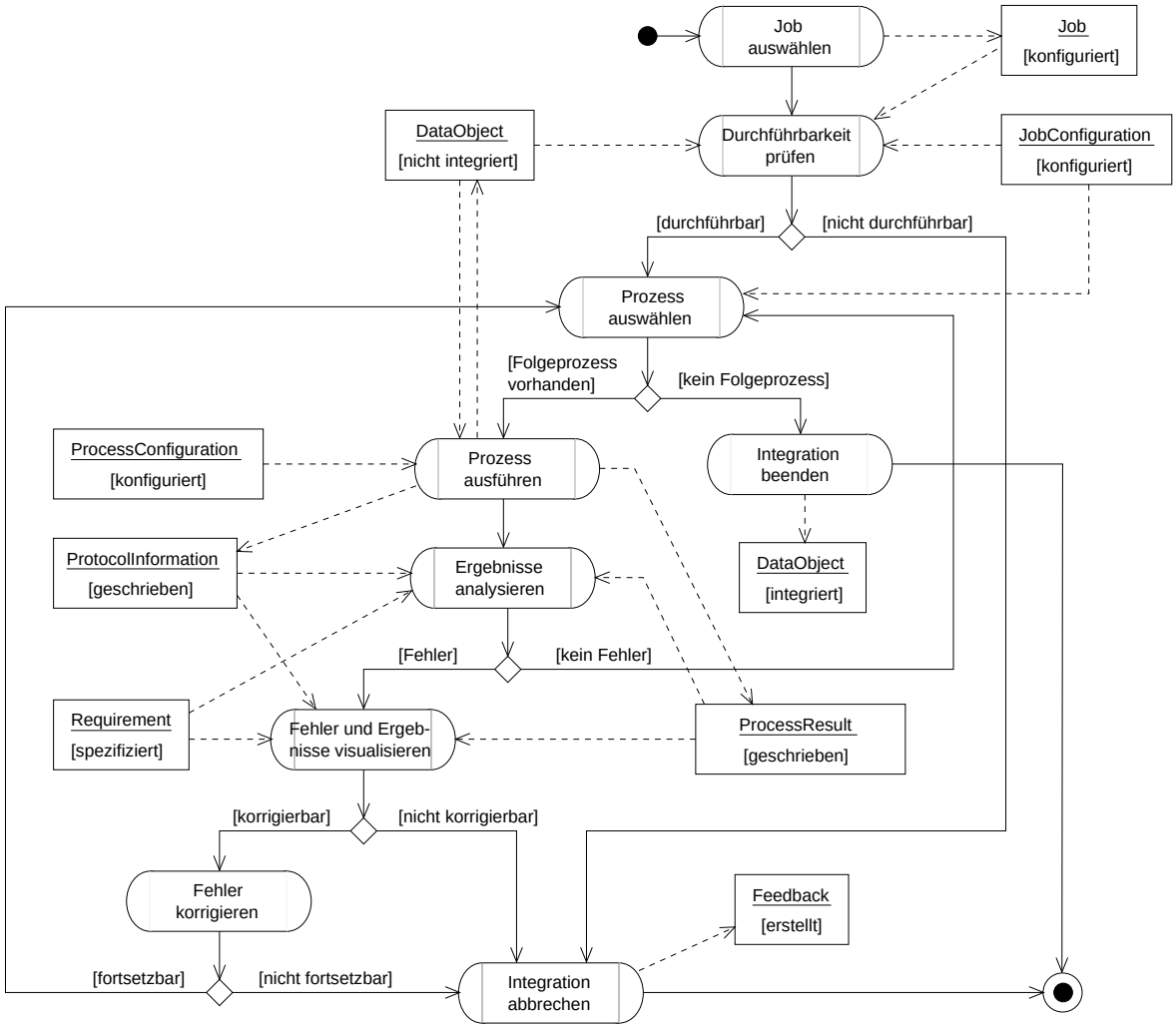


Abbildung 9.3: UML-Aktivitätsdiagramm zu Anwendungsfall 4 (aus [Ade01], revidiert)

Sind alle genannten Bedingungen erfüllt, wird mit der Ausführung des Jobs begonnen. Diese orientiert sich an den Vorgaben des in Kapitel 7 definierten Vorgehensmodells zur Datenintegration und wird daher an dieser Stelle nicht näher beschrieben. Jeweils nach Beendigung einer Phase und der Auswertung der Ergebnisse und Protokollinformationen wird die nachfolgend auszuführende Phase bestimmt und angestoßen. In diesem Zusammenhang sind folgende Analyseaufgaben durchzuführen:

1. *Prüfen, ob in der beendeten Phase Fehler aufgetreten sind.* Dazu wird das Vorliegen von Prozessergebnissen kontrolliert und eine zentrale Ablagestelle für Fehlerinformationen ausgewertet.

2. *Prüfen, ob Prozessergebnisse den Erwartungen entsprechen.* Dazu werden die Resultate von Messprozessen des CLIQ-Vorgehensmodells analysiert.

Im Fehlerfall werden die Prozessergebnisse für den Systemnutzer visualisiert. Dieser entscheidet, welche Maßnahmen zur Problembehebung zu ergreifen sind. Im Allgemeinen muss der Integrationsprozess nach der Durchführung korrigierender Maßnahmen in derjenigen Phase wieder aufgesetzt werden, in der die Problemursache lag. Liegen für die Ausführung dieser Phase notwendige Zwischenergebnisse nicht mehr vor, so muss der Integrationsprozess u. U. abgebrochen (inkl. Löschen der bisher berechneten Zwischenergebnisse) und neu gestartet werden.

Wird ein Prozesspunkt erreicht, an dem laut CLIQ-Vorgehensmodell ein interaktiver Eingriff des Systemnutzers erforderlich ist (z. B. Beseitigung von Inkonsistenzen), wird die automatisierte Ausführung unterbrochen und muss vom Systemnutzer zu gegebener Zeit explizit fortgesetzt werden.

Die Integration ist erfolgreich beendet, wenn der letzte Schritt des Integrationsprozesses (Phase 9, Freigabe von Datenprodukten) fehlerfrei abgeschlossen werden konnte.

Anwendungsfall 5: Protokollinformationen und Messergebnisse analysieren

Abbildung 9.4 zeigt das auf der Basis von Anwendungsfall 5 entwickelte Aktivitätsdiagramm. Im Einzelnen sollen zumindest die folgenden, über die Leistungsfähigkeit des DQMS Aufschluss gebenden Analysetätigkeiten softwareseitig unterstützt werden:

1. Analyse der Rohdatenqualität bzgl. einer Integrationsaufgabe über die Zeit auf Basis von Ergebnissen aus SPC-Prozessen und Konsistenzprüfungen, nach Datenquellen differenziert, inkl. Feedback an Datenquellen im Falle von signifikanten Trends.

2. Analyse der Zeiteffizienz der Durchführung von Integrationsaufgaben in Abhängigkeit von den jeweils zu integrierenden Datenvolumina auf Basis von Protokollinformationen.

3. Analyse von Datenqualitätstrends im Zielbereich über die Zeit auf Basis von Messergebnissen zu Qualitätsplänen bzw. Indikatorplänen.

Hat der Systemnutzer eines dieser Analyseverfahren ausgewählt, muss er es zunächst parametrieren, z. B. durch Angabe eines zu untersuchenden Jobs oder eines zu betrachtenden Zeitraums. Anschließend wird das ausgewählte Verfahren zur Ausführung gebracht. Die aus der automatisierten Analyse resultierenden Daten werden visuell aufbereitet (z. B. als Diagramme oder Tabellen) und dem Systemnutzer präsentiert.

Je nach Analyseergebnis kann der Systemnutzer bei Bedarf korrigierende Maßnahmen einleiten, um z. B. in zukünftigen Ausführungsläufen zu einem Job bestimmten, an Hand von Analyseergebnissen erkannten Negativtrends entgegenzuwirken. Dabei stehen folgende Möglichkeiten zur Auswahl:

1. *Konfigurationen ändern*
   Zur Änderung von Konfigurationen wird auf die bei Anwendungsfall 3 beschriebene Planungskomponente zurückgegriffen. Danach erfolgt ein Rücksprung zur Aktivität „Maßnahme festlegen“, in der der Systemnutzer weitere Maßnahmen festlegen oder aber eine neue Analyse anfordern kann (Rücksprung zur Aktivität „Analyseverfahren auswählen“).

2. *Feedback an Datenquellen liefern*
   Sind als Ursache für negative Trends oder auffällige Fehlerhäufigkeiten nicht die Prozesse des CLIQ-Vorgehensmodells, sondern die Lieferanten von Daten identifiziert worden, so ist ein Feedback an die betroffenen Datenquellen zu liefern, in dem auf die Problemursache hingewiesen und – soweit möglich – Hinweise zur Problembeseitigung gegeben werden. Anschließend stehen dem Systemnutzer die unter 1. aufgeführten Optionen zur Auswahl. Der DQM Controller hat die Rückkopplungen mit den Quellen zu protokollieren und ggf. – z. B. falls eine Quelle den Hinweisen nicht nachkommt und wiederholt mangelhafte Daten liefert – eine Terminierung vorzunehmen (entsprechende Folgemaßnahmen erforderlich, z. B. Abkopplung einer unzuverlässigen Quelle).

3. *Analysekomponente verlassen*
   Sind nach dem Ermessen des Systemnutzers keine weiteren korrigierenden Maßnahmen erforderlich oder müssen Maßnahmen eingeleitet werden, die von der CLIQWorkbench nicht explizit unterstützt werden (z. B. Änderungen am Datenschema oder an Prozessabläufen), kann die Analysekomponente verlassen werden.

Ausgehend von den spezifizierten Anforderungen wird im folgenden Abschnitt der Entwurf der einzelnen Pakete des DQM Controllers vorgenommen.

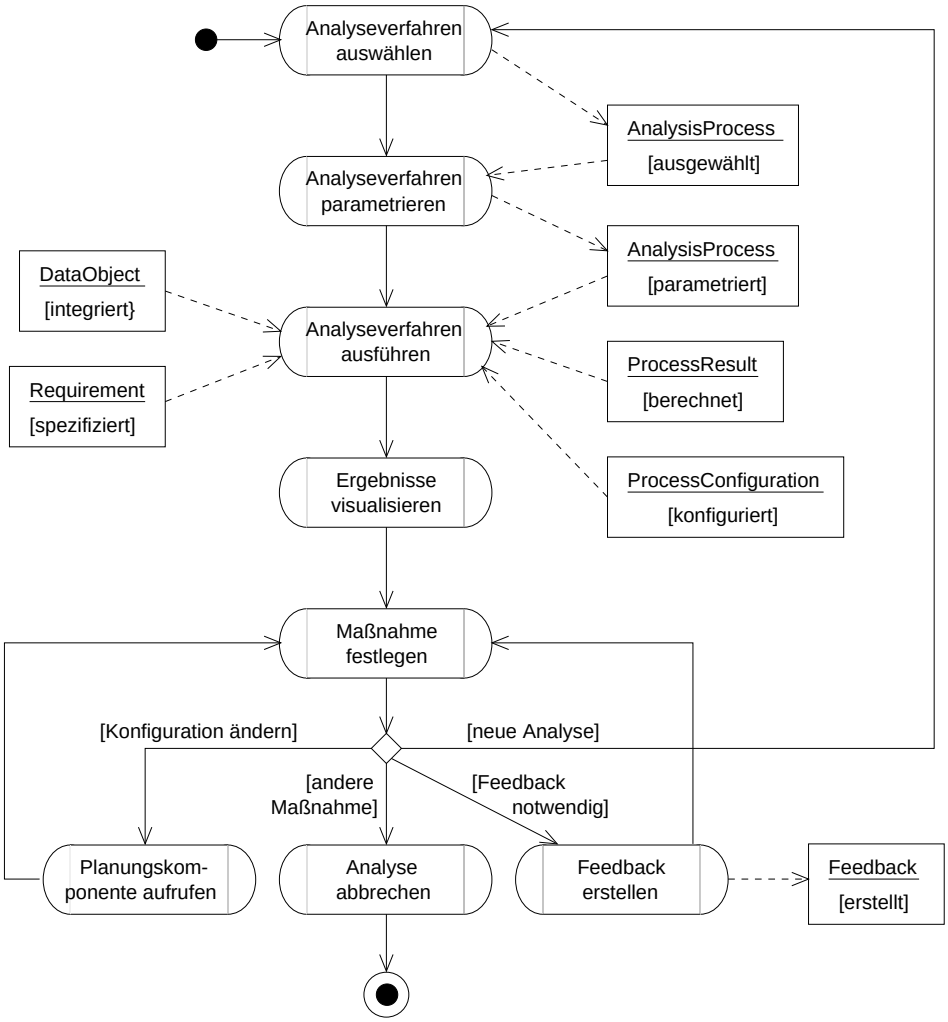


Abbildung 9.4: UML-Aktivitätsdiagramm zu Anwendungsfall 5 (aus [Ade01], revidiert)

9.3.2 Architektur und Datenstrukturen

In gewisser Weise entspricht die Funktionalität des DQM Controllers der eines Workflow Management Systems (WfMS). WfMS bieten die Möglichkeit, Prozesse bzw. Workflows, die mit Hilfe graphischer oder textueller Notationssprachen beschrieben sind, entweder direkt auszuführen oder in Programmcode umzusetzen. Die in die Ausführung des Workflows involvierten Softwarekomponenten müssen über geeignete Schnittstellen verfügen, die wiederum dem WfMS in geeigneten Spezifikationen bekannt zu machen sind.

In dieser Arbeit wurde eine Neuentwicklung des DQM Controllers der Verwendung eines kommerziellen WfMS vorgezogen, da sich der DQM Controller so exakt auf die Schnittstellen der übrigen

Komponenten der CLIQWorkbench zuschneiden ließ. Mittelfristig sollte jedoch auch in Erwägung gezogen werden, den DQM Controller durch ein kommerzielles WfMS zu ersetzen.

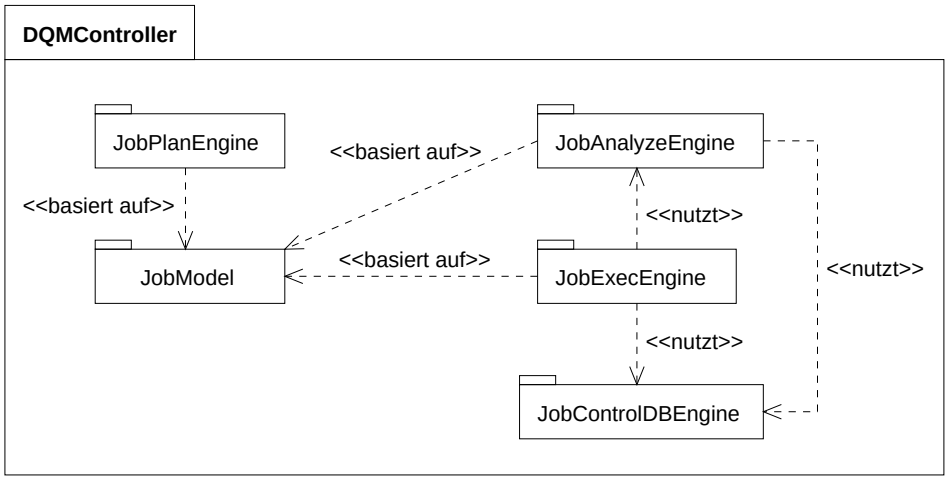


Abbildung 9.5: UML-Paketdiagramm zum DQM Controller

Abbildung 9.5 gibt einen Überblick über die Pakete des DQM Controllers. Die in Abschnitt 9.3.1 besprochenen Anwendungsfälle (Konfigurieren, Ausführen, Analysieren) spiegeln sich direkt in korrespondierenden Paketen wider (*JobPlanEngine, JobExecEngine, JobAnalyzeEngine*). Diese werden durch ein Hilfspaket zum Datenbankzugriff (*JobControlDBEngine*) und ein Paket mit grundlegenden Datenstrukturen (*JobModel*) ergänzt.

Paket *JobModel*

Das Paket *JobModel* (siehe Abb. 9.6) enthält die zur Beschreibung eines Jobs notwendigen Klassen. Alle Klassen dieses Pakets wurden direkt aus dem Modell der Business-Klassen (siehe Abb. 8.2) übernommen.

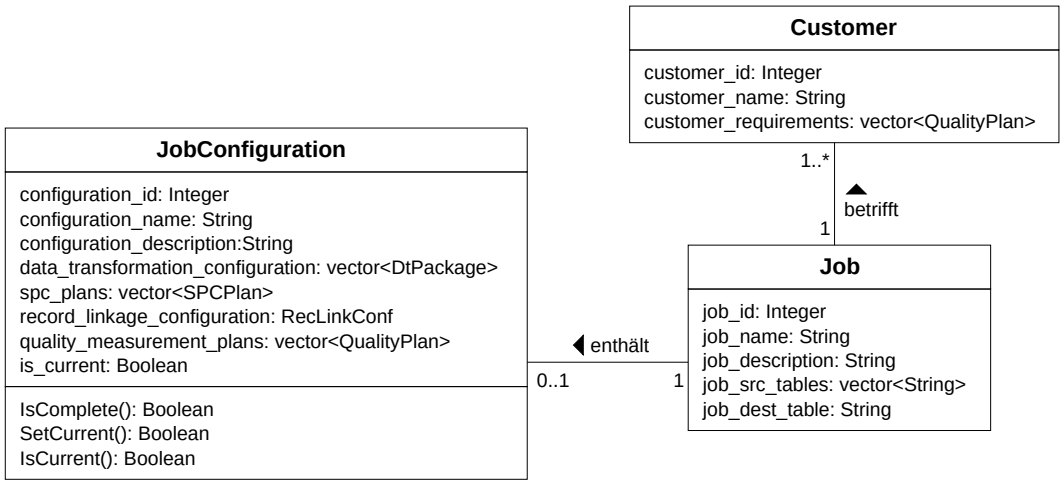


Abbildung 9.6: UML-Klassendiagramm zum Paket *JobModel* (aus [Ade01], revidiert)

Im Folgenden werden die Klassen des Pakets in alphabetischer Reihenfolge vorgestellt.

- *Customer*
  Die Klasse *Customer* repräsentiert Kunden, die an einem Datenprodukt, das durch die Ausführung eines Jobs erstellt werden soll, interessiert sind. Anforderungen, die ein Kunde an ein Datenprodukt stellt, werden über das Attribut *customer_requirements* als Menge von sog. *Qualitätsplänen* (Klasse *QualityPlan*, siehe Abschnitt 9.5.2) referenziert. Dabei wird – wie auch in vielen der nachfolgend beschriebenen Klassen – von der Containerklasse *vector* der *Standard Template Library (STL)* [MS98] Gebrauch gemacht. Die übrigen Attribute der Klasse *Customer* sind nur beschreibender Natur (Name und Kundennummer).

- *Job*
  Die Attribute der Klasse *Job* dienen der Beschreibung einer Integrationsaufgabe. Zur Definition eines Jobs gehören im Wesentlichen eine Menge von Quellrelationen (Attribut *job_src_tables*) und eine Zielrelation (Attribut *job_dest_table*). Ein *Job* kann eine Konfiguration besitzen (Assoziation zu *JobConfiguration*) und betrifft eine Menge von Kunden (Assoziation zu *Customer*).

- *JobConfiguration*
  Die Klasse *JobConfiguration* verwaltet in ihren Attributen alle für die Durchführung eines Jobs notwendigen Konfigurationen zu den einzelnen Phasen des Datenintegrationsprozesses (zu *DtPackage*, *SPCPlan* und *RecLinkConf* siehe Abschnitte 9.4.2, 9.5.2 und 9.6.2). Die Methode *IsComplete()* gibt Auskunft darüber, ob eine Konfiguration vollständig ist. Dies ist genau dann der Fall, wenn folgende Bedingungen gelten (konjunktive Verknüpfung), wobei sich jede der Bedingungen auf genau eine (konfigurierbare) Phase des CLIQ-Vorgehensmodells bezieht und über die Zugehörigkeit einer Konfiguration C zu einem Job J der Bezug zu einer Zielrelation und zu einer Teilmenge von Kunden gegeben ist:

  – Für jede Quellrelation R in *J.job_src_tables* existiert ein Transformationspaket in *C.data_transformation_configuration*, das sich auf R bezieht (Phase 2, Vereinheitlichung der Repräsentation).
  – Es existiert mindestens eine SPC-Konfiguration in *C.spc_plans* (Phase 3, Statistische Prozesskontrolle).
  – Es existiert eine Record Linkage-Konfiguration in *C.record_linkage_configuration* (Phase 5, Record Linkage).
  – Zu jedem mit dem Job J assoziierten Kunden K existiert mindestens ein Qualitätsplan in *C.quality_measurement_plans*, der sich auf K bezieht (Phase 7, Qualitätsmessung und -analyse).

  Eine vollständige Konfiguration kann mit Hilfe der Methode *SetCurrent()* aktiviert werden, so dass sie in nachfolgenden Ausführungsläufen des Jobs verwendet wird. Die bisher aktivierte Konfiguration wird dabei historisiert, d. h. deaktiviert. Mittels *IsCurrent()* lässt sich der Aktivierungsstatus einer Konfiguration ermitteln. Die übrigen, hier nicht näher erläuterten Methoden der Klasse dienen dem Setzen, Löschen und Verändern von einzelnen Prozesskonfigurationen.

Paket *JobPlanEngine* **(Planungskomponente)**

Die Planungskomponente in Form des Pakets *JobPlanEngine* dient der Steuerung von Dialogklassen (siehe Abschnitt 9.9), die das Erstellen und Ändern von Konfigurationen zu einem Job ermöglichen.

Die in der Planungskomponente definierten Jobs und Konfigurationen werden mit Hilfe von Methoden der Klasse *JobControlDB* aus dem Paket *JobControlDBEngine* in einer relationalen Datenbank abgelegt. Nähere Informationen zum zu Grunde liegenden Datenschema finden sich in [Ade01].

Paket *JobExecEngine* **(Ausführungskomponente)**

Das Paket *JobExecEngine* stellt die Ausführungskomponente des Controllers dar und enthält Klassen zur qualitätsgelenkten Durchführung von Integrationsaufgaben. Diese Klassen werden im Folgenden – alphabetisch sortiert – kurz charakterisiert. Abbildung 9.7 zeigt das zugehörige Klassendiagramm.

- *CLIQProcess*
  Ein *CLIQProcess*-Objekt repräsentiert einen Prozess, der einer einzelnen Phase des CLIQ-Vorgehensmodells entspricht und von der zuständigen Komponente der CLIQWorkbench ausgeführt wird. Die abstrakte Klasse *CLIQProcess* kapselt die konkreten Prozesse und erlaubt somit ein einfaches Austauschen sowie Hinzufügen von Prozessen (siehe auch Abschnitt 12.1.3).

- *CLIQProcessConfiguration*
  Hierbei handelt es sich um eine abstrakte Klasse, deren Methode *GetConfiguration()* von abgeleiteten Klassen zu überschreiben ist. Damit kann für jeden Prozess des CLIQ-Vorgehensmodells, der eine Konfiguration benötigt, eine eigene Klasse entwickelt werden, die genau die jeweils erforderlichen Konfigurationsangaben verwalten kann. Erzeugt von einem *JobThread*, repräsentiert ein Objekt einer von *CLIQProcessConfiguration* abgeleiteten Klasse einen Teilausschnitt der Konfiguration zu einem Job. Auf die Konfigurationen, die die einzelnen Prozesse des Vorgehensmodells benötigen, wird im Zusammenhang mit der Vorstellung der zuständigen CLIQWorkbench-Komponenten näher eingegangen (siehe Abschnitte 9.4 bis 9.8).

- *CLIQProcessInfo*
  Eine Instanz dieser Klasse enthält in ihren Attributen Informationen zum Aufbau einer Ablaufumgebung für ein *CLIQProcess*-Objekt, insbesondere Angaben zum Datenbankzugriff. Ein *CLIQProcessInfo*-Objekt wird einmalig von einem *JobThread* erzeugt und dann an *JobStep*-Objekte als Referenz übergeben. Diese können mit Hilfe der übergebenen Informationen selbstständig Datenbankverbindungen aufbauen.

- *CLIQProcessResult*
  Analog zu *CLIQProcessConfiguration* kapselt die ebenfalls abstrakte Klasse *CLIQProcessResult* die von der jeweils zuständigen CLIQWorkbench-Komponente berechneten Ergebnisse zu einem mit einer Phase des CLIQ-Vorgehensmodells assoziierten Prozess. Aus den jeweils von *CLIQProcessResult* abgeleiteten Klassen können dann gezielt die (je nach *CLIQProcess* ggf. unterschiedlich repräsentierten) Ergebnisinformationen ausgelesen werden. In der Basisklasse werden ausschließlich Protokollinformationen (*messages*), Fehlermeldungen (*errors*) sowie Angaben zu Ausführungszeit (*process_start*, *process_end*) und Anzahl verarbeiteter Tupel (*tuple_count*) verwaltet.

- *JobExec*
  Diese Klasse kapselt die Funktionalität des Pakets *JobExecEngine*. Durch Aufruf der Methode *ExecuteJobThread()* wird an Hand der übergebenen Kennung ein Job (inkl. seiner aktiven Konfiguration) aus der Datenbank geladen und als *JobThread* instantiiert, wobei intern u. A. ein *JobRunInfo*-Objekt (siehe unten) als Parameter übergeben wird. Der *JobThread* wird daraufhin gestartet, vorausgesetzt, die dafür erforderlichen Ressourcen (z. B. eine Relation im Zielbereich) sind verfügbar (vgl. Abschnitt 7.14).

  Die Methode *RestartJobThread()* verhält sich analog zu *ExecuteJobThread()*. Allerdings darf hier ein Aufruf nur dann erfolgen, wenn sich der Job nicht in der Ausführung befindet und ein zuvor mittels *AbortJobThread()* abgebrochener Job (z. B. bedingt durch einen Programmabbruch) erneut gestartet werden soll. Dazu werden die bereits in der Datenbank befindlichen Informationen in das assoziierte *JobRunInfo*-Objekt geladen, und der *JobThread* setzt die Ausführung des Jobs fort. Die Methode *ResumeJob()* lässt einen zuvor mit *PauseJobThread()* in

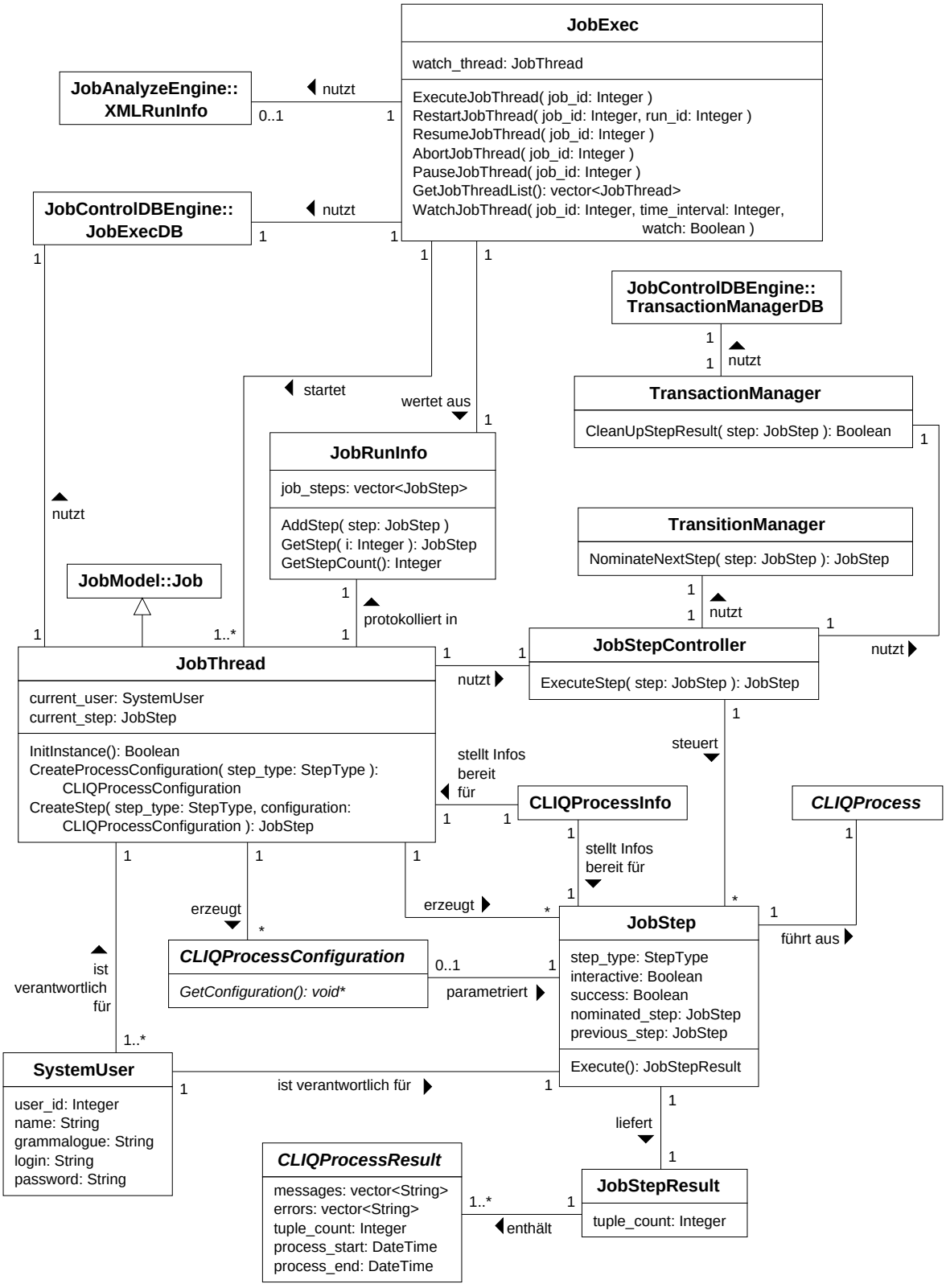


Abbildung 9.7: UML-Klassendiagramm zum Paket *JobExecEngine* (aus [Ade01], revidiert)

einen Ruhezustand versetzten *JobThread* mit der Bearbeitung der Integrationsaufgabe fortfahren. Diese Methode muss aufgerufen werden, um z. B. nach einer interaktiven Phase die Verantwortung über die Ausführung der Integrationsaufgabe an den *JobThread* zurückzugeben.

JobExec verwaltet eine Menge von gestarteten *JobThreads*, auf die über *GetJobThreadList()* zugegriffen werden kann. Das Attribut *watch_thread* bezeichnet einen ggf. zu überwachenden Thread, also einen Thread, dessen Ergebnisse nach jedem Schritt aufbereitet und visualisiert werden sollen. Dieser kann durch Aufruf der Methode *WatchJobThread()* festgelegt werden, sofern sich der entsprechende Job in der Liste der gestarteten Threads befindet. Der Parameter *time_interval* legt dabei die Zeitintervalle fest, in denen das *JobRunInfo*-Objekt des Thread ausgewertet werden soll.

Die Auswertung des *JobRunInfo*-Objekts erfolgt mit Hilfe des assoziierten *XMLRunInfo*-Objekts (siehe dazu den Unterabschnitt zur Analysekomponente).

- *JobRunInfo*
  Diese Klasse dient der Protokollierung von Laufzeitinformationen und der Kommunikation zwischen einem gestarteten Thread (Klasse *JobThread*) und demjenigen Objekt, von dem der Thread instantiiert wurde. Auf ein *JobRunInfo*-Objekt können sowohl der *JobThread* als auch das aufrufende Objekt über entsprechende Synchronisierungsverfahren zugreifen (Methoden *AddStep(), GetStep(), GetStepCount()*).

  Ein Objekt der Klasse *JobRunInfo* enthält zur Laufzeit einer Integrationsaufgabe die gesammelten Protokollinformationen und Messergebnisse der bisher abgeschlossenen Schritte (Attribut *job_steps*) eines Jobs. Nach einem Abbruch einer Integrationsaufgabe kann das Objekt an Hand der in der Datenbank persistent abgelegten Informationen rekonstruiert werden. Bereits gespeicherte Zwischenergebnisse von nicht zu Ende geführten Integrationsschritten können erkannt und aus der Datenbank entfernt werden. Die Integrationsaufgabe kann dann an der Stelle fortgesetzt werden, an der die Protokollierung im *JobRunInfo*-Objekt endet.

- *JobStep*
  Ein Objekt dieser Klasse repräsentiert einen Schritt der Durchführung eines Jobs (entspricht der Ausführung einer Phase des CLIQ-Vorgehensmodells). Es ist dazu mit je einer Instanz der Klassen *CLIQProcessConfiguration* und *CLIQProcessInfo* assoziiert. Durch Aufruf der Methode *Execute()* wird der jeweils in dem *JobStep* spezifizierte *CLIQProcess* mit der entsprechenden Konfiguration ausgeführt. Dabei ist es u. U. notwendig, die Konfiguration weiter in einzelne Elemente aufzuspalten und den *CLIQProcess* jeweils mit diesen Elementen (z. B. einzelne Qualitätspläne) wiederholt auszuführen. Die während der Ausführung berechneten Ergebnisse und generierten Protokollinformationen werden in einem Objekt der Klasse *JobStepResult* abgelegt. Dieses wird von der *Execute()*-Methode zurückgeliefert und intern gespeichert. Das Attribut *interactive* gibt an, ob es sich um einen automatisierten oder einen interaktiven Schritt handelt. Im Attribut *success* wird gespeichert, ob die Ausführung des Schritts erfolgreich war oder nicht (gesetzt durch den *JobStepController*). Die Phase des CLIQ-Vorgehensmodells, auf die sich der *JobStep* bezieht, ist im Attribut *step_type* (vom Aufzählungstyp *StepType*) angegeben. Ein *JobStep* kennt seinen Vorgänger (Attribut *previous_step*) sowie seinen Nachfolger (Attribut *nominated_step*) in der Ausführungssequenz zu einer Integrationsaufgabe. Damit wird der Zugriff auf Ergebnisse eines früher oder später ausgeführten *JobStep* ermöglicht.

- *JobStepController*
  Mit Hilfe der assoziierten Objekte der Klassen *TransitionManager* und *TransactionManager* steuert der *JobStepController* die Ausführung von *JobSteps*. Mittels Aufruf der Methode *ExecuteStep()* durch einen *JobThread* wird die *Execute()*-Methode eines bereits mit einer Konfiguration versehenen *JobStep* aufgerufen. Lässt sich aus dem hieraus resultierenden Ergebnis (siehe Attribute der Klasse *CLIQProcessResult*) ablesen, dass es während der Ausführung des Schritts zu technischen Fehlern gekommen ist, so sind bereits gespeicherte Zwischenergebnisse

durch Aufruf der Methode *CleanUpStepResult()* der Klasse *TransactionManager* aus der Datenbank zu löschen. Ansonsten bestimmt *ExecuteStep()* mit Hilfe der Methode *NominateNextStep()* des referenzierten *TransitionManager*-Objekts den als nächstes auszuführenden Schritt.

- *JobStepResult*
  Ein *JobStepResult* verwaltet die Ergebnisse eines durchgeführten *JobStep*. Die einzelnen, ggf. von mehrfachen Aufrufen desselben *CLIQProcess* stammenden Ergebnisse (z. B. für unterschiedliche Qualitätspläne) werden in Objekten abgelegt, deren Klasse von *CLIQProcessResult* abgeleitet ist. Das Attribut *tuple_count* speichert die kumulierte Anzahl der in den einzelnen *CLIQProcess*-Objekten verarbeiteten Tupel.

- *JobThread*
  Diese zentrale Klasse des Ausführungspakets ist von der Klasse *Job* aus dem Paket *JobModel* abgeleitet und stellt Methoden für die thread-basierte Ausführung eines Jobs bereit. Als Threads werden in diesem Kontext quasi parallel[1] ablaufende Steuerflüsse innerhalb eines Programms bezeichnet. Durch die Verwendung von Threads ist es z. B. möglich, mehrere Jobs (quasi) parallel auszuführen und (quasi) zeitgleich die Planung eines neuen Jobs oder die Analyse abgeschlossener Jobs vorzunehmen, wobei jede Aktivität in einem eigenen Thread abläuft.

  Durch Aufruf von *JobThread::CreateProcessConfiguration()* wird vor der Ausführung eines Jobs die im jeweiligen *Job*-Objekt referenzierte Job-Konfiguration an Hand ihrer Attribute in Teilkonfigurationen zerlegt (Subklassen von *CLIQProcessConfiguration*), die direkt von den CLIQWorkbench-Komponenten, die für einzelne Phasen des CLIQ-Vorgehensmodells zuständig sind (siehe Abschnitte 9.4 bis 9.8), verarbeitet werden können. Aus einer Teilkonfiguration wird dann mittels *JobThread::CreateStep()* der erste Ausführungsschritt des Jobs (Klasse *JobStep*) bestimmt. Die zentrale Methode *InitInstance()* steuert mit Hilfe des referenzierten *JobStepController*-Objekts den Ablauf der Datenintegration.

  Durch Aufruf der Methode *ExecuteStep()* des *JobStepController* wird die Ausführung eines *JobStep* angestoßen. Vor diesem Aufruf wird der Beginn des Ausführungsschritts in der Datenbank protokolliert. Dazu wird eine Methode des referenzierten *JobExecDB*-Objekts verwendet, die die entsprechenden Informationen in die Datenbank überträgt. Das Ende der Ausführung wird ebenfalls in der Datenbank protokolliert, sofern keine technischen Fehler (z. B. Ausfall der Datenbankverbindung) zu einem Abbruch des Schrittes geführt haben. Technische Fehler werden in Objekten der Klasse *JobRunInfo* protokolliert. Der Systemnutzer hat im Einzelfall zu entscheiden, welche korrektiven Maßnahmen durchzuführen sind, bevor der Prozess wiederaufgesetzt werden kann.

  Eine Referenz auf den von der Methode *ExecuteStep()* ausgeführten Prozess (Objekt der Klasse *JobStep*) wird unter Beachtung der sicherzustellenden Synchronisation (siehe Erläuterungen zu *JobRunInfo*) in das referenzierte *JobRunInfo*-Objekt eingefügt. In dem *JobStep*-Objekt ist der vom *JobStepController* gemäß dem CLIQ-Vorgehensmodell identifizierte Nachfolgeschritt referenziert. Dieser kann mittels Aufruf von *JobThread::CreateProcessConfiguration()* mit einer Konfiguration versehen werden. Ist eine Interaktion mit einem Systemnutzer erforderlich, stellt der *JobThread* seine Aktivität zunächst ein und gibt per Signalaustausch über *JobRunInfo* die Verantwortung an dasjenige Objekt zurück, das den Thread gestartet hat. Nutzerinteraktionen sind wiederum über Threads parallelisierbar, so dass z. B. mehrere Systemnutzer zeitgleich inkonsistente Tupel nachbearbeiten können. Nach Abschluss des interaktiven Prozesses kann der *JobThread* reaktiviert werden. Dieser fährt dann mit einer durch das Vorgehensmodell bestimmten Phase fort.

  Das Objekt der Klasse *JobRunInfo* muss in dem aufrufenden Objekt angelegt und an das *JobThread*-Objekt übergeben werden, damit eine synchronisierte Kommunikation möglich ist.

[1] Das „quasi“ gilt insbesondere für Rechner mit nur einem Prozessor.

Sind keine Protokollinformationen darin gespeichert, wird beim Start eines *JobThread* von einer von Beginn an (Phase 1 des Vorgehensmodells) auszuführenden Integrationsaufgabe ausgegangen. Sind hingegen Protokollinformationen enthalten, wird eine zuvor abgebrochene Integrationsaufgabe fortgesetzt.

Ein Thread wird dann beendet, wenn die Integrationsaufgabe entweder vollständig durchgeführt (Abschluss der Phase 9 des Vorgehensmodells) oder explizit durch einen Systemnutzer abgebrochen wurde.

- *SystemUser*
  Diese Klasse entspricht der gleichnamigen Business-Klasse aus Abschnitt 8.2.2. In ihren Attributen werden Name, Namenskürzel, Login und Passwort von Systemnutzern gespeichert. Objekte dieser Klasse werden von Objekten der Klassen *JobThread* (der „aktuelle“ Nutzer) und *JobStep* (der für den Schritt verantwortliche Nutzer) referenziert.

- *TransactionManager*
  Die Klasse *TransactionManager* stellt Funktionalität zum Entfernen von Zwischenergebnissen und Protokollinformationen eines aufgrund von technischen Fehlern oder explizit durch den Systemnutzer abgebrochenen *JobStep* bereit. An Hand eines *JobStep*-Objekts wählt die Methode *CleanUpStepResult()* die zu löschenden Daten aus und übergibt diese an die entsprechenden Methoden des assoziierten *TransactionManagerDB*-Objekts.

- *TransitionManager*
  Die Methode *NominateNextStep()* der Klasse *TransitionManager* analysiert die Ergebnisse eines zuvor durchgeführten Schritts und bestimmt daraufhin den nachfolgend auszuführenden Schritt (in Form eines neu erzeugten *JobStep*-Objekts) auf der Grundlage der in Abb. 7.3 dokumentierten Logik für den Übergang zwischen Phasen des CLIQ-Vorgehensmodells.

Die Präzisierung der einzelnen Prozesskonfigurationen wird im Zusammenhang des Entwurfs der jeweils zuständigen Komponenten der CLIQWorkbench vorgenommen (siehe Abschnitte 9.4 bis 9.8).

Paket *JobAnalyzeEngine* (Analysekomponente)

Das Paket *JobAnalyzeEngine* enthält Klassen, die zum einen für eine dateibasierte Protokollierung der Ausführung von Jobs und zum anderen für die Durchführung von Analysen auf diesen Protokollen benötigt werden. Abbildung 9.8 zeigt die wesentlichen, für das Verständnis des Zusammenspiels wichtigen Attribute und Methoden der Klassen des Pakets. Im Folgenden wird näher auf die einzelnen Klassen eingegangen (in alphabetischer Reihenfolge).

- *DiagramEngine*
  Diese Klasse erlaubt es, aus gegebenen Messergebnissen und zugehörigen Qualitätsplänen oder SPC-Plänen Diagramme zu erstellen. Diese werden mit Hilfe der extern angebundenen Applikation *Microsoft Excel* erzeugt (siehe auch Abschnitt 10.1) und unter einem gewünschten Namen dateibasiert abgelegt. Für die einzelnen Diagrammtypen und die jeweils als Eingabe zu verwendenden Datenstrukturen werden unterschiedliche Methoden angeboten (zu den Klassen *SPCPlan*, *SPCXChartEntry*, *SPCParetoChartEntry*, *QualityPlan* und *QualityMeasurement* siehe Abschnitt 9.5.2).

- *JobAnalyze*
  Die von *XMLEngine* abgeleitete Klasse *JobAnalyze* bietet Methoden an, mit denen sich die in Abschnitt 9.3.1 genannten Analyseverfahren durchführen und Analyseergebnisse in XML-Dokumenten aufbereiten lassen. Im Folgenden wird eine Zuordnung der Analyseverfahren zu den in Abb. 9.8 dargestellten Methoden von *JobAnalyze* vorgenommen:

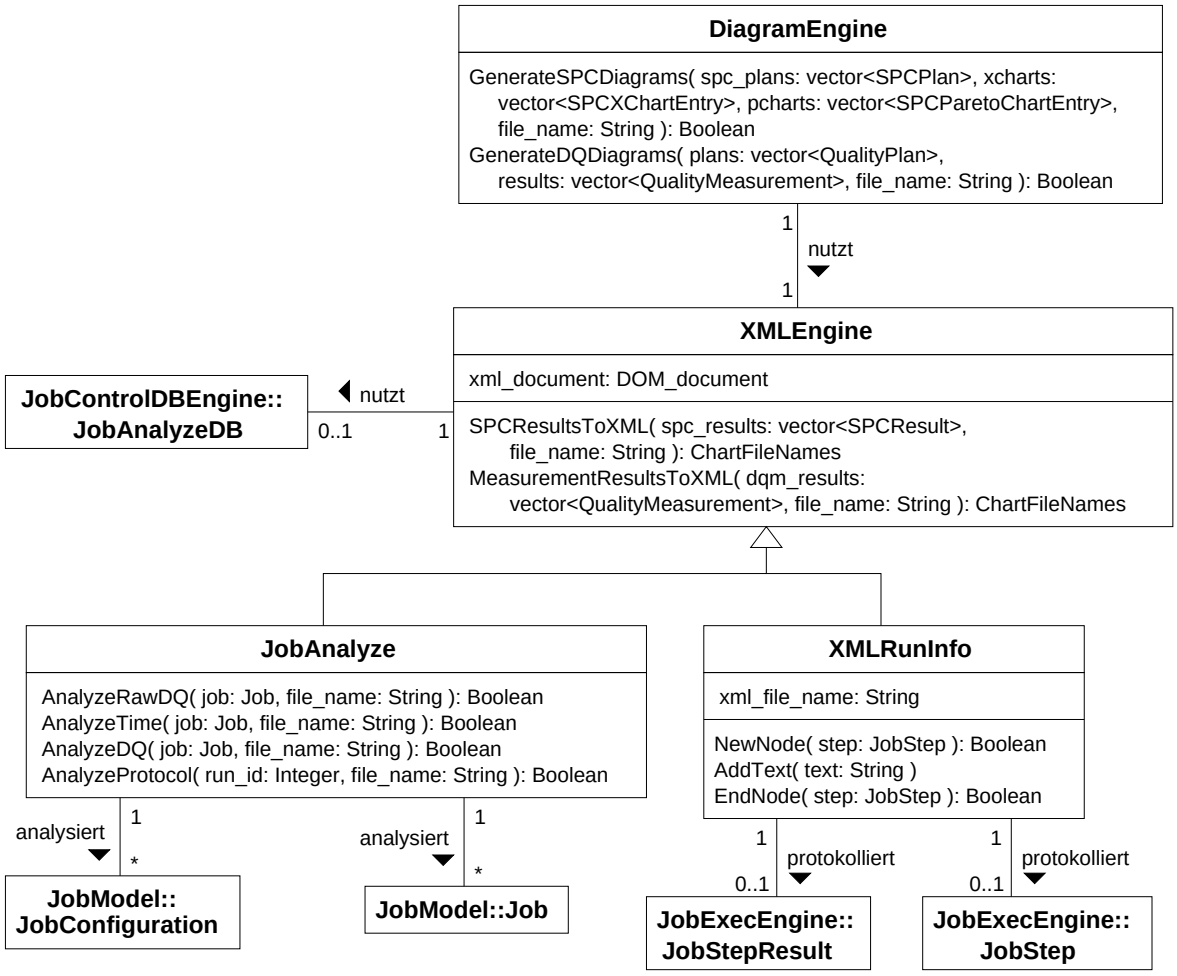


Abbildung 9.8: UML-Klassendiagramm zum Paket *JobAnalyzeEngine* (aus [Ade01], revidiert)

– *AnalyzeRawDQ(job: Job, file_name: String): Boolean*
Diese Methode dient zur Analyse der Qualität der von einer Quelle gelieferten Rohdaten über die Zeit. Aus der Datenbank werden diejenigen Messergebnisse aus Statistischen Prozesskontrollen (Phase 3) und Konsistenzprüfungen (Phase 4) selektiert, die im Rahmen der bisher zum übergebenen Job abgeschlossenen Ausführungsläufe angefallen sind. Die selektierten Ergebnisse werden dann – über die Zeit aufgetragen und nach SPC- bzw. Konsistenzprüfungen differenziert – in Diagramme eingetragen (via *DiagramEngine*) und in einer XML-Datei abgelegt (via *XMLEngine*).

– *AnalyzeTime(job: Job, file_name: String): Boolean*
Der Aufruf dieser Methode stößt eine Analyse der Zeiteffizienz des übergebenen Jobs an. Mit Hilfe des referenzierten *JobAnalyzeDB*-Objekts werden die zu abgeschlossenen Ausführungsläufen des Jobs in der Datenbank protokollierten Bearbeitungszeiträume selektiert. Die Ergebnisse werden – differenziert nach jeweils verwendeten Konfigurationen – in einem 3D-Balkendiagramm dargestellt, das zu einem Ausführungslauf die benötigte Zeit zeigt. Um die Zeiträume und damit die Konfigurationen vergleichbar zu machen, werden in einem zweiten Diagramm relative Ausführungszeiten (pro gegebener Tupelanzahl) dargestellt. Beide Diagramme werden in einer Datei gespeichert.

– *AnalyzeDQ(job: Job, file_name: String): Boolean*
Diese Methode ermöglicht eine Analyse der Datenqualität im Zielbereich. Mit Hilfe der durch die Klasse *JobAnalyzeDB* bereitgestellten Funktionalität werden die auf abgeschlossene Ausführungsläufe des übergebenen Jobs und damit eine festgelegte Zielre-

lation bezogenen, in Phase 7 (Qualitätsmessung und -analyse) der Datenintegration ermittelten Messergebnisse selektiert. Diese werden über die Zeit – aufgeschlüsselt nach Datenqualitätsmerkmalen – in einem Diagramm aufbereitet (via *DiagramEngine*) und in einem XML-Dokument gespeichert.

Aus den in der Datenbank abgelegten Protokollinformationen und Messergebnissen (für Details siehe [Ade01]) lässt sich jederzeit das zu einem Durchlauf eines Jobs geführte Protokoll rekonstruieren. Durch Aufruf der Methode *AnalyzeProtocol()* wird mit Hilfe eines *XMLRunInfo*-Objekts und indirekt eines *DiagramEngine*-Objekts ein zur übergebenen *run_id* gehörendes Protokoll erstellt und in einer Datei abgelegt.

- *XMLEngine*
  Diese zentrale Klasse nutzt die C++-Klassenbibliothek *Xerces*[Apa01] zum Parsen, Generieren, Manipulieren und Validieren von XML-Dokumenten. Das jeweils bearbeitete Dokument ist vom Xerces-Typ *DOM_document* und wird im Attribut *xml_document* abgelegt. Mit Hilfe der Methoden *SPCResultsToXML()* und *MeasurementResultsToXML()* können Messergebnisse, die im Rahmen eines Datenintegrationsprozesses gemäß dem CLIQ-Vorgehensmodell ermittelt wurden, in XML-Dokumenten aufbereitet werden (zur Datenstruktur *SPCResult* siehe Abschnitt 9.5.2). Intern erzeugen die Methoden XML-Dateien, die von der als Parameter übergebenen Datei mit dem Namen *file_name* referenziert und deren automatisiert generierte Namen in der Datenstruktur *ChartFileNames* zurückgeliefert werden.

- *XMLRunInfo*
  Mittels *XMLRunInfo* können bereits zur Ausführungszeit eines Jobs Protokollinformationen aufbereitet werden. Die Methode *NewNode()* macht dazu die Ausführung eines neuen *JobStep* bekannt, *AddText()* fügt eine einfache Textnachricht in das Protokoll zum aktuellen *JobStep* ein, und *EndNode()* übernimmt die Protokollinformationen zu einem *JobStep* in das mitgeführte Protokoll. Den Protokollen liegt folgende Document Type Definition (DTD) zu Grunde:

```
<!-- Ein Protokoll besteht aus einer Folge von Schritten. -->
<!ELEMENT cliq_protocol     (step*) >

<!-- Ein Protokoll hat eine ID und einen Namen. -->
<!ATTLIST cliq_protocol
          run_id            CDATA #REQUIRED
          job_name          CDATA #REQUIRED >

<!-- Ein Schritt besteht aus einer Folge von Nachrichten, -->
<!-- einem OK-Flag und einem Ergebnis. -->
<!ELEMENT step              (msg*, ok_value, result) >

<!-- Ein Schritt hat eine ID und einen Namen. -->
<!ATTLIST step
          id                CDATA #REQUIRED
          step_name         CDATA #REQUIRED >

<!-- Eine Nachricht besteht aus einer Folge von Eintraegen. -->
<!ELEMENT msg              (entry*) >

<!-- Eintraege, OK-Flags und Ergebnisse sind Terminale. -->
<!ELEMENT entry            (#PCDATA) >
<!ELEMENT ok_value         (#PCDATA) >
<!ELEMENT result           (#PCDATA) >

<!-- Ein Ergebnis hat einen Link und eine Beschreibung. -->
<!ATTLIST result
```

```
link            CDATA #REQUIRED
text            CDATA #REQUIRED >
```

XMLRunInfo-Objekte werden durch die Klasse *JobAnalyze* aus diesem Paket und durch die Klasse *JobExec* aus dem Paket *JobExecEngine* angesprochen.

Paket *JobControlDBEngine*

Das Paket *JobControlDBEngine* (siehe Abb. 9.9) stellt Funktionalität für den Zugriff auf die in der zu Grunde liegenden relationalen Datenbank verwalteten Konfigurationen, Messergebnisse und Protokollinformationen zu Integrationsprozessen bereit:

- *JobControlDB*
  Diese Klasse bietet Basisfunktionalität für den Datenbankzugriff. Mit den Methoden der Klasse können sowohl Jobs als auch Konfigurationen (inkl. der einzelnen Prozesskonfigurationen) gespeichert, geladen, manipuliert und gelöscht werden. Die in Abb. 9.9 aufgeführten Methoden zu dieser Klasse sind weitgehend selbsterklärend.

- *JobExecDB*
  Die Klasse *JobExecDB* erweitert die Klasse *JobControlDB* um Methoden, die insbesondere den Zugriff auf Laufzeitinformationen zum Datenintegrationsprozess ermöglichen. *RunStartToDB()* liefert zu einer gegebenen Konfiguration eine Durchlaufkennung zurück. Die Beendigung eines Durchlaufs (*run_id*) wird der Datenbank durch den Systemnutzer (*user_id*) mittels *RunEndToDB()* „mitgeteilt“, wobei im Parameter *finished* übergeben wird, ob der Durchlauf abgebrochen wurde oder nicht. Die Methoden *StepStartToDB()* und *StepEndToDB()* arbeiten analog, beziehen sich aber auf einzelne Prozessschritte. *StepFromDB()* schließlich liest Angaben zu einem Prozessschritt aus der Datenbank aus.

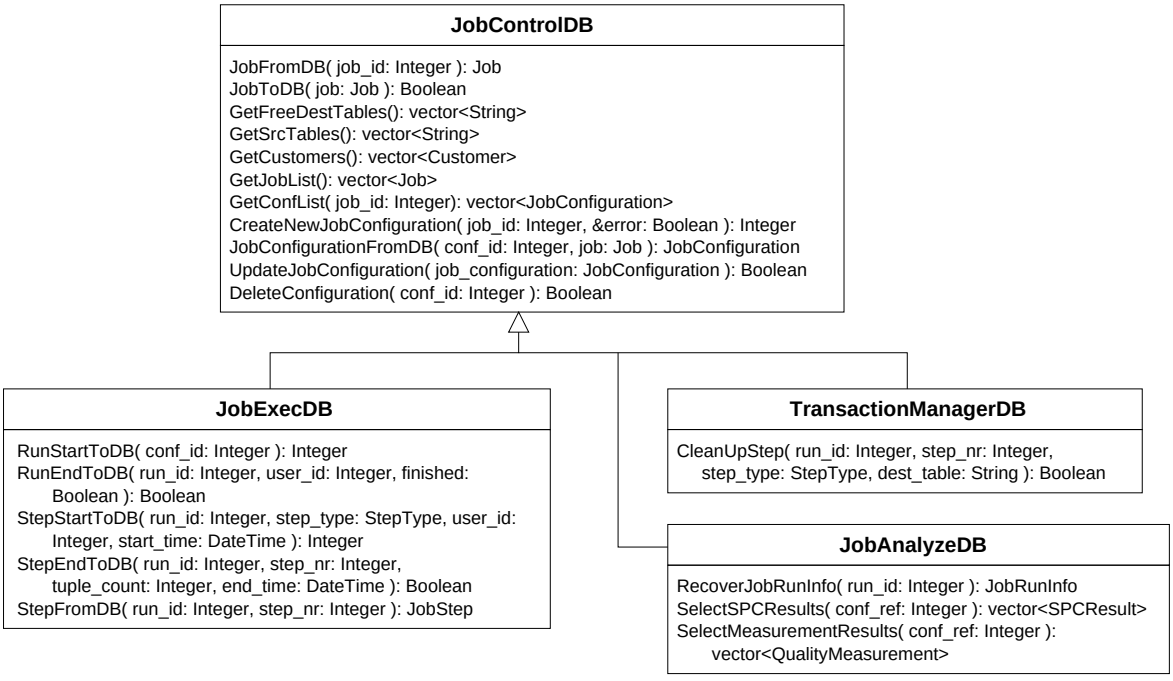


Abbildung 9.9: UML-Klassendiagramm zum Paket *JobControlDBEngine* (aus [Ade01], revidiert)

- *JobAnalyzeDB*
  Diese Klasse erweitert die Klasse *JobControlDB* um Methoden zur Selektion von historisiert verwalteten Messergebnissen, Laufzeitinformationen und Konfigurationen zu Analysezwecken. Die in Abb. 9.9 aufgeführten Methoden zu dieser Klasse sind weitgehend selbsterklärend (zu *SPCResult* und *QualityMeasurement* siehe Abschnitt 9.5.2).

- *TransactionManagerDB*
  Die Klasse *TransactionManagerDB* stellt die Methode *CleanUpStep()* zum Löschen von Teilergebnissen bereit, um nach dem Abbruch eines Datenintegrationsprozesses einen konsistenten Datenbankzustand wiederherzustellen. Im Rahmen der Erläuterungen zur Ausführungskomponente wurde genauer darauf eingegangen.

Beziehungen von Klassen aus *JobControlDBEngine* zu Klassen aus anderen Paketen wurden bereits in den Klassendiagrammen zu diesen Paketen dokumentiert und sind daher in Abb. 9.9 nicht noch einmal aufgeführt.

9.4 Data Migrator

Der Data Migrator dient der softwaretechnischen Unterstützung von Phase 2 des CLIQ-Vorgehensmodells (Vereinheitlichung der Repräsentation von Daten). An der Konzeption des Data Migrators war Jürgen Meister (Mitarbeiter im CLIQ-Projekt) maßgeblich beteiligt. Über die nachfolgenden Ausführungen hinausgehende Entwurfsdetails können [MH01] entnommen werden.

9.4.1 Operative Anforderungen

Die Hauptaufgabe des Data Migrators besteht in der Spezifizierung und Durchführung von Transformationen zur Vereinheitlichung der Repräsentation von Rohdaten, die in Quelldatenpuffern vorgehalten werden, und dem Ablegen der vereinheitlichten Daten im Transformationsbereich. Die im Rahmen der Vereinheitlichung anfallenden Typen von Transformationen wurden bereits in Abschnitt 7.4 diskutiert.

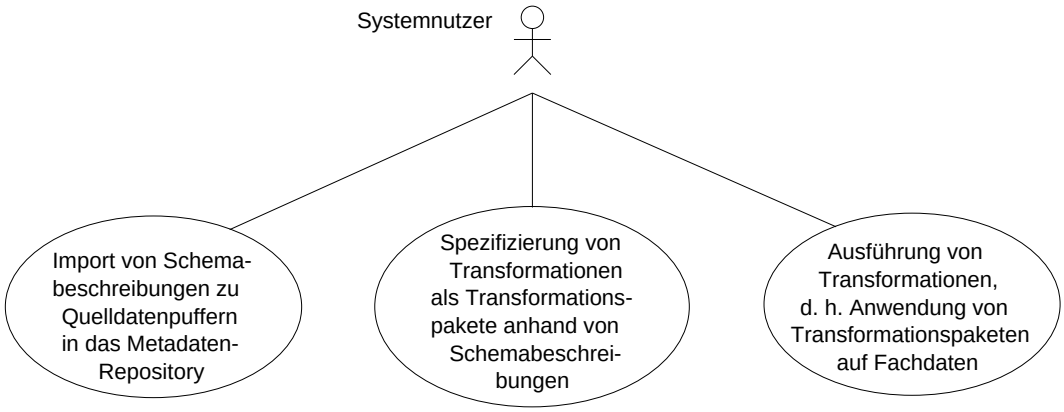


Abbildung 9.10: UML-Anwendungsfalldiagramm zum Data Migrator

Eine weitere Aufgabe des Data Migrators ist die Unterstützung inkrementeller Updates. Dazu sind insbesondere Verknüpfungen zwischen den von den Quellen gelieferten lokalen und den vom Data Migrator erzeugten globalen Schlüsseln zu verwalten. Weiterhin sind tupelbezogene Metadaten, die

im Rahmen der weiteren Datenintegration benötigt werden, durch den Data Migrator geeignet zu initialisieren.

Für die Durchführung der Datenextraktion ist der Data Migrator hingegen *nicht* zuständig. Es wird davon ausgegangen, dass pro Datenquelle ein (in dieser Arbeit nicht betrachteter) Monitor existiert, der seit dem vorangegangenen Extraktionsvorgang neu erzeugte, geänderte und gelöschte Datensätze erkennt, aus der Quelle extrahiert und im zugehörigen Quelldatenpuffer ablegt.

Insgesamt werden in dieser Arbeit drei Anwendungsfälle unterschieden (siehe Abb. 9.10):

- *Import von Schemabeschreibungen*
  Hier wird die Schemabeschreibung eines Quelldatenpuffers bestimmt und im Metadaten-Repository abgelegt. Die Schemainformationen werden später für die Spezifizierung von Transformationen benötigt.
- *Spezifizierung von Transformationen*
  Ausgehend von den Schemabeschreibungen zu einem Quelldatenpuffer und zum Transformationsbereich wird durch den Systemnutzer festgelegt, wie die Attribute der Quellrelationen auf die Attribute der Zielrelation abzubilden sind. Diese Abbildungsvorschriften werden in einem sog. *Transformationspaket* abgelegt. Zusätzlich werden intern automatisiert diejenigen Bestandteile des Transformationspakets generiert, die zur Unterstützung inkrementeller Updates benötigt werden.
- *Ausführung von Transformationen*
  Die Ausführung eines Transformationspakets wird durch einen Systemnutzer bzw. durch den DQM Controller angestoßen. Das Transformationspaket greift wiederum auf die Schemabeschreibungen im Metadaten-Repository zurück, um die spezifizierten Transformationen auf die Daten des Quelldatenpuffers anzuwenden und die transformierten Daten im Transformationsbereich abzulegen.

9.4.2 Architektur und Datenstrukturen

Der Data Migrator basiert auf den *Data Transformation Services* (DTS) von Microsoft [AL99] – ihrerseits auf dem *Component Object Model (COM)* [EE98, Mal99a] und *OLE DB* [Nal01] aufbauend (siehe auch Abschnitt 10.2.3) – und dem von diesen verwendeten *Data Transformations Model* des *Open Information Model (OIM)* [Met99a], einem Standard zur Repräsentation von Metadaten, insbesondere für DWS. Diese Entwurfsentscheidung reduziert den Entwurfs- und Implementierungsaufwand des Data Migrators erheblich, da bereits modellierte und implementierte Basisdienste zur Datenintegration genutzt werden können. Die DTS stellen eine Infrastruktur für die Spezifizierung und Ausführung von Datentransformationen zur Verfügung. Der „direkte“ Einsatz der DTS (ohne eine darauf aufsetzende Schicht) ist aufgrund folgender Probleme nicht sinnvoll:

- Die mitgelieferten Front End-Werkzeuge (siehe Abschnitt 10.2.3) unterstützen nur einen Teil der Funktionalität der DTS.
- Der Systemnutzer könnte Transformationspakete spezifizieren, die nicht zu der in Abschnitt 7.4 festgelegten Spezifikation von Datentransformationen im CLIQ-Vorgehensmodell konform sind.

Aus diesen Gründen wird die DTS-Funktionalität in dieser Arbeit durch die Klassen des Data Migrators gekapselt (Präfix *Dt*, siehe auch Abb. 9.11). Im Folgenden wird erläutert, wie der Data Migrator die in Abschnitt 9.4.1 skizzierten Anwendungsfälle unterstützt.

Import von Schemabeschreibungen

Die Klassen zum Metadatenimport gehören zum Paket *MrRepository*, welches Standardzugriffe auf das Metadaten-Repository kapselt. Für weitere Details siehe Abschnitt 9.10.

Spezifizierung von Transformationen

Für jeden Quelldatenpuffer, aus dem Daten in eine vorgegebene Zielrelation überführt werden sollen, wird genau ein Transformationspaket als Objekt der Klasse *DtPackage* definiert (siehe Abb. 9.11). Dazu werden zunächst automatisiert OLE DB-Verbindungen zum assoziierten Quelldatenpuffer sowie zum Transformationsbereich aufgebaut. Ein *DtPackage* lässt sich laden (Methode *Open()*), bearbeiten (Methoden *CreateTaskSet()* und *DeleteTaskSet()*), ausführen (Methode *Execute()* unter Nutzung von *CreateSrcConnection()* und *CreateDestConnection()*) und speichern (Methode *Save()*).

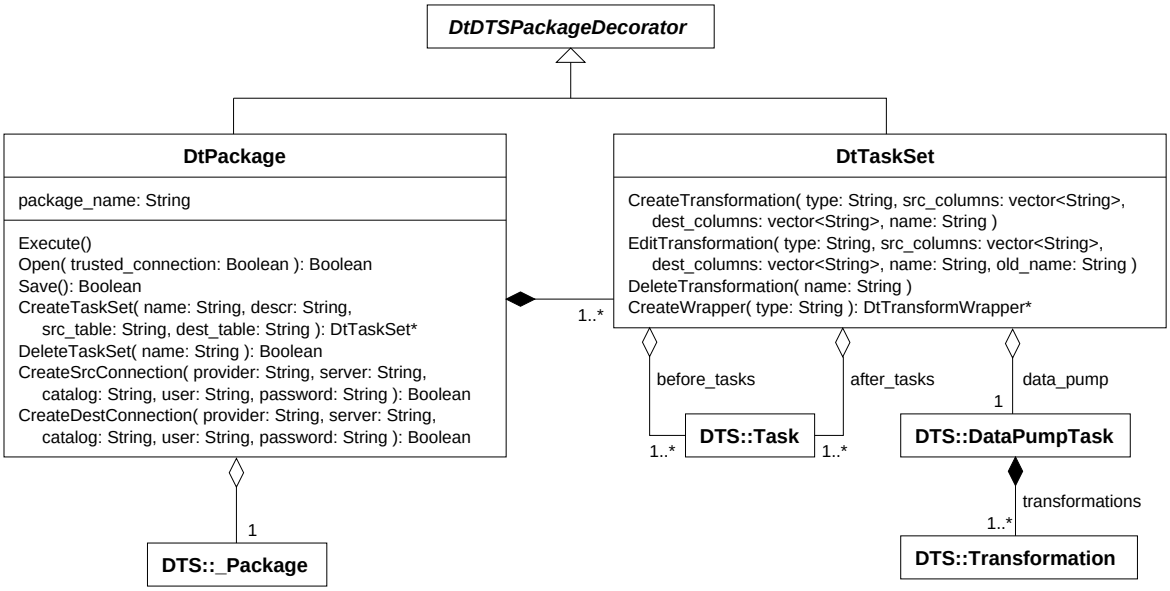


Abbildung 9.11: UML-Klassendiagramm zum Data Migrator

Innerhalb eines *DtPackage* wird mittels der Methode *CreateTaskSet()* für jede QDP-Relation, die in die Zielrelation eingeht, interaktiv ein sog. *Task Set* definiert (siehe Abb. 9.12), welches als Objekt der Klasse *DtTaskSet* realisiert ist und beschreibt, wie die Attribute der QDP-Relation auf die Attribute der Zielrelation abzubilden sind. Die Transformationen eines *DtTaskSet* können mit Hilfe der Methoden *CreateTransformation()* und *EditTransformation()* bearbeitet werden. In beiden Methoden kommt die Hilfsmethode *CreateWrapper()* zum Einsatz, die in Abhängigkeit vom jeweiligen Transformationstyp eine Wrapper-Klasse instantiiert (siehe dazu Abschnitt 9.4.3).

Zentrales Element eines *DtTaskSet* ist ein sog. *DataPumpTask*, in den durch den Systemnutzer Abbildungsvorschriften eingebettet werden. Zusätzlich wird in jedem *DataPumpTask* implizit eine Transformation zum Setzen von global eindeutigen Primärschlüsseln in der Zielrelation spezifiziert. Einem *DataPumpTask* werden automatisch bestimmte Tasks vor- und ggf. nachgeschaltet (Assoziationen *before_tasks* und *after_tasks*). Diese enthalten Spezifikationen zur Unterstützung von inkrementellen Updates. Im einfachsten Fall wird genau ein vorgeschalteter Task instantiiert, der sich um das Aktualisieren von Metarelationen zur Vorbereitung auf die bevorstehende Datenmigration kümmert (siehe dazu Abschnitt 9.11). Durch den Konstruktor der Klasse *DtTaskSet* wird das Grundgerüst des Task Set zusammen mit der Default-Transformation zum Setzen des globalen Schlüssels instantiiert.

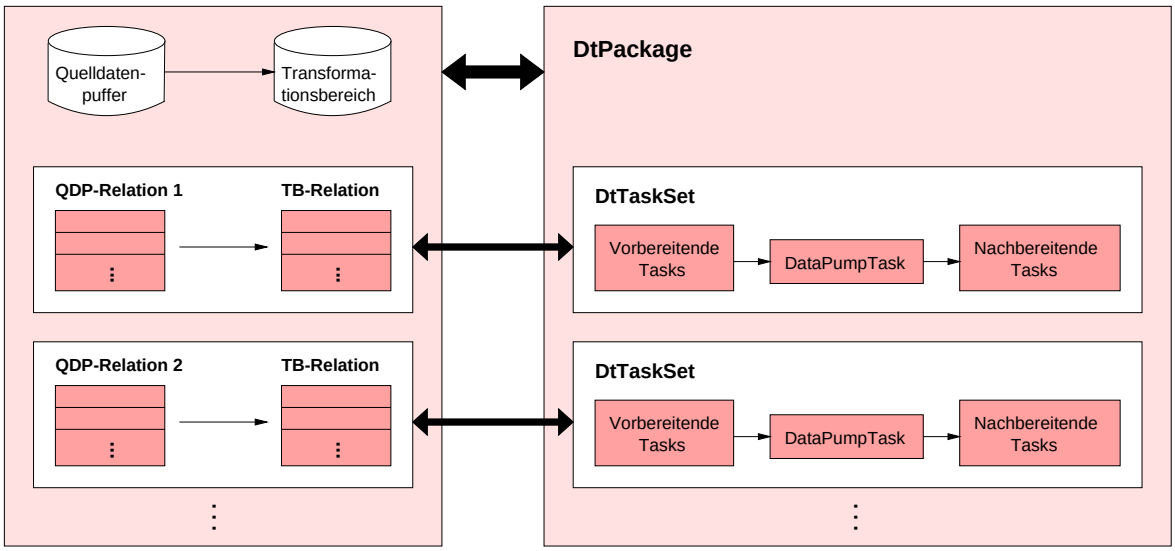


Abbildung 9.12: Grundgerüst eines *DtPackage*

Die abstrakte Basisklasse *DtDTSPackageDecorator* implementiert Hilfsmethoden zum Abfragen und Verändern von DTS-Paketen.

Ausführung von Transformationen

Bei der Ausführung eines Transformationspakets – angestoßen durch Aufruf der Methode *DtPackage::Execute()* – werden die darin spezifizierten Transformationsvorschriften auf die in den Relationen des assoziierten Quelldatenpuffers vorliegenden Tupel angewandt. Transformationspakete entsprechen damit Filtern, die zwischen Quelldatenpuffer und Transformationsbereich geschaltet sind. Jede Transformation liest eine oder mehrere Attributausprägungen aus einer Quellrelation aus und überführt diese in einen oder mehrere Attributwerte eines neuen Tupels der Zielrelation.

9.4.3 Einbindung neuer Transformationsverfahren

Die DTS erlauben es, zusätzlich zu vorgegebenen Transformationsvorschriften (Kopieren von Attributwerten und Verwendung von ActiveX-Skripten) auch nutzerdefinierte oder durch ein externes ETL-Werkzeug bereitgestellte Abbildungen „einzuklinken“ (siehe Abb. 9.13).

Zur Einbindung eines neuen Transformationsverfahrens in den Data Migrator ist mit Hilfe der ActiveX Template Library (ATL) ein sog. *In Process-COM-Server* [EE98] mit folgenden Schnittstellen zu implementieren:

- *IDispatch:* Allgemeine Schnittstelle, deren Implementierung automatisiert durch die ATL erfolgt.
- *IDTSDataPumpTransform:* Eine COM-Schnittstelle, die Funktionalität zur Transformation von ausgewählten Attributen eines Quelltupels in ausgewählte Attribute eines Zieltupels anbietet. Eigenentwickelte Transformationsverfahren müssen diese Schnittstelle implementieren, damit sie mit Objekten der Klasse *DTS::DataPumpTask* zusammenarbeiten können.
- Eine nutzerdefinierte Schnittstelle, die eine Parametrierung des neu eingebundenen Verfahrens ermöglicht.

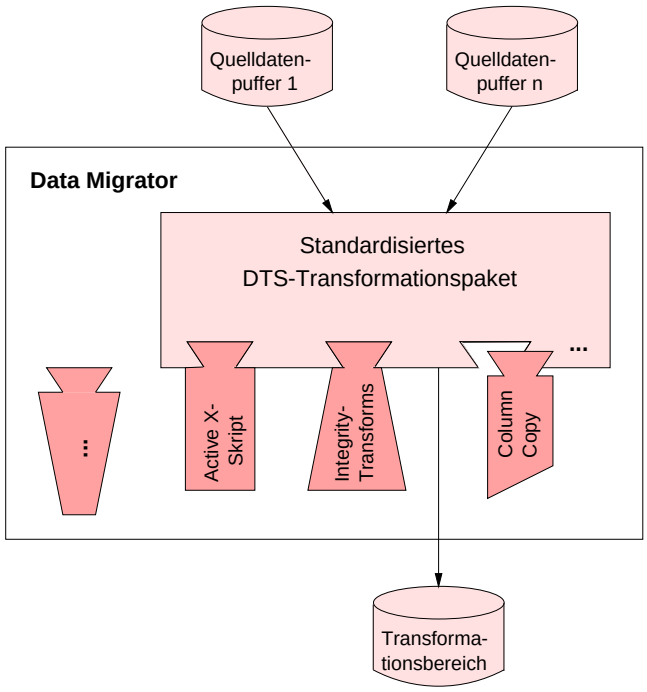


Abbildung 9.13: Einbindung von Transformationsverfahren in den Data Migrator

Transformationsverfahren im Data Migrator können sich in der Art ihrer Parametrierung unterscheiden. So ist z. B. einer ActiveX-Transformation ein Skript zuzuordnen, welches bei Aufruf der Methode *Execute()* auszuführen ist. Um den Aufwand für die Einbindung eines neuen Verfahrens zu minimieren, werden Transformationsverfahren in sog. *Dekorierer* (siehe [GHJV01]) eingebettet, die die Zuordnung von Verfahren zu einem *DataPumpTask* und das Setzen von Parametern für den jeweiligen Verfahrenstyp vereinheitlichen. Durch die abstrakte Dekorierer-Klasse *DtTransformWrapper* werden die spezifizierten Verfahren mit dem *DataPumpTask* verknüpft sowie die Dialoge zur interaktiven Parametrierung der Verfahren gesteuert. Zur Einbindung eines neuen Transformationstyps sind folgende Erweiterungen des Data Migrators erforderlich:

1. Ableiten einer Klasse von *DtTransformWrapper*. Folgende Methoden sind dabei zu reimplementieren:

   - *PrePlausiCheck()* zur Prüfung der Anwendbarkeit des Verfahrens auf die gewählten Attributkombinationen. Bestimmte Transformationstypen (siehe unten) erfordern beispielsweise, dass es genau ein Quell- und genau ein Zielattribut gibt.
   - *SetParameters()* zur Parametrierung des Verfahrens mit Hilfe zugeschnittener Dialoge, z. B. ein Editorfenster für ActiveX-Transformationen.
   - *PostPlausiCheck()* zur abschließenden Plausibilitätsprüfung des Transformationsverfahrens, z. B. Syntaxprüfung für ActiveX-Skripte, die mittels *SetParameters()* spezifiziert wurden.

2. Bekanntmachung des neuen Wrappers gegenüber den *DtTaskSets* durch Erweiterung der Methode *DtTaskSet::CreateWrapper()*.

Die Methoden *DtTransformWrapper::CreateTransformation()* und *DtTransformWrapper::EditTransformation()* legen die allgemeine Vorgehensweise zum Erzeugen bzw. Bearbeiten von Transformationen fest. Dazu gehört insbesondere die Aufrufsequenz der Methoden *PrePlausiCheck()*, *SetParameters()* und *PostPlausiCheck()*.

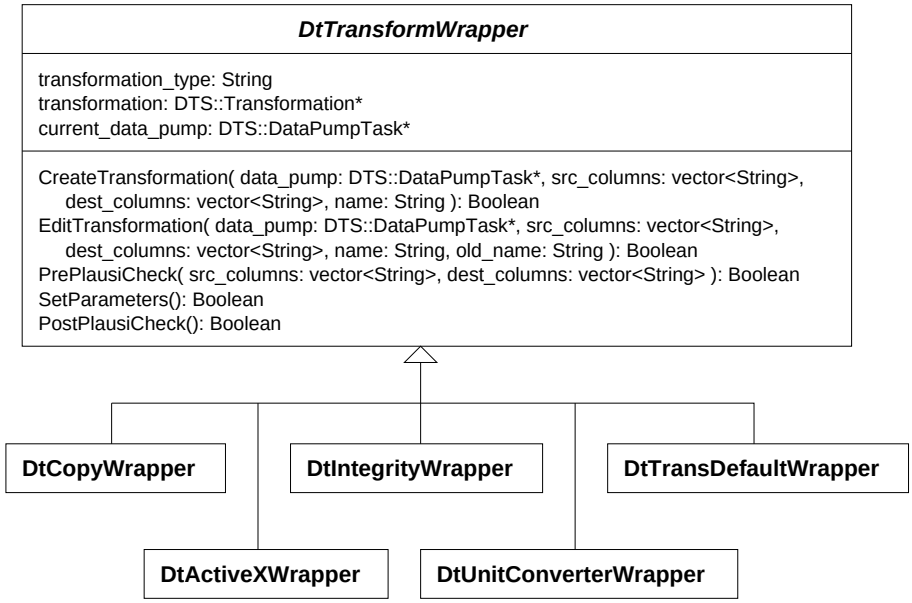


Abbildung 9.14: UML-Klassendiagramm zu Wrapper-Klassen im Data Migrator

Zur Zeit unterstützt der Data Migrator folgende Transformationstypen in Form von Spezialisierungen der Klasse *DtTransformWrapper* (siehe auch Abb. 9.14):

- *DtCopyWrapper*
  In diese Kategorie fallen alle Transformationen, in denen Attributwerte aus einer QDP-Relation unverändert in die spezifizierte Zielrelation übertragen werden (entspricht der identischen Abbildung).

- *DtActiveXWrapper*
  In diese Kategorie fallen alle Transformationen, die als ActiveX-Skripte realisiert sind. Die Transformationslogik wird hier in einer mit *Microsoft Visual Basic* verwandten, interpretierten Programmiersprache von Hand ausprogrammiert.

- *DtIntegrityWrapper*
  In diese Kategorie fallen alle Transformationen, die Zugriff auf Funktionalität des ETL-Werkzeugs *Vality Integrity* [Val02] (siehe Abschnitt 10.2.4) erfordern. Dieses Werkzeug, das eine Sammlung von Transformationsfunktionen zur Verfügung stellt (insbesondere zur Vereinheitlichung von Adressangaben), wurde in dieser Arbeit exemplarisch für andere externe Werkzeuge angebunden.

- *DtUnitConverterWrapper*
  In diese Kategorie fallen alle Transformationen, die auf die frei verfügbare Funktionsbibliothek *UnitConversion* (K. Stuempfle, Edgewood, MD, USA) zugreifen. Diese enthält Funktionen zur Umrechnung von Maßeinheiten.

Neben den genannten Wrapper-Klassen existiert noch die Klasse *DtTransDefaultWrapper*, die durch Nutzung der COM-Schnittstelle *IDispatch* die Spezifizierung beliebiger Transformationen erlaubt. Diese Flexibilität wird allerdings durch eine unkomfortable (weil COM-nahe) Benutzungsschnittstelle erkauft.

9.5 Measurement Engines

Die in diesem Abschnitt zu entwerfende Softwarekomponente soll den Nutzer in denjenigen Bereichen des DQM unterstützen, die mit der Messung von Datenqualität zu tun haben. Dies betrifft die Phasen 1 (Prüfung von Schlüssel- und referentieller Integrität), 3 (Statistische Prozesskontrolle), 4 (Domänenspezifische Konsistenzprüfung/Nachbearbeitung) und 7 (Qualitätsmessung und -analyse) des CLIQ-Vorgehensmodells (siehe Kapitel 7).

9.5.1 Operative Anforderungen

In Bezug auf SPC soll es möglich sein, statistische Kennzahlen zu einzelnen Attributen einer Relation des Transformationsbereichs zu berechnen und diese (bzw. deren Historie) in X- sowie Pareto-Charts graphisch aufzubereiten (vgl. Abschnitt 7.5). Bei Unter- oder Überschreiten spezifizierter Schwellenwerte soll das System den Nutzer automatisch benachrichtigen.

Bezüglich Phase 4 soll das System Tupel einer Relation im Transformationsbereich automatisiert – unter Einbeziehung angemessen repräsentierten domänenspezifischen Wissens – auf Inkonsistenzen untersuchen können. Gefundene Inkonsistenzen sollen so dokumentiert werden, dass der Nutzer sie im Rahmen der Nachbearbeitung leicht lokalisieren kann. Die Nachbearbeitung selbst soll ebenfalls softwaretechnisch unterstützt werden, indem die inkonsistenten Tupel auf einer graphischen Benutzungsoberfläche editierbar gemacht werden. Zur Unterstützung von Phase 1 des Vorgehensmodells soll eine analoge Lösung angeboten werden, d. h. eine automatisierte Suche nach Integritätsverletzungen (in den einzelnen Quelldatenpuffern) und deren Protokollierung, gefolgt von einer interaktiven Nachbearbeitung der betroffenen Tupel.

Im Hinblick auf Phase 7 muss das System automatisiert den Erfüllungsgrad operationalisierter Datenqualitätsanforderungen bestimmen können. Diese können in Qualitäts- und Indikatorplänen gemäß Abschnitt 7.9 spezifiziert sein. Die darin bezeichneten Datenobjekte im Konsolidierungs- bzw. Zielbereich sind mit Hilfe der in Kapitel 6 definierten Metriken und Messverfahren im Hinblick auf unterschiedliche Datenqualitätsmerkmale zu messen (im Falle von Qualitätsplänen) bzw. an Hand von domänenspezifischen Indikatoren zu bewerten (im Falle von Indikatorplänen). Das System hat die Messergebnisse mit den in den Plänen angegebenen Sollwerten zu vergleichen und den Nutzer insbesondere bei Nicht-Erfüllung von Anforderungen zu benachrichtigen.

9.5.2 Architektur und Datenstrukturen

Ausgehend von den im vorigen Abschnitt spezifizierten Anforderungen wurde im Rahmen dieser Arbeit eine Softwarekomponente entwickelt, welche – den vier genannten Funktionsbereichen entsprechend – aus vier Paketen besteht: Während das Paket *MeasurementEngine* Funktionalität zum Ausführen von Qualitätsplänen (d. h. zur Messung der Ist-Datenqualität und zum Vergleich mit der spezifizierten Soll-Datenqualität, jeweils bzgl. unterschiedlicher Qualitätsmerkmale) bereitstellt, realisiert das Paket *SPCEngine* eine Statistische Prozesskontrolle gemäß Abschnitt 7.4. Domänenspezifische Inkonsistenzen werden in dem Paket *DomainSpecificMeasurementEngine*, schlüsselbezogene Integritätsverletzungen in dem Paket *KeyIntegrityMeasurementEngine* behandelt.

Die Spezifizierung von Qualitäts- und SPC-Plänen erfolgt nicht in der Messkomponente selbst, sondern wird vom DQM Controller (siehe Abschnitt 9.3) übernommen. Die vier genannten Pakete werden in den folgenden Unterabschnitten näher beschrieben.

Paket *MeasurementEngine*

Das Paket *MeasurementEngine* enthält eine gleichnamige Klasse, die, wie Abb. 9.15 zeigt, von den Klassen *DBInterface* und *MrRepository* (siehe Abschnitt 9.10) abgeleitet ist und sich deren Funktionalität zum Zugriff auf die zu messenden Datenobjekte bzw. zum Auslesen und Ablegen von Metainformationen zunutze macht. Ein *MeasurementEngine*-Objekt verwaltet eine Menge von Messergebnissen (Attribut *qmeasmts*) zu einem Qualitätsplan (Attribut *qplan*). Messergebnisse werden durch die Methode *ExecutePlan()* berechnet. Je nach Datenqualitätsmerkmal, welches laut Plan zu betrachten ist, ruft *ExecutePlan()* ihrerseits eine (private) Methode auf, die das mit dem Merkmal assoziierte Messverfahren (siehe Abschnitt 6.1) implementiert.

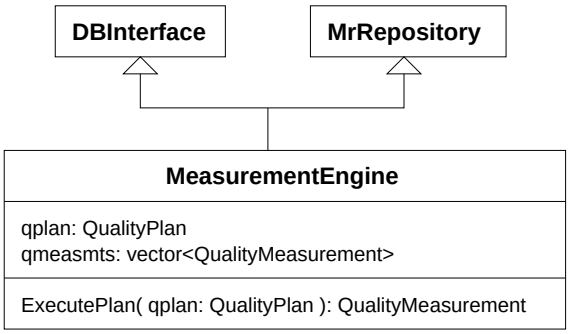


Abbildung 9.15: UML-Klassendiagramm zur *MeasurementEngine*

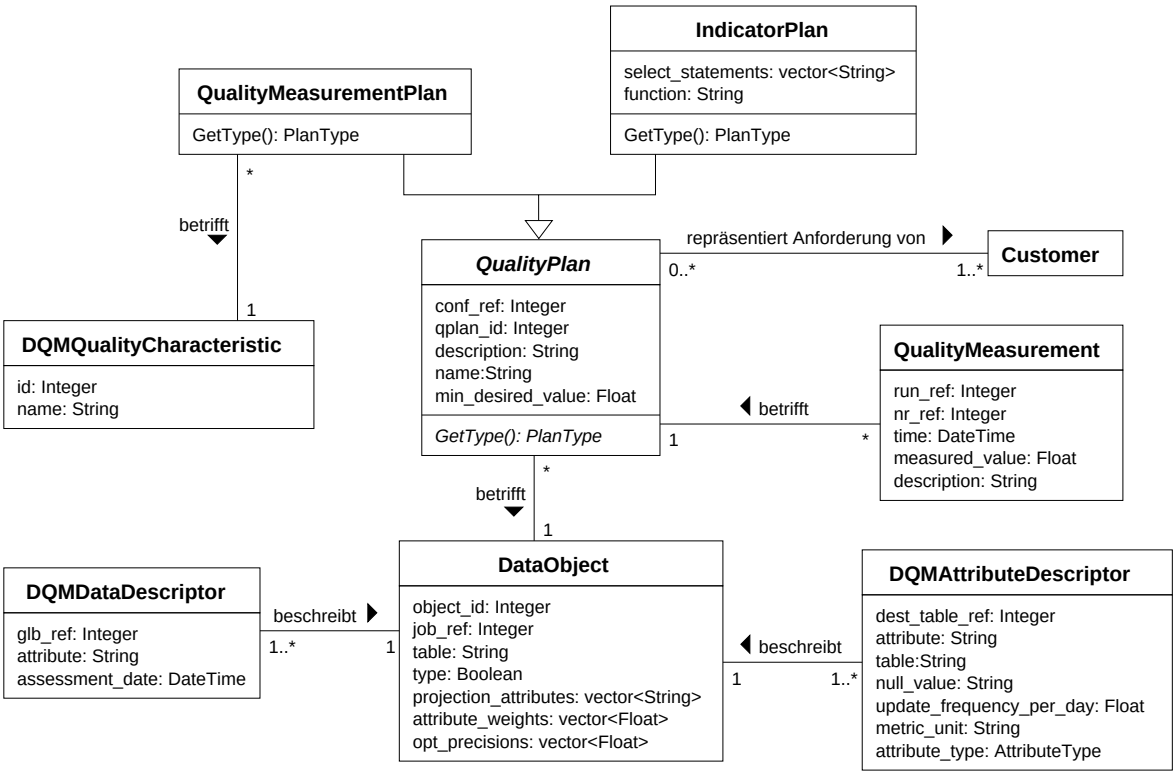


Abbildung 9.16: UML-Klassendiagramm zu Datenstrukturen der *MeasurementEngine*

Abbildung 9.16 gibt einen Überblick über die von der Klasse *MeasurementEngine* verwendeten objektorientierten Datenstrukturen. Qualitätsanforderungen von Kunden (Klasse *Customer* im

JobModel-Paket) an ein gegebenes Datenobjekt (Klasse *DataObject*) werden als Objekte einer von der abstrakten Klasse *QualityPlan* abgeleiteten Klasse beschrieben. In einem *QualityPlan* wird zu einer Anforderung ein Sollwert verwaltet (Attribut *min_desired_value*), der in einer Messung mindestens erreicht werden muss, damit die Anforderung als erfüllt gilt. Gemäß den Vorgaben des CLIQ-Vorgehensmodells werden zwei Typen von Plänen unterschieden (vgl. Abschnitt 7.9), die als Subklassen von *QualityPlan* modelliert werden: Die Klasse *QualityMeasurementPlan* entspricht Qualitätsplänen, die sich auf je ein Datenqualitätsmerkmal beziehen, während die Klasse *IndicatorPlan* der Spezifizierung von domänenspezifischen Datenqualitätsindikatoren an Hand arithmetischer Verknüpfungen (Attribut *function*) von SQL-Kommandos (Attribut *select_statements*) dient (vgl. Abschnitt 7.9.2). Der Typ eines Qualitätsplans (Aufzählungstyp *PlanType*) kann zur Laufzeit mittels der Methode *GetType()* abgefragt werden.

Wird ein Qualitätsplan durch Aufruf von *MeasurementEngine::ExecutePlan()* aktiviert und damit eine Qualitätsmessung auf dem spezifizierten Datenobjekt durchgeführt, so resultiert dies in einem Messergebnis (Klasse *QualityMeasurement*), das u. A. den Zeitpunkt der Messung (Attribut *time*) und den Messwert (Attribut *measured_value*) enthält. Je nach betrachtetem Qualitätsmerkmal greift das Messverfahren auf Metadaten zurück, die entweder attributspezifisch (Klasse *DQMAttributeDescriptor*, z. B. Update-Häufigkeit und Maßeinheit) oder datenspezifisch (Klasse *DQMDataDescriptor*, insb. Befunddatum) sein können. Für Beispiele siehe Abschnitt 11.2. Attributgewicht (vgl. g_j in Abschnitt 6.1) und „optimale“ Stelligkeit gemäß Abschnitt 6.1.5 gelten nicht als Attributdeskriptoren, sondern werden als anforderungsabhängige Eigenschaften einem Datenobjekt zugeordnet (Attribute *attribute_weights* und *opt_precisions*).

Paket *SPCEngine*

Zentrales Element des Pakets *SPCEngine* ist die gleichnamige, von *MeasurementEngine* (siehe voriger Abschnitt) abgeleitete Klasse. Diese stellt Methoden zur Berechnung und Auswertung statistischer Kennzahlen, differenziert nach Datenquellen, bereit. Die Methode *ComputeStatistics()* nimmt – ausgehend von einer als Objekt der Klasse *SPCPlan* (siehe unten) repräsentierten SPC-Konfiguration – die Berechnung auf dem im *SPCPlan* spezifizierten Datenobjekt vor (siehe Abb. 9.17). Die von *ComputeStatistics()* zurückgelieferte Datenstruktur *SPCResult* enthält dann eine Menge von Werten, die direkt in X- und Pareto-Charts abgelegt werden können.

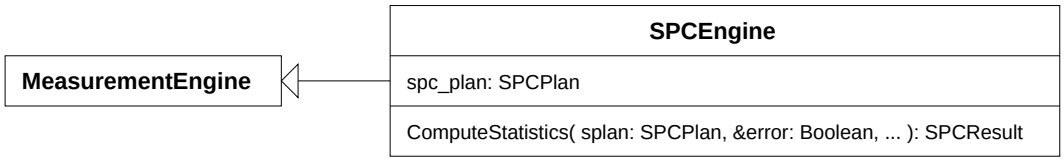


Abbildung 9.17: UML-Klassendiagramm zur *SPCEngine*

Die von *SPCEngine* verwendeten Datenstrukturen zeigt Abb. 9.18. Jeder *SPCPlan* enthält Angaben zu erwarteten statistischen Kennzahlen (Attribute *mean* und *standard_deviation*). Diese können manuell vorgegeben oder zur Laufzeit aus den Daten des Zielbereichs berechnet werden. Weiterhin sind pro Kennzahl untere und obere Kontrollgrenzen zu spezifizieren, bei deren Unter- bzw. Überschreitung ein interaktiver Eingriff erfolgen soll. Da ausschließlich attributbezogenes SPC unterstützt wird, muss über das Attribut *projection_attributes* der Klasse *DataObject* (siehe auch Abb. 9.16) eine Projektion auf ein einziges Attribut der in *table* festgelegten Relation (des Transformationsbereichs) erfolgen.

Die Ergebnisse einer SPC-Messung werden in Objekten der Klassen *SPCXChartEntry* und *SPCParetoChartEntry* gespeichert. Diese realisieren die in Abschnitt 7.5 beschriebenen Kontrollkarten. Die

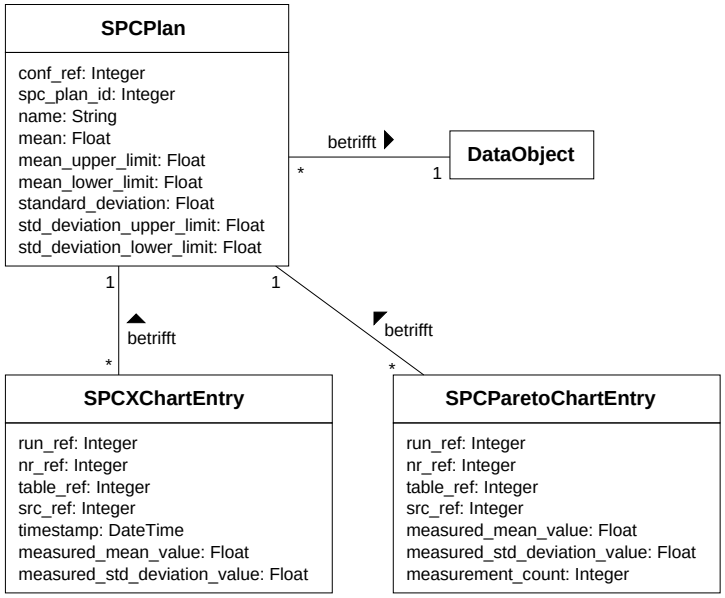


Abbildung 9.18: UML-Klassendiagramm zu Datenstrukturen der *SPCEngine*

graphische Aufbereitung der Kontrollkarten wird vom DQM Controller übernommen. Dieser erzeugt über die COM-Schnittstelle von Microsoft Excel entsprechende Diagrammobjekte, die schließlich in XML-Dokumenten referenziert und dann visualisiert werden (siehe Abschnitt 9.3.2).

Paket *DomainSpecificMeasurementEngine*

Domänenspezifische Konsistenzprüfungen werden in dedizierten Klassen (eine pro Domäne) realisiert, die von der (domänenübergreifenden) Klasse *MeasurementEngine* abgeleitet sind, in Abb. 9.19 angedeutet durch die Klasse *DomainSpecificMeasurementEngine<x>* für eine Domäne *<x>* (Beispiele: *<x> = „CarManufacturingData“*, *<x> = „TumorData“*, *<x> = „AccountingData“*). Jede dieser Subklassen kapselt domänenspezifisches Wissen, das unterschiedlich repräsentiert sein kann (vgl. Abschnitt 7.6). In jedem Fall muss es jedoch eine Methode *ExecuteConsTuple(object: DataObject, log: vector<String>): vector<Float>* geben, die die gleichnamige (abstrakte) Methode der Klasse *MeasurementEngine* implementiert (siehe Abb. 9.19). Diese Methode liefert für ein übergebenes Datenobjekt der Granularität Tupel zu jeder erkannten Inkonsistenz (möglicherweise mehrere pro Tupel) den Schweregrad sowie im Referenzparameter *log* eine Beschreibung der einzelnen Inkonsistenzen zurück.

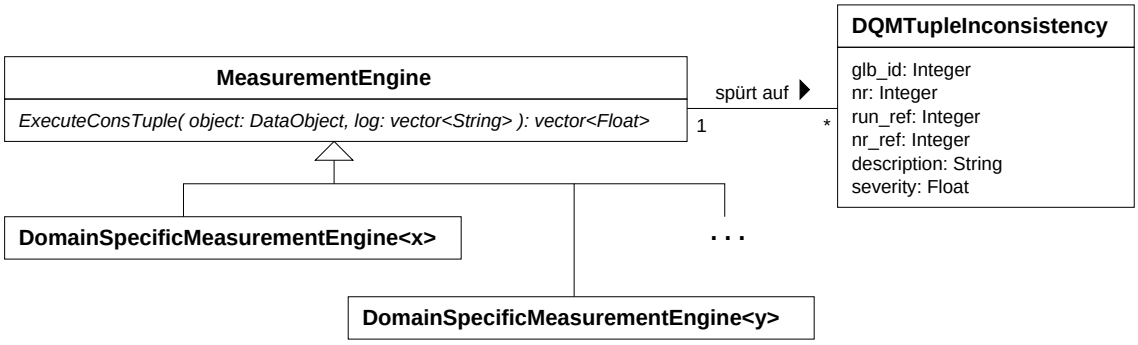


Abbildung 9.19: UML-Klassendiagramm zur *DomainSpecificMeasurementEngine*

Die Resultate von Konsistenzprüfungen werden in Objekten der Klasse *DQMTupleInconsistency* verwaltet. Diese Objekte stellen die Grundlage für die interaktive Nachbearbeitung von inkonsistenten Tupeln dar. Diese wird durch das Paket *InconsReview* unterstützt (siehe Abb. 9.20). *InconsReview* nutzt das im Rahmen dieser Arbeit entwickelte Paket *PerspectiveWall* zur Visualisierung von Tupeln. Datenbankzugriffe werden durch das Paket *InconsReviewDB* gekapselt.

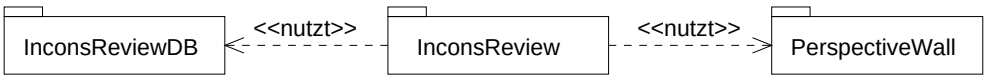


Abbildung 9.20: UML-Paketdiagramm zur Nachbearbeitungskomponente

An dieser Stelle wird – beispielhaft für andere Repräsentationsformen von Domänenwissen – die Anbindung von (Geschäfts-)Regeln mit Hilfe des regelverarbeitenden Systems ILOG Rules [Ilo02] (siehe auch Abschnitt 10.2.6) demonstriert. Die ILOG Rules-Funktionalität ist durch *DomainSpecificMeasurementEngine<x>* geeignet zu kapseln.

ILOG Rules kann Konsistenzprüfungen ausschließlich auf objektorientierten Datenstrukturen durchführen. Daher muss für jede Relation, deren Extension geprüft werden soll, einmalig eine korrespondierende Klasse entwickelt werden. Alle Relationenattribute, auf die sich Regeln beziehen können, sind auf die Klassenstruktur abzubilden (unter entsprechender Anpassung der Datentypen). Domänenspezifische Prüfverfahren, die aus ILOG Rules heraus angestoßen werden sollen, können als Methoden der Klasse implementiert werden.

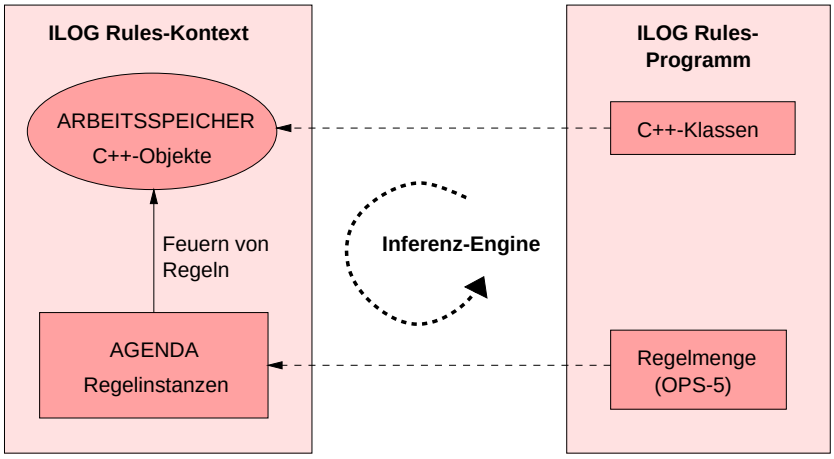


Abbildung 9.21: Ausführung eines ILOG Rules-Programms [Ilo97]

Zusätzlich sollte die Klasse eine Methode *void AddInfo(severity: Float, log: String)* beinhalten, die jeweils bei Konsistenzverletzungen – also beim „Feuern“ einer Regel – im Aktionsteil der Regel aufgerufen wird und den Schweregrad der Inkonsistenz quantifiziert (über den Parameter *severity*) sowie einen beschreibenden Text (über den Parameter *log*) spezifiziert. Dazu ist der Bedingungsteil einer Regel so zu formulieren, dass die Regel im Falle einer Inkonsistenz „feuert“. Die von *AddInfo()* übergebenen Werte werden in (privaten) Attributen zwischengespeichert und später als Rückgabewerte der Methode *ExecuteConsTuple()* (siehe oben) verwendet.

Regeln werden bei ILOG Rules in OPS-5–Notation [BFKM85] formuliert und zu Regelmengen zusammengefasst. Klassenstruktur und Regelmenge bilden zusammen ein ILOG Rules-Programm (siehe Abb. 9.21). Zur Laufzeit wird zu dem Programm ein sog. *ILOG Rules-Kontext* aufgebaut. Dazu werden die zu prüfenden Daten aus der Datenbank geladen und tupelweise in Objekten der korrespondierenden Klasse abgelegt. Jedes Objekt, das auf Konsistenz geprüft werden soll, wird nun

einzeln in den sog. *ILOG Rules-Arbeitsspeicher* (Bestandteil eines Kontexts) eingefügt, in dem die eigentliche Prüfung stattfindet. Zu denjenigen Regeln, deren Bedingungsteil durch das aktuell im Arbeitsspeicher befindliche Objekt erfüllt ist, werden Regelinstanzen erzeugt und in eine sog. *Agenda* geschrieben. Bei mehr als einer Regelinstanz in der Agenda wird nach bestimmten Kriterien (Regelpriorität, Verweildauer in der Agenda etc.) eine ausgewählt. Für diese Regelinstanz wird der Aktionsteil ausgeführt. Anschließend wird die Regelinstanz aus der Agenda entfernt, der Inhalt der Agenda neu berechnet und erneut eine Regelinstanz zum „Feuern“ ausgewählt (ein Tupel kann mehr als eine Inkonsistenz aufweisen). Dieser Vorgang wiederholt sich so lange, bis die Agenda keine Regelinstanzen mehr enthält (wird durch den Inferenzmechanismus von ILOG Rules geprüft). An Hand der mittels *AddInfo()* protokollierten Schweregrade der Regelverletzungen kann über die in Kapitel 6 beschriebenen Metriken die Konsistenz des betrachteten Datenobjekts bewertet werden. Die während der Prüfung aufgezeichneten beschreibenden Texte können für Rückfragen an die Datenquelle bzw. zur Nachbearbeitung von Tupeln herangezogen werden.

Paket *KeyIntegrityMeasurementEngine*

Das Paket *KeyIntegrityMeasurementEngine* besteht ausschließlich aus einer gleichnamigen, von *MeasurementEngine* abgeleiteten Klasse. Diese nutzt die von der Basisklasse bereitgestellte Funktionalität zur Suche nach schlüsselbezogenen Integritätsverletzungen (gemäß Abschnitten 6.1.8 und 6.1.9). Analog zum Paket *DomainSpecificMeasurementEngine* werden gefundene Verletzungen tupelbezogen protokolliert. Zur interaktiven Nachbearbeitung der betroffenen Tupel greift die *KeyIntegrityMeasurementEngine* auf die (Sub-)Pakete *InconsReview* und *InconsReviewDB* des Pakets *DomainSpecificMeasurementEngine* zurück.

9.6 Data Linker

Der Data Linker hat die Aufgabe, Tupel, die dieselbe Entität der Diskurswelt beschreiben, einander automatisiert zuzuordnen (Record Linkage, Phase 5 des CLIQ-Vorgehensmodells) und anschließend miteinander zu verschmelzen (Merging, Phase 6), um unbeabsichtigte Redundanz im Zielbereich zu vermeiden.

9.6.1 Operative Anforderungen

Beim Record Linkage ist darauf zu achten, dass Synonym- und Homonymfehler minimiert werden: Einerseits muss der Data Linker fehlertolerant genug sein, um die Zusammengehörigkeit zweier Tupel trotz Abweichungen in den Attributwerten (bedingt z. B. durch Tippfehler oder über die Zeit variierende Merkmalsausprägungen) zu erkennen, andererseits muss die Trennschärfe so hoch sein, dass nicht zusammengehörige Tupel auch bei ähnlichen Attributausprägungen nicht fälschlicherweise zusammengeführt werden.

Tupel, die Updates von bereits im Zielbereich gespeicherten Tupeln darstellen (inkrementelle Updates), sollen an Hand ihrer Primärschlüssel mit den jeweiligen Tupeln des Zielbereichs zusammengeführt werden, da bei Verwendung eines solchen schlüsselbasierten Linkage-Verfahrens Zuordnungsfehler ausgeschlossen sind und außerdem die in einem anschließenden, berechnungstechnisch ungleich aufwändigeren ähnlichkeitsbasierten Linkage zu betrachtenden Datenvolumina reduziert werden können (vgl. Abschnitt 7.7). Tupel, die aus einem inkrementellen Update hervorgehen, sind im Konsolidierungsbereich abzulegen und bei einem nachfolgend durchzuführenden ähnlichkeitsbasierten Linkage zu berücksichtigen.

In einem zweiten Arbeitsschritt sind einander zugeordnete Tupel miteinander zu verschmelzen (Merging, Phase 6 des CLIQ-Vorgehensmodells). Dieser Vorgang erfordert z. T. interaktive Entscheidungen darüber, welche Attributwerte der Ausgangstupel in das Zieltupel übernommen werden sollen. Die Zuordnungen sind daher graphisch so aufzubereiten, dass zu übernehmende Werte auf einfache Weise (z. B. durch Anklicken mit der Maus) ausgewählt werden können. Außerdem soll das System den Nutzer zumindest insofern unterstützen, dass triviale Entscheidungen (z. B. falls alle Werte zu einem Attribut übereinstimmen oder falls alle Werte bis auf einen *NULL* sind) automatisiert vorgenommen werden. Mittels einer Implementierung von Auswahlkriterien wie Quellenpriorität, Häufigkeit oder Zeitnähe von Attributwerten kann der Automatisierungsgrad weiter gesteigert werden. Allerdings ist über deren Anwendbarkeit im (domänenspezifischen) Einzelfall zu entscheiden.

Tupel, die aus einer Verschmelzung von Tupeln aus Transformationsbereich (neu angelieferte Tupel), Konsolidierungsbereich (beim schlüsselbasierten Linkage zusammengeführte Tupel) und Zielbereich (bereits konsolidierte Tupel) hervorgegangen sind, sind gemäß CLIQ-Vorgehensmodell im Konsolidierungsbereich abzulegen.

9.6.2 Architektur und Datenstrukturen

Der Data Linker setzt sich – den zwei im vorigen Abschnitt beschriebenen Arbeitsschritten entsprechend – aus zwei Paketen zusammen, nämlich *RecLink* (Arbeitsschritt 1) und *RecLinkReview* (Arbeitsschritt 2), wie in Abb. 9.22 dargestellt. *RecLink* wiederum besteht aus einem Paket zum schlüsselbasierten Record Linkage (*KeyRecLink*) und einem Paket zum probabilistischen Record Linkage (*ProbRecLink*).

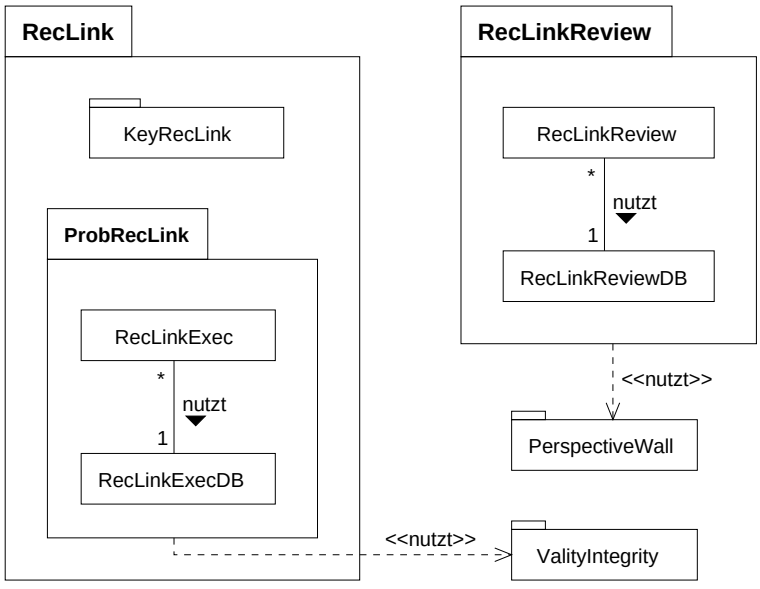


Abbildung 9.22: UML-Paketdiagramm zum Data Linker

Auf eine detaillierte Betrachtung des *KeyRecLink*-Pakets wird verzichtet. Die folgenden Unterabschnitte gehen näher auf die (im Vergleich zu *KeyRecLink* ungleich komplexeren) Pakete *ProbRecLink* und *RecLinkReview* ein.

Paket *ProbRecLink*

Im *ProbRecLink*-Paket wird für die Zuordnung von Tupeln ein probabilistisches Record Linkage-Verfahren (wie in [Jar89] beschrieben) eingesetzt. Diese Entwurfsentscheidung wurde aufgrund von

mehrjährigen Erfahrungen getroffen, die im Projekt CARLOS [AFH+97] gewonnen wurden. Analog zur Vorgehensweise in CARLOS wurde der verwendete Record Linkage-Algorithmus einem kommerziellen ETL-Werkzeug (Vality Integrity [Val02], siehe Abschnitt 10.2.4) entnommen. Aufbauend auf diesem Algorithmus wurden Klassen entwickelt, die die Integrity-Funktionalität geeignet kapseln (Klasse *RecLinkExec*), um eine Austauschbarkeit des Algorithmus zu gewährleisten, und in die vorliegende Systemumgebung (Integrationsdatenbank und Metadaten-Repository) einbetten (Klasse *RecLinkExecDB*).

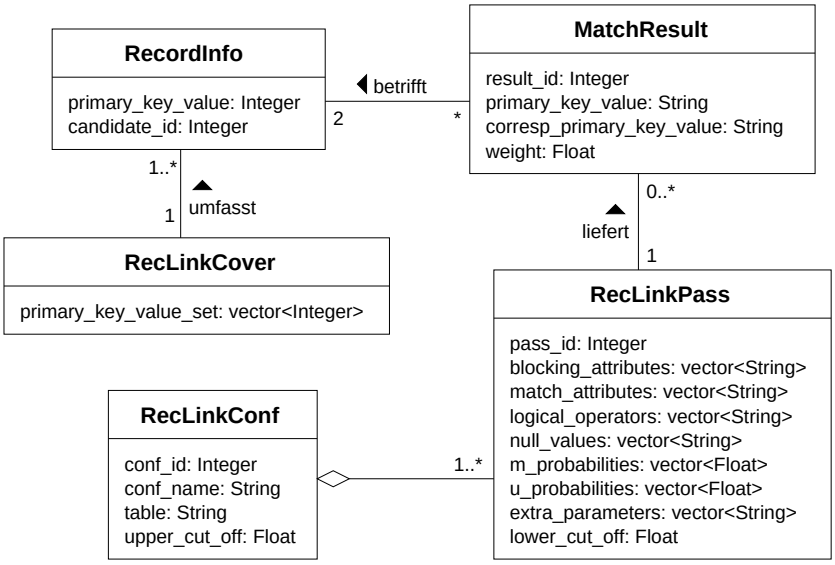


Abbildung 9.23: UML-Klassendiagramm zu Datenstrukturen des Pakets *ProbRecLink*

Abbildung 9.23 zeigt die objektorientierten Datenstrukturen des Pakets *ProbRecLink*. Eine zentrale Bedeutung fällt der Klasse *RecLinkConf* zu. Jedes Objekt dieser Klasse beschreibt eine über die *JobPlanEngine* (siehe Abschnitt 9.3.2) spezifizierte Konfiguration zum probabilistischen Record Linkage. Diese legt eine Relation (streng genommen eine Sicht über gleichnamige Relationen aus Transformations-, Konsolidierungs- und Zielbereich), in der nach Duplikaten gesucht werden soll, sowie einen oberen Schwellenwert fest. Weiterhin wird in einer Record Linkage-Konfiguration eine (nichtleere) Menge von „Umläufen“ (Klasse *RecLinkPass*) spezifiziert. Jedes *RecLinkPass*-Objekt beschreibt Block- und Matchvariablen für den jeweiligen Umlauf. Für jede Matchvariable werden weiterhin der zu verwendende Vergleichsoperator (inkl. ggf. erforderlichen Zusatzparametern), Schätzwerte für m- und u-Wahrscheinlichkeiten (Erklärung siehe Abschnitt 7.7.2), Werte, die einer fehlenden Angabe entsprechen (z. B. „X“, -1 etc.) sowie ein unterer Schwellenwert spezifiziert.

Auf der Basis einer solchen Konfiguration kann nun ein probabilistischer Record Linkage-Prozess angestoßen werden. Zur Laufzeit wird die Klasse *RecordInfo* zur Verwaltung temporärer Prozessinformationen genutzt. Zu jedem am Record Linkage-Prozess beteiligten Tupel der in der Konfiguration spezifizierten Relation (bzw. Sicht) wird dazu genau ein korrespondierendes Objekt der Klasse *RecordInfo* angelegt. In einem Umlauf gefundene Zuordnungen zwischen Tupeln (potentielle Duplikate, deren Matchgewicht oberhalb des unteren Schwellenwerts liegt) werden als Objekte der Klasse *MatchResult* abgelegt. Konkret werden die Primärschlüsselwerte der beiden beteiligten Tupel sowie das Matchgewicht (siehe oben) gespeichert. Ist die Suche nach Tupelpaaren abgeschlossen, werden die *MatchResult*-Objekte auf transitive Beziehungen untersucht. Die so berechneten transitiven Abschlüsse (Matchgruppen) zu jedem Match werden als Mengen von Primärschlüsselwerten in Objekten der Klasse *RecLinkCover* abgelegt und anschließend mit Tupelzuordnungen, die im Rahmen des schlüsselbasierten Linkage bestimmt wurden, verschmolzen.

Paket *RecLinkReview*

Die in *ProbRecLink* berechneten, als Objekte der Klasse *RecLinkCover* repräsentierten Tupelzuordnungen können von *RecLinkReview* direkt weiterverarbeitet werden. Diese vollständig neu entwickelte Komponente realisiert in der gleichnamigen Klasse eine graphische Aufbereitung von einander zugeordneten Tupeln. Dabei wird auf das Paket *PerspectiveWall* zurückgegriffen (vgl. Abb. 9.22), das eine flexible Darstellung von beliebigen Tupeln ermöglicht und auch bei der Nachbearbeitung von inkonsistenten Tupeln Verwendung findet (siehe Abschnitt 9.5.2). Datenbank- und Repository-Zugriffe werden durch die Klasse *RecLinkReviewDB* gekapselt.

Bei der Verarbeitung der Tupelzuordnungen wird wie folgt vorgegangen: Jede Matchgruppe wird zunächst automatisiert daraufhin überprüft, ob für alle Paare der enthaltenen Tupel Matchgewichte berechnet wurden (abzulesen aus den zugehörigen *MatchResult*-Objekten), die oberhalb des in der Record Linkage-Konfiguration (Klasse *RecLinkConf*) spezifizierten oberen Schwellenwerts liegen. Ist dies der Fall, so werden die Zuordnungen konsolidiert. Dabei werden die beteiligten Tupel so zu einem Tupel miteinander verschmolzen, dass das Zieltupel die „besten“ Informationen enthält. Nicht automatisiert aufzulösende Konflikte zwischen beteiligten Attributwerten erfordern manuelle Eingriffe. Gleiches gilt für Matchgruppen, in denen Matches enthalten sind, deren Gewichte unterhalb des oberen Schwellenwertes liegen. Hier muss entschieden werden, ob die Tupel tatsächlich dieselbe Entität beschreiben oder nicht. Bei einelementigen Matchgruppen (keine potentiellen Matchpartner vorhanden) stimmen Ziel- und Ausgangstupel überein.

Die aus Verschmelzungen hervorgegangenen Tupel werden schließlich in den Konsolidierungsbereich geschrieben, die Matchgruppen selbst sowie die darin enthaltenen Tupel aus dem Transformationsbereich werden gelöscht.

9.7 Data Auditor (MEDAS)

Der Data Auditor soll Phase 8 des CLIQ-Vorgehensmodells (Lenkung von Datenprodukten/Qualitätsverbesserung) unterstützen. Die Komponente wurde unter dem Namen MEDAS (Metadata-Based Data Auditing System) von Thomas Wilkens im Rahmen seiner Diplomarbeit [Wil00] entwickelt.

9.7.1 Operative Anforderungen

Aufgabe des Data Auditors ist es, weitgehend automatisiert nach Inkonsistenzen in Datenbeständen zu suchen und die Suchergebnisse so aufzubereiten, dass sie von einem Systemnutzer analysiert werden können. Sich dabei bestätigende Inkonsistenzen sollen auf einer graphischen Oberfläche interaktiv beseitigt werden können (im Sinne einer Datenbereinigung), wobei der Data Auditor Prognosewerte für bisher inkonsistente oder fehlende Angaben vorschlagen soll.

9.7.2 Architektur und Datenstrukturen

Die zentrale Entwurfsentscheidung bestand darin, die Suche nach Inkonsistenzen mit Hilfe von Data Mining-Algorithmen zu realisieren, um einerseits einen hohen Automatisierungsgrad zu erreichen und andererseits nicht auf umfangreiches Domänenwissen angewiesen zu sein.

Im Folgenden wird kurz auf die Architektur von MEDAS eingegangen (siehe Abb. 9.24). Für weitere Details sei auf [Wil00] verwiesen.

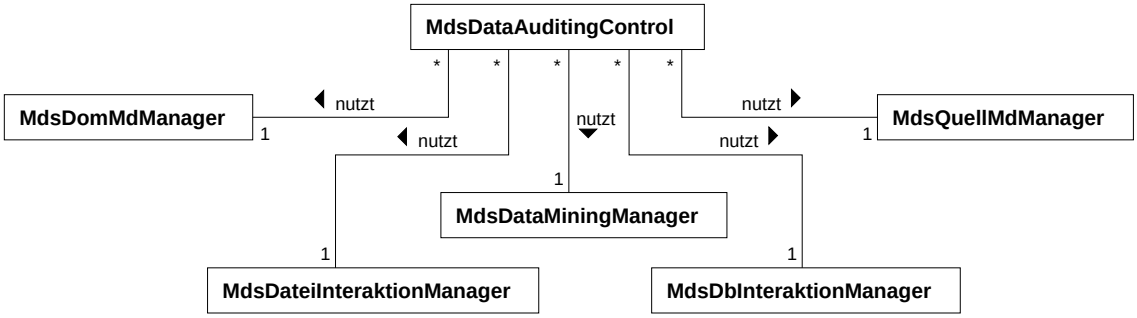


Abbildung 9.24: UML-Klassendiagramm zu MEDAS [Wil00]

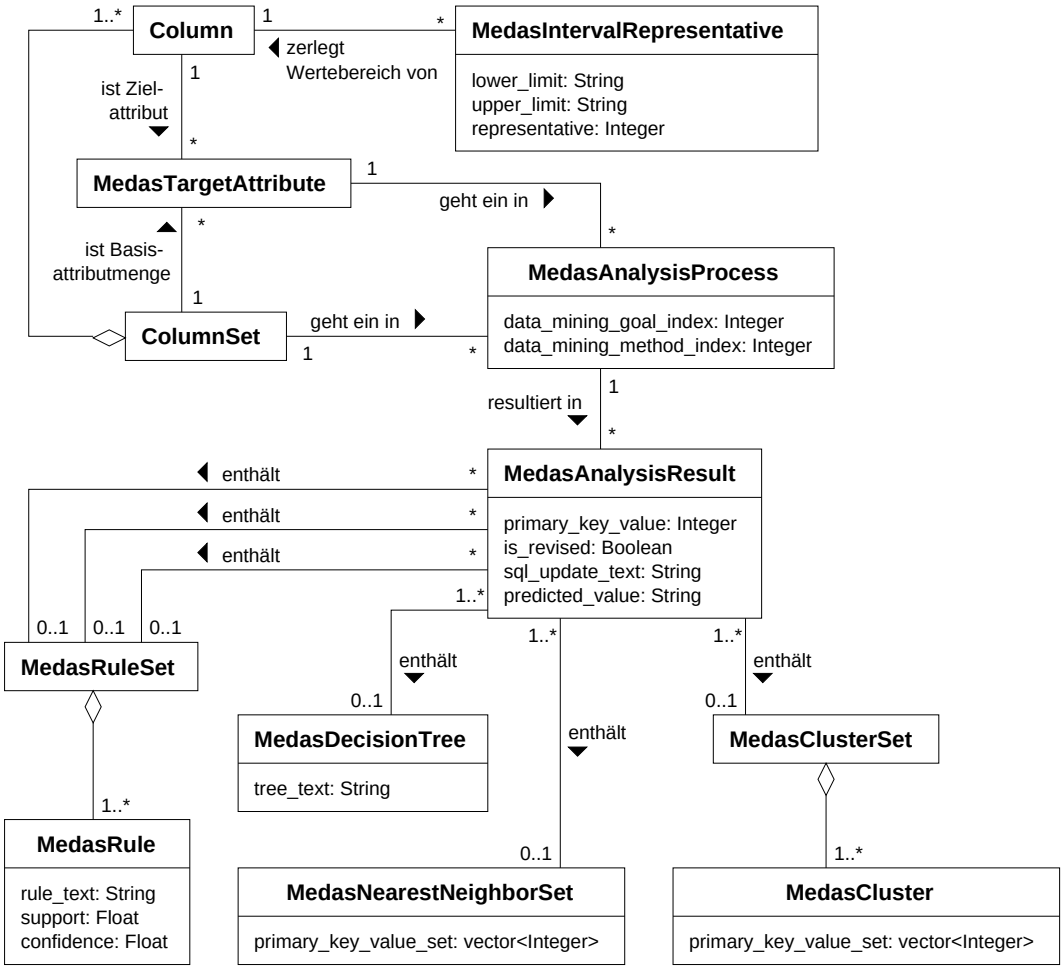


Abbildung 9.25: UML-Klassendiagramm zu Datenstrukturen in MEDAS [Wil00]

Zentrales Element ist die Klasse *MdsDataAuditingControl*. Diese ist für die Steuerung der Abläufe des Analyse- und Nachbearbeitungsprozesses zuständig und kommuniziert mit allen anderen beteiligten Klassen. Insbesondere nutzt *MdsDataAuditingControl* die Funktionalität der Klasse *MdsDataMiningManager*, welche den Zugriff auf eine angebundene Data Mining-Bibliothek (MLC++, siehe Abschnitt 10.2.5) kapselt. Die Klassen *MdsDateiInteraktionManager* und *MdsDbInteraktionManager* stellen Zugriffsfunktionen zum Dateisystem (einige Data Mining-Algorithmen arbeiten dateibasiert) bzw. zu einem DBMS (auf YADI [Fri96] basierend, siehe Abschnitt 10.2.1) zur Verfügung.

Die Klassen *MdsDomMdManager* und *MdsQuellMdManager* schließlich sorgen für den Zugriff auf domänenspezifische (hier: regelbasierte) bzw. schemabezogene Metadaten.

Abbildung 9.25 zeigt die objektorientierten Datenstrukturen, auf denen MEDAS arbeitet. Eine Instanz der Klasse *MedasAnalysisProcess* beschreibt dabei einen Data Mining-Prozess auf der Basis einer voreingestellten Konfiguration. In dieser ist insbesondere das Data Mining-Ziel (Klassifikation, Prognose etc.) und das zum Erreichen des Ziels zu verwendende Data Mining-Verfahren (Entscheidungsbäume, neuronale Netze etc.) festgelegt. Des weiteren muss spezifiziert sein, in welchem Attribut nach Inkonsistenzen gesucht werden soll (sog. *Zielattribut*, Klasse *MedasTargetAttribute*) und welche Attribute zur Bewertung des Zielattributs herangezogen werden sollen (sog. *Basisattribute*, Klasse *ColumnSet*). Für Attribute mit großen Wertebereichen empfiehlt es sich – falls eine lineare Ordnung existiert – Intervalle von Werten zu bilden und jedes Intervall durch einen Vertreterwert zu repräsentieren (Klasse *MedasIntervalRepresentative*), um die Laufzeiteffizienz und die Interpretierbarkeit von Analyseergebnissen zu erhöhen.

Die Durchführung eines Analyseprozesses resultiert in einer Menge von Analyseergebnissen (Klasse *MedasAnalysisResult*). Jedes Analyseergebnis bezieht sich auf einen (potentiell inkonsistenten oder unvollständigen) Datensatz, der durch seinen Primärschlüsselwert beschrieben ist. Je nach spezifiziertem Data Mining-Ziel (siehe oben) enthält das Ergebnis einen vorausgesagten Wert (Attribut *predicted_value*), den MEDAS zur Korrektur bzw. Vervollständigung des jeweiligen Datensatzes im Zielattribut vorschlägt. Diese kann mit Hilfe eines SQL-Update-Kommandos angestoßen werden, was im Analyseergebnis protokolliert wird (Attribut *is_revised*). Schließlich sind einem Analyseergebnis Datenstrukturen zugeordnet, die im Rahmen des Data Mining-Prozesses erstellt wurden und sich daher je nach verwendetem Data Mining-Verfahren unterscheiden (zu den Verfahren siehe z. B. [BHM98, BS97a]):

- *Regelinduktion*: Hier wird eine Menge von Assoziationsregeln berechnet (Klassen *MedasRuleSet* und *MedasRule*).
- *Entscheidungsbäume*: Hier wird ein Entscheidungsbaum berechnet (Klasse *MedasDecisionTree*).
- *k-nächste-Nachbarn*: Hier werden die „nächsten Nachbarn“ (gemäß eines gegebenen Abstandsmaßes) zu einem Datensatz berechnet und über ihre Primärschlüsselwerte gespeichert (Klasse *MedasNearestNeighborSet*).
- *Clustering*: Hier wird eine Menge von Datensatzclustern berechnet, wobei jedes Cluster über eine Menge von Primärschlüsselwerten definiert ist (Klassen *MedasClusterSet* und *MedasCluster*).

Für nähere Informationen zu den einzelnen Verfahren und deren Anwendung im Kontext der Datenbereinigung sei auf [Wil00] verwiesen.

9.8 Release/Retraction Handler

Der Release/Retraction Handler ist für die zueinander inversen Phasen 9 (Datenfreigabe) und 10 (Datenrücknahme) des CLIQ-Vorgehensmodells verantwortlich. Das Paket enthält die beiden Klassen *DataReleaseHandler* und *DataRetractionHandler* (siehe Abb. 9.26), die von der gemeinsamen abstrakten Basisklasse *DataMoveHandler* abgeleitet sind. Letztere stellt Hilfsattribute und -methoden (z. B. zum Lesen und Schreiben von Tupeln sowie zum Setzen von Metaattributen – in Abb. 9.26

nicht dargestellt) zur Verfügung, die von beiden Subklassen gemeinsam genutzt werden können. Diese Subklassen implementieren jeweils eine *Execute()*-Methode, die die freizugebenden bzw. zurückzunehmenden Tupel vom Konsolidierungs- in den Zielbereich (bzw. umgekehrt) verschiebt und die tupelbezogenen Metaattribute (siehe Abschnitt 9.11) entsprechend den Vorgaben des Vorgehensmodells (siehe Abschnitte 7.11 und 7.12) setzt.

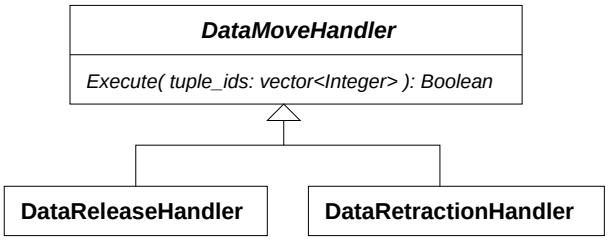


Abbildung 9.26: UML-Klassendiagramm zum Release/Retraction Handler

9.9 CLIQWorkbench Console

Die CLIQWorkbench Console realisiert die graphische Benutzungsoberfläche der CLIQWorkbench. Sie enthält Dialogklassen, über die der Systemnutzer mit der CLIQWorkbench kommuniziert. Eine Auswahl von Dialogfenstern, die durch diese Klassen realisiert wurden, wird im Rahmen der in Abschnitt 11.3 erläuterten Fallstudie präsentiert. Die Attribute und Methoden der Dialogklassen betreffen fast ausschließlich die Verwaltung der graphischen Elemente der Dialoge und werden daher nicht näher erläutert.

9.10 Database/Repository Interface

Das Database/Repository Interface stellt Funktionalität zum lesenden und schreibenden Zugriff auf die der CLIQWorkbench zu Grunde liegenden Datenbanken (Fach- und Metadaten) bereit. Das Paket enthält die beiden Subpakete *MrRepository* (siehe Abb. 9.27) und *DBInterface* (ohne Abbildung). Diese sind grundsätzlich unabhängig von den übrigen CLIQWorkbench-Paketen und lassen sich auch in anderen datenbankbasierten Anwendungen einsetzen.

Das Paket *MrRepository* beinhaltet Hilfsklassen zur Handhabung des Microsoft Repository, welches von der CLIQWorkbench zur Verwaltung von schemabezogenen Metadaten genutzt wird (siehe Abschnitt 10.2.2). Die gleichnamige (Singleton-)Klasse (siehe dazu [GHJV01]) verwaltet die Verbindung zwischen Anwendung (hier: CLIQWorkbench) und Repository. Die statische Methode *GetRepository()* liefert eine Referenz auf die einzige Instanz der Klasse. Beim erstmaligen Aufruf dieser Methode wird die Verbindung zum Repository aufgebaut. Die einzelnen im Repository verwalteten Informationsmodelle (insbesondere die des OIM, siehe Abschnitt 10.2.2) werden über sog. *Typbibliotheken* angesprochen. Die Methode *GetTypeLibs()* liefert eine Liste aller verfügbaren Typbibliotheken, die Methode *GetTypeLib()* einen Zeiger auf eine bestimmte Typbibliothek. *GetRootObjectInterface()* schließlich gibt einen Zeiger auf das Wurzelobjekt des (hierarchisch aufgebauten) Repository zurück.

MrRepository wird von den Klassen *MrDbmRepository* und *MrDbmPackage* genutzt. Erstere ist für lesende, letztere für schreibende Zugriffe auf das sog. *Relational Database Schema Model* (Submodell des *Database and Warehousing Model* des OIM) zuständig. Die in Abb. 9.27 angedeutete Art des Zugriffs (siehe Methodenparameter) reflektiert die hierarchische Struktur des Informationsmodells,

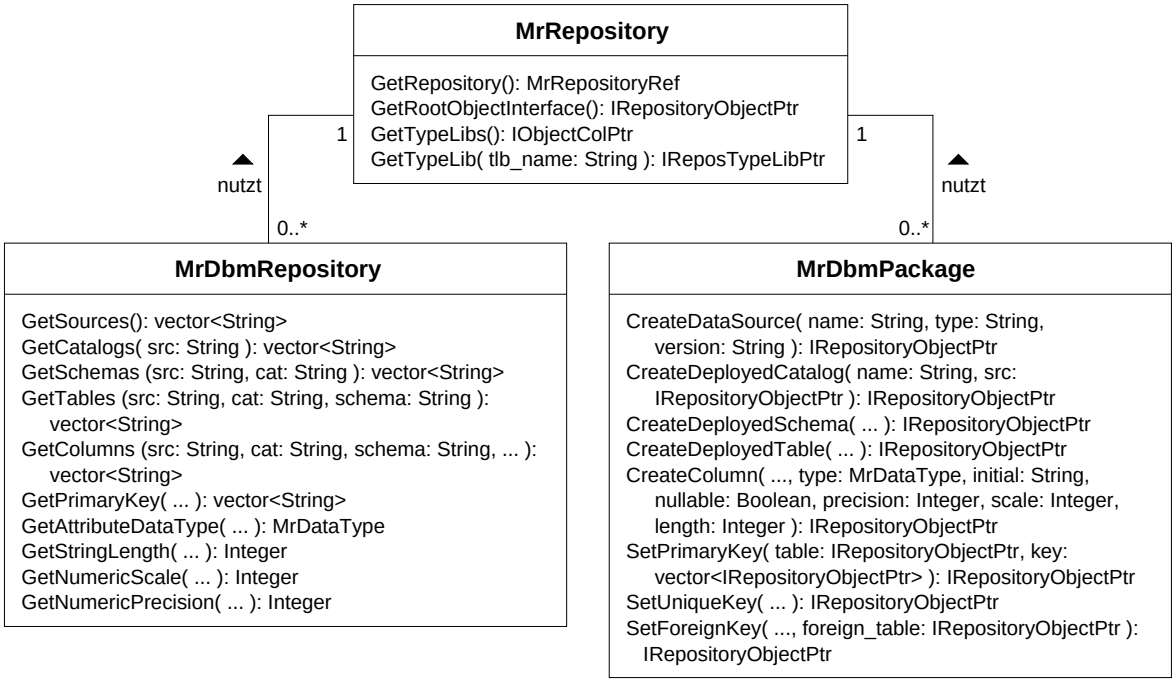


Abbildung 9.27: UML-Klassendiagramm zum Paket *MrRepository*

in der ausgehend von Quellen über Kataloge, Schemata und Relationen bis zu Attributen und deren Charakteristika, insbesondere Schlüsseleigenschaften und Datentyp (Aufzählungstyp *MrDataType*), „hinabgestiegen“ werden kann. Diejenigen Bezeichner mit Präfix „I“ stellen COM-Schnittstellen dar.

Das Paket *DBInterface* besteht lediglich aus einer Klasse gleichen Namens. Diese bietet grundlegende Funktionen zum Auf- und Abbau von Verbindungen zu relationalen DBMS (über Konstruktoren bzw. Destruktoren), zum Transaktionsmanagement (Commit und Rollback) sowie zur Abfrage von Datenbankstatus und Fehlermeldungen.

9.11 Metarelationen zur Unterstützung inkrementeller Updates

Datenänderungen in einer an das DWS angebundenen Quelle müssen – soweit der betroffene Datenbestand für das DWS relevant ist – in das DWS propagiert werden (sog. *inkrementelle Updates*). In diesem Abschnitt werden (relationale) Schemastrukturen spezifiziert, die die Durchführung inkrementeller Updates im DWS unterstützen. Die Erläuterung dieser Strukturen ist für das Verständnis des Datenintegrationsprozesses unerlässlich. Das relationale Schema der CLIQWorkbench umfasst darüber hinaus noch eine Reihe weiterer Relationen, die im Wesentlichen mit den in den Abschnitten 9.3 bis 9.7 beschriebenen Klassen korrespondieren. Da diese Relationen „nur“ der persistenten Speicherung von Laufzeitobjekten dienen, werden sie an dieser Stelle nicht näher betrachtet. Eine Zusammenstellung aller Relationenschemata der CLIQWorkbench ist in [Ade01] enthalten.

Die in Kapitel 7 eingeführten Datenhaltungskomponenten Quelldatenpuffer (QDP), Transformationsbereich (TB), Konsolidierungsbereich (KB) und Zielbereich (ZB) lassen sich auf logischer Ebene als unterschiedliche Datenbanken auffassen. Bei der Implementierung können die Bereiche wahlweise als einzelne physische Datenbanken oder als eine gemeinsame Datenbank (zur Unterscheidung Postfixe notwendig, siehe unten) realisiert werden. Die Relationen in Transformations-, Konsolidierungs- und Zielbereich werden unter dem Begriff *Integrationsdatenbank* zusammengefasst.

Die Relationen der CLIQWorkbench lassen sich wie folgt klassifizieren:

- *Fachrelationen* speichern (domänenspezifische) Fachdaten, die entweder Rohdaten (in einem Quelldatenpuffer), vollständig integrierte Daten (im Zielbereich) oder Zwischenergebnisse des Integrationsprozesses (in Transformations- oder Konsolidierungsbereich) darstellen.
- *Metarelationen* speichern Metadaten, die zur Durchführung von Integrationsprozessen erforderlich sind (z. B. Prozesskonfigurationen) oder von diesen zur Laufzeit erzeugt werden (z. B. Zuordnungen zwischen zusammengehörigen Tupeln und Prozessprotokolle). In Abschnitt 10.2.2 wird begründet, warum nicht alle Metadaten im Repository verwaltet werden.

Im Folgenden werden Metarelationen zur Speicherung von tupelbezogenen Metadaten im Kontext inkrementeller Updates modelliert – differenziert nach Quelldatenpuffer (Abschnitt 9.11.1) und Integrationsdatenbank (Abschnitt 9.11.2).

9.11.1 Quelldatenpuffer

Jeder QDP enthält eine beliebige Anzahl von Fachrelationen, die sich aus den jeweiligen Quellschemata ergeben. Diese Fachrelationen werden durch das Postfix *_QDP* gekennzeichnet. Zur Abstraktion von konkreten Fachrelationen wird im Folgenden der Platzhalter **_QDP* verwendet. Um von heterogen modellierten (ggf. zusammengesetzten) Primärschlüsselattributen in den Fachrelationen zu abstrahieren und damit die Handhabung von QDP-Daten zu vereinfachen, wird in **_QDP* das künstliche numerische Primärschlüsselattribut *local_id* eingeführt (siehe Beispiel in Abb. 9.28).

Zuordnungen zwischen den proprietären, quellenspezifischen und den einheitlichen, künstlich erzeugten Schlüsseln werden durch die Metarelation **_LU* (Lookup) verwaltet. Zu jeder Fachrelation im QDP existiert eine solche Metarelation zum Schlüssel-Lookup. Diese ermöglicht zum einen, im Falle eines inkrementellen Update (Änderung oder Löschung eines schon im DWS gespeicherten Tupels) zu dem gelieferten proprietären Schlüssel der Quelle den im DWS verwendeten, stellvertretenden lokalen Schlüssel zu ermitteln. Zum anderen ist gewährleistet, dass im Falle eines tupelbezogenen Feedback an eine Datenquelle die Originalschlüssel – nur diese sind der Quelle bekannt – rekonstruiert werden können. Die Attribute von **_LU* stellen zwar eine Teilmenge der Attribute von **_QDP* dar, die gespeicherten Informationen sind jedoch nur temporär redundant, da in **_QDP* nur temporär Daten gehalten werden (bis sie in Schritt 2 des CLIQ-Vorgehensmodells in den Transformationsbereich überführt werden). In **_LU* werden hingegen alle vormals vorgenommenen Zuordnungen zwischen proprietären Quellschlüsseln und von der CLIQWorkbench vergebenen lokalen Schlüsseln permanent gespeichert.

Weiterhin wird davon ausgegangen, dass jedes zu integrierende Tupel einen (im Rahmen der Datenextraktion gesetzten) Historisierungsstatus mitführt, der im Attribut *hist_state* von **_QDP* abgelegt ist (vgl. Abschnitt 7.2). *hist_state* kann einen der folgenden Werte annehmen:

- *'N' (New)*: Das Tupel wurde in der Datenquelle neu angelegt.
- *'H' (Historize)*: Das Tupel wurde in der Datenquelle geändert, wobei mindestens ein zu historisierendes Attribut betroffen war. Ein Attribut ist genau dann zu historisieren, wenn es sich um ein Fakt- oder ein dimensionales Attribut handelt. Faktattribute bilden Würfelzellen im multidimensionalen Datenmodell [TKS01] (Umsatz, Anzahl Krankheitsfälle etc.). Dimensionale Attribute spezifizieren die Position von assoziierten Fakten im multidimensionalen Datenraum.
- *'O' (Overwrite)*: Das Tupel wurde in der Datenquelle geändert, wobei ausschließlich beschreibende Attribute betroffen waren. Beschreibende Attribute dienen der näheren Beschreibung von Fakten, z. B. Erhebungsmethode, Diagnosesicherung etc.
- *'D' (Delete)*: Das Tupel wurde in der Datenquelle gelöscht.

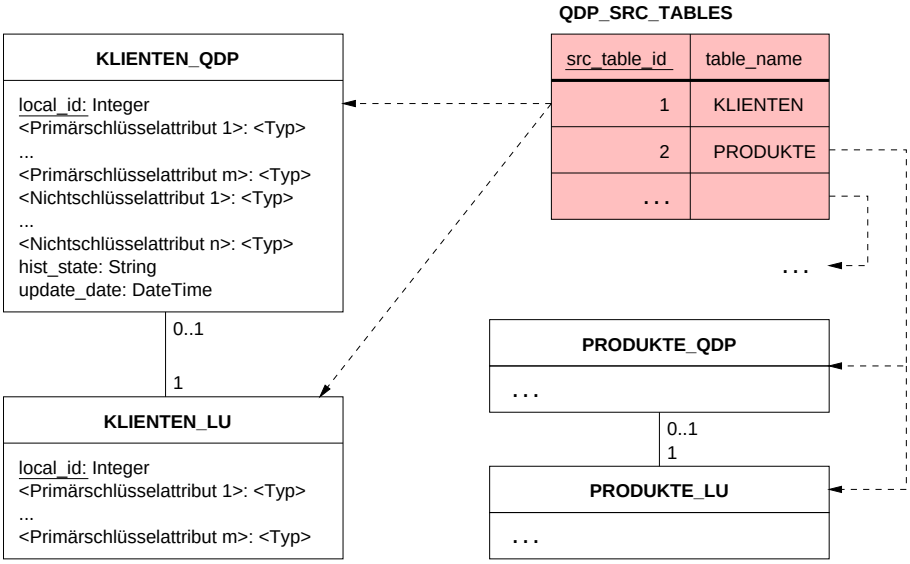


Abbildung 9.28: Aufbau eines Quelldatenpuffers

Der Historisierungsstatus wird im Rahmen des Datenintegrationsprozesses beim Merging zusammengehöriger Datensätze (Phase 6 des CLIQ-Vorgehensmodells) sowie bei der Aktualisierung des Zielbereichs (Phase 9) ausgewertet. Im Attribut *update_date* wird derjenige Zeitpunkt festgehalten, an dem das Datenbankereignis in der Quelle (Erfassung, Änderung, Löschung) erfolgte.

Die Metarelation *QDP_SRC_TABLES* repräsentiert das „Inhaltsverzeichnis“ eines QDP; jede Fachrelation des QDP ist hier – versehen mit einer eindeutigen Kennung – registriert.

9.11.2 Integrationsdatenbank

Abbildung 9.29 zeigt den Aufbau der Integrationsdatenbank. Wie bereits in Kapitel 7 erläutert, sind die Schemata von Transformations-, Konsolidierungs- und Zielbereich identisch. Die Namen der in den Bereichen vorkommenden Fachrelationen werden in der Metarelation *DEST_TABLES* verwaltet. Zur einfacheren Unterscheidung wird im Folgenden an den Namen einer Fachrelation jeweils eines der drei Postfixe *_TB*, *_KB* oder *_ZB* angehängt, je nachdem, ob eine Fachrelation im Transformations-, Konsolidierungs- oder Zielbereich gemeint ist.

Von den konkreten Namen der Fachrelationen in den drei Bereichen wird durch die Platzhalternotation **_TB*, **_KB* bzw. **_ZB* abstrahiert. Alle (Fachdaten-)Tupel weisen eine (in Phase 2 des CLIQ-Vorgehensmodells erzeugte) numerische Kennung auf (Attribut *glb_id*). Eine solche Kennung ist eindeutig innerhalb der Integrationsdatenbank. Wird ein Tupel im Laufe des Datenintegrationsprozesses vom Transformations- in den Konsolidierungsbereich oder vom Konsolidierungs- in den Zielbereich übertragen, so behält es seine Kennung bei. Durch die eindeutige Kennung werden Primärschlüsselkonflikte, die bei der Integration von Daten aus unterschiedlichen Quellen auftreten könnten, vermieden.

Zu jedem Tripel von äquivalenten Fachrelationen, z. B. *(KUNDEN_TB, KUNDEN_KB, KUNDEN_ZB)* in Abb. 9.29, werden zwei Metarelationen definiert, die durch die Postfixe *_ES* (Entity State) und *_EH* (Entity History) als Erweiterung des originären Namens (hier: *KUNDEN_ES* und *KUNDEN_EH*) gekennzeichnet sind. Als Platzhalter für diese Metarelationen werden im Folgenden die Bezeichnungen **_ES* und **_EH* verwendet.

Jedes (Metadaten-)Tupel in **_ES* beschreibt den Status eines über die Kennung *glb_id* bezeichneten (Fachdaten-)Tupels in der Integrationsdatenbank an Hand einer Menge von Statusattributen. Diese

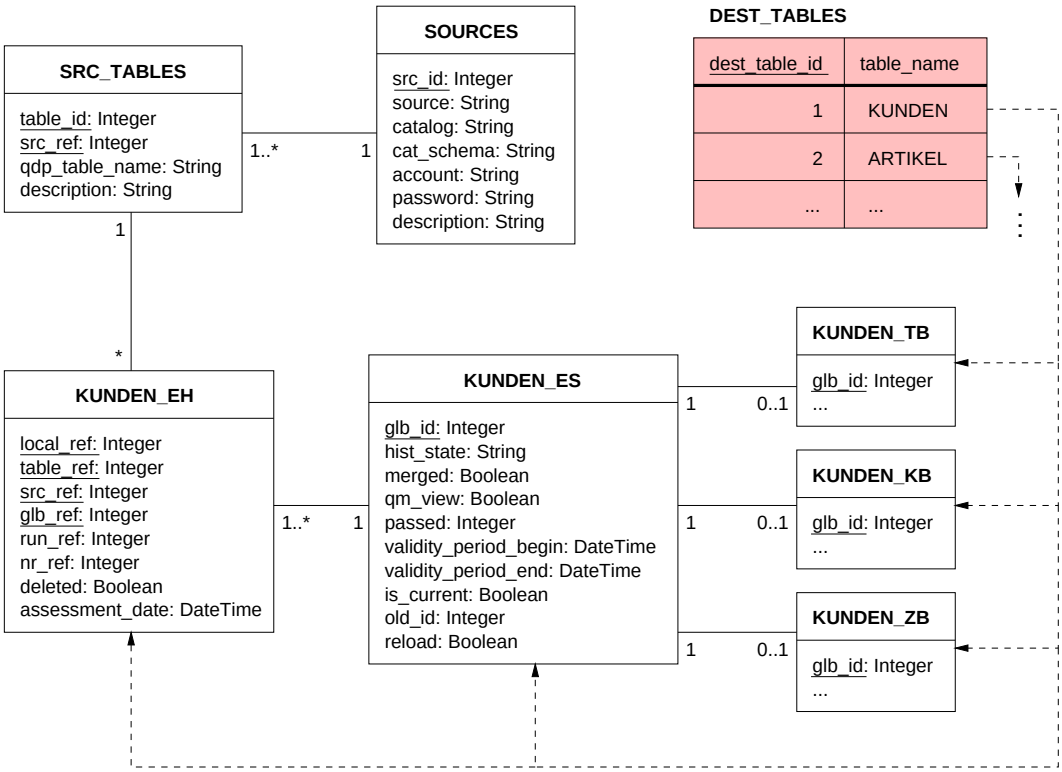


Abbildung 9.29: Aufbau der Integrationsdatenbank

lassen sich differenzieren in Attribute, die im Datenintegrationsprozess verwendet werden (sog. *interne Statusattribute*), und Attribute, die bei der Auswertung der Fachdaten zu berücksichtigen sind und im Datenintegrationsprozess gesetzt werden (sog. *externe Statusattribute*). Die einzelnen Attribute werden nun kurz charakterisiert (vgl. Abschnitt 7.2):

- Interne Statusattribute
  - *hist_state:* Entspricht dem gleichnamigen Attribut in **_QDP*.
  - *merged:* Kennzeichnet, ob ein Tupel aus einer Zusammenführung mehrerer Tupel entstanden ist.
  - *qm_view:* Kennzeichnet, ob ein Tupel (in Konsolidierungs- oder Zielbereich) im Rahmen von Datenqualitätsmessungen und -analysen (Phase 7) zu berücksichtigen ist.
  - *passed:* Gibt an, ob alle Datenobjekte, in denen das assoziierte Tupel enthalten ist, die gestellten Qualitätsanforderungen erfüllen oder nicht.
- Externe Statusattribute
  - *validity_period_begin:* Kennzeichnet den Beginn des Gültigkeitszeitraums eines Tupels.
  - *validity_period_end:* Kennzeichnet das Ende des Gültigkeitszeitraums eines Tupels.
  - *is_current:* Kennzeichnet, ob ein Tupel aktuell oder bereits historisiert ist.
  - *old_id:* Falls das assoziierte Tupel ein inkrementelles Update eines bereits historisierten („alten“) Tupels darstellt, wird hier auf dessen Kennung verwiesen und damit die Tupelhistorie nachgeführt.
  - *reload:* Kennzeichnet, ob das Tupel neu in ein an den Zielbereich angebundenes Data Warehouse geladen werden muss.

Die Metarelation **_EH* dient der Verwaltung von Historien von (Fachdaten-)Tupeln. Unter Anderem werden hier Zuordnungen zwischen (künstlich erzeugten) lokalen Schlüsseln (*local_ref*) – inkl. Angabe der jeweiligen Quelle (*src_ref*) und Quellrelation (*table_ref*) – und global eindeutigen Schlüsseln (*glb_ref*) gespeichert. An Hand dieser Zuordnungen können neu zu integrierende Tupel, die Updates von bereits in der Integrationsdatenbank vorliegenden Tupeln darstellen, auf einfache Weise mit den zu überschreibenden bzw. zu historisierenden Tupeln assoziiert werden. Weiterhin werden in **_EH* Angaben zum assoziierten Ausführungslauf eines Jobs (*run_ref, nr_ref*), zum Befunddatum (*assessment_date*, im Gegensatz zum *update_date* in **_QDP* dauerhaft gespeichert) sowie zum Löschstatus von Tupeln (*deleted*) abgelegt. Letzterer gibt an, ob das Tupel in der jeweiligen Datenquelle gelöscht worden ist. Diese Angabe wird im Rahmen der Historisierung von Tupeln im Zielbereich benötigt, da ein ZB-Tupel, das aus mehreren Quelltupeln (durch Merging) hervorgegangen ist, erst dann historisiert werden darf, wenn *alle* beteiligten Quelltupel in den jeweiligen Quellen gelöscht wurden.

Die Metarelation *SOURCES* weist für jeden an die CLIQWorkbench angebundenen Quelldatenpuffer ein Tupel auf, welches Zugangsdaten sowie eine Beschreibung des QDP enthält.

Die Metarelation *SRC_TABLES* verwaltet zu jeder relevanten QDP-Relation eine eindeutige Kennung (*table_id*), einen Verweis auf den zugehörigen QDP (*src_ref*), den Relationennamen (*qdp_table_name*) sowie eine Beschreibung (*description*).

Die Verwendung der einzelnen Metarelationen im Datenintegrationsprozess wird im Rahmen der in Abschnitt 11.3 beschriebenen Fallstudie demonstriert.

9.12 Zusammenfassung

Das im vorliegenden Kapitel entworfene, als CLIQWorkbench bezeichnete Softwaresystem ist modular aufgebaut: Die durch das CLIQ-Vorgehensmodell vorgeschriebenen DQM-Maßnahmen werden durch die Komponenten Data Migrator, Data Linker, Data Auditor, Measurement Engines und Release/Retraction Handler realisiert. Die Ansteuerung dieser Komponenten erfolgt über einen zentralen DQM Controller. Vervollständigt wird die Softwarearchitektur durch die CLIQWorkbench Console, die eine graphische Benutzungsoberfläche bereitstellt, und das Database/Repository Interface, das Zugriffe auf die Fachdatenbank und das Metadaten-Repository kapselt. Die folgende Aufzählung fasst die wichtigsten Entwurfsentscheidungen bzgl. der CLIQWorkbench zusammen:

- Sowohl die Datenintegration als auch deren Konfigurierung erfolgt unter Verwendung von Threads, um eine quasi-parallele Ausführung nebenläufiger Prozesse zu ermöglichen.
- Zur Analyse von Daten im Hinblick auf Qualitätsmängel werden – wo angemessen – innovative Verfahren aus den Bereichen SPC, Probabilistisches Record Linkage und Data Mining eingebunden.
- Eine Anpassung des prinzipiell domänenunabhängigen Systems an domänenspezifische Gegebenheiten kann über Prozesskonfigurationen und Metadaten vorgenommen werden. Zur Durchführung von Konsistenzprüfungen ist zusätzlich eine gegebene Klasse (*DomainSpecificMeasurementEngine*) zu spezialisieren.
- Zur Protokollierung von Integrationsläufen wird XML als Metasprache eingesetzt.
- Soweit möglich, wird auf existierende Softwarekomponenten (Microsoft Data Transformation Services, Microsoft Excel, ILOG Rules, Vality Integrity, MLC++, Xerces, UnitConversion) zurückgegriffen, um den Entwurfs- und Implementierungsaufwand zu reduzieren und die Robustheit des Gesamtsystems zu erhöhen.

Kapitel 10

Implementierung der CLIQWorkbench

In Abschnitt 10.1 wird zunächst ein Überblick über die Softwarearchitektur des implementierten CLIQWorkbench-Prototypen gegeben. Abschnitt 10.2 geht näher auf dabei eingesetzte Fremdkomponenten ein, während Abschnitt 10.3 die CLIQWorkbench aus Sicht der komponentenbasierten Softwareentwicklung untersucht. Das Kapitel schließt mit einer Zusammenstellung von Angaben zum Umfang der Implementierung in Abschnitt 10.4. An den Implementierungsarbeiten zur CLIQ-Workbench waren Thomas Aden (als Hilfskraft, Diplomand und Mitarbeiter), Jürgen Meister (als Mitarbeiter), Ina Wentzlaff (als Hilfskraft) und Thomas Wilkens (als Diplomand) beteiligt.

10.1 Überblick

Abbildung 10.1 zeigt die Schichtenarchitektur des in dieser Arbeit implementierten Softwaresystems. Diese setzt sich aus neu implementierten und wiederverwendeten Komponenten zusammen.

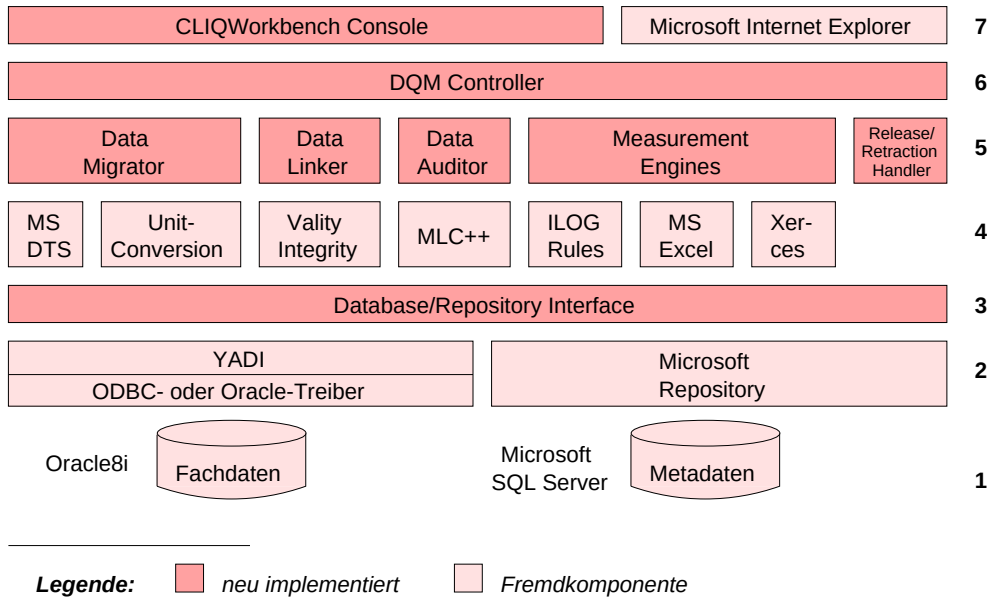


Abbildung 10.1: Schichtenarchitektur der CLIQWorkbench

In Schicht 1 befinden sich die Datenhaltungskomponenten. In der aktuellen Version des Prototypen werden die Fachdaten in Oracle 8i-Datenbanken gehalten (Quelldatenpuffer, Transformations-, Konsolidierungs- und Zielbereich). Metadaten werden vom Microsoft SQL Server 7 verwaltet. Schicht 2

enthält Schnittstellen zu den Datenhaltungskomponenten. Dabei erfolgt der Fachdatenzugriff über das im OFFIS entwickelte Datenbank-Interface YADI (aufbauend auf einem entsprechenden Treiber), der Metadatenzugriff über das Microsoft Repository. Beide Komponenten werden vom Database/Repository Interface (vgl. Abschnitt 9.10) in Schicht 3 gekapselt.

Die Schichten 5 und 6 stellen den Kern der CLIQWorkbench dar: Der DQM Controller in Schicht 6 (vgl. Abschnitt 9.3) realisiert das in Kapitel 7 entwickelte Vorgehensmodell zur Datenintegration. Die Komponenten in Schicht 5 (Data Migrator, Data Linker, Data Auditor, Measurement Engines, Release/Retraction Handler) sind jeweils für einzelne Schritte des CLIQ-Vorgehensmodells verantwortlich, wie in den Abschnitten 9.4 bis 9.8 beschrieben. Dabei machen sie sich die Funktionalität der z. T. kommerziellen Werkzeuge und Bibliotheken aus Schicht 4 zunutze. Dazu gehören die Microsoft Data Transformation Services, die Bibliothek UnitConversion, das ETL-Werkzeug Vality Integrity, die Data Mining-Klassenbibliothek MLC++, die Regel-Engine von ILOG Rules, Microsoft Excel sowie die Klassenbibliothek Xerces. Ausgewählte Fremdkomponenten werden in Abschnitt 10.2 noch näher erläutert. Das Front End der CLIQWorkbench bilden die CLIQWorkbench Console (vgl. Abschnitt 9.9) und der Microsoft Internet Explorer in Schicht 7.

Die Implementierung aller neu entwickelten CLIQWorkbench-Komponenten erfolgte mit Microsoft Visual C++ 6.0 unter Verwendung der Microsoft Foundation Classes (MFC) [Jon99] zur Realisierung der graphischen Benutzungsoberfläche.

10.2 Fremdkomponenten der CLIQWorkbench

In diesem Abschnitt werden ausgewählte Fremdkomponenten, auf die bei der Implementierung des in dieser Arbeit entwickelten Softwaresystems zurückgegriffen wurde, näher vorgestellt. Auf Details zu den Microsoft-Werkzeugen Excel (genutzt zur Erzeugung von Diagrammen mit Messergebnissen sowie zur Berechnung von Qualitätsindikatoren) und Internet Explorer (genutzt zur Visualisierung von im Laufe des Datenintegrationsprozesses generierten XML-Dateien) wird verzichtet. Gleiches gilt für die Bibliotheken UnitConversion (genutzt zur Umrechnung von Maßeinheiten) und Xerces (genutzt zum Erzeugen von XML-Dokumenten), die in der Arbeit nur eine untergeordnete Rolle spielen.

10.2.1 YADI

Bei YADI (Yet Another Database Interface) [Fri96] handelt es sich um eine am Institut OFFIS entwickelte C++-Klassenbibliothek zum Zugriff auf relationale DBS über einen DBMS-spezifischen Treiber oder ODBC. Im Folgenden wird an Hand von einfachen Codefragmenten die Verwendung von YADI-Methoden demonstriert. Auf (in der Praxis notwendige) Fehlerabfragen wird dabei verzichtet.

Das folgende Codefragment baut eine Verbindung zu einer Oracle-Datenbank auf. Dabei wird zunächst eine Instanz eines sog. *Treibermanagers* erzeugt (Klasse *JdaYadiFactory*). Mit der Methode *NewSession()* des Treibermanagers – parametriert mit entsprechenden Zugangsinformationen – wird im nächsten Schritt die eigentliche Datenbankverbindung (Session) erstellt. Dabei spezifiziert der Parameter *dbms* das anzubindene DBMS („Ora“ für Oracle, „ODBC“ für ein beliebiges ODBC-fähiges DBMS).

```
JdaYadiFactory* factory = new JdaYadiEnv;
JdaDbSession* session = factory->NewSession(
      login, password, connect, dbms );
```

In jeder Session kann es mehrere sog. *Cursors* geben, die jeweils ein SQL-Kommando zur Zeit verwalten können. Um also – wie im folgenden Beispiel – ein SQL-SELECT-Kommando absetzen zu können, muss zunächst mittels der Methode *JdaDbSession::NewCursor()* ein Cursor (Klasse *JdaHiLevelCursor*) erzeugt werden. Die Methode *JdaHiLevelCursor::ExecuteArbitrary()* wird dann zum Absetzen der Anfrage verwendet. Sie liefert das (mengenwertige) Anfrageergebnis als Objekt der Klasse *JdaValueArrayListList* zurück.

```
JdaHiLevelCursor* cursor = session->NewCursor();
JdaValueArrayListList* result = cursor->ExecuteArbitrary(
      "SELECT plz, wohnort FROM kunden" );
```

Das Anfrageergebnis wird von YADI in mehrere Blöcke aufgespalten, falls die Anzahl der Tupel die a priori gegebene maximale Blockgröße übersteigt. Mit der Methode *JdaValueArrayListList::Entry()* kann ein Block ausgelesen und mit *JdaValueArrayListList::Next()* zum nachfolgenden Block navigiert werden, wie im folgenden Codefragment dargestellt. Erstgenannte Methode liefert einen Zeiger auf ein Objekt der Klasse *JdaAbstractValue* zurück. Dieses entspricht einer Liste von Tupeln. Die Anzahl der Tupel kann mittels *JdaAbstractValue::CurrentElements()* bestimmt werden. Der Zugriff auf den ersten Attributwert des i-ten Tupels der Liste erfolgt mittels *val→GetV...()[i]*, wobei die konkrete Methode vom Datentyp des jeweiligen Attributs abhängt (z. B. *GetVint()* für einen Integer-Wert). Mit *JdaAbstractValue::Next()* kann zum nächsten Attributwert eines Tupels navigiert werden. Die Reihenfolge der Attributwerte in einem Tupel ist durch die SELECT-Klausel festgelegt (im Beispiel *plz* vor *wohnort*).

```
while( result ){
  JdaAbstractValue* val = result->Entry();
  num = val->CurrentElements();
  for (unsigned long i = 0; i < num; i++){
    plz = val->GetVint()[i];
    wohnort = val->Next()->GetVstring()[i];
    ...  //Weiterverarbeitung von plz und wohnort
  } //for
  result = result->Next();
} //while
```

Durch das Löschen der Cursor- und Session-Objekte wird eine Datenbankverbindung wieder abgebaut:

```
delete cursor;
delete session;
```

10.2.2 Microsoft Repository und OIM

Das Microsoft Repository [BHS+97, BBC+99] ist ein Werkzeug zum objektorientierten Metadatenmanagement. Metadaten werden im Microsoft SQL Server persistent gespeichert. Der Zugriff erfolgt über eine Repository Engine (siehe Abb. 10.2). Diese implementiert ein sog. *Repository Type Information Model (RTIM)*. Dabei handelt es sich um ein (Metameta-)Modell zur Beschreibung von Metamodellen. Bei Auslieferung (mit dem SQL Server) „kennt“ das RTIM die Metamodelle des OIM [Met99a]. Die Entscheidung zugunsten des OIM wurde infolge einer vergleichenden Untersuchung von OIM und dem Konkurrenzstandard *Common Warehouse Metamodel (CWM)* [Obj00] getroffen.

Für das OIM sprach, dass dessen Spezifikation zum Zeitpunkt der Entwurfsentscheidung wesentlich weiter fortgeschritten war als die des CWM und dass es bereits eine Implementierung dieser Spezifikation gab. Die Weiterentwicklung des OIM wurde Ende 2000 eingestellt; die OMG arbeitet zur Zeit daran, Konzepte des OIM in das CWM zu integrieren. Ausführliche Vergleiche zwischen OIM und CWM können in [VVS00] und [JR00] nachgelesen werden.

Schreibende Zugriffe auf die Metamodelle des Repository (insbesondere durch Entwurfswerkzeuge) erfolgen über COM [EE98, Mal99a]. Lesende Zugriffe sind sowohl über COM als auch über SQL möglich. Weiterhin bietet das Microsoft Repository eine XML-Schnittstelle zum Import und Export von Metadaten.

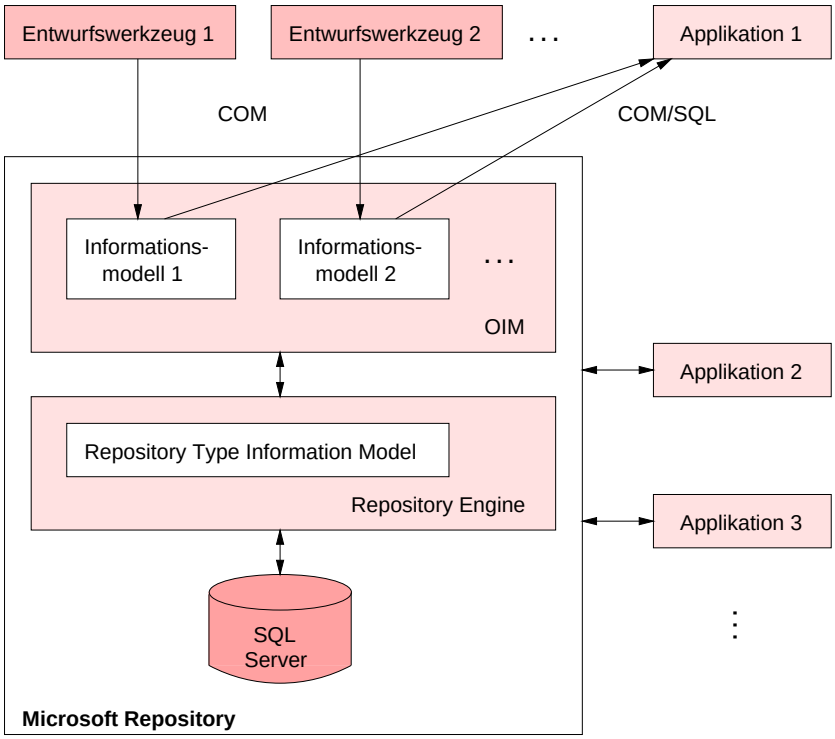


Abbildung 10.2: Microsoft Repository

Das OIM weist eine hierarchische Struktur auf (siehe Abb. 10.3). Es gibt verschiedene sog. *Informationsmodelle*, die wiederum jeweils aus Submodellen bestehen. Die Informationsmodelle decken verschiedene Bereiche eines IT-Systems ab, z. B. die Analyse- und Entwurfsdokumentation, Komponentenbeschreibungen sowie Schema- und Transformationsspezifikationen. Nicht alle Informationsmodelle sind so ausgereift, dass sie für den Praxiseinsatz tauglich sind. Die CLIQWorkbench verwendet daher ausschließlich das *Relational Database Schema Model* (vgl. Abschnitt 9.10) sowie das *Data Transformations Model* (vgl. Abschnitt 9.4) des OIM.

Laut Systemdokumentation kann das OIM wie folgt erweitert und so auf verschiedene Anwendungsgebiete zugeschnitten werden:

- *Manuell* durch Programmierung der Erweiterungen direkt im RTIM, was aufwändig und fehleranfällig ist.
- *Werkzeugunterstützt* durch das sog. *Model Development Kit (MDK)*. Das MDK ist ein Add-In für Rational Rose und erlaubt die graphische Modellierung neuer Metamodelle mit der UML [Obj01]. Nach einem Syntaxtest können sog. *Verteilungsdateien* (Installationsdateien, Typbibliotheken, Beispielcode) generiert werden, die dann zu einer gegebenen Applikation dazugebunden werden können.

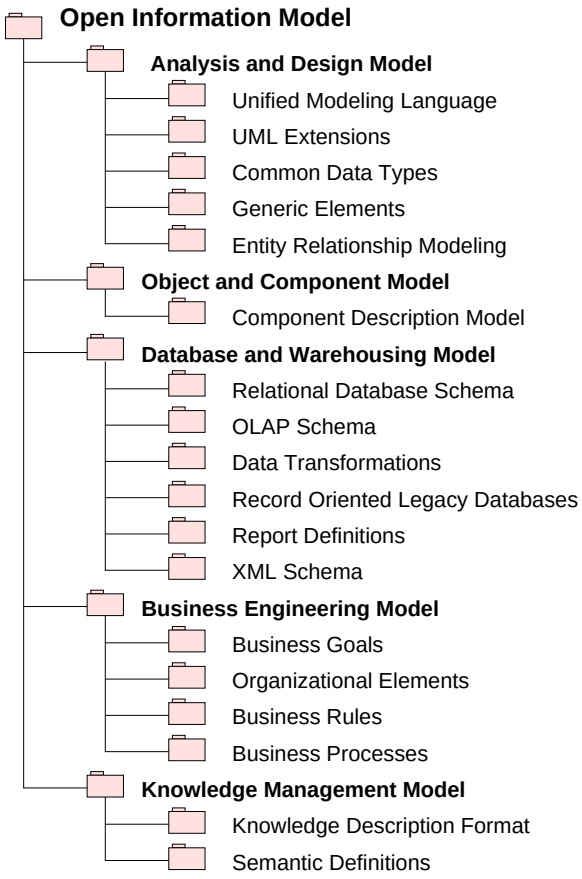


Abbildung 10.3: Informationsmodelle des OIM [Met99a]

In der Praxis funktioniert die werkzeugunterstützte Erweiterung des OIM aufgrund von Unzulänglichkeiten der MDK-Implementierung jedoch nicht. Die für das DQM benötigten Metamodelle mussten daher – soweit sie nicht bereits durch das OIM abgedeckt sind – unter Umgehung des Microsoft Repository implementiert werden. Provisorisch wurde eine relationale Modellierung vorgenommen. Die einzelnen Relationenschemata sind in [Ade01] zu finden.

10.2.3 Microsoft Data Transformation Services (DTS)

Die Microsoft Data Transformation Services [AL99] unterstützen die Extraktion von Daten aus heterogenen Quellen und ihre Transformation in Zieldatenstrukturen. Die Transformationen beschränken sich auf klassische Attribut-Mappings, wobei die Abbildungsvorschriften beliebig komplex sein können. Neben einer Reihe vordefinierter Funktionen hat der Nutzer die Möglichkeit, ActiveX-Skripte einzubinden und darüber ggf. auch die Funktionalität von Drittanbieterwerkzeugen zu nutzen. Die Anbindung der Quell- und Zieldatenbanken erfolgt über OLE DB. Spezifikationen von Transformationsprozessen sind in den DTS als *Transformationspakete* organisiert (vgl. Abschnitt 9.4). Jedes Transformationspaket besteht aus einer Menge von Transformationsabbildungen, Constraints und Verknüpfungen zu Datenbanken. Es kann gespeichert und beliebig oft ausgeführt werden. Dabei erfolgt intern eine Abbildung auf das Data Transformations Model des OIM (siehe Abschnitt 10.2.2). Die Ausführung eines Transformationspakets ist transaktionsorientiert; über Transaktionen werden Log-Dateien geführt.

Abbildung 10.4 zeigt die einzelnen Bestandteile der DTS. Zur Erzeugung von Transformationspaketen stehen der *Import/Export Wizard* sowie der *DTS Designer* zur Verfügung. Während ersterer sich

zur schnellen, teilautomatisierten Spezifizierung einfacher Transformationen eignet, lassen sich mit letzterem auch komplexe Transformationspakete interaktiv erstellen. Die Pakete werden mit Hilfe des sog. *DTS Object Model* (einer COM-Schnittstelle) angesteuert und so den Ausführungswerkzeugen zugänglich gemacht. Hierzu gehören das Kommandozeilen-Utility *DTSRun*, das graphische Front End *DTSRunUI* sowie *SQLAgent* (streng genommen kein DTS-Modul) zum Scheduling von Ausführungsjobs.

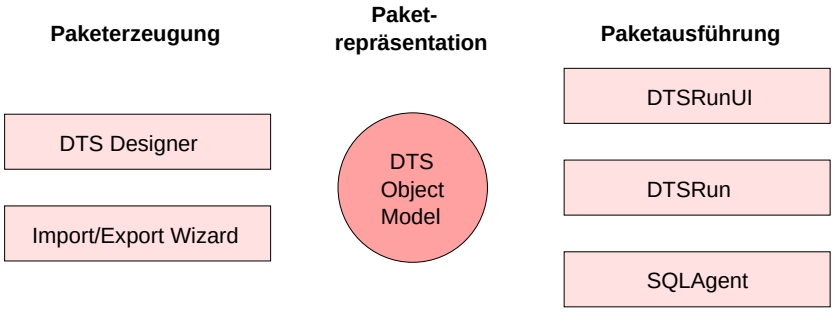


Abbildung 10.4: Module der Data Transformation Services

Die DTS werden vom Data Migrator benutzt, um Daten aus den Quelldatenpuffern in den Transformationsbereich zu übertragen und dabei zu vereinheitlichen (vgl. Abschnitt 9.4). Dabei wird die DTS-Funktionalität direkt über COM angesteuert, da nur über diese Schnittstelle der gesamte Funktionsumfang der DTS zur Verfügung steht; die in Abb. 10.4 skizzierten Front End-Module kommen nicht zum Einsatz.

10.2.4 Vality Integrity

Bei Vality Integrity [Val00, Val02] handelt es sich um ein ETL-Werkzeug, dessen Besonderheit ein ausgereiftes probabilistisches Record Linkage-Verfahren ist. Integrity setzt sich aus zwei Modulen zusammen, die vergleichbare Funktionalität in unterschiedlichen Umgebungen bieten (und inzwischen auch separat vertrieben werden): Während das sog. *Data Reengineering Environment* eine graphische Benutzungsoberfläche zur Spezifizierung (proprietäre Notation) und zum Monitoring von ETL-Prozessen zur Verfügung stellt, besteht das sog. *Integrity Real Time* aus Funktionsbibliotheken mit Application Programming Interfaces (APIs) für C, Java und Cobol, die vom Anwendungsentwickler direkt in eigene Applikationen eingebunden werden können. In der CLIQWorkbench wird ausschließlich Integrity Real Time (über das C-API) genutzt. Die von Integrity angebotene Funktionalität umfasst folgende vier Bereiche:

- Analyse von Datenbeständen
- Vereinheitlichung heterogener Datenbestände (insb. für Adressangaben)
- Probabilistisches Record Linkage auf vereinheitlichten Datenbeständen
- Merging von einander zugeordneten Tupeln.

Das Analysemodul von Integrity ist im Wesentlichen auf die Untersuchung von Schemainformationen ausgerichtet, betrifft also weniger die Datenintegration selbst und wird daher in CLIQ nicht verwendet. Die Funktionalität der Module zur Vereinheitlichung und zum Record Linkage kommt in den CLIQWorkbench-Komponenten Data Migrator und Data Linker zum Einsatz (siehe Abschnitte 9.4 und 9.6). Die Merging-Funktionalität von Integrity ist hingegen unzureichend und wird daher von der CLIQWorkbench nicht genutzt. Stattdessen wurde im Rahmen dieser Arbeit das Paket *RecLinkReview* zur interaktiven Nachbearbeitung von Tupelzuordnungen entwickelt (siehe Abschnitt 9.6.2).

Im Hinblick auf die Vereinheitlichung der Datenrepräsentation überschneidet sich der Funktionsbereich von Integrity mit dem der Microsoft DTS (siehe Abschnitt 10.2.3). Im Data Migrator werden die „besten“ Eigenschaften beider Komponenten kombiniert: Transformationen zwischen Attributwerten werden mit Hilfe der DTS spezifiziert, wobei über die Einbindung von ActiveX-Skripten auf die Integrity-Funktionalität zur Vereinheitlichung von Adressangaben zugegriffen werden kann.

10.2.5 MLC++

MLC++ (Machine Learning Library in C++) [KSD97] stellt eine objektorientierte Bibliothek von Data Mining-Algorithmen dar. Bei der Konzipierung von MLC++ wurde besonderer Wert auf eine hohe Vergleichbarkeit der von den verschiedenen Algorithmen gelieferten Ergebnisse gelegt, wodurch die Auswahl eines geeigneten Algorithmus für einen gegebenen Datenbestand vereinfacht wird. MLC++ stellt eine Workbench zur Verfügung, die derartige Vergleiche unterstützt sowie die Entwicklung von hybriden Algorithmen ermöglicht. MLC++ richtet sich also weniger an Endanwender, sondern eher an Softwareentwickler, die Unterstützung bei der Umsetzung einer gegebenen Data-Mining-Aufgabe benötigen.

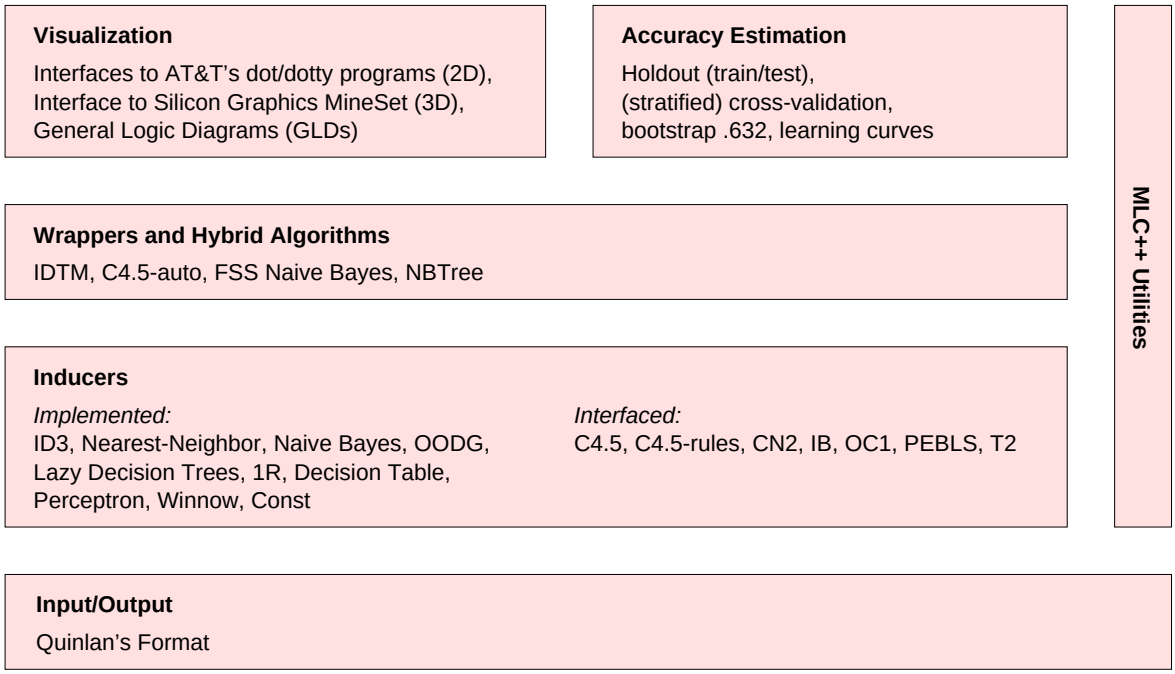


Abbildung 10.5: Module von MLC++

MLC++ besteht aus den in Abb. 10.5 dargestellten Modulen. Als einheitliches Ein- und Ausgabeformat wird das Format von Quinlan [Qui93] unterstützt. Auf Eingabedaten können eine Reihe von Data Mining-Algorithmen (sog. *Inducer*) angewandt werden, u. A. zur Induktion von Entscheidungsbäumen, Assoziationsregeln und Nearest-Neighbour-Clustern. Es ist zu unterscheiden zwischen Algorithmen, die direkt in MLC++ implementiert sind, und Algorithmen (z. T. kommerziell, z. B. C4.5), zu denen lediglich eine Schnittstelle bereitgestellt wird. Durch die objektorientierte Struktur von MLC++ ist es möglich, Algorithmen als Black Boxes zu kapseln und Berechnungsergebnisse zu kombinieren. Ausgewählte Wrapper und hybride Algorithmen sind in MLC++ bereits enthalten. Für Endanwender ohne Programmierkenntnisse wird neben dem objektorientierten API noch eine Menge sog. *MLC++ Utilities* bereitgestellt. Dabei handelt es sich um Kommandozeilenprogramme, die ausgewählte Funktionen aus MLC++ anbieten. Zur Visualisierung von Berechnungsergebnissen unterstützt MLC++ den Datenexport an verschiedene Werkzeuge von Drittanbietern. Schließlich stellt

MLC++ Funktionen zur oben angesprochenen vergleichenden Bewertung von Algorithmen für eine gegebene Aufgabe zur Verfügung.

Im Prototypen der CLIQWorkbench wird bisher lediglich der ID3-Algorithmus zur Induktion von Entscheidungsbäumen genutzt; dieser wurde an den Data Auditor angebunden (vgl. Abschnitt 9.7).

10.2.6 ILOG Rules

ILOG Rules [Ilo02] stellt C++-Funktionsbibliotheken zur Verfügung, mit denen Inferenzen auf regelbasierten Wissensbasen durchgeführt werden können. Dazu wurde in ILOG Rules der von Charles Forgy entwickelte *RETE-Algorithmus* (beschrieben u. A. in [RN95]) zum Pattern Matching objektorientiert implementiert. In Abschnitt 9.5.2 wurde bereits erläutert, wie domänenspezifisches Wissen mittels ILOG Rules zur automatisierten Konsistenzprüfung verwendet werden kann. An dieser Stelle wird diese Anbindung nun kurz aus Implementierungssicht betrachtet.

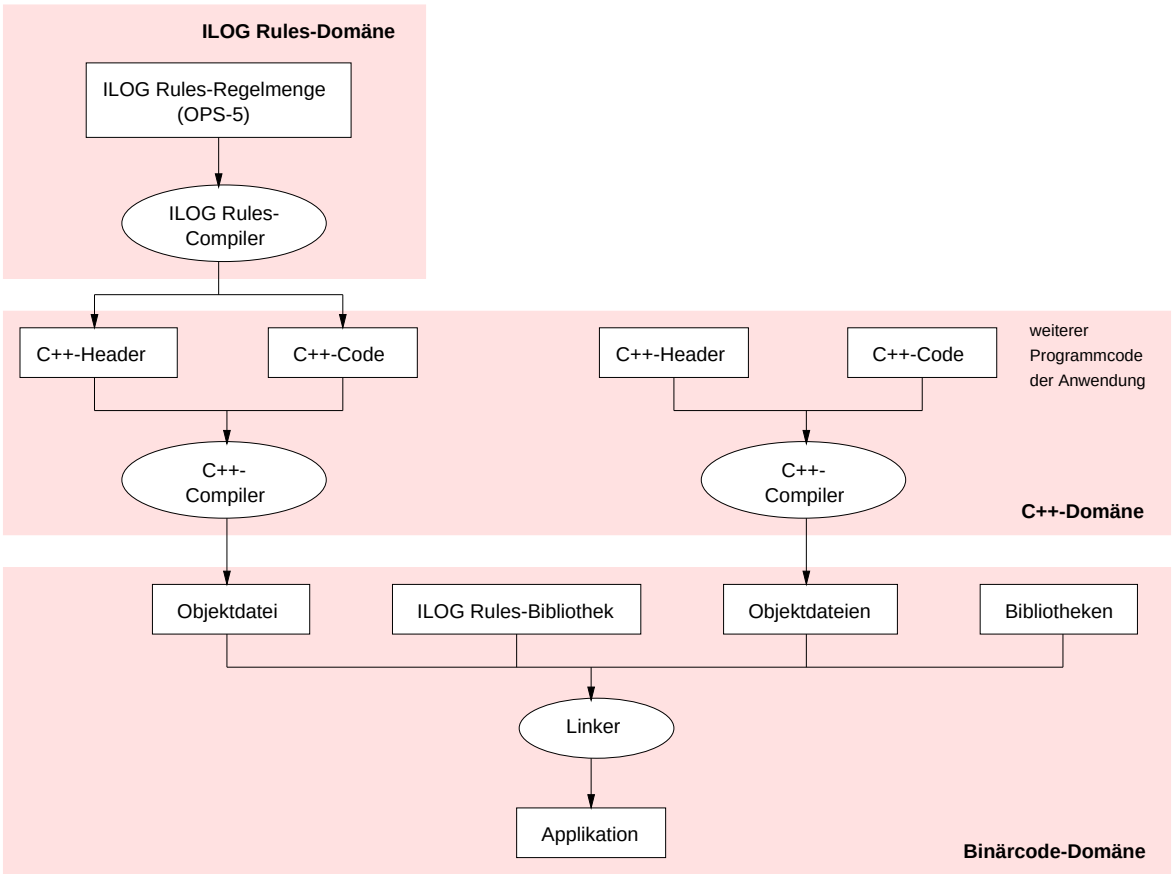


Abbildung 10.6: Schritte zur Erstellung einer ILOG Rules-basierten Applikation [Ilo97]

Wie Abb. 10.6 zeigt, läuft die Anbindung in mehreren Schritten ab: Eine Quelldatei mit den in OPS-5 notierten Regeln wird zunächst vom ILOG Rules-Compiler in eine C++-Header- und eine C++-Code-Datei umgewandelt. Aus diesen können dann, analog zu den C++-Dateien der eigentlichen Applikation, mit einem herkömmlichen C++-Compiler Objektdateien erstellt werden. Im letzten Schritt werden die Objektdateien mit der ILOG Rules-Bibliothek, die die RETE-Implementierung bereitstellt, und weiteren applikationsspezifischen Bibliotheken zu einer ausführbaren Anwendung gelinkt.

10.3 Die CLIQWorkbench aus Sicht der komponentenbasierten Softwareentwicklung

Die komponentenbasierte Softwareentwicklung hat in den vergangenen Jahren zunehmend an Bedeutung gewonnen. Dabei werden erprobte Softwarekomponenten systematisch wiederverwendet, um Kostenersparnisse und eine verbesserte Softwarequalität zu erreichen. Eine *Softwarekomponente* (kurz: *Komponente*) ist nach [Szy98] definiert als „eine Kompositionseinheit, die nur vertraglich spezifizierte Schnittstellen und explizite Kontextabhängigkeiten aufweist. Eine Komponente kann unabhängig eingesetzt werden und unterliegt der Komposition durch dritte Parteien“. In diesem Abschnitt des Kapitels werden Entwurf und Implementierung der CLIQWorkbench aus dem Blickwinkel der komponentenbasierten Softwareentwicklung beleuchtet.

Zunächst ist zu unterscheiden zwischen gegebenen Komponenten, die in der CLIQWorkbench wiederverwendet werden (siehe Abschnitt 10.2), und Komponenten, die im Rahmen dieser Arbeit entwickelt wurden und ihrerseits in anderen Softwaresystemen wiederverwendbar sind. Die Auswahl der Komponenten des ersten Typs erfolgte teils aufgrund von Erfahrungen aus anderen OFFIS-Projekten, teils aufgrund von Literatur- und Internetrecherchen. Zu den Komponenten des zweiten Typs gehören der Data Migrator, der Data Linker, der Data Auditor, die Measurement Engines, der Release/Retraction Handler sowie das Database/Repository Interface. Während letztgenannte Komponente ein klassisches objektorientiertes API als ausschließliche Schnittstelle aufweist, verfügen die übrigen Komponenten zusätzlich über eine datenbankorientierte Schnittstelle, die durch das in [Ade01] dokumentierte relationale Datenschema sowie das Relational Database Schema Model des OIM (siehe auch Abschnitt 10.2.2) spezifiziert ist. Die (Fach- bzw. Meta-)Datenbank dient den Komponenten dabei sowohl als Datenquelle (lesende Zugriffe, Parametrierung) als auch als Datensenke (schreibende Zugriffe, Speichern von Ergebnissen). DQM Controller und CLIQWorkbench Console werden nicht als Komponenten im Sinne der o. a. Definition angesehen, da sie quasi das Front End der Applikation darstellen (Steuerungsmodul und graphische Benutzungsoberfläche).

Die neu entwickelten Komponenten der CLIQWorkbench basieren nicht auf einem speziellen Komponentenmodell wie COM [EE98] oder Enterprise JavaBeans [Mon00] (für einen Überblick über gängige Komponentenmodelle siehe [TR00]), nutzen aber ihrerseits COM-Komponenten (konkret: Microsoft DTS und Repository). Die Kopplung zwischen den einzelnen Komponenten wurde minimiert, um ihre Wiederverwendung in anderen Softwaresystemen zu ermöglichen. Zwischen den Komponenten bestehen ausschließlich „uses“-Beziehungen und keine Vererbungsbeziehungen, so dass das sog. *Fragile Base Class Problem* [Szy98] („Kann eine Klasse weiterentwickelt werden, ohne dass unabhängig entwickelte Subklassen davon beeinflusst und so ungewollte Abhängigkeiten zwischen Komponenten eingeführt werden?“) vermieden wird.

Die interne Kohäsion (Grad der Bindung zwischen den Elementen einer Komponente) ist im Data Migrator, im Data Auditor und im Release/Retraction Handler sehr hoch. Diese Komponenten haben einen in sich abgeschlossenen, kaum weiter zerlegbaren Aufgabenbereich. Data Linker und Measurement Engines hingegen sind weniger kohäsiv. Sie könnten ggf. noch in feinere Bausteine zerlegt werden, z. B. in separate Komponenten für schlüsselbasiertes und probabilistisches Record Linkage bzw. für SPC, Konsistenzprüfungen und sonstige Qualitätsmessungen.

Da die Komponenten metadatengestützt sind, lassen sie sich flexibel für unterschiedliche Domänen konfigurieren. Dem Leitsatz „Maximizing reuse minimizes use“ [Szy98] folgend, sind die Einsatzbereiche der Komponenten allerdings eingeschränkt. Im Folgenden wird kurz skizziert, welche Einsatzbereiche jeweils in Frage kommen:

- *Data Migrator:* Anwendungen, in denen ein Mapping zwischen unterschiedlichen Datenstrukturen erforderlich ist, z. B. im Rahmen einer Enterprise Application Integration (EAI)

- *Data Linker:* Anwendungen zum Abgleich sich inhaltlich überlappender Datenbestände, z. B. in Krankheitsregistern
- *Data Auditor:* Anwendungen zur Analyse gegebener Datenbestände im Hinblick auf Auffälligkeiten, z. B. im Rahmen von Reengineering-Maßnahmen
- *Measurement Engines:* Anwendungen zur Bestimmung allgemeiner Charakteristika von Datenbeständen, z. B. im Rahmen von Reengineering-Maßnahmen
- *Release/Retraction Handler:* Beliebige Anwendungen, in denen Daten von Datenbank zu Datenbank verschoben werden müssen
- *Database/Repository Interface:* Beliebige datenbank- bzw. repositorybasierte Anwendungen.

Wie in [Has02] beschrieben, eignen sich für die Kommunikation zwischen Komponenten häufig ereignisgesteuerte Lösungen, da diese mit einer losen Kopplung zwischen Komponenten auskommen. Dabei legen „Erzeugerobjekte“ Informationen (Ereignisse) auf einen von allen angebundenen Komponenten gemeinsam genutzten *Ereigniskanal.* „Verbraucherobjekte“ müssen entweder bei Vorliegen neuer, für sie relevanter Ereignisse benachrichtigt werden (*Push-Modell*) oder den Kanal regelmäßig abfragen (*Pull-Modell*), um in geeigneter Weise reagieren zu können. In der CLIQWorkbench gibt es keinen solchen Ereigniskanal. Stattdessen werden Datenverarbeitungsprozesse zentral durch den DQM Controller gesteuert. Ein gemeinsames Systemframework (mit Diensten wie Persistenz, Verteilung etc.) kommt in der CLIQWorkbench nicht zum Einsatz. Allerdings lässt sich das Database/Repository Interface als Anbieter eines Persistenzdienstes auffassen, der von den einzelnen Komponenten gemeinsam genutzt wird.

Die Wiederverwendung blieb bei der Entwicklung der CLIQWorkbench im Wesentlichen auf die Programmierebene beschränkt (Nutzung von existierenden Funktions- und Klassenbibliotheken bzw. Einbindung von Komponenten über COM und andere Middleware). Eine Wiederverwendung auf Entwurfsebene (Einbeziehung von Entwurfsmustern, Anwendungsframeworks etc., siehe dazu z. B. [Has02]) fand hingegen nicht statt. Auf sehr abstrakter Ebene kann von einer Wiederverwendung der durch die OOD-Methode nach [CY91b] „vorgegebenen“, aus vier Hauptkomponenten (Human Interaction, Task Management, Problem Domain, Data Management) bestehenden Architektur gesprochen werden, wie in Abschnitt 9.2 dargestellt.

Durch die Nutzung bereits existierender Komponenten konnte bei der Entwicklung der CLIQWorkbench auf die Reimplementierung folgender Funktionalität verzichtet werden:

- Zugriff auf die relationale Datenbank (YADI)
- Basisdienste zur Spezifizierung und Ausführung von Datentransformationen (Microsoft DTS)
- Umrechnung von Maßeinheiten (UnitConversion)
- Adressvereinheitlichung und probabilistisches Record Linkage (Vality Integrity)
- Regelbasiertes Schließen (ILOG Rules)
- Erzeugung von Diagrammen und deren Visualisierung (Microsoft Excel und Internet Explorer)
- Erzeugung von XML-Dokumenten (Xerces)
- Basisdienste zur Entwicklung graphischer Benutzungsoberflächen (MFC).

Aus dieser Wiederverwendung resultierte eine erhebliche, allerdings schwer zu quantifizierende Kostenersparnis sowie eine Verbesserung der Stabilität des Gesamtsystems (im Vergleich zu einer vollständigen Neuentwicklung).

Insgesamt gestaltete sich die Integration der einzelnen Komponenten im Rahmen der vorliegenden Arbeit relativ problemlos. Dies war im Wesentlichen dadurch bedingt, dass keine Rücksicht auf a

priori existierende oder sogar zur Entwicklungszeit operative Systeme genommen werden musste, wie es häufig bei der Integration von Legacy-Systemen oder Enterprise-Resource-Planning–Systemen der Fall ist.

Als Fazit lässt sich festhalten, dass die CLIQWorkbench durchaus wesentliche Merkmale eines komponentenbasierten Softwaresystems aufweist, auch wenn dies beim Systementwurf nicht explizit im Vordergrund stand. Für den Praxiseinsatz der CLIQWorkbench ist der modulare Aufbau essentiell, da zu erwarten ist, dass in Organisationen, die die Einführung eines DQMS planen, eine Reihe von infrastrukturellen und geschäftsprozessbezogenen Rahmenbedingungen existieren, die einen „off the shelf“-Einsatz der CLIQWorkbench unmöglich machen (vgl. Evaluierungsszenarien in den Abschnitten 12.3 und 12.2). Stattdessen muss angestrebt werden, einzelne Komponenten, die die datenqualitätsbezogenen Kernprobleme einer Organisation betreffen (z. B. Inkonsistenzen oder Duplikate im Datenbestand), gemäß den konkreten Bedürfnissen der Organisation zu konfigurieren. Schrittweise kann dann durch Hinzufügen weiterer Komponenten der CLIQWorkbench ein umfassendes DQMS in der Organisation etabliert werden.

10.4 Umfang der Implementierung

Tabelle 10.1 (siehe folgende Seite) gibt einen Überblick über den Umfang der Implementierung der CLIQWorkbench. Teile des Codes von Dialogklassen der *CLIQWorkbench Console* wurden automatisch generiert.

10.5 Zusammenfassung

In der prototypischen Implementierung der CLIQWorkbench kommen eine Reihe von Fremdkomponenten zum Einsatz. Deren Funktionalität wird durch die neu entwickelten Komponenten gekapselt. Wie im vorliegenden Kapitel gezeigt wurde, lässt sich die CLIQWorkbench (mit Einschränkungen) als komponentenbasiertes Softwaresystem auffassen.

Komponente	**Paket**	**Klassen**	**Codezeilen**
DQM Controller	JobModel	3	786
	JobPlanEngine	8	1.661
	JobExecEngine	38	4.923
	JobAnalyzeEngine[1]	6	1.990
	JobControlDBEngine	9	4.975
	Gesamt	64	14.335
Data Migrator	DtTransform	18	4.886
	DbmExtension	11	1.227
	Gesamt	29	6.113
Measurement Engines	Measurement Engines	17	4.353
	InconsReview (+ DB)	10	2.331
	PerspectiveWall	16	5.203
	Gesamt	43	11.887
Data Linker	KeyRecLink	6	962
	ProbRecLink	9	2.644
	RecLinkReview	8	2.564
	Gesamt	23	6.170
Data Auditor (MEDAS)		21	7.051
Release/Retraction Handler[1]		5	774
CLIQWorkbench Console	ControllerDialogsMFC	8	1.118
	Data Migrator GUI	14	2.345
	Measurement Engines GUI	18	3.702
	RecLinkMFC	9	1.858
	MEDAS GUI	16	4.365
	Hauptprogramm	6	1.871
	Gesamt	71	15.259
Database/Repository Interface	DBInterface	2	381
	MrRepository	11	4.083
	RepInterface	1	309
	Gesamt	14	4.773
Gesamt		**270**	**66.362**

Tabelle 10.1: Umfang der Implementierung der CLIQWorkbench (Stand: Dezember 2001)

[1] Noch nicht vollständig implementiert.

Zur Homepage der Dissertation

Teil III

Evaluierung

Kapitel 11

CLIQ@EKN: Eine Fallstudie

Das vorliegende Kapitel dokumentiert eine Fallstudie, in der die in dieser Arbeit entwickelten Konzepte und deren softwaretechnische Implementierung auf die Anwendungsdomäne des Epidemiologischen Krebsregisters Niedersachsen (EKN) angewandt werden. Ziel dieser Fallstudie ist es, im Sinne eines „Proof of Concepts“ zu zeigen, dass der CLIQ-Ansatz in einer konkreten Anwendungsumgebung tragfähig ist.

In Abschnitt 11.1 wird zunächst eine kurze Einführung in die gewählte Anwendungsdomäne gegeben. Aus einer Betrachtung des gegenwärtig im EKN verfolgten Ansatzes zur Datenintegration wird dabei Bedarf für einen Einsatz des CLIQ-Vorgehensmodells und der CLIQWorkbench im EKN abgeleitet. Abschnitt 11.2 skizziert das Anwendungsszenario der Fallstudie. Deren Durchführung wird in Abschnitt 11.3 beschrieben. In Abschnitt 11.4 erfolgt abschließend eine kritische Bewertung der gewonnenen Ergebnisse.

11.1 Das Epidemiologische Krebsregister Niedersachsen

Bevor in Abschnitt 11.1.2 auf die aktuelle Vorgehensweise zur Datenintegration im EKN eingegangen wird, wird zunächst ein kurzer Überblick über die epidemiologische Krebsregistrierung in Deutschland gegeben. In Abschnitt 11.1.3 werden die im EKN eingesetzten Softwarewerkzeuge im Hinblick auf ihren Bezug zum CLIQ-Ansatz untersucht.

11.1.1 Epidemiologische Krebsregistrierung in Deutschland

Die zentralen Aufgaben eines epidemiologischen Krebsregisters bestehen in der bevölkerungsbezogenen Registrierung von Krebserkrankungen, der Analyse zeitlicher und räumlicher Trends, der Ermittlung von Risikogruppen und -faktoren sowie der Bereitstellung von Daten für Studien [JPM+91]. Die Grundlage für die Erfüllung dieser Aufgaben bildet eine möglichst vollständige und valide Sammlung von Krebsfällen, wobei jeder Krebsfall nur einmal im Register geführt werden darf, auch wenn er von verschiedenen Stellen (Kliniken, Nachsorgeleitstellen, Gesundheitsämter, Pathologen, niedergelassene Ärzte etc.) mehrfach gemeldet wird.

Die Meldung eines Krebsfalles besteht aus *personenidentifizierenden* (Name, Vorname, Geburtsdatum etc.) und *epidemiologischen* Daten (Diagnosezeitpunkt, Tumorcodierung etc.). Bei der Erfassung, Verarbeitung und Übermittlung dieser Daten ist dem Datenschutz, insbesondere dem informationellen Selbstbestimmungsrecht der Patienten, Rechnung zu tragen. Das Gesetz über Krebsregister (KRG) [Deu94] hat die Bundesländer verpflichtet, bis zum 1.1.1999 epidemiologische Krebsregister einzurichten. Einige Konzepte und Verfahren zur Aufbau- und Ablauforganisation epidemiologischer

Krebsregister werden im KRG zwingend vorgeschrieben (Registrierungsmodell, Minimaldatensatz etc.), andere kann jedes Bundesland frei festlegen (Meldemodell, Abrechnungsmodell etc.). Entscheidend ist, dass eine registerübergreifende Auswertbarkeit der Daten gewährleistet sein muss.

Das KRG sieht ein Registrierungsmodell vor, welches auf zwei unabhängigen, institutionell getrennten Stellen basiert:

- *Vertrauensstelle*
  Die Vertrauensstelle sammelt patientenbezogene Krebsmeldungen von Ärzten, teilweise über Nachsorgeleitstellen und klinische Register, überführt sie in ein einheitliches Format und prüft sie auf Vollständigkeit und Konsistenz, wobei Auffälligkeiten zu einer Rückkopplung mit Meldern führen. Weiterhin verschlüsselt die Vertrauensstelle die personenidentifizierenden Daten und übermittelt die Chiffrate anschließend – zusammen mit den im Klartext verbleibenden epidemiologischen Daten – an die Registerstelle. Eine Dechiffrierung von personenidentifizierenden Daten – etwa zur Durchführung von Studien – ist nur nach Erfüllung bestimmter, wohldefinierter Auflagen erlaubt.

- *Registerstelle*
  Die Registerstelle speichert die chiffrierten personenidentifizierenden und die epidemiologischen Meldungsdaten dauerhaft, es sei denn, ein Patient legt Widerspruch gegen die Speicherung seiner Daten ein. Beziehen sich mehrere Meldungen auf denselben Krebsfall, so werden sie zusammengeführt und zu einem epidemiologisch auswertbaren Datensatz verdichtet. In Zweifelsfällen können Unklarheiten – zeitlich befristet – durch Rückfragen an die Vertrauensstelle beseitigt werden. Auf dem konsolidierten, kumulierten Datenbestand führt die Registerstelle explorative Analysen zur Aufdeckung zeitlicher und räumlicher Trends sowie zur Ermittlung von Risikogruppen und -faktoren durch.

11.1.2 Status Quo der Datenintegration im EKN

Für das EKN [ABH+99], in dessen Meldebereich ca. 7,9 Mio. Personen leben (Stand: Dezember 1999), gelten folgende Mengengerüste: Jährlich werden ca. 40.000 Neuerkrankungen erwartet, also etwa 5 Fälle pro 1000 Einwohnern. Das EKN hat – nach Abschluss des bis 2003 andauernden Stufenaufbaus – pro Jahr ca. 60.000 klinische Meldungen und 80.000 Meldungen von Pathologen zu verarbeiten, was einer Mehrfachmeldequote von 3,5 entspricht. Hinzu kommen ca. 87.000 Todesbescheinigungen (darunter auch Nicht-Krebsfälle).

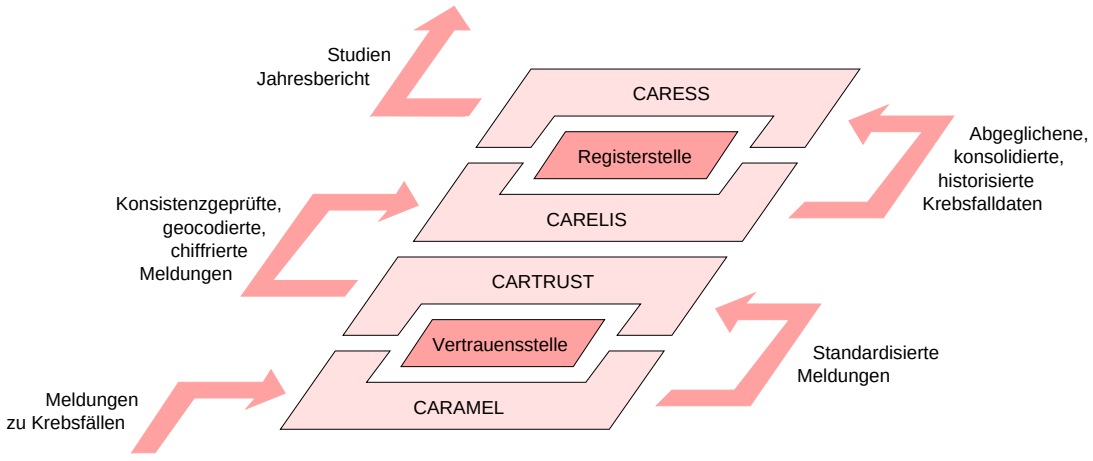


Abbildung 11.1: Der CARTools-Werkzeugkasten

Aufgrund der genannten Datenvolumina und der Komplexität der Datenverarbeitungsprozesse benötigt das EKN eine durchgängige Unterstützung seiner Dokumentations-, Kommunikations- und Auswertungsprozesse durch datenbankgestützte Softwarewerkzeuge. Ein wichtiger Aspekt, der werkzeugseitig abgedeckt sein muss, ist die Integration von Daten aus heterogenen Quellen. Im Rahmen des Projekts CARLOS (Cancer Registry Lower-Saxony) [ABH+99] wird in OFFIS seit 1993 an der Entwicklung von Softwarewerkzeugen zur epidemiologischen Krebsregistrierung gearbeitet. Dabei ist ein als CARTools [AFH+97] bezeichneter „Werkzeugkasten“ entstanden, der im EKN eingesetzt wird und aus vier Komponenten besteht (siehe auch Abb. 11.1). Im Folgenden werden die für die vorliegende Arbeit relevanten Features der CARTools skizziert:

- *CARAMEL* dient der Anbindung von heterogenen Meldequellen an die Vertrauensstelle des EKN. Die in elektronischer Form eingehenden Meldungen werden dabei in das Format des EKN-Datenschemas gemäß [RW99] überführt.
- *CARTRUST* führt Konsistenzprüfungen auf den neu in der Vertrauensstelle eingegangenen Meldungen durch und bietet Funktionalität zur interaktiven Nachbearbeitung von Meldungen im Falle von unvollständigen oder inkonsistenten Angaben (auch bei Rückfragen aus der Registerstelle). Des weiteren wird eine Geocodierung von Patientenadressen (Abbildung auf Gemeindekennziffern und Gauß-Krüger-Koordinaten als Vorbereitung für epidemiologische Auswertungen) sowie eine Chiffrierung von personenidentifizierenden Daten vorgenommen. Derartig aufbereitete Meldungen werden schließlich an die Registerstelle transferiert.
- *CARELIS* ist für die Integration von neu in der Registerstelle eingetroffenen Meldungen in den Registerdatenbestand zuständig. Dieser Vorgang setzt sich aus einem probabilistischen Record Linkage (an Hand von „pseudonymisierten“ personenbezogenen Angaben, sog. *Kontrollnummern*) und einem teils automatisierten, teils interaktiven Merging-Prozess zusammen. Dabei werden Meldungen, die sich auf denselben Krebsfall beziehen (dieselbe Person, derselbe Tumor) zu einem epidemiologisch auswertbaren Datensatz zusammengefasst; die Einzelmeldungen, die in das Merging eingehen, bleiben unverändert und werden zu Dokumentationszwecken dauerhaft in der Registerdatenbank gespeichert.
- *CARESS* ermöglicht umfassende Auswertungen der Registerdatenbank zur Unterstützung von Aufgabenbereichen wie Inzidenzmonitoring, Durchführung von Überlebenszeit- und Clusteranalysen auf der Basis eines einheitlichen Raumbezugs, Berichterstellung und Export von Daten an die Forschung, Qualitätssicherung im Register durch spezielle Indikatoren sowie Hypothesenbildung über Risikofaktoren von Krebs durch die vergleichende Einbeziehung raumbezogener und soziodemographischer Hintergrunddaten.

Das EKN eignet sich in besonderer Weise für eine Evaluierung des in dieser Arbeit entwickelten Ansatzes, und zwar aus folgenden Gründen:

- Die Anwendungsdomäne ist hinreichend komplex, aber noch überschaubar.
- Die Aufbau- und Ablauforganisation des EKN ist mit der eines typischen DWS vergleichbar, wie in Abschnitt 11.1.3 gezeigt wird.
- Die Qualität der an das EKN gelieferten Daten ist oft mangelhaft.

11.1.3 Abbildung von CLIQ auf das EKN

Die Datenverarbeitungsprozesse, die im EKN mit Hilfe der CARTools durchgeführt werden, entsprechen denen eines typischen DWS – ausgehend von heterogenen Quelldaten bis hin zu Auswertungen auf einem integrierten Datenbestand. Damit lässt sich grundsätzlich auch die Aufbau- und

Ablauforganisation des CLIQ-Ansatzes auf das EKN abbilden. Abbildung 11.2 zeigt die wesentlichen Assoziationen zwischen dem CLIQ-Vorgehensmodell (vgl. Kapitel 7) – inkl. der involvierten Datenhaltungskomponenten – und den CARTools-Modulen.

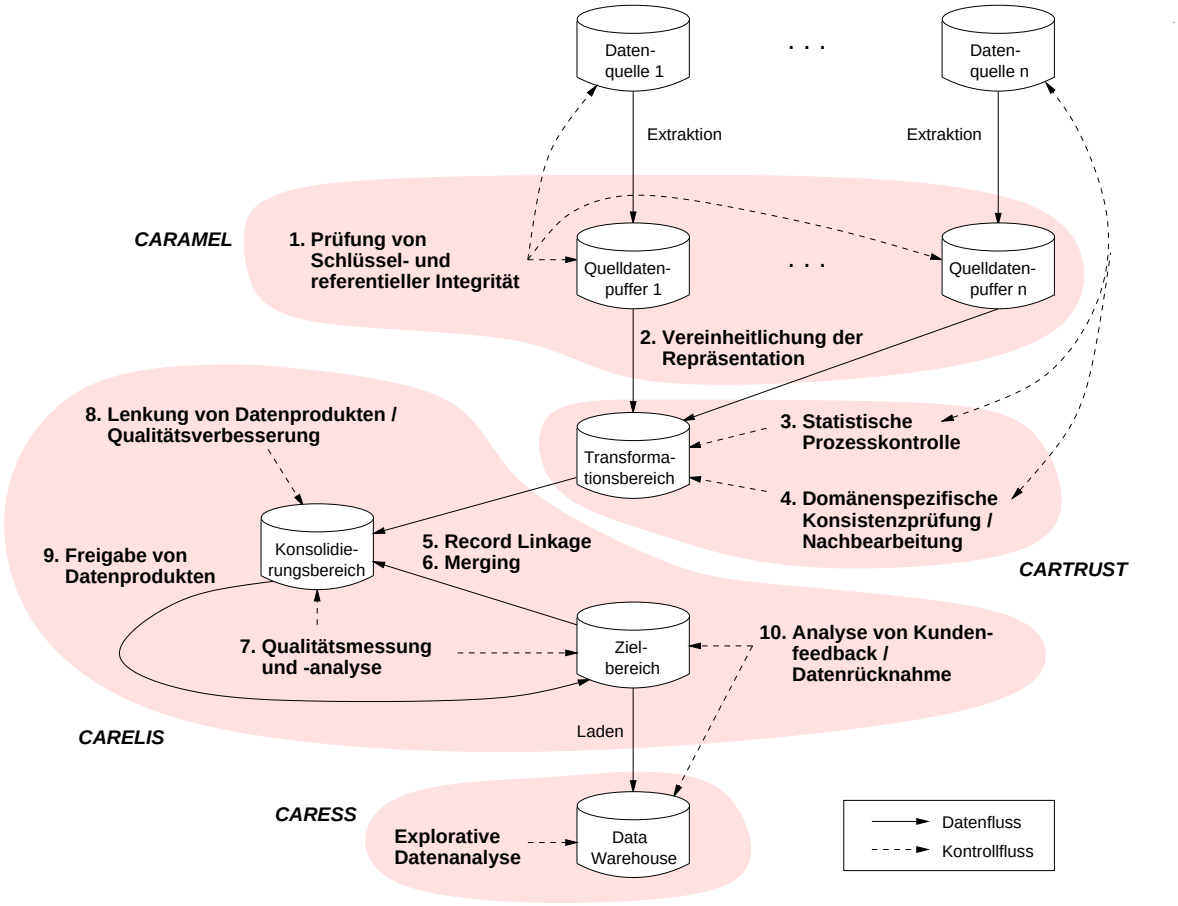


Abbildung 11.2: Assoziationen zwischen CLIQ und den CARTools

Man erkennt, dass durchaus Entsprechungen zwischen den Aufgabenbereichen der einzelnen CARTools und den Phasen des CLIQ-Vorgehensmodells existieren. Allerdings sind die jeweiligen Konzepte naturgemäß nicht kongruent: Einerseits bieten die CARTools domänenspezifische Funktionalität (Geocodierung, Chiffrierung etc.), die im (domänenunabhängigen) CLIQ-Vorgehensmodell nicht betrachtet wird. Andererseits enthält der CLIQ-Ansatz Konzepte zum DQM, die in den CARTools nur ansatzweise oder gar nicht wiederzufinden sind, wie die folgende phasenorientierte Gegenüberstellung zeigt:

1. *Prüfung von Schlüssel- und referentieller Integrität:* Derartige Prüfungen werden von den CARTools nicht unterstützt. Dies hat zur Folge, dass Verletzungen der Schlüsselintegrität und referentiellen Integrität in Quelldatenpaketen oft erst in späteren Verarbeitungsschritten auffallen, indem sie zum Abbruch der Verarbeitung führen. Nach der händischen Beseitigung einer Integritätsverletzung muss der abgebrochene Verarbeitungsschritt nach dem „Trial and Error“-Prinzip von Neuem angestoßen werden. Die Effizienz der Datenintegration kann so bei unzuverlässigen Datenquellen erheblich beeinträchtigt werden.

2. *Vereinheitlichung der Repräsentation:* Diese Aufgabe wird von CARAMEL durchgeführt. Allerdings sind die Transformationsspezifikationen – zumindest in der zum Zeitpunkt der Evaluierung vorliegenden Version – relativ unflexibel (mehr oder weniger hart codiert), was die Anbindung neuer Meldequellen und die Anpassung an geänderte Meldeformate erschwert.

3. *Statistische Prozesskontrolle:* Die CARTools unterstützen bisher keine SPC-Maßnahmen. Stattdessen müssen neu eingegangene Datenpakete durch manuelle Begutachtung durch einen Sachbearbeiter auf grobe Auffälligkeiten (Beispiel: alle Werte eines Attributs sind NULL) untersucht werden. Qualitätsmängel, die hier übersehen werden, fallen oft erst nach abgeschlossener Datenintegration auf, im ungünstigsten Fall erst im Rahmen von Auswertungen des Registerbestandes.

4. *Domänenspezifische Konsistenzprüfung/Nachbearbeitung:* Hier bieten die CARTools umfassende Unterstützung. Formalisierte Geschäftsregeln werden mittels ILOG Rules (siehe auch Abschnitt 10.2.6) auf ihre Einhaltung geprüft. Gefundene Inkonsistenzen können in einer komfortablen (allerdings wiederum hart codierten) Benutzungsoberfläche interaktiv begutachtet und korrigiert werden.

5. *Record Linkage:* Ein schlüsselbasiertes Record Linkage ist im EKN nicht möglich, da inkrementelle Updates nicht als solche gekennzeichnet sind. Stattdessen weist jede eingehende Meldung eine eigene Kennung auf und wird damit zunächst als neuer Krebsfall betrachtet. Im Rahmen eines probabilistischen Record Linkage (unter Nutzung des Werkzeugs Vality Integrity, vgl. Abschnitt 10.2.4) wird dann überprüft, ob der in der Meldung beschriebene Krebsfall bereits in der Registerdatenbank gespeichert ist oder nicht.

6. *Merging:* Für die Zusammenführung von Meldungsdaten bietet das Werkzeug CARELIS eine komfortable graphische Benutzungsoberfläche. Die Zusammenstellung der „besten“ (d. h. der validesten oder spezifischsten) Angaben aus den eingehenden Meldungen erfolgt teilautomatisiert mit Hilfe eines auf ILOG Rules (siehe Abschnitt 10.2.6) basierenden Regelwerks, das Domänenwissen, z. B. über den Informationsgehalt von Angaben oder über attributspezifische Quellenprioritäten, repräsentiert.

7. *Qualitätsmessung und -analyse:* Eine explizite Spezifizierung von Qualitätsanforderungen an die bereitzustellenden Daten ist in den CARTools nicht vorgesehen. Gleiches gilt für die Messung der Datenqualität bzgl. allgemeiner Qualitätsmerkmale. CARESS bietet allerdings Funktionalität zur Berechnung domänenspezifischer Qualitätsindikatoren (vgl. Indikatorpläne in Abschnitt 7.9.2) wie DCO-Rate oder M/I-Index. Diese findet allerdings im Gegensatz zum CLIQ-Vorgehensmodell erst nach der Datenfreigabe statt, quasi als Datenauswertung speziellen Typs.

8. *Lenkung von Datenprodukten:* Eine systematische Qualitätslenkung gibt es im EKN nicht. Weisen die Werte von Qualitätsindikatoren auf Qualitätsmängel hin, so liegt es im Ermessen der zuständigen Sachbearbeiter, wie mit den mangelhaften Daten zu verfahren ist.

9. *Freigabe von Datenprodukten:* Im EKN gelten Daten als freigegeben, sobald sie den Merging-Schritt passiert haben. Sie können dann in die auswertungsorientierten Datenstrukturen geladen werden.

10. *Auswertung von Kundenfeedback/Datenrücknahme:* Diese Aspekte werden im EKN nicht softwaretechnisch unterstützt. Wiederum liegt es im Ermessen der zuständigen Sachbearbeiter, aus dem Feedback von Datenauswertern Maßnahmen für eine Qualitätsverbesserung abzuleiten.

Die in Abschnitt 8.2.1 identifizierten Rollen und Anwendungsfälle (siehe insb. Abb. 8.1) lassen sich ebenfalls auf das EKN abbilden:

- Die Rolle der *Organisationsleitung* wird im EKN vom niedersächsischen Ministerium für Frauen, Arbeit und Soziales (MFAS) sowie von Ärzten in Vertrauens- und Registerstelle übernommen. Diese ermitteln in Zusammenarbeit mit den Kunden des EKN (siehe unten) Anforderungen an Daten(-produkte) (Anwendungsfall 1), bestimmen, welche Datenquellen mit welcher

Priorität an das EKN anzubinden sind (Anwendungsfall 2), nehmen Kundenfeedback entgegen und leiten bei Bedarf Optimierungsmaßnahmen ein (Anwendungsfälle 5 und 6).

- *Kunden* des EKN im engeren Sinne sind die Epidemiologen der Registerstelle sowie Forscher und Forschungsinstitute, die epidemiologische Studien auftragsweise durchführen. An sie werden integrierte Daten, die möglichst den von ihnen gestellten Anforderungen (Anwendungsfall 1) entsprechen sollten, zum Zwecke der Auswertung ausgeliefert (Anwendungsfall 4). Mangelhafte, d. h. nicht auswertbare Daten(-produkte) können zurückgegeben werden (Anwendungsfall 7). Im weiteren Sinne ist auch die Bevölkerung Niedersachsens, die letztlich von den Ergebnissen der Auswertung von EKN-Daten profitieren soll, als Kunde zu sehen.

- *Datenquellen (Lieferanten)* des EKN sind z. B. Kliniken, Nachsorgeleitstellen, Gesundheitsämter, Pathologen und niedergelassene Ärzte. Sie werden im Rahmen eines Auswahlprozesses (Anwendungsfall 2) an das EKN angebunden und beginnen anschließend mit der Lieferung von Rohdaten. Im Rahmen des Datenintegrationsprozesses ist bei Bedarf eine Kommunikation zwischen Datenquellen und Systemnutzern (siehe unten) in Form von Rückfragen und/oder Feedback erforderlich (Anwendungsfall 4).

- Die Rolle des *Systemnutzers* nehmen im EKN medizinische Dokumentare in Vertrauens- und Registerstelle wahr. Sie besitzen nicht nur domänenspezifische (d. h. medizinische), sondern auch technische Kenntnisse und sind für den gesamten Prozess der Datenintegration verantwortlich, von der Anbindung von Datenquellen (Anwendungsfall 2) über die Konfigurierung, Ausführung und Lenkung von Integrationsprozessen (Anwendungsfälle 3, 4 und 5) bis hin zu der Entgegennahme von Kundenfeedback (Anwendungsfall 6) und der Rücknahme von mangelhaften Daten(-produkten) zum Zwecke der nachträglichen Qualitätsverbesserung (Anwendungsfall 7).

Die obigen Gegenüberstellungen deuten darauf hin, dass die Aufbau- und Ablauforganisation des EKN für eine Umsetzung des CLIQ-Vorgehensmodells bzw. einen Einsatz der CLIQWorkbench geeignet ist. Bevor in Abschnitt 11.3 die Integration typischer Datenpakete mit Meldungen zu Krebsfällen in die Registerdatenbank des EKN mit Hilfe der CLIQWorkbench demonstriert wird, soll zunächst das dabei betrachtete Anwendungsszenario skizziert werden.

11.2 Das Anwendungsszenario der Fallstudie

Der Fallstudie liegen – in Abweichung von den realen Gegebenheiten im EKN (siehe [RW99]) – folgende Annahmen zu Grunde:

- Pro Meldungstupel wird genau ein Tumor gemeldet.

- Es gibt keine sog. „Best Of“-Relationen (vgl. [RW99]), in die verdichtete Daten geschrieben werden. Stattdessen findet die Verdichtung direkt auf Meldungsebene statt.

- Von für die Fallstudie irrelevanten Details wird abstrahiert (z. B. verschiedene Meldemodi, Anamnesen etc.).

Als Zielrelation wird eine Relation *MELDUNGEN* definiert, die Angaben zu einem Tumor enthält, ergänzt um einige Angaben zum Patienten. *MELDUNGEN* stellt damit ein „Konglomerat“ aus verschiedenen Relationen des (für den Umfang der Fallstudie zu komplexen) EKN-Registerstellenschemas [RW99] dar. Tabelle 11.1 zeigt das Relationenschema von *MELDUNGEN* (bzw. *MELDUNGEN_TB*,

MELDUNGEN_KB und *MELDUNGEN_ZB* gemäß der in Abschnitt 9.11.2 eingeführten Postfixnotation). Die im EKN nicht betrachteten Attribute *koerpergroesse* und *koerpergewicht* werden aufgenommen (und mit Zufallswerten belegt), um die Umrechnung von Maßeinheiten demonstrieren zu können.

Attributname	**Datentyp**	**Beschreibung**
glb_id	NUMBER(10)	Global eindeutiger Primärschlüssel (Surrogat)
geschlecht	NUMBER(1)	Geschlecht des Patienten (1 = männlich, 2 = weiblich)
geburtsjahr	NUMBER(4)	Geburtsjahr des Patienten (vierstellig)
wohnort	NUMBER(8)	Wohnort des Patienten als Gemeindekennziffer
diagnose_monat	NUMBER(2)	Monat (1-12) der Diagnose
diagnose_jahr	NUMBER(4)	Jahr der Diagnose (vierstellig)
diagnose	CHAR(6)	Diagnosecode nach ICD [Deu95]
diagnose_version	NUMBER(2)	ICD-Version zum Diagnosecode
histologie	NUMBER(4)	Histologiecode nach ICD-O [Wor90]
histologie_version	CHAR(2)	ICD-O–Version zum Histologiecode
lokalisation	CHAR(6)	Lokalisationscode nach dem Tumorlokalisationsschlüssel zur ICD-O [Wor90]
lokalisation_version	NUMBER(2)	Version des Tumorlokalisationsschlüssels
dignitaet	NUMBER(1)	Malignitätsgrad nach ICD-O [Wor90]
seite	NUMBER(1)	Körperseite des Tumorsitzes (rechts, links, beidseits, trifft nicht zu, fehlende Angabe/unbekannt), codiert
diagnosesicherung	NUMBER(1)	Beste Sicherung der Diagnose (klinisch, zytologisch, histologisch etc.), codiert
ausbreitung	NUMBER(1)	Ausbreitung des Tumors, codiert
grading	NUMBER(2)	Differenzierungsgrad des Tumors, codiert
t	CHAR(2)	Tumorausbreitung nach TNM [Int97]
ct	NUMBER(1)	C-Faktor (ähnlich Diagnosesicherung) der Tumorausbreitung nach TNM
n	CHAR(1)	Ausbreitung in die Lymphknoten nach TNM
m	CHAR(1)	Vorliegen von Fernmetastasen nach TNM
tnm_art	NUMBER(1)	Art des TNM (z. B. y-pTNM, cTNM, pTNM), codiert
tnm_version	NUMBER(2)	Version und Auflage des TNM-Schlüssels (z. B. 41 für 4. Version, 1. Auflage)
koerpergroesse	NUMBER(3)	Körpergröße in cm
koerpergewicht	NUMBER(3)	Körpergewicht in kg

Tabelle 11.1: Relationenschema von *MELDUNGEN*

Zur Relation *MELDUNGEN* werden die in Tab. 11.2 aufgeführten attributbezogenen Metadaten (Klasse *DQMAttributeDescriptor*, siehe Abschnitt 9.5.2, plus „optimale“ Stelligkeit nach Abschnitt 6.1.5) spezifiziert. Bei allen Attributen mit Ausnahme von *glb_id* sind Nullwerte erlaubt. Als Datendeskriptoren (Klasse *DQMDataDescriptor*, siehe Abschnitt 9.5.2) für die Zeitnähe werden die Angaben zum Diagnosedatum gewählt.

In der Fallstudie werden vereinfachend nur zwei Quelldatenpuffer betrachtet, die zwei Meldequellen entsprechen. Weitere Quelldatenpuffer bzw. Meldequellen können analog angebunden werden. Beide Puffer speichern Daten, die für die Zielrelation *MELDUNGEN_ZB* bestimmt sind. Quelldaten seien dabei in den Relationen *MELDUNGEN1_QDP* und *MELDUNGEN2_QDP* abgelegt.

Attributname	Nullwert	„Optimale“ Stelligkeit	Update-Häufigkeit pro Tag	Maßeinheit	Verhalten bei Update
geschlecht	9	1	0	–	Ü[1]
geburtsjahr	0	1	0	–	Ü
wohnort	0	4[2]	0.00026[3]	–	H
diagnose_monat	0	1	0.0052	–	H
diagnose_jahr	0	1	0.0052	–	H
diagnose	D48.9[4]	3	0.0052	–	H
diagnose_version	0	1	0.0052	–	H
histologie	8000[5]	1	0.0052	–	H
histologie_version	0	1	0.0052	–	H
lokalisation	C80.9[5]	3	0.0052	–	H
lokalisation_version	0	1	0.0052	–	H
dignitaet	9	1	0.001	–	H
seite	9	1	0.0052	–	H
diagnosesicherung	9	1	0.0052	–	Ü
ausbreitung	9	1	0.0052	–	H
grading	9	1	0.0052	–	H
t	X	1	0.0052	–	H
ct	9	1	0.0052	–	Ü
n	X	1	0.0052	–	H
m	X	1	0.0052	–	H
tnm_art	9	1	0.0052	–	H
tnm_version	0	1	0.0052	–	H
koerpergroesse	0	1	0[6]	cm	Ü
koerpergewicht	0	1	0.0027	kg	H

Tabelle 11.2: Attributbezogene Metadaten zu *MELDUNGEN*

Die Schemata dieser Relationen stimmen bis auf folgende Ausnahmen mit dem Schema von *MELDUNGEN_ZB* bzw. *MELDUNGEN* (siehe Tab. 11.1) überein:

- Beide Relationen haben ein Primärschlüsselattribut *local_id* an Stelle von *glb_id*.
- Beide Relationen weisen ein Attribut *meldung_id* (Datentyp NUMBER(10)) auf, welches in der jeweiligen Datenquelle als (proprietäres) Primärschlüsselattribut genutzt wird.
- Beide Relationen enthalten die (Meta-)Attribute *hist_state* und *update_date* gemäß den Ausführungen in Abschnitt 9.11.1.
- Das Attribut *geschlecht* sei in *MELDUNGEN1_QDP* als ’M’ und ’W’ codiert (Datentyp CHAR(1), Nullwert ’X’).

[1] ’Ü’ = Überschreiben, ’H’ = Historisieren (vgl. Abschnitt 9.11.1). Updates auf Attributen mit Änderungshäufigkeit 0 werden als Korrektur einer vormals mangelhaften Angabe gewertet.

[2] Gemeindekennziffern sind maximal 8-stellig: 1. und 2. Stelle Bundesland, 3. Stelle Regierungsbezirk, 4. und 5. Stelle Landkreis, 6. bis 8. Stelle (Samt-)Gemeinde. Daraus ergibt sich eine maximale Stelligkeit von 4, die in diesem Kontext gleichzeitig die „optimale“ Stelligkeit darstelle.

[3] Entspricht etwa einem Update bzw. Umzug in 10 Jahren.

[4] Nach ICD 10 [Deu95].

[5] Nach ICD-O, 2. Auflage [Wor90].

[6] Hier wird davon ausgegangen, dass alle Patienten erwachsen sind.

- Das Diagnosedatum sei in *MELDUNGEN1_QDP* als Attribut *diagnose_datum* vom Datentyp DATE repräsentiert (an Stelle von *diagnose_monat* und *diagnose_jahr*).
- Die Werte im Attribut *koerpergroesse* seien in *MELDUNGEN1_QDP* in *inch* statt in *cm* angegeben.

Aufgrund der genannten Annahmen sowie aus organisatorischen und datenschutzrechtlichen Gründen wird die Fallstudie nicht direkt auf der Registerdatenbank des EKN, sondern auf einer separaten (Test-)Datenbank durchgeführt.

Im Rahmen der Fallstudie sind zwei Datenpakete (eines pro Quelldatenpuffer) in den Zielbereich zu integrieren. Diese Datenpakete wurden auf der Basis realer Datenlieferungen an das EKN zusammengestellt, indem datenschutzrechtlich kritische (insb. personenidentifizierende) Angaben entfernt und Anpassungen gemäß den zu Beginn dieses Abschnitts beschriebenen Annahmen vorgenommen wurden. Zu Beginn der Fallstudie liegen die Quelldaten in den QDP-Relationen *MELDUNGEN1_QDP* und *MELDUNGEN2_QDP* vor (je 200 Tupel); *MELDUNGEN_TB* enthält zu diesem Zeitpunkt 459 bereits integrierte Tupel. Derartig kleine Mengengerüste entsprechen zwar nicht gängigen Praxisbedingungen (der Registerbestand des EKN umfasst derzeit 120.000 Meldungen), sind jedoch für den angestrebten „Proof of Concepts“ ausreichend. Mit Hilfe der Komplexitätsabschätzungen in Abschnitt 7.15 lässt sich prinzipiell eine „Hochrechnung“ der Laufzeit auf größere Datenvolumina vornehmen. In der Studie wird jedoch darauf verzichtet, da insbesondere für die zeitkritischen Phasen 5 (Record Linkage) und 7 (Qualitätsmessung und -analyse) des CLIQ-Vorgehensmodells bisher keine hinreichend exakte Größenordnung der asymptotischen Komplexität angegeben werden konnte.

Die Fallstudie wird auf einem Windows2000-PC mit AMD Athlon-Prozessor (1,2 GHz) und 512 MB Hauptspeicher durchgeführt. Als DBMS kommen Oracle 8.1.7 (für Fachdaten) und Microsoft SQLServer 7 (für Metadaten) zum Einsatz; der Zugriff erfolgt jeweils über ein lokales Netzwerk.

Trotz der genannten vereinfachenden Annahmen gewährleistet das beschriebene Szenario, dass das Ziel der Fallstudie, nämlich die Demonstration der prinzipiellen Anwendbarkeit des CLIQ-Vorgehensmodells und der CLIQWorkbench in einer konkreten Anwendungsdomäne, erreicht werden kann.

11.3 Datenintegration im EKN mit Hilfe der CLIQWorkbench

Im Folgenden wird die Durchführung der auf dem im vorigen Abschnitt vorgestellten Anwendungsszenario basierenden Fallstudie dokumentiert. Dabei ist zwischen Konfigurierungsschritten und Integrationsschritten zu unterscheiden. In den *Konfigurierungsschritten* wird der Datenintegrationsprozess spezifiziert. Die *Integrationsschritte* beinhalten die Arbeitsschritte zur Integration von Datenpaketen aus den Quelldatenpuffern in den Zielbereich. Deren Ausgestaltung ergibt sich aus den Ergebnissen der Konfigurierungsschritte.

Die beiden nachfolgenden Abschnitte beschreiben die einzelnen Konfigurierungs- und Integrationsschritte an Hand des Anwendungsszenarios aus Abschnitt 11.2. Dabei werden die Phasen 1 (Prüfung von Schlüssel- und referentieller Integrität) und 10 (Auswertung von Kundenfeedback/Datenrücknahme) des CLIQ-Vorgehensmodells ausgeklammert, da ihre softwaretechnische Umsetzung zum Zeitpunkt der Durchführung der Fallstudie noch nicht abgeschlossen war.

Nach dem Start der CLIQWorkbench erscheint zunächst ein Dialog, in dem sich der Systemnutzer gegenüber der (Fach-)Datenbank sowie dem Metadaten-Repository authentifizieren muss (siehe Abb. 11.3). Diese Authentifizierung ist Voraussetzung für die Durchführung von Konfigurierungs- und Integrationsschritten.

Abbildung 11.3: Authentifizierung gegenüber (Fach-)Datenbank und Repository

11.3.1 Konfigurierungsschritte

Ein Konfigurierungsschritt bezieht sich immer auf eine einzelne Phase des CLIQ-Vorgehensmodells (siehe Kapitel 7). Die Reihenfolge der Ausführung der Konfigurierungsschritte spielt keine Rolle; es müssen jedoch alle Schritte durchlaufen worden sein, bevor Integrationsschritte angestoßen werden können.

Abbildung 11.4 zeigt den Dialog zum Erstellen einer Konfiguration zu einer Integrationsaufgabe. Von diesem kann zu den einzelnen Konfigurierungsschritten verzweigt werden (siehe Menüleiste). Im vorliegenden Fall wird eine erste Konfiguration (laufende Kennung 1) zu einer Integrationsaufgabe (d. h. zu einem Job) angelegt, die sich auf die Relation *MELDUNGEN* bezieht.

Phase 2: Vereinheitlichung der Repräsentation

Da die Schemata der Relationen *MELDUNGEN1_QDP* und *MELDUNGEN2_QDP* von dem Schema der Relation *MELDUNGEN_TB* abweichen, sind entsprechende Abbildungsvorschriften zu spezifizieren, die in Task Sets (vgl. Abschnitt 9.4.2) gruppiert werden. In Abb. 11.5 ist beispielhaft dargestellt, wie eine Abbildungsvorschrift zwischen dem Attribut *diagnose_datum* aus *MELDUNGEN1_QDP* und den Attributen *diagnose_jahr* und *diagnose_monat* aus *MELDUNGEN_TB* festgelegt wird. Die dafür erforderliche Zerlegung von Datumswerten wird bisher prototypisch durch ein ActiveX-Skript realisiert (siehe Abb. 11.6). Die Abbildung von Werten des Attributs *geschlecht* von 'M' und 'W' auf 1 und 2 wird analog spezifiziert.

Zwischen dem Attribut *koerpergroesse* aus *MELDUNGEN1_QDP* und dem gleichnamigen Attribut aus *MELDUNGEN_TB* ist eine Umrechnung von *inch* nach *cm* vorzunehmen. Diese wird in einem separaten Dialog spezifiziert (siehe Abb. 11.7).

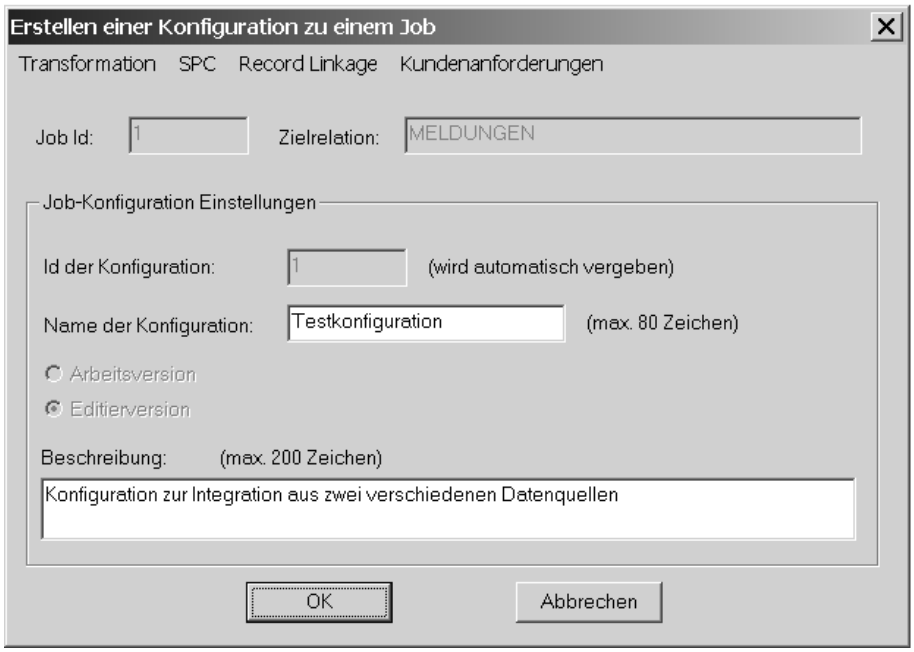


Abbildung 11.4: Erstellen einer Konfiguration zur Integrationsaufgabe

Die Werte aller anderen Attribute von *MELDUNGEN1_QDP* – mit Ausnahme der Metaattribute sowie der lokalen Schlüsselattribute *local_id* und *meldung_id*, deren Werte nicht in den Transformationsbereich zu überführen sind – können aufgrund der jeweils identischen Attributschemata unverändert übernommen werden (Transformationsverfahren „Spalten kopieren“).

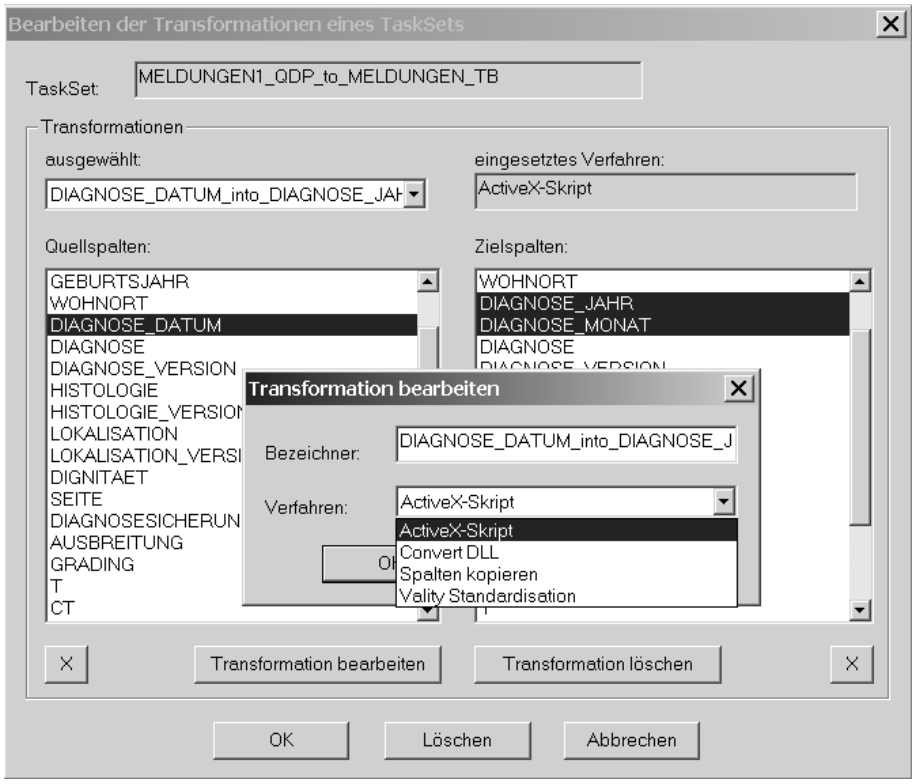


Abbildung 11.5: Auswahl eines Transformationsverfahrens im Data Migrator

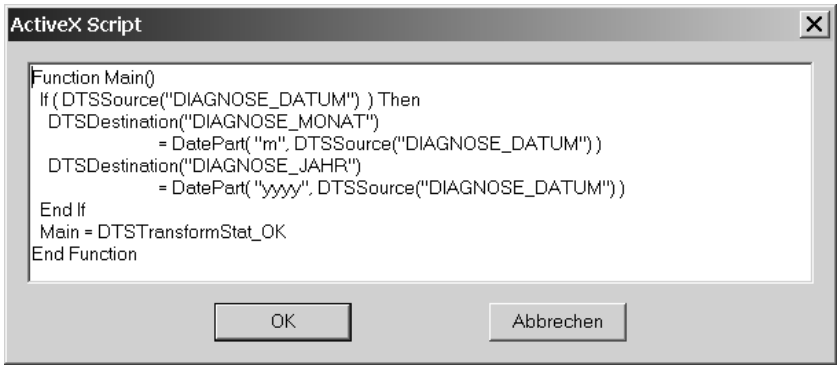


Abbildung 11.6: ActiveX-Skript zur Zerlegung von Datumswerten

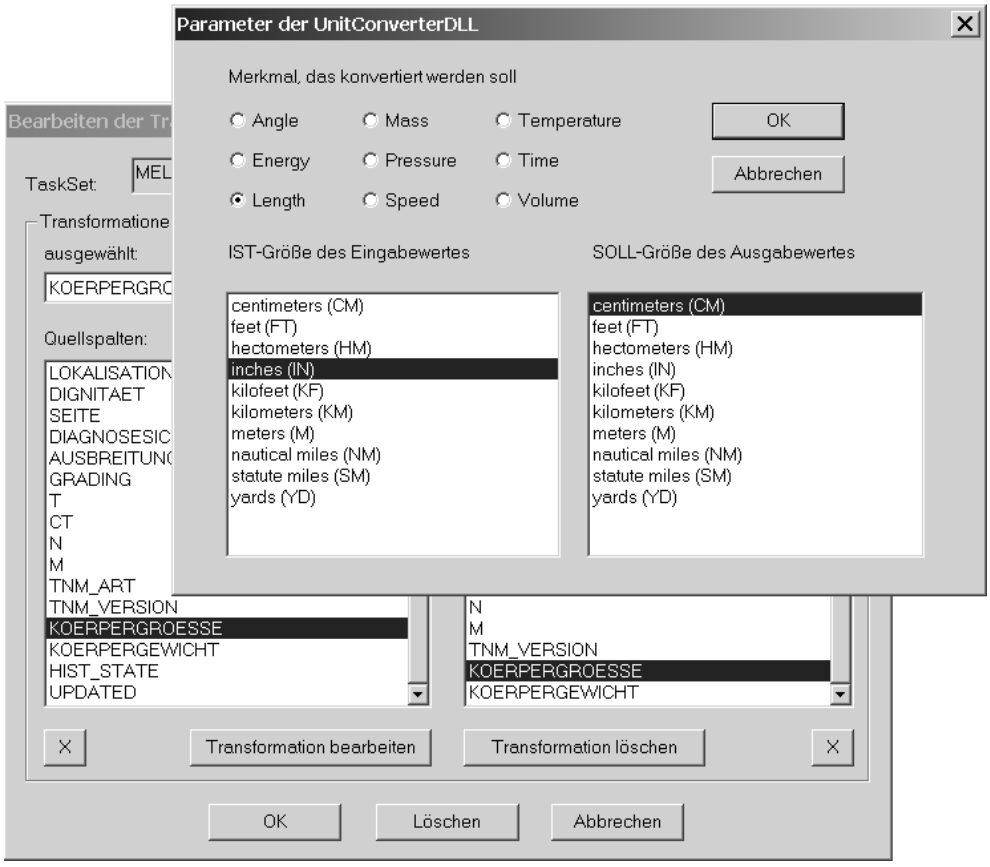


Abbildung 11.7: Festlegung der Umrechnung von Maßeinheiten im Data Migrator

Phase 3: Statistische Prozesskontrolle

Im Rahmen der Fallstudie werden beispielhaft für die Attribute *geschlecht*, *diagnose_monat* und *seite* der Relation *MELDUNGEN_TB* SPC-Pläne spezifiziert, um sie im Zuge von Integrationsprozessen gezielt im Hinblick auf Auffälligkeiten überwachen zu können. Abbildung 11.8 zeigt den zugehörigen Dialog der CLIQWorkbench. Dieser ermöglicht neben der Spezifizierung eines zu betrachtenden Datenobjekts auch die Festlegung von erwarteten statistischen Kennzahlen (Mittelwert, Standardabweichung) und zugehörigen Kontrollgrenzen.

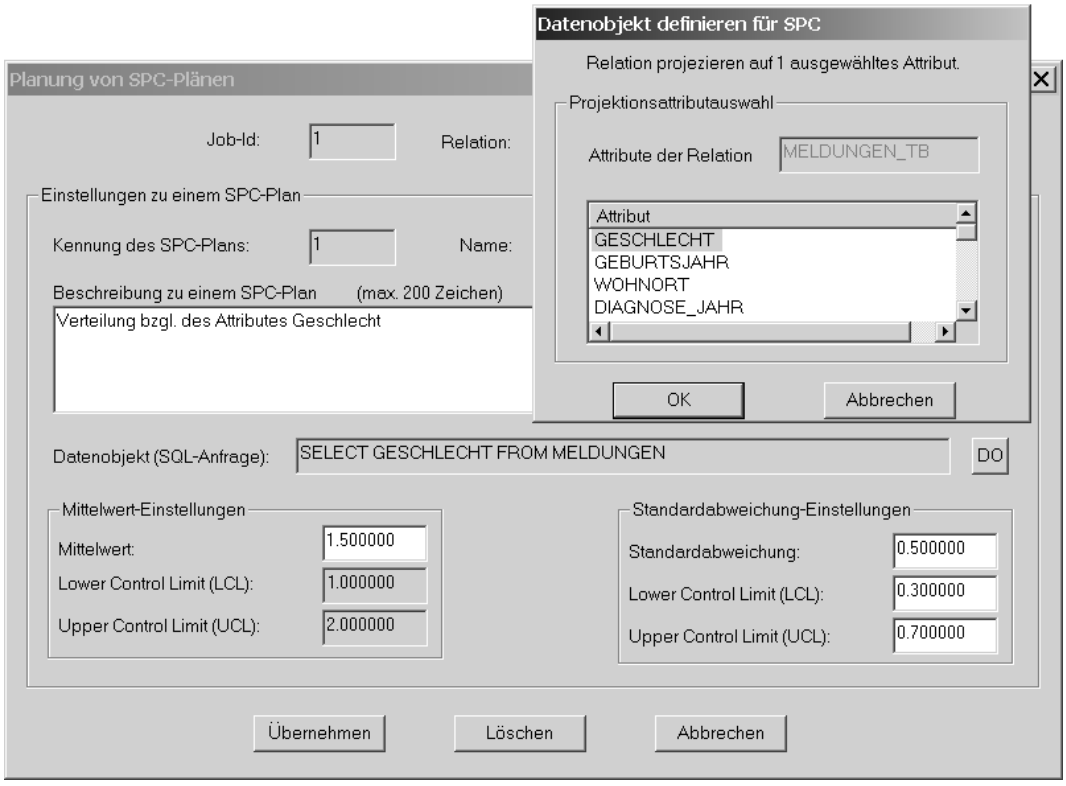


Abbildung 11.8: Planung der Statistischen Prozesskontrolle

Phase 4: Domänenspezifische Konsistenzprüfung/Nachbearbeitung

Die Konfigurierung dieser Phase wird – zumindest in der aktuellen Version des CLIQWorkbench-Prototypen – nicht zur Laufzeit über einen Fensterdialog vorgenommen, sondern erfordert das Hinzubinden eines vorcompilierten ILOG Rules-Regelwerks, wie in den Abschnitten 9.5.2 und 10.2.6 erläutert. Für die Fallstudie wird das im EKN eingesetzte, 37 einzelne Regeln umfassende Regelwerk zur Prüfung der Konsistenz von medizinischen Tumorangaben übernommen. Sechs Regeln, die Attribute betreffen, die in der Fallstudie unberücksichtigt bleiben, werden deaktiviert. Der Aktionsteil der verbleibenden Regeln wird jeweils um die Rückgabe eines Schweregrads sowie einer textuellen Beschreibung erweitert, wie in Abschnitt 7.6 gefordert (siehe auch dortige Beispielregel).

Phase 5: Record Linkage

Der probabilistische Record Linkage-Prozess der in der Fallstudie betrachteten Integrationsaufgabe wird so konfiguriert, dass sowohl Trennschärfe als auch Fehlertoleranz Berücksichtigung finden. Konkret werden über den zuständigen Dialog der CLIQWorkbench zwei Umläufe spezifiziert, die auf unterschiedlichen Block- und Matchvariablen beruhen. Die Spezifikation des ersten Umlaufs ist in Abb. 11.9 dargestellt. Im zweiten Umlauf werden die Attribute *diagnose_monat, diagnose_jahr* und *histologie* als Blockvariablen sowie *wohnort, geschlecht* und *geburtsjahr* als Matchvariablen herangezogen. Diese Attributauswahl basiert auf Erfahrungen aus dem EKN.

Phase 6: Merging

Merging-Prozesse werden in der aktuellen Version des Prototypen interaktiv durchgeführt und erfordern keine gesonderte Konfigurierung.

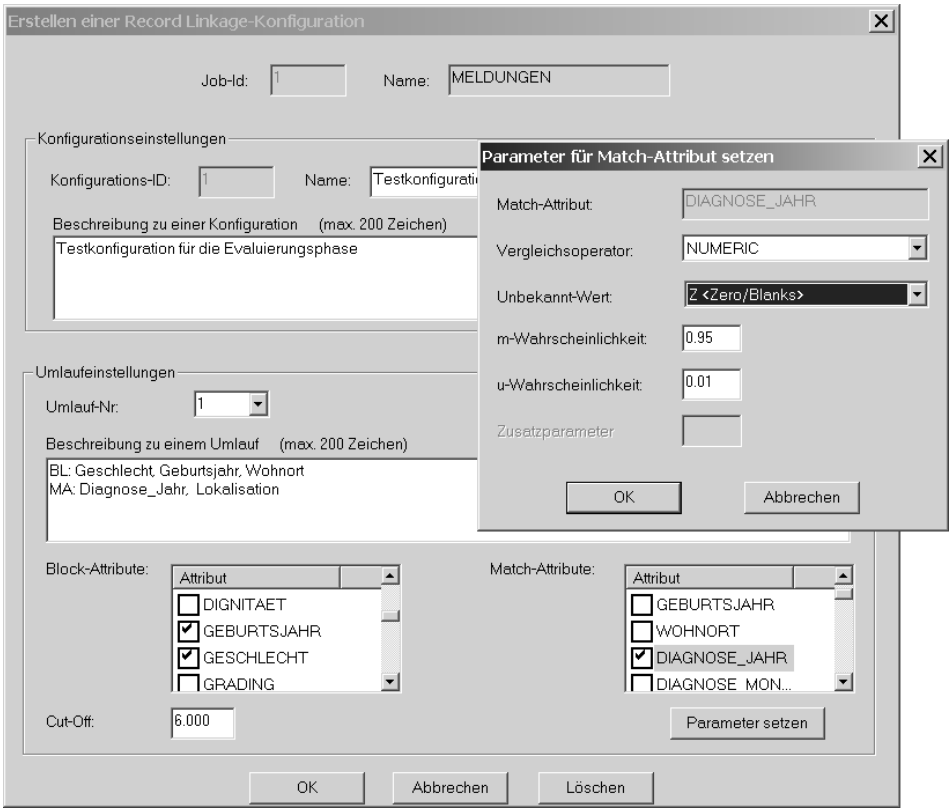


Abbildung 11.9: Spezifizierung eines probabilistischen Record Linkage-Umlaufs

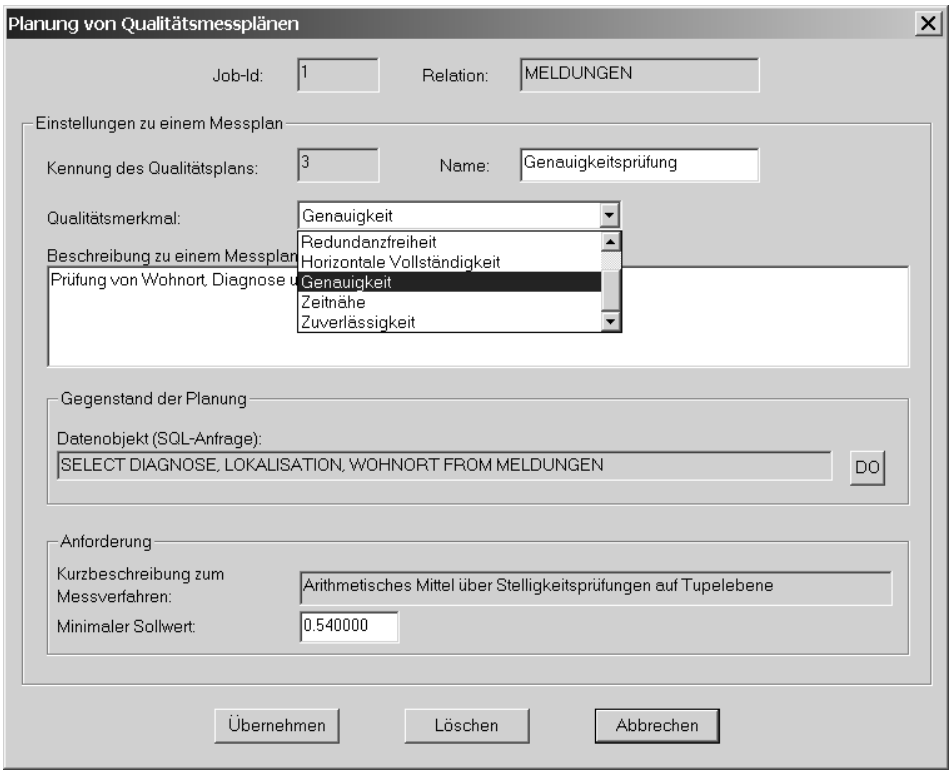


Abbildung 11.10: Spezifizierung eines Qualitätsplans

Phase 7: Qualitätsmessung und -analyse

Im ersten Teil dieses Konfigurierungsschritts werden Anforderungen an Meldungen im Zielbereich hinsichtlich der Qualitätsmerkmale Konsistenz, Redundanzfreiheit, Vollständigkeit und Genauigkeit spezifiziert, partiell eingeschränkt auf einzelne Attribute. Abbildung 11.10 zeigt exemplarisch den Qualitätsplan zum Merkmal Genauigkeit. In diesem Fall wird eine Projektion auf diejenigen Attribute aus *MELDUNGEN* vorgenommen, deren Ausprägungen in unterschiedlicher Stelligkeit auftreten können (vgl. Tab. 11.2), nämlich *diagnose, lokalisation* und *wohnort*. Jedes der ausgewählten Attribute kann mit einer Gewichtung versehen werden, die mit dem Einfluss des Attributs auf die Qualitätsbewertung korreliert. Die Auswahl und Gewichtung von Attributen wird in Abb. 11.11 demonstriert. Die Festlegung der minimalen Sollwerte in den einzelnen Qualitätsplänen erfolgt auf Basis von Erfahrungswerten – soweit verfügbar.

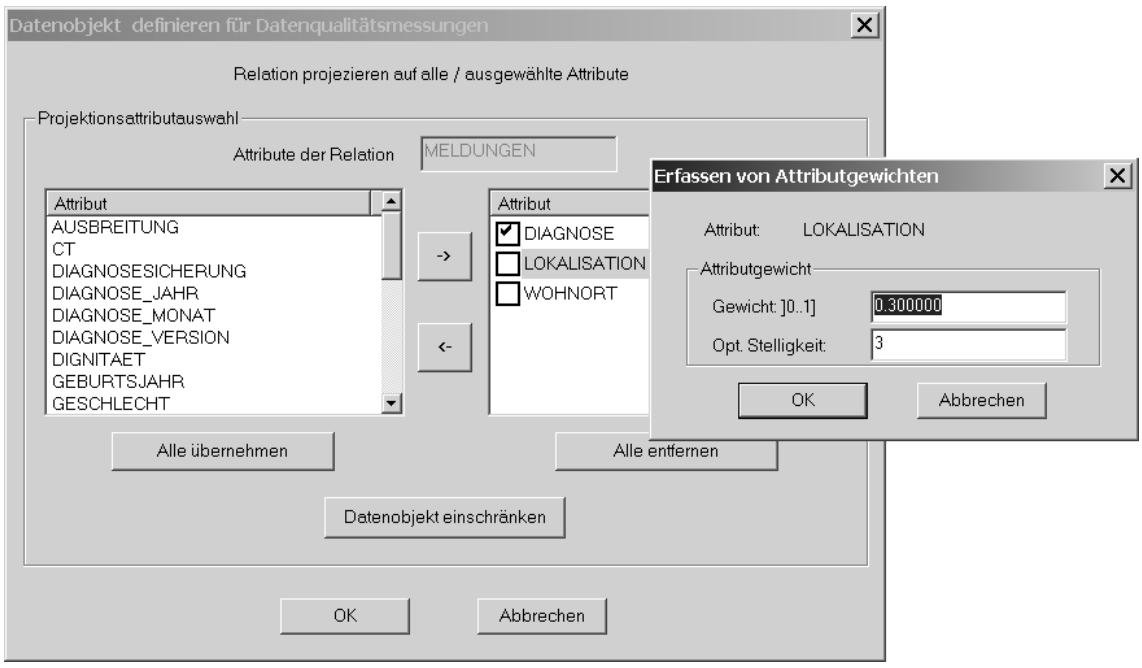


Abbildung 11.11: Spezifizierung eines gewichteten Datenobjekts für einen Qualitätsplan

Der zweite Teil des Konfigurierungsschritts besteht in der Spezifizierung eines domänenspezifischen Indikatorplans gemäß Abschnitt 7.9.2. Als exemplarischer Indikator wird der HV-Anteil (Anteil der Fälle mit histologisch abgesicherter Tumordiagnose) gewählt. Abbildung 11.12 zeigt den zugehörigen Indikatorplan. Dabei sind *HV* und *GES* Variablen, die durch SELECT-Statements definiert sind und mittels arithmetischer Operatoren zu einer Messfunktion verknüpft werden können.

Phase 8: Lenkung von Datenprodukten/Qualitätsverbesserung

Bezüglich Phase 8 des CLIQ-Vorgehensmodells wäre lediglich der Data Auditor zu konfigurieren. Da dieser bereits in [HW00] evaluiert worden ist, wird er aus der Fallstudie ausgeklammert.

Phase 9: Freigabe von Datenprodukten

Die Freigabe von Daten muss nicht explizit konfiguriert werden.

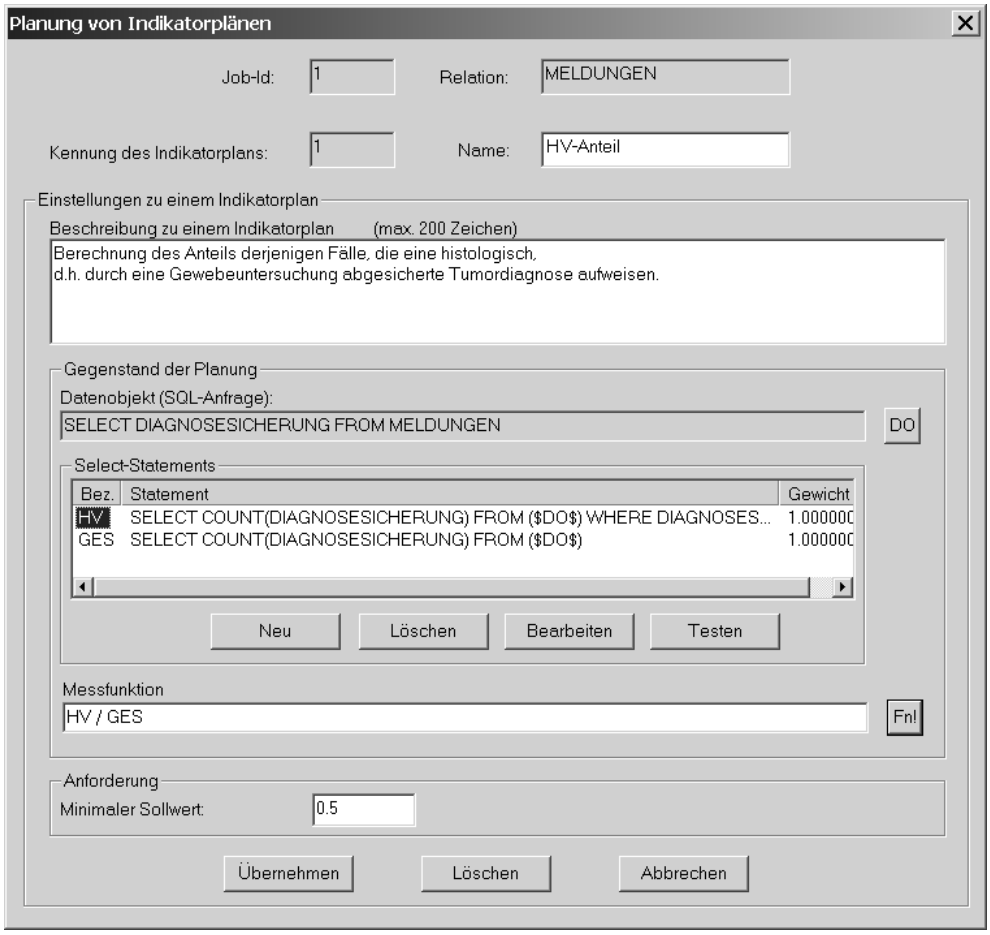


Abbildung 11.12: Spezifizierung eines Indikatorplans

11.3.2 Integrationsschritte

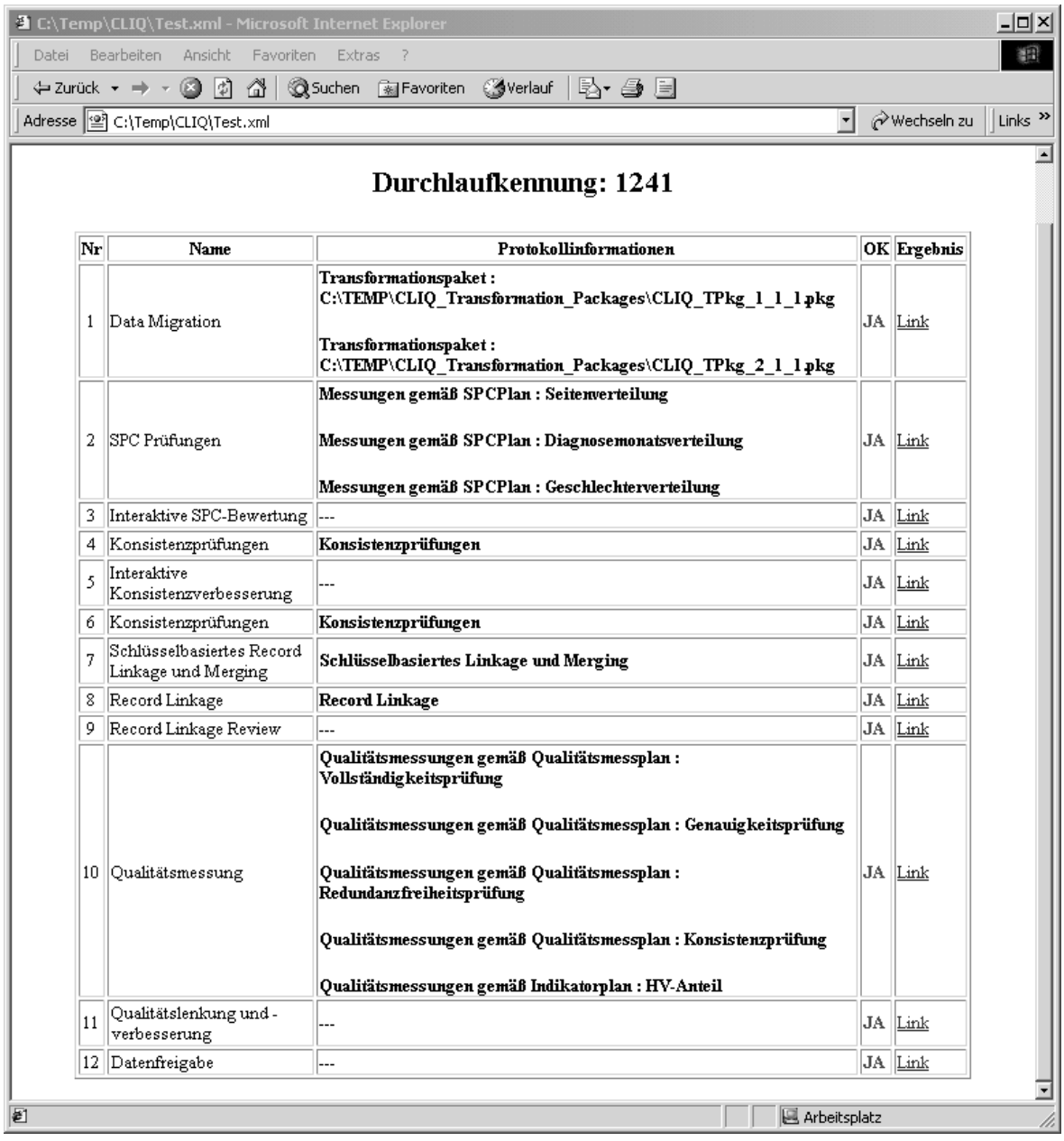
Bei der Durchführung von Integrationsschritten wird das CLIQ-Vorgehensmodell (siehe Kapitel 7) unter Verwendung der in den Konfigurierungsschritten festgelegten Spezifikationen auf die neu zu integrierenden Daten angewandt. Die Datenintegration findet dabei – gesteuert durch den DQM Controller – weitestgehend automatisiert im Hintergrund statt. Ist ein interaktiver Eingriff erforderlich, wird die automatisierte Bearbeitung unterbrochen und der Systemnutzer über die durchzuführenden Maßnahmen informiert.

Zu jedem abgeschlossenen Schritt werden Protokollinformationen in der Datenbank abgelegt, für den Systemnutzer mittels XML und XSL aufbereitet und im Hauptfenster der CLIQWorkbench – in den folgenden Screenshots alternativ im Microsoft Internet Explorer – dargestellt. Abbildung 11.13 zeigt das im Rahmen der Fallstudie generierte Protokoll nach Abschluss der Datenintegration. Auf die Darstellung der einzelnen, sukzessive erweiterten Zwischenprotokolle wird aus Platzgründen verzichtet. Die Hyperlinks in der rechten Spalte ermöglichen eine Navigation zu Subprotokollen einzelner Ausführungsschritte. Diese werden in den folgenden Unterabschnitten im Detail vorgestellt.

Phase 2: Vereinheitlichung der Repräsentation

Alle 400 Tupel aus *MELDUNGEN1_QDP* und *MELDUNGEN2_QDP* können gemäß der spezifizierten Abbildungsvorschriften automatisiert in eine einheitliche Repräsentation überführt werden. Dies

war zu erwarten, da die zur Fallstudie herangezogenen Daten von Meldern stammen, die ihrerseits zumindest sicherstellen, dass vorgegebene Datentypen bzw. Wertebereiche eingehalten werden. Diese Eigenschaft sollte als Mindestanforderung an alle an ein DWS anzubindenden Quellen gestellt werden, ist in der Praxis allerdings nicht immer gegeben.



Durchlaufkennung: 1241

Nr	Name	Protokollinformationen	OK	Ergebnis
1	Data Migration	**Transformationspaket : C:\TEMP\CLIQ_Transformation_Packages\CLIQ_TPkg_1_1_1.pkg** **Transformationspaket : C:\TEMP\CLIQ_Transformation_Packages\CLIQ_TPkg_2_1_1.pkg**	JA	Link
2	SPC Prüfungen	**Messungen gemäß SPCPlan : Seitenverteilung** **Messungen gemäß SPCPlan : Diagnosemonatsverteilung** **Messungen gemäß SPCPlan : Geschlechterverteilung**	JA	Link
3	Interaktive SPC-Bewertung	---	JA	Link
4	Konsistenzprüfungen	**Konsistenzprüfungen**	JA	Link
5	Interaktive Konsistenzverbesserung	---	JA	Link
6	Konsistenzprüfungen	**Konsistenzprüfungen**	JA	Link
7	Schlüsselbasiertes Record Linkage und Merging	**Schlüsselbasiertes Linkage und Merging**	JA	Link
8	Record Linkage	**Record Linkage**	JA	Link
9	Record Linkage Review	---	JA	Link
10	Qualitätsmessung	**Qualitätsmessungen gemäß Qualitätsmessplan : Vollständigkeitsprüfung** **Qualitätsmessungen gemäß Qualitätsmessplan : Genauigkeitsprüfung** **Qualitätsmessungen gemäß Qualitätsmessplan : Redundanzfreiheitsprüfung** **Qualitätsmessungen gemäß Qualitätsmessplan : Konsistenzprüfung** **Qualitätsmessungen gemäß Indikatorplan : HV-Anteil**	JA	Link
11	Qualitätslenkung und -verbesserung	---	JA	Link
12	Datenfreigabe	---	JA	Link

Abbildung 11.13: Überblick über das Protokoll der durchgeführten Integrationsaufgabe

Phase 3: Statistische Prozesskontrolle

Für die 400 neu zu integrierenden Tupel werden gemäß der in Abschnitt 11.3.1 beschriebenen Spezifikation statistische Prozesskontrollen durchgeführt. Exemplarisch ist in Abb. 11.14 derjenige Protokollausschnitt dargestellt, der den X-Chart (oben) sowie den Pareto-Chart (unten) zu den Mittelwerten im Attribut *geschlecht* enthält.

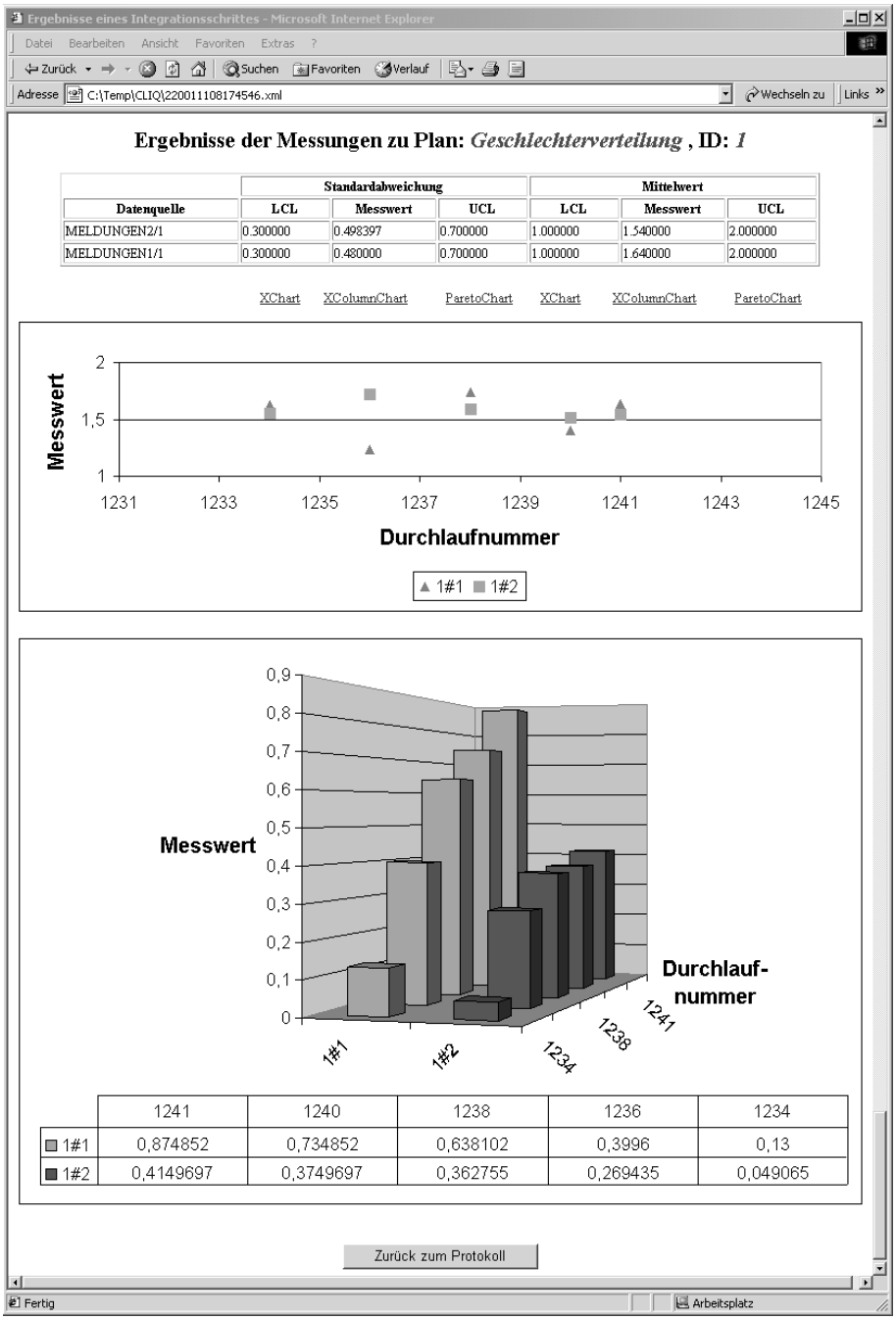


Abbildung 11.14: X-Chart und Pareto-Chart zu den Mittelwerten im Attribut *geschlecht*

Im X-Chart erkennt man, dass die gemessenen Mittelwerte aus den zurückliegenden Integrationsläufen (1234 bis 1241) bei *MELDUNGEN1_QDP* (Dreiecke) um den erwarteten Mittelwert schwanken, während sie bei *MELDUNGEN2_QDP* (Quadrate) eher oberhalb des erwarteten Wertes liegen. In beiden Fällen bewegen sich die Werte innerhalb der spezifizierten Kontrollgrenzen, so dass kein Eingriff in den Prozess erforderlich ist.

Im Pareto-Chart (kumulierte Abweichungen) fällt auf, dass die Abweichungen in *MELDUNGEN1_QDP* durchgängig größer sind als in *MELDUNGEN2_QDP* und dass sich die Messwerte in *MELDUNGEN2_QDP* im Laufe der Zeit (d. h. im Zuge der Integrationsläufe 1234 bis 1241) immer mehr an den erwarteten Mittelwert annähern (geringe Zunahme des kumulierten Wertes).

Aus Platzgründen wird auf die Darstellung der SPC-Diagramme zur Standardabweichung verzichtet. Gleiches gilt für die Diagramme zu den ebenfalls überwachten Attributen *diagnose_monat* und *seite*. Bei diesen werden folgende Beobachtungen gemacht:

- Alle Tupel weisen im Attribut *seite* den Wert 9 („unbekannt“) auf (d. h. Mittelwert 9, Standardabweichung 0), was darauf hindeutet, dass entweder ein Übertragungsfehler vorliegt oder der entsprechende Melder dieses Merkmal nicht erfasst. In der Praxis wäre diese Beobachtung ein Anlass, den Melder zu kontaktieren und ggf. eine Neulieferung der mangelhaften Daten zu veranlassen. Im Rahmen der Fallstudie werden hingegen keine korrigierenden Maßnahmen getroffen.
- Im Attribut *diagnose_monat* werden überdurchschnittlich hohe Mittelwerte (nahe der oberen Kontrollgrenze) und unterdurchschnittlich niedrige Standardabweichungen (nahe der unteren Kontrollgrenze) gemessen. Es stellt sich heraus, dass diese Auffälligkeiten darin begründet sind, dass das der Fallstudie zu Grunde liegende Datenpaket ausschließlich Meldungen aus dem Quartal IV/1999 enthält.

Phase 4: Domänenspezifische Konsistenzprüfung/Nachbearbeitung

Nach Abschluss der statistischen Prozesskontrollen werden die Tupel im Transformationsbereich mit Hilfe des Regelwerks aus dem EKN auf Inkonsistenzen geprüft. Das Ergebnis dieser Untersuchung fasst der in Abb. 11.15 dargestellte Protokollausschnitt zusammen. Die hohe Anzahl an Inkonsistenzen ist darin begründet, dass die Diagnose in allen Tupeln mit ’D48.9’ („unbekannt“) codiert ist (wäre auch über SPC feststellbar gewesen). Hierbei handelt es sich allerdings nicht um einen schwerwiegenden Datenqualitätsmangel, da die Diagnosecodierung aus den Angaben zur Lokalisation, zur Histologie sowie zur Dignität mit einer speziellen Konvertierungssoftware nachträglich berechnet werden kann.

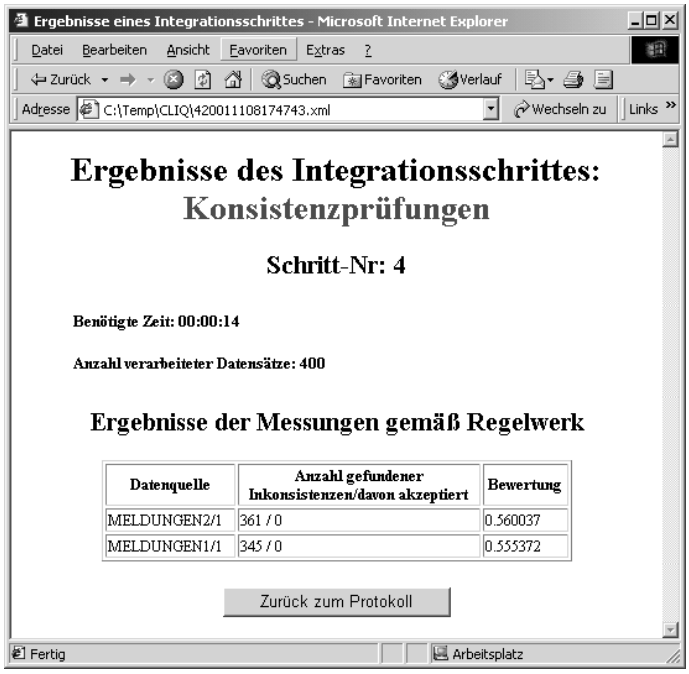


Abbildung 11.15: Überblick über die Ergebnisse der Konsistenzprüfungen

Nach Durchführung dieser Konvertierung und anschließender Wiederholung der Konsistenzprüfungen wird mit der interaktiven Nachbearbeitung der verbliebenen Inkonsistenzen begonnen (siehe

Abb. 11.16). Zu jeder inkonsistenten Meldung werden dabei die automatisiert erzeugten Fehler- und Warnmeldungen angezeigt. Die Meldungen werden von einer Domänenexpertin editiert; vorgenommene Änderungen werden persistent gemacht. Aus Zeitgründen werden allerdings Inkonsistenzen, zu denen keine Fehlermeldung, sondern lediglich eine Warnung (geringer Schweregrad) generiert worden ist, nicht nachbearbeitet.

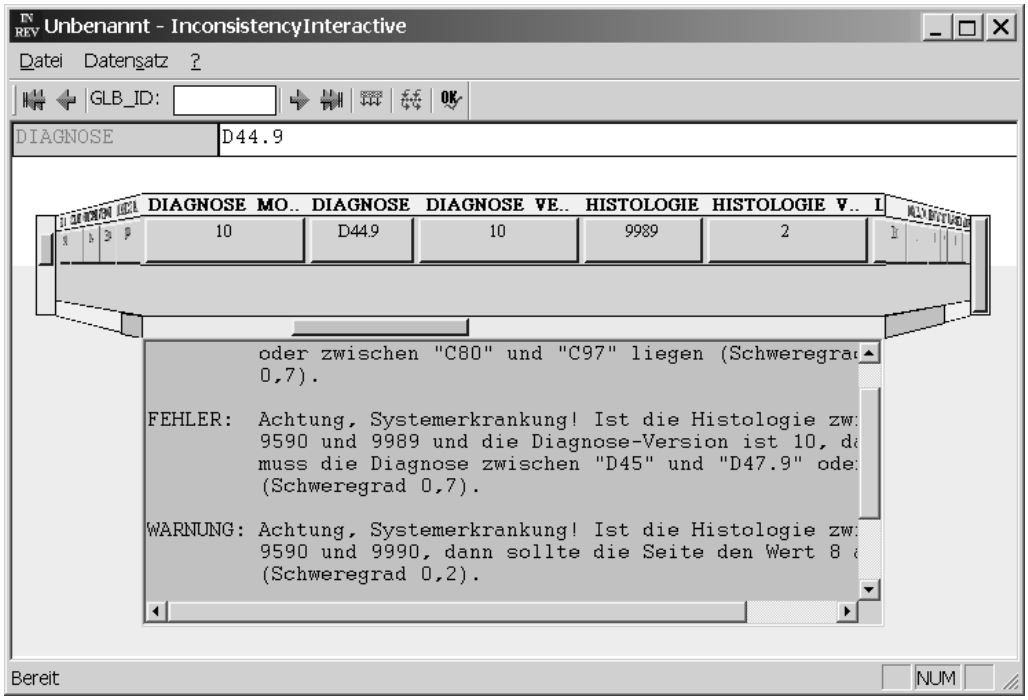


Abbildung 11.16: Interaktive Nachbearbeitung von inkonsistenten Meldungen

Phase 5: Record Linkage

Nach der automatisierten Durchführung des schlüsselbasierten Linkage (vgl. Schritt 7 des Protokolls in Abb. 11.13) – hier wurden einige Meldungen mit Historisierungsstatus 'H' (vgl. Abschnitt 9.11.1) mit korrespondierenden Meldungen aus dem Zielbereich zusammengeführt – wird vom DQM Controller der probabilistische Record Linkage-Prozess angestoßen. Dessen Resultat zeigt der in Abb. 11.17 dargestellte Protokollausschnitt: Insgesamt werden 88 Matchgruppen (vgl. Abschnitt 7.7.2) gebildet, die insgesamt 313 Tupel enthalten.

Phase 6: Merging

Die im vorigen Schritt gebildeten Matchgruppen müssen nun interaktiv konsolidiert werden. Dazu ist für jede Gruppe zu bestimmen, welche der enthaltenen Meldungen sich auf denselben Krebsfall beziehen. Diese müssen zu einem Zieltupel verschmolzen werden. Abbildung 11.18 zeigt eine dreielementige Matchgruppe, die zwei neu zu integrierende Meldungen aus dem Transformationsbereich (TB) und eine Meldung aus dem Zielbereich (ZB) umfasst. Die drei Meldungen sind sich sehr ähnlich (in der Abbildung nur zum Teil zu erkennen), allerdings gibt es Unterschiede in einzelnen Attributwerten. So weicht die Angabe zum Geburtsjahr in Meldung 22932 (1921) um ein Jahr von den Angaben der anderen beiden Meldungen ab, und in der Meldung 22909 fehlt die Angabe zum Wohnort. Durch Anklicken einzelner Werte kann der Systemnutzer diejenigen Angaben bestimmen, die in das Zieltupel übernommen werden sollen.

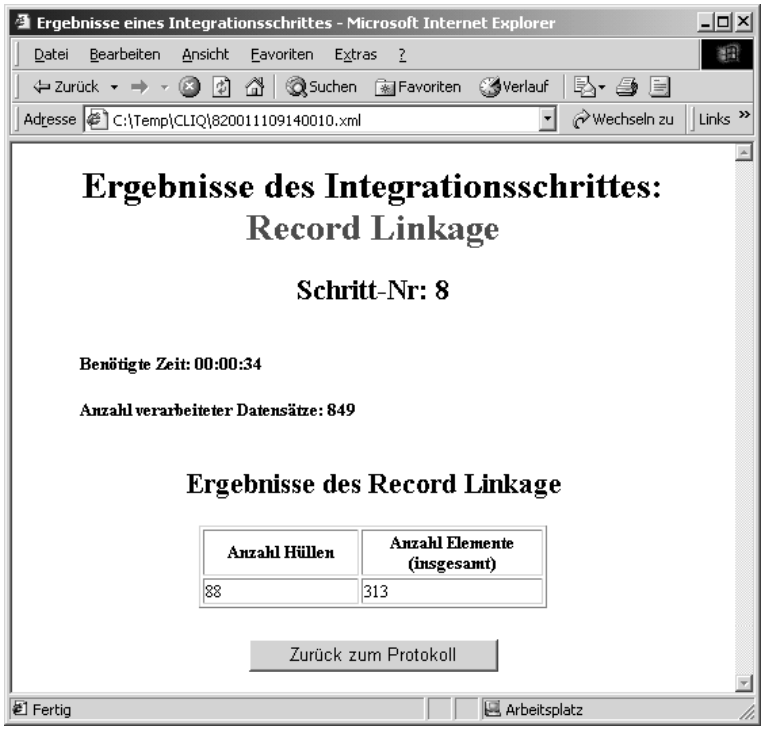


Abbildung 11.17: Überblick über die Ergebnisse des probabilistischen Record Linkage

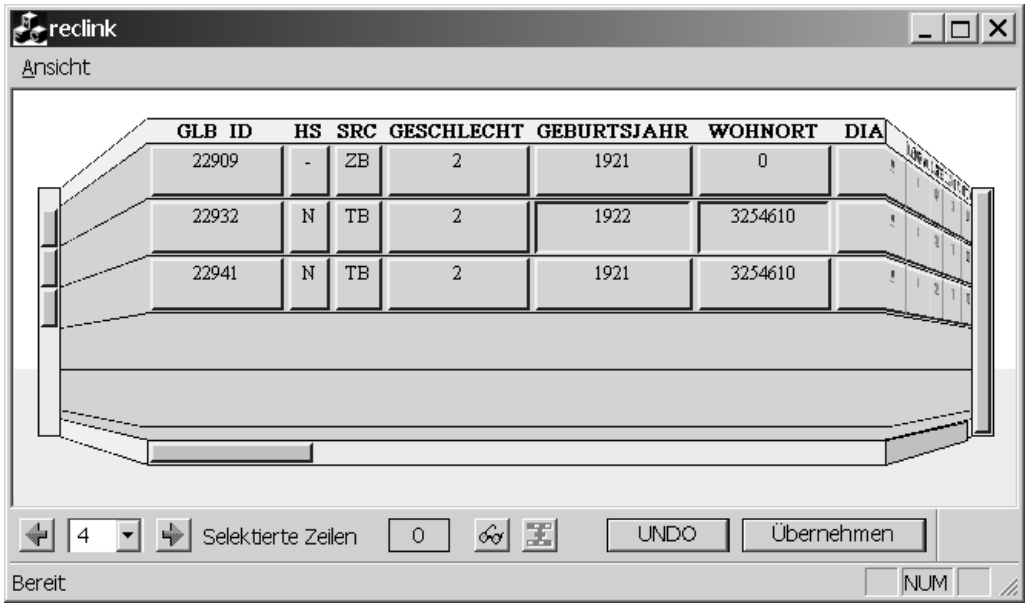


Abbildung 11.18: Interaktive Aufbereitung von Matchgruppen

Diejenigen Tupel, die in eine Verschmelzung eingehen sollen, sind durch Anklicken der Buttons am linken Rand der „Perspective Wall“ (vgl. Abschnitt 9.6.2) auszuwählen. Aus einer Verschmelzung hervorgegangene Tupel legt die CLIQWorkbench im Konsolidierungsbereich ab. Aus Zeitgründen werden in der Fallstudie nicht alle in den Matchgruppen enthaltenen Meldungen aufbereitet, sondern z. T. als einzelne Tupel in den Konsolidierungsbereich übernommen.

Phase 7: Qualitätsmessung und -analyse

Ziel dieses Schritts ist es, die im Rahmen der Konfigurierung der Integrationsaufgabe (siehe Abschnitt 11.3.1) spezifizierten Qualitätsanforderungen im Hinblick auf ihren Erfüllungsgrad zu überprüfen. Die Ergebnisse dieser Prüfungen sind in Abb. 11.19 dargestellt. Man erkennt, dass alle Messwerte oberhalb der geforderten Sollwerte liegen, so dass alle Qualitätspläne als erfüllt gelten.

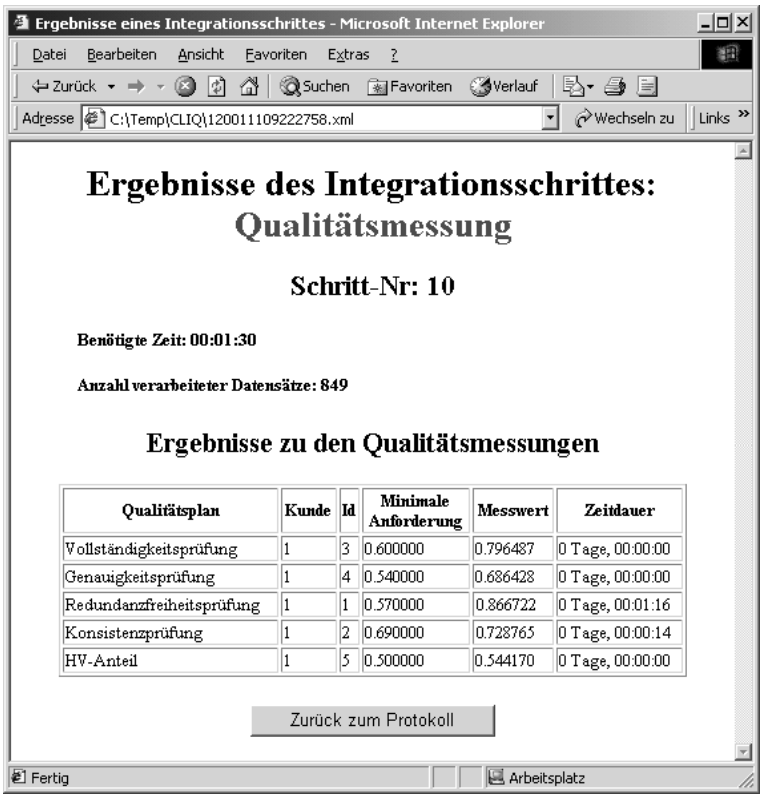


Abbildung 11.19: Überblick über die Ergebnisse der Datenqualitätsmessungen

Dass die Messwerte für Konsistenz und Redundanzfreiheit unter dem maximal erreichbaren Wert von 1 liegen, ist auf die Tatsache zurückzuführen, dass in den Phasen 4 und 6 nicht alle Inkonsistenzen bzw. Matchgruppen bearbeitet wurden. Die relativ lange Laufzeit der Messung zur Redundanzfreiheit ergibt sich daraus, dass die in Abschnitt 7.15.7 vorgeschlagene Laufzeitoptimierung zum Zeitpunkt der Durchführung der Fallstudie noch nicht umgesetzt war, d. h. es wurde zur Bestimmung des Messwertes ein Record Linkage auf dem gesamten Zielbereich vorgenommen.

Phase 8: Lenkung von Datenprodukten/Qualitätsverbesserung

Da alle a priori spezifizierten Qualitätsanforderungen auch nach einer Übertragung der neu integrierten Meldungen in den Zielbereich weiterhin erfüllt sein werden (siehe voriger Schritt), sind keine Lenkungsmaßnahmen (Vergabe von Nutzungsauflagen, ursachenorientierte Maßnahmen, Aussonderung/Neuanforderung o. Ä.) erforderlich.

Phase 9: Freigabe von Datenprodukten

Als letzter Schritt der Datenintegration wird die Freigabe der nun qualitätsgesicherten Meldungen initiiert. Da diese vollständig im Hintergrund abläuft, sind in den Abbildungen 11.20 und 11.21 Aus-

schnitte der Metarelationen *MELDUNGEN_EH* und *MELDUNGEN_ES* im Zustand nach Durchführung der Freigabe dargestellt. Exemplarisch werden im Folgenden die in den Abbildungen umrahmten Tupel im Detail erläutert:

- Die beiden Tupel mit $local_ref = 523$ in *MELDUNGEN_EH* (graue Umrahmung in Abb. 11.20) beschreiben eine Meldung mit Befunddatum 1.3.2000 und ein zugehöriges inkrementelles Update mit Befunddatum 1.6.2000. Beide Tupel stammen aus derselben Quelle (und Quelltabelle) und sind in unterschiedlichen Integrationsläufen verarbeitet worden (1229 bzw. 1234). Ihnen sind in Phase 1 des Vorgehensmodells unterschiedliche globale Schlüssel zugeordnet worden. In Abb. 11.21 wird deutlich, dass das Update den Historisierungsstatus 'H' aufweist, d. h. dass eine Historisierung des ursprünglichen Tupels erforderlich war. Dabei wurde das ursprüngliche Tupel deaktiviert ($is_current := 0$). Als Ende des Gültigkeitszeitraumes wurde das Befunddatum des Update eingetragen. Des weiteren wurde das historisierte Tupel aus der Sicht über diejenigen Tupel, die in Qualitätsmessungen eingehen, gestrichen ($qm_view := 0$). Schließlich wurde das aktualisierte Tupel über das Attribut *old_id* mit dem historisierten Tupel „verzeigert".

- Die beiden Tupel mit $glb_ref = 23.117$ in *MELDUNGEN_EH* (schwarze Umrahmung in Abb. 11.20) beschreiben zwei Meldungen aus unterschiedlichen Quellen (1 und 2), die zu einem Tupel im Zielbereich verschmolzen wurden (daher der gemeinsame globale Schlüssel). Folglich ist in *MELDUNGEN_EH* das Metaattribut *merged* auf 1 gesetzt. Als Beginn des Gültigkeitszeitraumes wurde das am längsten zurückliegende Befunddatum der in die Verschmelzung eingegangenen Meldungen gewählt. Diese Meldungen (lokale Schlüssel 525 und 180 in *MELDUNGEN_EH*) wurden nach dem Merging gelöscht. Die zugehörigen Metadaten bleiben jedoch erhalten, um zukünftige inkrementelle Updates direkt dem aus der Verschmelzung hervorgegangenen Tupel im Zielbereich zuordnen zu können.

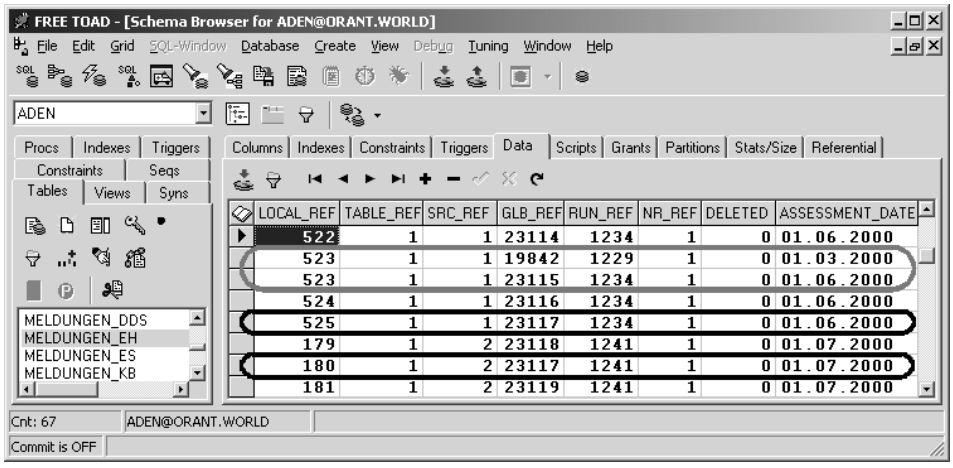


Abbildung 11.20: Ausschnitt aus *MELDUNGEN_EH*

Aus den in den Abbildungen 11.20 und 11.21 dargestellten Tupeln lässt sich weiterhin Folgendes ablesen: Alle beschriebenen Meldungen konnten im ersten Anlauf (d. h. ohne Neuanforderung o. Ä.) integriert werden ($nr_ref = 1$). Löschungen sind bei diesen Meldungen bisher nicht aufgetreten ($deleted = 0$). Des weiteren sind alle Qualitätsanforderungen an Datenobjekte, die die bezeichneten Meldungen einbeziehen, zum aktuellen Zeitpunkt erfüllt ($passed = 1$). Alle integrierten Meldungen sind noch in nachgeschaltete, analyseorientierte Datenstrukturen (z. B. ein Data Warehouse) – soweit vorhanden – zu übertragen ($reload = 1$, in der Fallstudie nicht betrachtet).

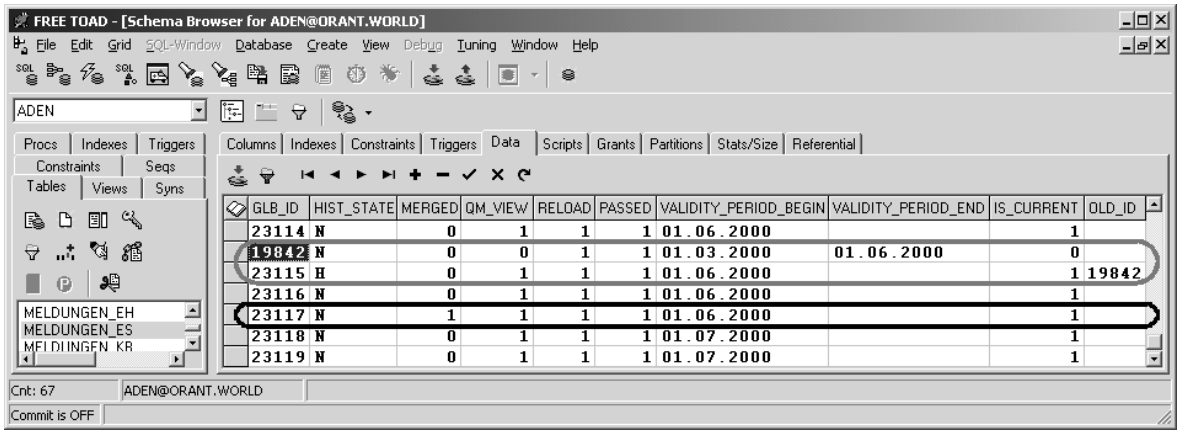


Abbildung 11.21: Ausschnitt aus *MELDUNGEN_ES*

11.4 Zusammenfassung und Bewertung der Ergebnisse der Fallstudie

In der in diesem Kapitel dokumentierten Fallstudie wurden das CLIQ-Vorgehensmodell und die CLIQWorkbench in der Anwendungsdomäne des EKN zum Einsatz gebracht. Dazu wurden zunächst Beziehungen zwischen den Randbedingungen im EKN und den im Rahmen dieser Arbeit getroffenen Annahmen hergestellt. Darauf aufbauend wurde ein einfaches Anwendungsszenario spezifiziert, das die Integration zweier kleiner, heterogener Datenpakete mit Meldungen zu Krebsfällen in einen ebenfalls kleinen, bereits integrierten Datenbestand vorsieht. Diese Integrationsaufgabe wurde mit Hilfe der CLIQWorkbench konfiguriert und schließlich zur Ausführung gebracht.

Als Fazit der Fallstudie lässt sich Folgendes festhalten:

- Die prinzipielle Anwendbarkeit des CLIQ-Vorgehensmodells sowie die Erfüllung der operativen (funktionalen) Anforderungen an die CLIQWorkbench konnten demonstriert werden.
- Die entwickelten Konzepte ließen sich in einfacher Weise auf den Anwendungsbereich des EKN zuschneiden. Die Konfigurierung und Durchführung der Integrationsaufgabe erwies sich als relativ problemlos.
- Die interaktiven DQM-Maßnahmen (insbesondere Nachbearbeitung von Inkonsistenzen und Merging) stellten den mit Abstand zeitaufwändigsten Teil des ansonsten weitgehend automatisierten Integrationsprozesses dar. Allerdings darf nicht übersehen werden, dass auf diese Weise wertvolles Domänenwissen in den Prozess eingebracht werden konnte.
- Das Nachhalten detaillierter Metainformationen zu Integrationsprozessen bildet die Grundlage für eine Nachvollziehbarkeit von Integrationsprozessen. Die XML/XSL-basierte Aufbereitung derartiger Informationen stellte sich als wertvolles Instrument zur automatisierten Prozessdokumentation heraus.
- Das als Voraussetzung für die Datenintegration erforderliche Metadaten-Reengineering erwies sich als relativ aufwändig. Insbesondere die Akquise von Wissen über Korrelationen zwischen Attributen war schwierig. Derartiges Wissen war z. B. unabdingbar für die Erkenntnis, dass das Fehlen spezifischer Diagnoseangaben nicht zwangsläufig einen Datenqualitätsmangel darstellt.
- Das Laufzeitverhalten der CLIQWorkbench ist bisher für praxistypische Datenvolumina nur bedingt geeignet. Die im Rahmen der Fallstudie auftretenden Datenmengen konnten zwar im

unteren Minutenbereich integriert werden (ca. 2.5 Minuten für 400 Tupel unter Ausklammerung der interaktiven DQM-Maßnahmen), für die Verarbeitung größerer Mengengerüste dürfte jedoch eine weitere Optimierung der Laufzeit erforderlich sein. Zur detaillierten Analyse des Laufzeitverhaltens der CLIQWorkbench in einzelnen Phasen des CLIQ-Vorgehensmodells in Abhängigkeit von den Datenvolumina können prinzipiell die in Abschnitt 7.15 vorgenommenen Komplexitätsabschätzungen herangezogen werden, allerdings liegen bisher noch nicht für alle Phasen des Vorgehensmodells hinreichend exakte Abschätzungen vor, so dass von einer „Hochrechnung“ abgesehen wurde.

Abschließend bleibt anzumerken, dass in einer Fallstudie wie dieser zwangsläufig gewisse vereinfachende Annahmen gemacht werden müssen (hier z. B. Verzicht auf Durchführung von Qualitätslenkungsmaßnahmen). Dadurch bleiben ggf. bestimmte praxisrelevante Probleme unberücksichtigt.

Zur Homepage der Dissertation

Kapitel 12

Weitere Evaluierungsmaßnahmen

In diesem Kapitel werden die in Teil II dieser Arbeit erarbeiteten Konzepte – zusätzlich zu der in Kapitel 11 dokumentierten Fallstudie – bzgl. verschiedener Kriterien evaluiert. In Abschnitt 12.1 erfolgt ein Abgleich der CLIQWorkbench mit den gestellten Anforderungen. Die Abschnitte 12.2 und 12.3 skizzieren Evaluierungsaktivitäten, die in Zusammenarbeit mit der SAP AG bzw. der DaimlerChrysler AG stattfanden. Im erstgenannten Fall ging es um die Bewertung der Integrierbarkeit der CLIQWorkbench in das SAP Business Warehouse, im zweitgenannten Fall um die Anwendbarkeit des CLIQ-Vorgehensmodells auf eine unternehmensinterne Fahrzeugdatenbank.

12.1 Abgleich der CLIQWorkbench mit den Anforderungen

In diesem Abschnitt wird kritisch überprüft, inwieweit das in dieser Arbeit entwickelte Softwaresystem den in Kapitel 8 gestellten Anforderungen genügt. Dabei werden Anforderungen wiederum gemäß [SR98] klassifiziert.

12.1.1 Operative Anforderungen

Im Folgenden wird auf die in Abschnitt 8.1.1 angegebene Klassifikation von operativen Anforderungen an die CLIQWorkbench zurückgegriffen.

- *Planung von Datenintegrationsprozessen:* Zur Konfigurierung von Integrationsprozessen dient in der CLIQWorkbench das Paket *JobPlanEngine* des DQM Controllers. Dieses stellt Dialogfenster bereit, mit denen sich alle Phasen des in Kapitel 7 vorgestellten Vorgehensmodells – soweit erforderlich – konfigurieren lassen.
- *Durchführung von Datenintegrationsprozessen:* Integrationsprozesse werden in der CLIQWorkbench durch das Paket *JobExecEngine* des DQM Controllers gesteuert. Dieses Paket stößt die für die eigentliche Datenintegration zuständigen Komponenten wie Data Migrator, Data Linker etc. an. Der gesamte Integrationsprozess von den Quelldatenpuffern bis zum Zielbereich wird durch die CLIQWorkbench durchgängig unterstützt. Wo erforderlich, werden Interaktionen mit einem Systemnutzer vorgenommen.
- *Analyse von Datenintegrationsprozessen:* Integrationsprozesse können mit Hilfe des Pakets *JobAnalyzeEngine* des DQM Controllers analysiert werden. Dieses Paket bietet Verfahren an, mit denen prozess- und datenbezogene Metainformationen ausgewertet und graphisch aufbereitet werden können.

12.1.2 Technische Anforderungen

Die CLIQWorkbench ist modular aufgebaut. Insbesondere wurde die weitgehend plattformunabhängige Funktionalität zur Datenintegration von der plattformspezifischen Funktionalität zur Nutzerinteraktion (graphische Benutzungsoberfläche) getrennt, um eine einfache Portierung des Systems zu gewährleisten. Abstriche bei der Portierbarkeit müssen allerdings aufgrund der Verwendung der plattformspezifischen Technologien COM und ActiveX in einzelnen Teilmodulen (Repository Interface bzw. Data Migrator) gemacht werden.

Die Kapselung der DBMS- und Repository-spezifischen Funktionalität in dem Paket *DBInterface*, den submodulspezifischen Paketen *JobControlDBEngine*, *RecLinkExecDB*, *RecLinkReviewDB* etc. sowie dem Paket *MrRepository* stellt ferner die Austauschbarkeit der zu Grunde liegenden Datenhaltungskomponenten sicher.

12.1.3 Qualitätsanforderungen

Zum Erfüllungsgrad der Anforderungen an die Robustheit und Benutzungsfreundlichkeit der CLIQWorkbench kann an dieser Stelle keine Bewertung erfolgen, da dafür Erfahrungen aus dem praktischen Einsatz des Systems – über die in Kapitel 11 dokumentierte Fallstudie hinaus – vorliegen müssten. Im Hinblick auf die Laufzeiteffizienz von Datenintegrationsprozessen auf Basis des CLIQ-Vorgehensmodells wurden Komplexitätsabschätzungen durchgeführt, deren Ergebnisse bereits in Abschnitt 7.15 beschrieben wurden.

Der Forderung nach Erweiterbarkeit des Systems wurde bei dem Entwurf der CLIQWorkbench wie folgt Rechnung getragen: Um z. B. einen neuen Teilprozess zu integrieren, muss – im Sinne eines objektorientierten Anwendungsframeworks – von der abstrakten Klasse *CLIQProcess* eine weitere Subklasse abgeleitet werden, die den Prozess beschreibt. Ein- und Ausgabeschnittstellen des neuen Prozesses können durch Spezialisierungen von *CLIQProcessConfiguration* bzw. *CLIQProcessResult* spezifiziert und geeignet gekapselt werden. Die Klasse *TransitionManager* ist dahingehend zu erweitern, dass der neue Teilprozess in das bereits existierende System von in Beziehung stehenden Prozessen eingebunden wird. Schließlich ist das Paket *JobPlanEngine* ggf. um Dialogklassen zur Konfigurierung des Teilprozesses und die Klasse *TransactionManager* ggf. um Funktionalität zum Löschen von Zwischenergebnissen des Teilprozesses zu ergänzen.

Betrachtet man die CLIQWorkbench im Hinblick auf Automatisierbarkeit, so stellt man fest, dass interaktive Eingriffe in den Datenintegrationsprozess im Wesentlichen auf die folgenden Bereiche beschränkt sind:

- Konfigurierung
  - Spezifizierung von Datenintegrationsprozessen (einmalig pro Zielrelation)
  - Spezifizierung von Datenqualitätsanforderungen (einmalig pro Kunde)
- Durchführung
  - Beseitigung von schlüsselbezogenen Integritätsverletzungen (Phase 1)
  - Ergreifen von Maßnahmen infolge einer Über- oder Unterschreitung von Kontrollgrenzen im Rahmen der SPC (Phase 3)
  - Beseitigung von domänenspezifischen Inkonsistenzen (Phase 4)
  - Auflösung von Attributwertkonflikten beim Verschmelzen von einander zugeordneten Tupeln (Phase 6)
  - Ergreifen von Maßnahmen zur Lenkung mangelhafter Datenprodukte (Phase 8).

Diese interaktiven Elemente sind so in die CLIQWorkbench integriert, dass ein möglichst reibungsloser Ablauf der Datenintegration gewährleistet ist. So führt z. B. bei der automatisierten Suche nach domänenspezifischen Inkonsistenzen die Entdeckung einer Inkonsistenz nicht zur Unterbrechung des Suchprozesses. Stattdessen werden Inkonsistenzen zunächst nur protokolliert, so dass sie später batchorientiert, ggf. auch parallel (durch mehrere Sachbearbeiter) abgearbeitet werden können. Bei schlüsselbezogenen Integritätsverletzungen sowie bei der Zuordnung und Verschmelzung von redundanten Tupeln wird analog verfahren.

Die CLIQWorkbench ist – wie in Abschnitt 8.1.3 gefordert – weitgehend domänenunabhängig. Eine Anpassung an eine konkrete Domäne kann wie folgt vorgenommen werden:

- Spezifizierung von (attributspezifischen) Metadaten wie Update-Häufigkeit oder speziellen Nullwerten (Klasse *DQMAttributeDescriptor*)
- Spezifizierung von Prozesskonfigurationen unter Berücksichtigung domänenspezifischer Charakteristika, z. B. Werteverteilungen von Attributen bei der Auswahl von Blockvariablen zum Record Linkage (Phase 5)
- Einbindung von domänenspezifischem Wissen über das Paket *DomainSpecificMeasurementEngine* (Phase 4)
- Spezifizierung von domänenspezifischen Qualitätsindikatoren (Phase 7).

Die Integrierbarkeit der CLIQWorkbench bzw. des CLIQ-Vorgehensmodells in ein gegebenes DWS wird in den Abschnitten 12.2 und 12.3 an Hand des SAP Business Warehouse bzw. am Beispiel einer Datenbankanwendung bei DaimlerChrysler diskutiert.

12.1.4 Wartungsanforderungen

Es wurden keine Wartungsanforderungen an die CLIQWorkbench gestellt.

12.1.5 Realisierungsanforderungen

Wie in Abschnitt 8.1.5 gefordert, wurde die CLIQWorkbench gemäß der OOA/OOD-Methodik unter Verwendung der Modellierungsnotation der UML entwickelt.

12.2 Einbindung der CLIQWorkbench in das SAP BW

Im Rahmen dieses Teils der Evaluierung wurde in Kooperation mit der SAP AG versucht, die CLIQWorkbench bzw. Teile davon in das SAP Business Warehouse (SAP BW) Version 2.0B [SAP97, Has00a, SSL01] zu integrieren. Dazu musste zunächst der Datenfluss im SAP BW analysiert werden (siehe Abb. 12.1).

Neu in das SAP BW zu integrierende Daten werden quellenseitig in eine sog. *Transferstruktur* überführt und in diesem Format an das SAP BW übertragen, wo sie in einer sog. *Persistent Staging Area* (*PSA*, eine pro Quelle) abgelegt werden. Mit Hilfe sog. *Übertragungsregeln* (Attribut-Mapping und einfache Konsistenzprüfungen) werden Daten von einer PSA in eine sog. *InfoSource* transferiert, deren Schema als *Kommunikationsstruktur* bezeichnet wird. Eine *InfoSource* beschreibt die Menge aller verfügbaren Daten zu einem Typ eines Geschäftsvorfalls. Im SAP BW wird zwischen Stamm- und Bewegungsdaten unterschieden. *Stammdaten* (nicht historisiert, z. B. Wohnorte von Kunden) werden

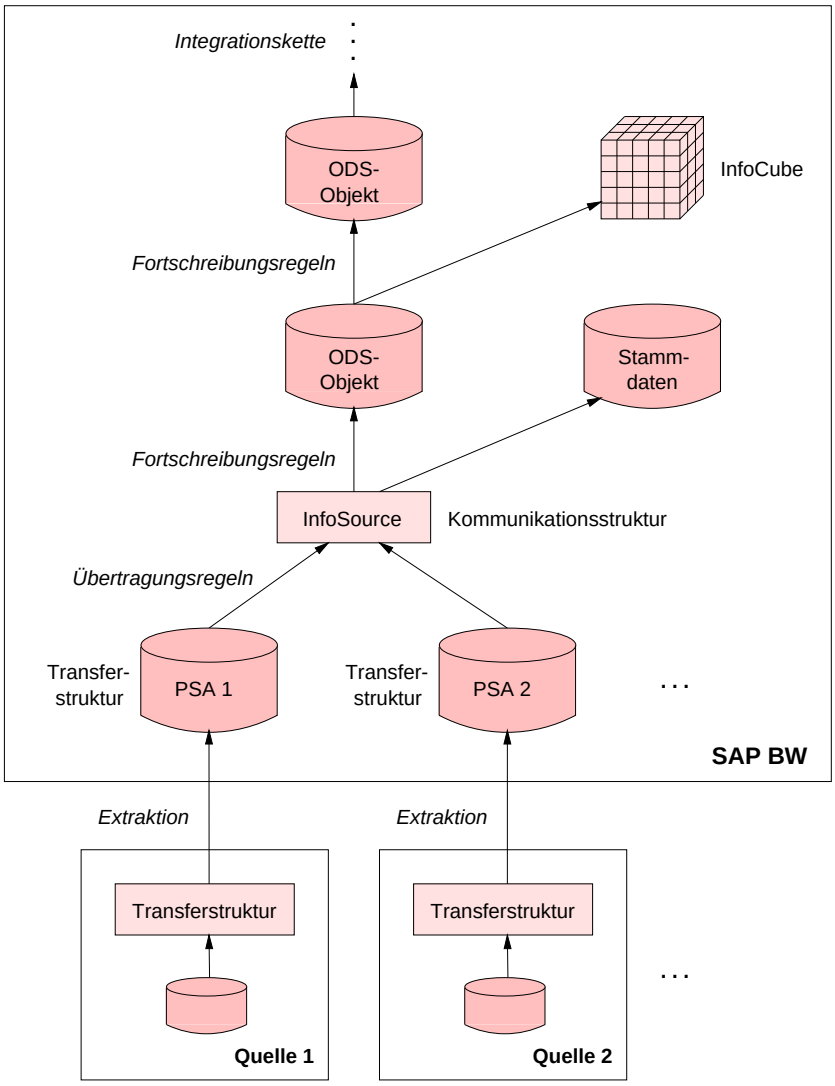


Abbildung 12.1: Datenfluss im SAP Business Warehouse 2.0B

in sog. *Stammdatentabellen* abgelegt und von den Analyseobjekten (ODS-Objekte und InfoCubes, siehe unten) gemeinsam genutzt, während *Bewegungsdaten* (Kennzahlen mit Merkmalsbezug) in jedem Analyseobjekt gespeichert werden, das diese benötigt. Unter Verwendung sog. *Fortschreibungsregeln* werden Daten von einer *InfoSource* in die Stammdatentabellen bzw. in *ODS-Objekte* geladen. Ein ODS-Objekt dient der Verwaltung von konsolidierten Detaildaten, einerseits zu Reporting-Zwecken und andererseits als Vorstufe zu den nachgeschalteten, verdichteten (multidimensionalen) *InfoCubes*. Von diesen ist wiederum ein Durchgreifen (Drill Through) auf die Detaildaten des zu Grunde liegenden ODS-Objekts möglich. Die Übertragung von Daten aus einem ODS-Objekt in einen InfoCube erfolgt wiederum mit Hilfe von Fortschreibungsregeln.

Eine Besonderheit des SAP BW stellt die Möglichkeit dar, mehrere ODS-Objekte mittels Fortschreibungsregeln zu einer sog. *Integrationskette* zu verknüpfen, um z. B. Daten aus unterschiedlichen Geschäftsprozessen zu integrieren.

Von Bedeutung ist weiterhin die interne Struktur eines ODS-Objekts, wie sie Abb. 12.2 zeigt: Eingehende Daten aus InfoSources oder vorgeschalteten ODS-Objekten werden zunächst in einem als *New Data* bezeichneten Puffer gesammelt. Zu bestimmten Zeitpunkten werden die im New Data-Bereich aufgelaufenen Daten dann in einem Schritt in den sog. *Active Data*-Bereich verschoben, wo sie zum Reporting bzw. zur Propagierung in nachfolgende ODS-Objekte oder InfoCubes zur Verfügung ste-

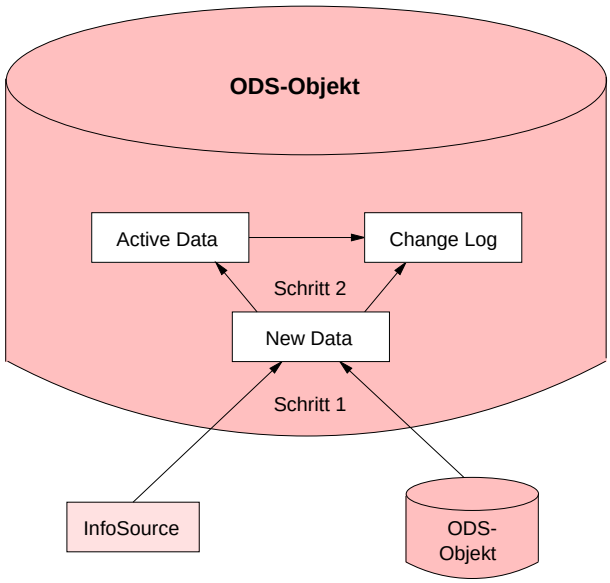


Abbildung 12.2: Aktualisierung von ODS-Objekten im SAP BW 2.0B

hen. Im Rahmen der Übertragung von New Data zu Active Data wird ein sog. *Change Log* erstellt, der zur Fortschreibung der nachfolgenden Objekte herangezogen wird (Delta Update). In der Version 3.0 des SAP BW wird der New Data-Bereich durch eine sog. *Activation Queue* ersetzt.

Das SAP BW unterstützt bisher nur Teilaspekte des DQM (siehe dazu [Dör00]), so dass eine (teilweise) Einbindung der CLIQWorkbench in das SAP BW sinnvoll erschien. Um zu ermitteln, ob und – wenn ja – wie eine derartige Einbindung erfolgen könnte, wurden zunächst die jeweils grundlegenden Begriffe (bzw. die dahinterstehenden Konzepte) aufeinander abgebildet (siehe Tab. 12.1). Es stellte sich heraus, dass es viele Entsprechungen zwischen beiden Architekturen gibt – insbesondere finden sich im SAP BW Pendants zu den CLIQ-Komponenten Quelldatenpuffer, Transformations-, Konsolidierungs- und Zielbereich, wenn auch im Falle des Konsolidierungsbereichs mit abweichender Zielsetzung ihrer Einführung (einfache Pufferung eingehender Daten).

SAP BW 2.0B	**CLIQWorkbench**
Transferstruktur	Relationenschema eines QDP
Persistent Staging Area	Quelldatenpuffer
InfoSource	Transformationsbereich
Kommunikationsstruktur	Relationenschema des TB
ODS-Objekt New Data	Konsolidierungsbereich
ODS-Objekt Active Data	Zielbereich
Übertragungsregel	Task Set (DTS)

Tabelle 12.1: Gegenüberstellung der Basiskonzepte von SAP BW 2.0B und CLIQWorkbench

Im nächsten Schritt wurde diskutiert, welche Teile der CLIQWorkbench für das SAP BW von Interesse sind. Tabelle 12.2 fasst die vorläufigen Ergebnisse dieser Überlegungen aus Implementierungssicht zusammen (nicht aufgeführte Module wurden nicht betrachtet).

Schließlich wurde für die als besonders interessant erachteten CLIQWorkbench-Module (Data Linker und Measurement Engines) ein Ansatz zur Integration ins SAP BW erarbeitet, wobei die in Tab. 12.1 dokumentierte Gegenüberstellung der Konzepte als Ausgangspunkt diente. Abbildung 12.3 zeigt die Kernideen des Ansatzes: Den InfoSources sowie den ODS-Objekten der Ebene 1 einer prinzipiell

CLIQWorkbench-Modul	Interessant für SAP BW?
DQM Controller	Nein, da auf CLIQ-Vorgehensmodell zugeschnitten
Data Migrator	Nein, da Funktionalität im SAP BW schon vorhanden
Measurement Engines	Ja, da Zusatzfunktionalität
Data Linker	Ja, da Zusatzfunktionalität
Data Auditor	Unklar, da vergleichbare kommerzielle Tools verfügbar

Tabelle 12.2: Vorläufige Auswahl von CLIQWorkbench-Modulen für das SAP BW

beliebig langen Integrationskette kommt eine besondere Bedeutung im Rahmen des DQM zu. Auf einer InfoSource werden statistische Prozesskontrollen sowie (ggf. tupelübergreifende) domänenspezifische Konsistenzprüfungen durchgeführt (vgl. die auf dem Transformationsbereich arbeitenden Phasen 3 und 4 des CLIQ-Vorgehensmodells). Die Daten der InfoSource werden anschließend mit den Daten des Active Data-Bereichs des ODS-Objekts der Ebene 1 abgeglichen (vgl. Record Linkage zwischen Transformations- und Zielbereich in Phase 5 des CLIQ-Vorgehensmodells) und ggf. miteinander verschmolzen, wobei die Ergebnisse dieses Merging-Prozesses zunächst in den New Data-Bereich des ODS-Objektes der Ebene 1 geschrieben werden (vgl. Schreiben der Merging-Ergebnisse in den Konsolidierungsbereich in Phase 6 des CLIQ-Vorgehensmodells).

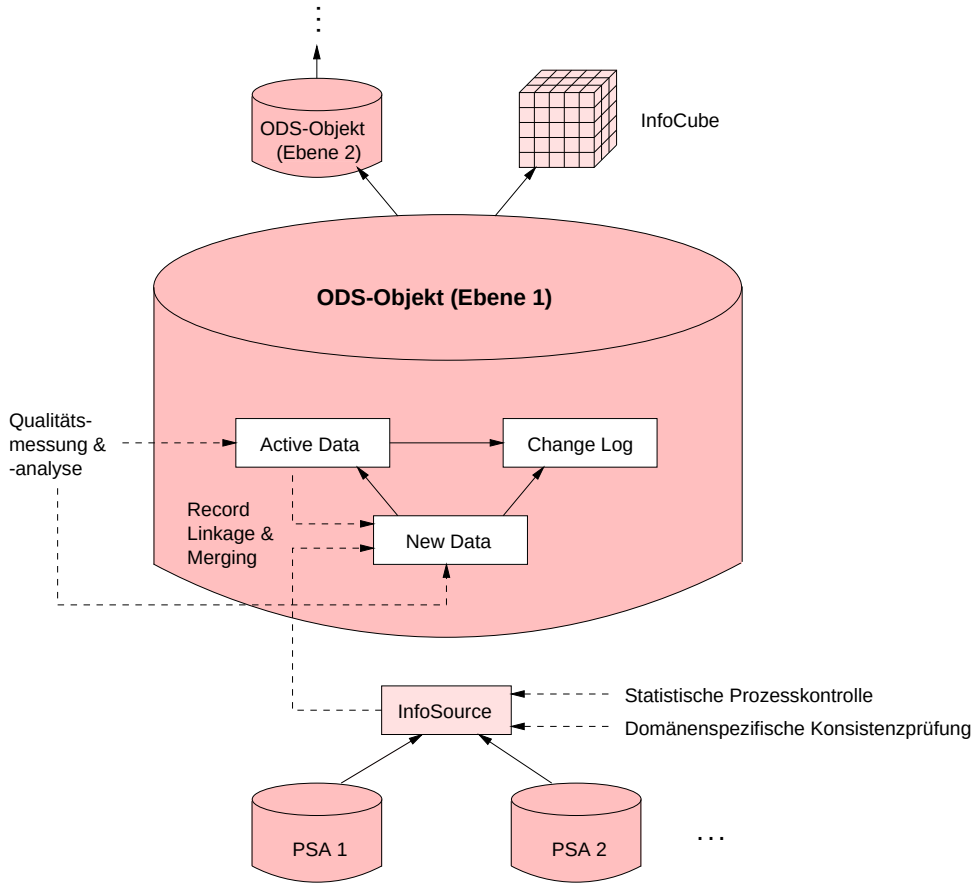


Abbildung 12.3: Ansatz zur Integration der CLIQWorkbench in das SAP BW 2.0B

Bevor die Daten in New Data aktiviert und damit freigegeben werden, sind sie zunächst – zusammen mit den bereits freigegebenen Daten in Active Data – Qualitätsmessungen zu unterziehen, um den Erfüllungsgrad von Qualitätsanforderungen (an den Active Data-Bereich), wie er sich nach einer

Aktivierung ergeben würde, zu bestimmen. Alle ODS-Objekte ab Ebene 2 sind vom DQM nicht mehr betroffen, da in ihnen ausschließlich bereits qualitätsgesicherte Daten fortgeschrieben werden. Gleiches gilt für InfoCubes.

Zur Abschätzung der tatsächlichen Kundennachfrage nach der von der CLIQWorkbench bereitgestellten Zusatzfunktionalität hat die SAP AG in Zusammenarbeit mit OFFIS und dem Fachbereich 4 (Wirtschafts- und Rechtswissenschaften) der Universität Oldenburg eine Diplomarbeit ausgeschrieben, in der eine empirische Untersuchung unter SAP BW-Kunden durchgeführt werden soll. Auf der Basis der Ergebnisse dieser Diplomarbeit soll über das weitere Vorgehen bzgl. der Integration der CLIQWorkbench in das SAP BW entschieden werden.

12.3 Evaluierung des CLIQ-Vorgehensmodells durch DaimlerChrysler

Im Kontext ihrer Forschungsaktivitäten im Bereich DQM (vgl. Abschnitt 5.2) hat die DaimlerChrysler AG die einzelnen Phasen des CLIQ-Vorgehensmodells (siehe Kapitel 7) im Hinblick auf ihre Anwendbarkeit in der unternehmensinternen Fahrzeugdatenbank QUIS (Qualitätsinformationssystem) beurteilt [GH01a]. Es handelte sich dabei um eine rein theoretische Evaluierung, deren Ergebnisse im Folgenden kurz zusammengefasst werden:

1. *Prüfung von Schlüssel- und referentieller Integrität:* Die Abbildung von Tupeln aus den heterogenen Vorsystemen von QUIS auf ein eindeutiges Schlüsselattribut war bereits durch die bestehenden Prozesse gewährleistet und somit nicht Gegenstand der Untersuchung.

2. *Vereinheitlichung der Repräsentation:* Auch die hier notwendigen Schritte waren durch die bestehenden Prozesse bereits implementiert. Exemplarisch ist die einheitliche Repräsentation von Kosten und Fahrzeuglaufleistung zu nennen.

3. *Statistische Prozesskontrolle:* Diese Phase hat für QUIS eine hohe Bedeutung, insbesondere zur Überwachung des fehlerfreien Ablaufs der komplexen, vorgelagerten Datenübertragungs- und Datentransformationsprozesse. Die im Rahmen dieser Phase implementierten Prüfverfahren wurden bereits in Abschnitt 5.2 vorgestellt.

4. *Domänenspezifische Konsistenzprüfung/Nachbearbeitung:* Zusätzlich zu bereits explizit bekannten Konsistenzregeln werden bei DaimlerChrysler mittels Data Mining-Verfahren (z. B. Entscheidungsbaumverfahren) zusätzliche (Geschäfts-)Regeln aus den historisierten QUIS-Daten ermittelt und in zukünftige Prüfungen integriert (vgl. Abschnitt 5.2.3).

5. *Record Linkage:* Da Fahrzeugdaten in allen beteiligten Quellsystemen eine eindeutige, fahrzeugspezifische Identifikationsnummer aufweisen, kann an Hand dieser Nummer ein einfaches schlüsselbasiertes Record Linkage vorgenommen werden. Ein probabilistisches Record Linkage ist nicht erforderlich.

6. *Merging:* Die Zusammenführung redundanter Tupel wird in QUIS bisher nicht softwaretechnisch unterstützt.

7. *Qualitätsmessung und -analyse:* Aufgrund der gegebenen Prozessabläufe ist es nicht möglich, die Qualität eines aktualisierten QUIS-Datenbestands bereits vor der Durchführung des inkrementellen Update zu berechnen bzw. abzuschätzen. Es kann derzeit somit (noch) nicht sichergestellt werden, dass sich die Datenqualität in QUIS durch ein Update nicht verschlechtert. Eine zukünftige Implementierung dieser Phase verspricht erhebliche Optimierungspotentiale, da dadurch Rollback-Läufe vermieden werden könnten.

8. *Lenkung von Datenprodukten:* Abhängig vom verursachenden Quellsystem sind aktuell zwei Lenkungsmaßnahmen realisiert:

   - Zurückweisung der mangelhaften Daten und Korrektur im Quellsystem mit anschließender Neuübertragung an QUIS
   - Korrektur bzw. Vervollständigung von Tupeln in QUIS.

9. *Freigabe von Datenprodukten:* Alle in QUIS verfügbaren Daten gelten als freigegeben, ggf. automatisch korrigierte oder vervollständigte Datensätze sind an Hand bestimmter Kennungen identifizierbar.

10. *Auswertung von Kundenfeedback/Datenrücknahme:* Dies erfolgt im Rahmen regelmäßig stattfindender Projektgruppentreffen und wird softwaretechnisch bisher nicht unterstützt.

Als Fazit lässt sich festhalten, dass die Anwendung des CLIQ-Vorgehensmodells auf die QUIS-Datenbank von DaimlerChrysler grundsätzlich als sinnvoll und praktikabel eingeschätzt wird. Allerdings mussten bzw. konnten einzelne Phasen aufgrund von systemspezifischen Besonderheiten (z. B. global eindeutige Kennzeichnung aller Quelltupel) nur teilweise umgesetzt werden (z. B. kein probabilistisches Record Linkage).

12.4 Zusammenfassung

Bei der Evaluierung des CLIQ-Vorgehensmodells und der CLIQWorkbench standen funktionale Aspekte wie Umfang der Unterstützung von DQM-Maßnahmen, Erweiterbarkeit, Automatisierbarkeit und Anpassbarkeit an gegebene Systemumgebungen im Vordergrund. Die Überprüfung eines Teils dieser Kriterien erfolgte in Kooperation mit externen Partnern (SAP AG und DaimlerChrysler AG). Aufgrund des prototypischen Zustands der Implementierung wurde auf eine leistungsorientierte Evaluierung der CLIQWorkbench verzichtet.

Zur Homepage der Dissertation

Teil IV

Zusammenfassung und Ausblick

Kapitel 13

Zusammenfassung und Ausblick

In Abschnitt 13.1 dieses Kapitels werden die Ergebnisse der Arbeit noch einmal zusammengefasst. Anschließend wird in Abschnitt 13.2 ein Ausblick auf zukünftige Forschungsarbeiten im Kontext des DQM gegeben.

13.1 Zusammenfassung

Taktische und strategische Entscheidungen in Unternehmen und anderen Organisationsformen werden immer häufiger auf der Basis von Datenanalysen durchgeführt. Die zunehmende Verbreitung von DWS und darauf aufsetzenden Anwendungen wie Customer Relationship Management oder Supply Chain Management bestätigt diese Entwicklung. Ein DWS erlaubt die flexible und effiziente Auswertung von aus heterogenen Quellen integrierten, möglicherweise historisierten Datenbeständen.

Während sowohl in der Forschung als auch in der Praxis intensiv an Techniken zur Datenauswertung gearbeitet wurde (multidimensionale Datenmodellierung, spezielle Indizierungsverfahren, OLAP- und Data Mining-Methoden), blieben Fragen der Datenintegration und Datenqualitätssicherung lange unberücksichtigt. Dabei ist leicht einzusehen, dass die Auswertung von qualitativ mangelhaften (d. h. inkonsistenten, unvollständigen, obsoleten etc.) Daten zwangsläufig zu qualitativ mangelhaften Ergebnissen führt, die nicht oder nur unter Vorbehalt zur Entscheidungsunterstützung herangezogen werden sollten („Garbage in, garbage out“).

Die vorliegende Arbeit hat sich daher zum Ziel gesetzt, grundlegende Konzepte eines Datenqualitätsmanagements herauszuarbeiten, darauf aufbauend eine Methodik für eine qualitätsgelenkte Datenintegration in DWS zu definieren sowie Ansätze für eine softwaretechnische Unterstützung der Anwendung dieser Methodik zu entwickeln und zu evaluieren. Diese Zielsetzung spiegelt sich unmittelbar in der Kapitelstruktur der Arbeit wider, wie Abb. 13.1 zeigt:

- Der in Kapitel 1 grob umrissene und motivierte Themenbereich wurde in den Kapiteln 3 bis 5 im Detail betrachtet. Dabei wurde zunächst der State of the Art von DWS untersucht. Anschließend wurden zentrale Datenqualitätsmerkmale identifiziert und Ursachen für Datenqualitätsmängel klassifiziert. Durch Anpassung gängiger QM-Konzepte an den Kontext der Datenintegration in einem DWS konnten die wesentlichen Funktionen des DQM identifiziert werden. Schließlich wurden verwandte Ansätze aus dem Umfeld des DQM vorgestellt und bewertet.
- Um Datenqualität „messbar“ zu machen und damit die Voraussetzung für ein effektives DQM zu schaffen, wurden in Kapitel 6 Metriken für verschiedene Datenqualitätsmerkmale definiert und durch Messverfahren operationalisiert. Für diejenigen Merkmale, bei denen eine nachträgliche Beseitigung von entdeckten Datenqualitätsmängeln möglich ist (z. B. Ein-

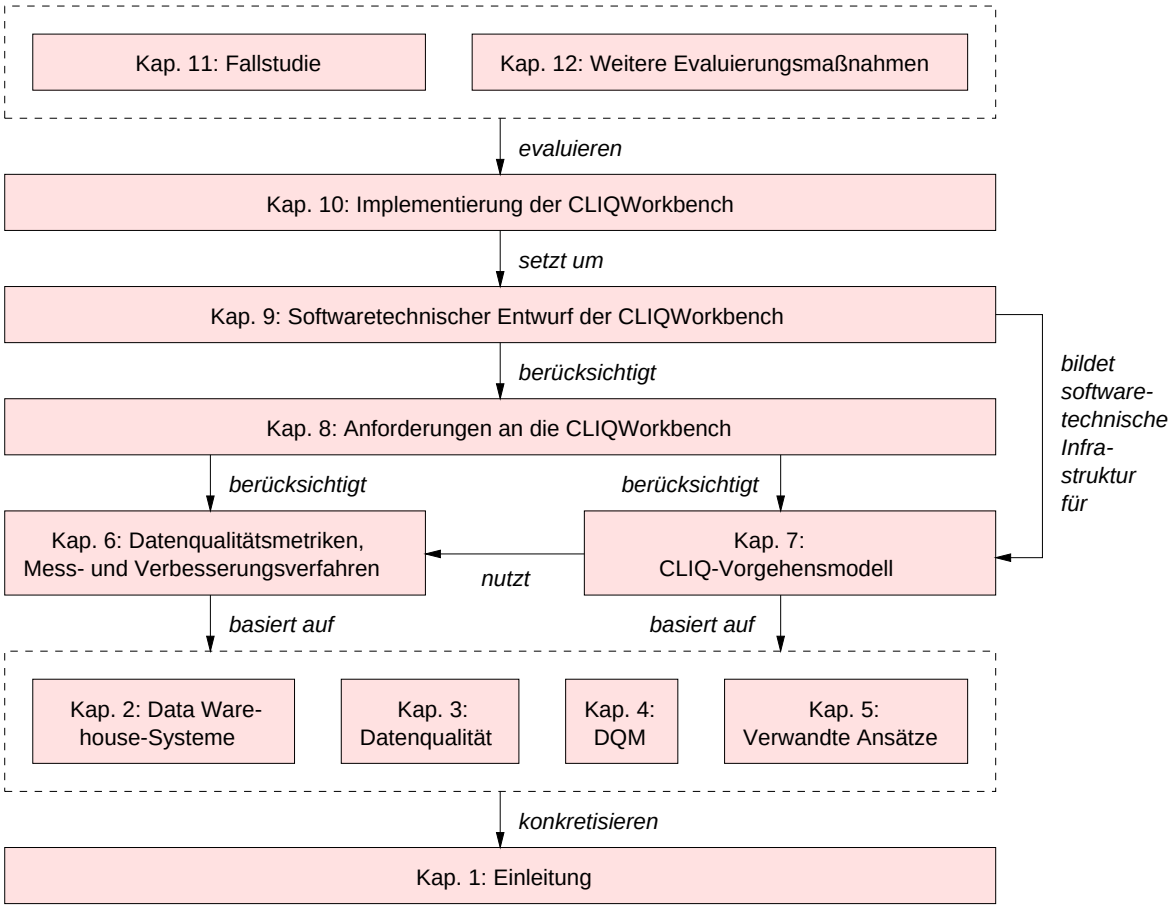


Abbildung 13.1: Rückblick auf die Kapitelstruktur dieser Arbeit

heitlichkeit, Konsistenz, Redundanzfreiheit), wurden darüber hinaus Ansätze zur Datenqualitätsverbesserung aufgezeigt.

- Einen zentralen Beitrag dieser Arbeit stellt das in Kapitel 7 beschriebene, 10-phasige Vorgehensmodell zur qualitätsgelenkten Datenintegration dar. Dieses auf den Anforderungen der Norm ISO 9001:2000 basierende Vorgehensmodell schafft die methodische Grundlage für hochgradig automatisierte, reproduzierbare und wartbare Integrationsprozesse mit qualitätsgesicherten Ergebnissen. Im Rahmen der Qualitätslenkung von Daten wird auf die in Kapitel 6 spezifizierten Metriken, Mess- und Verbesserungsverfahren zurückgegriffen.

- Zur technischen Umsetzung des Vorgehensmodells wurde in Kapitel 9 – basierend auf der in Kapitel 8 dokumentierten Anforderungsspezifikation – ein als CLIQWorkbench bezeichnetes Softwaresystem entworfen, welches die Planung, Durchführung und Analyse von Datenintegrationsprozessen unterstützt. Die prototypische Implementierung der CLIQWorkbench wurde in Kapitel 10 erläutert.

- Mit der Zielsetzung, einen „Proof of Concepts“ zu erbringen, wurde der in dieser Arbeit entwickelte Ansatz in den Kapiteln 11 und 12 evaluiert. Dabei wurde der Einsatz der CLIQWorkbench in einer im Kontext des Epidemiologischen Krebsregisters Niedersachsen durchgeführten Fallstudie demonstriert. Weiterhin erfolgte ein Abgleich der CLIQWorkbench mit den gestellten Anforderungen sowie eine Evaluierung einzelner Aspekte der erarbeiteten Lösung an Hand von externen Partnern (SAP AG und DaimlerChrysler AG).

13.2 Ausblick

In diesem Abschnitt wird der aktuelle Stand der Forschung zum DQM – unter besonderer Berücksichtigung des Beitrags der vorliegenden Arbeit – mit ausgewählten Zielsetzungen abgeglichen, um daraus weiteren Forschungsbedarf abzuleiten. Zu unterscheiden ist dabei zwischen Herausforderungen, denen sich die Forschungsgemeinde langfristig (d. h. mit einem Zeithorizont von mehr als drei Jahren) zu stellen hat, und Perspektiven, die sich daraus kurzfristig für die kommenden zwei bis drei Jahre ergeben.

13.2.1 Herausforderungen

Die übergreifende Herausforderung wird sein, DQM als gleichberechtigten Erfolgsfaktor in Organisationen zu etablieren, d. h. durchzusetzen, dass Datenqualität als wichtiges Organisationsziel festgeschrieben wird. Die in Abschnitt 3.1 vorgestellte Analogie zwischen Daten und Produkten, Datennutzern und Kunden, Datenquellen und Lieferanten sowie Datenintegrationsprozessen und Fertigungsprozessen muss dazu in die Praxis hineingetragen und von den jeweiligen Organisationen auch „gelebt“ werden. Langfristiges Ziel ist die Etablierung einer Zertifizierung von Daten bzw. datenverarbeitenden Systemen. Einen ersten Schritt in diese Richtung stellt das in Kapitel 7 dieser Arbeit entwickelte, ISO 9001:2000-basierte Vorgehensmodell zur Datenintegration dar.

Um das genannte Ziel erreichen zu können, müssen bestimmte Voraussetzungen erfüllt sein. Insbesondere müssen allgemein akzeptierte Metriken existieren, um Organisationen ein Hilfsmittel zur Bewertung von Datenqualität an die Hand zu geben. Für ausgewählte Qualitätsmerkmale wie Konsistenz, Vollständigkeit und Redundanzfreiheit sind bereits in dieser Arbeit Metriken entwickelt worden (siehe Kapitel 6). Problematisch sind allerdings „weiche“ Qualitätsmerkmale wie Relevanz und Verständlichkeit. Diese erfordern zum einen sehr umfangreiches Domänenwissen und sind damit – zumindest aus heutiger Sicht – praktisch ausschließlich durch einen menschlichen Experten zu bewerten. Das Merkmal Relevanz ist zudem hochgradig kontextsensitiv, während das Merkmal Verständlichkeit auch subjektive Einflüsse einbezieht. Für solche Merkmale müssen Metriken gefunden werden, die interaktive Elemente angemessen einbinden, ohne die Automatisierbarkeit der Messung völlig zu vernachlässigen. Dabei sollte möglichst Know-How aus anderen Disziplinen (z. B. Information Retrieval) genutzt werden. Grundsätzlich muss bei der Entwicklung von Metriken darauf geachtet werden, dass sie durch Messverfahren operationalisierbar sind und dass diese reproduzierbare Ergebnisse liefern.

Neben der reinen Definition von Metriken ist es weiterhin entscheidend, dass diese in der Praxis eine möglichst breite Unterstützung finden (im Sinne von Quasistandards). Hierzu sind geeignete Einführungsmethoden und Kostenmodelle zu entwickeln und zu evaluieren. Diese sind zur fundierten Argumentation für die DQM-Einführung gegenüber der Organisationsleitung unerlässlich.

Zur Sicherung einer hinreichend hohen Effizienz des DQM auch bei sehr großen Datenmengen sollten möglichst DBMS-nahe Operatoren zur Verfügung stehen (Beispiel: explizite Unterstützung des ETL-Prozesses in Oracle 9i). Massive Effizienzprobleme sind insbesondere beim DQM in föderierten und Web-basierten Informationssystemen zu erwarten, da hier zumindest ein Teil der DQM-Maßnahmen zur Anfragezeit durchgeführt werden muss. Dies erfordert ggf. die Entwicklung neuartiger Ansätze.

DQM ist eine organisationsweite, wenn nicht sogar organisationsübergreifende Aufgabe, die insbesondere auch die operativen Quellsysteme einzubeziehen hat. Ziel muss es sein, bereinigte und konsolidierte Daten aus einem DWS – soweit möglich und sinnvoll – in die operativen Systeme zurückzuleiten, um auch quellenseitig Datenqualitätsanforderungen erfüllen zu können. Die Integration von DQM-Maßnahmen in operative Abläufe von Organisationen ist noch weitgehend unerforscht.

Weitere Herausforderungen, die hier allerdings nicht näher betrachtet werden, sind:

- DQM auf semistrukturierten (z. B. XML-basierten) Daten
- Reengineering von Metadaten
- Interoperabilität von ETL-Werkzeugen.

13.2.2 Perspektiven

Bevor die im vorigen Abschnitt diskutierten Probleme angegangen werden können, ist es zunächst erforderlich, die bisher gewonnenen Forschungsergebnisse in der Praxis zu evaluieren, im Idealfall an großen, möglichst international tätigen Organisationen, die – bei positiver Evaluierung – eine Signalwirkung auf andere Organisationen ausüben können. Ziel der Evaluierung sollte also nicht ausschließlich ein „Proof of Concepts“ sein (vgl. Kapitel 11), sondern auch eine Etablierung der Forschungsergebnisse in der Praxis. Es ist außerdem zu erwarten, dass eine solche Evaluierung neue Anstöße für die Forschung hervorbringt.

Einen bedeutenden Beitrag könnte die Entwicklung von Ansätzen zur Minimierung des Spezifizierungsaufwandes von DQM-Maßnahmen für den Anwender darstellen. Es ist anzustreben, die Lücke zwischen der Verständniswelt des Anwenders und der technischen (datenbanknahen) Realisierung der DQM-Maßnahmen zu überbrücken und außerdem die Wartbarkeit und Wiederverwendbarkeit von Spezifikationen zu optimieren. Analog zur klassischen 3-Ebenen-Modellierung von Datenbanken könnte in diesem Kontext eine Unterscheidung zwischen einer konzeptuellen, einer logischen (z. B. deklarativ wie in [GFS+01]) und einer physischen Spezifikationsebene für DQM-Maßnahmen sinnvoll sein. Die Transformation zwischen den Ebenen sollte durch eine Methodik und/oder ein Softwarewerkzeug unterstützt werden.

Voraussetzung für die Entwicklung von hochautomatisierten, flexiblen Softwarewerkzeugen zum DQM ist ein umfassendes, flexibles und standardisiertes Metadatenmanagement. Mit der Spezifizierung des CWM-Standards durch die OMG [Obj00] ist ein erster Schritt in diese Richtung bereits getan. Es müssen nun jedoch dringend Werkzeuge folgen, die diesen Standard implementieren und damit die Basis für eine breite Nutzung des CWM bilden können.

Ein weiterer wichtiger Aspekt ist die Verbesserung der Erfassung von Nutzeranforderungen. Ein effektives DQM setzt voraus, dass bekannt ist, was der Kunde (Nutzer) will bzw. welche Anforderungen an Daten zu stellen sind, die für bestimmte Aufgaben verwendet werden sollen. In der gängigen Praxis wird auf die Erfassung von Datenqualitätsanforderungen häufig völlig verzichtet. Ein Grund dafür ist das Fehlen von Verfahren zur Operationalisierung von Anforderungen, die mit Hilfe gängiger Erhebungsmethoden ermittelt wurden. Ansätze aus den Bereichen Software Engineering (insb. Requirements Engineering) und BWL (insb. Quality Function Deployment, vgl. Abschnitt 4.4.1) könnten bei der Spezifizierung solcher Verfahren helfen.

Ferner sollte ein erhöhtes Augenmerk auf die automatisierte Korrektur von Inkonsistenzen gelegt werden, um die Effizienz des DQM auch bei großen Datenvolumina sicherzustellen. Erste Data Mining-basierte Ansätze haben hier bereits erfolgversprechende Ergebnisse geliefert [DJK00, HW00, HGG01]. Es besteht jedoch noch weiterer Forschungsbedarf, insbesondere im Hinblick auf die Einbindung domänenspezifischen Wissens und die Optimierung der Effizienz sowie des Precision & Recall solcher Verfahren. Bei sehr großen Datenmengen sind darüber hinaus spezielle Vorverarbeitungsschritte notwendig, z. B. zur Bewertung der Angemessenheit bzw. Reduzierbarkeit von Daten.

In modernen Organisationen stellen integrierte, qualitätsgesicherte Daten elementare Ressourcen für die taktische und strategische Entscheidungsunterstützung dar. Die in dieser Arbeit durchgängig verwendete Analogie zwischen Daten und Produkten sowie zwischen Datenintegrations- und Fertigungsprozessen sollte daher in Organisationen konsequent genutzt werden, um Konzepte und Methoden, die sich im Rahmen der Produktrealisierung etabliert haben, auch auf die Ressource „Daten“ anzuwenden. Als abschließendes Beispiel für eine solche Adaption etablierter Konzepte sei die Idee eines

Supply Chain Management für Datenintegrationsprozesse genannt. Insbesondere bei DWS mit externen Datenquellen, also einer organisationsübergreifenden „Datenlieferkette“ könnte so eine systematische Planung und Steuerung der Datenbewirtschaftung realisiert werden, bei der die Datennachfrage (gegeben durch Analyseerfordernisse), die Verarbeitungsressourcen für die Datenintegration sowie der Datennachschub von den Quellen aufeinander abgestimmt sind. Unter Hinzunahme der in der vorliegenden Arbeit entwickelten Lösungen zum Datenqualitätsmanagement wäre dies ein weiterer entscheidender Baustein zur zeitnahen, flexiblen Versorgung von DWS-Nutzern mit qualitativ hochwertigen Daten.

Zur Homepage der Dissertation

Anhänge und Verzeichnisse

Anhang A

Marktübersicht zu Softwarewerkzeugen für das DQM

In diesem Kapitel werden kommerzielle Softwarewerkzeuge vorgestellt, die im Bereich des DQM eingesetzt werden können. Die Auflistung soll einen Überblick geben und erhebt keinen Anspruch auf Vollständigkeit, da der Markt in diesem Bereich sehr dynamisch, heterogen und unübersichtlich ist. Die Angaben zu den Werkzeugen wurden z. T. aus den Web-Präsentationen der jeweiligen Anbieter extrahiert, z. T. aus Marktübersichten in der Fachliteratur (z. B. [Eng99]) übernommen. Sie können daher nur bedingt objektiv sein. Für die Richtigkeit der Angaben kann keine Gewähr übernommen werden.

Folgende (grobe) Kategorien von Werkzeugen können unterschieden werden (in Klammern die Kürzel, wie sie in den nachfolgenden Tabellen verwendet werden):

- *Auditing (A):* Analysewerkzeuge, die Daten aus Datenbanken oder Prozessen extrahieren, diese auf Qualitätsaspekte untersuchen und Ergebnisse in einem Bericht festhalten.
- *Cleansing (C):* Werkzeuge, die Daten aus heterogenen Quellen extrahieren, vereinheitlichen, transformieren, anreichern, oft im Rahmen einer Migration von Daten in ein DWS (ETL-Werkzeuge, vgl. Abschnitt 2.2).
- *Record Linkage (L):* Werkzeuge, die Duplikate in Datenbeständen aufspüren und diese durch Verschmelzen beseitigen (Subkategorie von C, vgl. Abschnitt 6.2.5).
- *Metadatenmanagement (M):* Werkzeuge, die z. B. Geschäftsregeln, Schemainformationen und Begriffstaxonomien verwalten und so zur Qualitätssicherung beitragen.
- *Namen/Adressen (N):* Werkzeuge zur Vereinheitlichung und Korrektur von Namen (Separierung von Namensbestandteilen etc.) und Adressen (Ergänzung/Korrektur von Postleitzahlen etc.), oft unter Zuhilfenahme von (gebietsbezogenen) Namens-/Adressregistern und geographischen Informationssystemen (Subkategorie von C).
- *Prävention (P):* Werkzeuge, die dem Auftreten von Datenqualitätsmängeln vorbeugen (Eingabekontrollen, Integritätsprüfungen etc.).
- *Regelentdeckung (R):* Werkzeuge, die nach Mustern und Regelmäßigkeiten in Datenbeständen suchen, diese als Geschäftsregeln formulieren und anderen Werkzeugen (z. B. denen der Kategorie A) zur Verfügung stellen.

Viele der im folgenden vorgestellten Werkzeuge gehören mehr als einer Kategorie an. Es dominieren die klassischen ETL-Werkzeuge zum Aufbau von Data Warehouses aus heterogenen Datenquellen

(Kategorie C). Überraschend hoch ist die Vielfalt an Werkzeugen zum Record Linkage und zur Nachbearbeitung von Namen und Adressen (Kategorien L und N). Werkzeuge zum Qualitätsaudit und zur Fehlerprävention sind hingegen klar in der Minderheit. Werkzeuge zur Unterstützung des DQM im Sinne des CLIQ-Ansatzes gibt es bisher nicht, allerdings können die dabei anfallenden Aufgaben zumindest teilweise durch eine Kombination verschiedener Werkzeuge (z. B. zur Migration, zum Record Linkage und zur Regelentdeckung) abgedeckt werden, wie in Kapitel 10 der vorliegenden Arbeit praktiziert.

A.1 Tabellarische Charakterisierung der Werkzeuge

Im Folgenden werden die einzelnen Werkzeuge – alphabetisch nach Anbietern sortiert – grob charakterisiert.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Acta Technology (Palo Alto, CA, USA) www.acta.com	ActaWorks Data Server	C	AIX, HP-UX, Solaris, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Applied Database Technology (Kirkland, WA, USA) www.adbtech.com	Data Mapper	C, M	Windows

Bemerkungen: Data Migration-Werkzeug für Oracle. Primär- und Fremdschlüssel sowie Check Constraints können aus dem Data Dictionary ausgelesen und mit zusätzlichen Validierungsregeln angereichert werden.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
ArkiData (Lombard, IL, USA) www.arkidata.com	Arkistra	A, C	Windows

Bemerkungen: Bietet Funktionalität für „what-if“-Szenarien und Reversibilität von Transformationen.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Ascential Software (Westboro, MA, USA) www.ascentialsoftware.com	DataStage XE, Quality Manager (siehe Abschnitt 5.3)	A, C, M	Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Augustyn + Company (Berkeley, CA, USA) www.dqms.com	DQMS, Data View	A, C	Windows

Bemerkungen: Verwaltet Zeitreihendaten und verwendet diese zum Datenqualitätstracking. Bietet weiterhin reversible Cleansing-Verfahren.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
The Client Server Factory (Toronto, Ontario, Canada) www.tcsf.com	PumpItUp	C	Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Coglin Mill (Rochester, MN, USA) www.coglinmill.com	Rodin	C, M	Windows, IBM iSeries, AS/400

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Cognos (Ottawa, Canada) www.cognos.com	Cognos Business Intelligence Platform	C, M	Windows, HP-UX

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Computer Associates (ehemals Platinum Technology) (Oakbrook Terrace, IL, USA) ca.com/products/decisionbase	InfoRefiner, InfoTransport, InfoPump	C, M	Mainframe, Unix, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Compuware (Farmington Hills, MI, USA) www.compuware.com	File-AID/Data Solutions, File-AID/Express	C	Unix, MVS, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Cross Access (Santa Clara, CA, USA) www.crossaccess.com	eXadas eData Engine, CrossXpress	C, M	Unix, Windows, OS/390

Anbieter	Produkt(e)	Kateg.	Plattform(en)
D2K (San Jose, CA, USA) www.d2k.com	Information Logistics Network (ILN)	C	Unix, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
DataFlux (SAS-Tochter) (Raleigh, NC, USA) www.dataflux.com	dfPower Studio, BlueFusion SDK	A, C, L, N	Windows

Bemerkungen: Ermöglicht quellenspezifische Datenqualitätsanalysen (zur Identifizierung von problematischen Quellen).

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Data Junction (Austin, TX, USA) www.datajunction.com	Data Junction Integration Suite	C, M	Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
DataMirror (Markham, Ontario, Canada) www.datamirror.com	Transformation Server	C	Windows, Unix, OS/390, OS/400

Anbieter	Produkt(e)	Kateg.	Plattform(en)
DataTools (Sydney, Australia) www.datatools.com.au	Twins III	C, N, L	Windows

Bemerkungen: Stellt Parser für Namen und Adressen sowie Funktionen zur Vereinheitlichung und Beseitigung von Duplikaten (inkl. teilautomatisiertem Merging) bereit.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Data Mentors (Tampa, FL, USA) www.datamentors.com	ValiData, DataFuse	A, C, L, N	Windows, Unix

Bemerkungen: ValiData unterstützt die Erstellung von attributspezifischen Werteverteilungen und tupelbezogenen Fehlerprotokollen. „Vertikale“ Datenanalyse zur Prüfung auf Homonym- und Synonymfehler (verursacht z. B. durch falsch konfigurierte Record Linkage-Software). DataFuse bereinigt Namens- und Adressdatenbanken.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
DBE Software (McLean, VA, USA) www.dbesoftware.com	DB-Examiner	M, P	Windows

Bemerkungen: Wissensbasierte Analyse von Datenbankschemata zum Aufspüren von Modellierungsfehlern.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Dedupe Technologies (Fareham, Hampshire, UK) www.dedupe.com	Dedupe (Component Library, Batch)	C, L	Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Evoke Software (San Francisco, CA, USA) www.evokesoft.com	Axio	A, C, L, M, P, R	Solaris, HP-UX, AIX, Windows

Bemerkungen: Dient der Analyse von Datenquellen (Inhalt, Schema, referentielle Integrität etc.) und dem Metadaten-Reengineering.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Evolutionary Technologies (Austin, TX, USA) www.eti.com	ETI*EXTRACT	C	Solaris, HP-UX, AIX, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Firstlogic (La Crosse, WI, USA) www.firstlogic.com	i. d. Centric Data Quality Suite, Postalsoft	C, L, N	Unix, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
FUZZY! Informatik AG (Ludwigsburg, Germany) www.fazi.de	Fuzzy! Double, Fuzzy! Post / Name / Bank / Branch etc.	C, N	Unix, Windows

Bemerkungen: Fuzzy-basiertes, auf Oracle aufbauendes System zur Erstellung von fehlertoleranten Suchsystemen; verschiedene Validierungssysteme als Add-Ons verfügbar.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Gamma Research (Huntsville, AL, USA) gamma-research.fws1.com	OCRProof	P	Windows

Bemerkungen: Vereinfacht die Verarbeitung von per OCR eingelesenen Dokumenten durch eine Gegenüberstellung von gescannten Bitmap-Bildern und OCR-konvertiertem Text und eine Qualitätsbewertung des OCR-Prozesses (Darstellung der Fehlerraten in Kontrollkarten).

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Gladstone Computer Services (Saugus, CA) www.happyrock.com	Gladstone Analysis Package (GAP), Gladstone Conversion Package (GCP)	A, C	MVS, VSE, AS/400, AIX, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Group 1 Software (Lanham, MD, USA) www.g1.com	DQ Plus, Code-1 Plus, Merge/Purge Plus	C, L, N	IBM Mainframe, AS/400

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Harland (Lake Mary, FL, USA) www.harland-orlando.com	Harland Warehouse Data-Prep System	C, L, N	MVS

Anbieter	Produkt(e)	Kateg.	Plattform(en)
helpIT Systems (Leatherhead, Surrey, UK) www.helpit.co.uk	matchIT Suite	C, L, N	Windows

Bemerkungen: Korrigiert Adressen an Hand von offiziellen Adressregistern.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
High-Tower (Irvine, CA, USA) www.high-tower.com	TowerView	A, P, R	IRIX, BSD, HP-UX, Solaris, Windows

Bemerkungen: Verfolgt und analysiert Änderungen in Datenbeständen.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Hopewiser (Cheshire, UK) www.hopewiser.co.uk	Probe, Rains, De-Dupe	C, L, N, P	AS/400, Windows, OS/2, Unix

Anbieter	Produkt(e)	Kateg.	Plattform(en)
HotData (Austin, TX, USA) www.hotdata.com	HotData 1-to-1	A, C, N	Windows

Bemerkungen: Online-Dienst zur Bereinigung und Anreicherung von Daten (Adressen, Telefonnummern, demographische Daten etc.) von unterschiedlichen Providern mittels Plug-Ins sowie zur Bereitstellung der Daten für Desktop-Applikationen.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Hummingbird (North York, Ontario, Canada) www.hummingbird.com	Genio Suite, Genio Miner	C, M, R	Windows, Solaris, AIX, HP-UX

Anbieter	Produkt(e)	Kateg.	Plattform(en)
IBM (San Jose, CA, USA) www.ibm.com/software/data	Visual Warehouse	C, M	AIX, OS/2, OS/400, Windows, Solaris

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Identex (Crawley, West Sussex, UK) www.identex.co.uk	Identex Data Quality Management	C, L, N, P	Unix, OS/400, MVS, Windows

Bemerkungen: Softwarewerkzeuge zur Bereinigung und Anreicherung von internationalen Namens- und Adressdaten mit Hilfe von globalen Registern.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Informatica (Palo Alto, CA, USA) www.informatica.com	PowerCenter	C, M	Windows, AIX, HP-UX, Solaris, Tru64

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Information Builders (New York, NY, USA) www.informationbuilders.com	WebFOCUS ETL	C	Windows, Unix, AS/400, OpenVMS

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Innovative Systems (Pittsburgh, PA, USA) www.innovativesystems.net	Innovative-Dictionary, Innovative-Review	C, L, N	MVS, Solaris, HP-UX, AIX, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Kismet Analytical (Arlington, VA, USA) www.kismeta.com	KisMeta Validator, Analyst	A, M, P	Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Metagenix (Durham, NC, USA) www.metagenix.com	MetaRecon	A, C, M, P	Windows, Linux, Solaris

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Metagon Technologies (Matthews, NC, USA) www.metagon.com	DQtransform	C	Windows, Unix

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Mobius (Rye, NY, USA) www.mobius-inc.com	INFOPAC-ABS	A, P	MVS

Bemerkungen: Werkzeug zur automatisierten „Balancierung“ von numerischen Daten über Applikationsgrenzen hinweg. Entdeckt „unbalancierte“ (d. h. integritätsverletzende) Situationen.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Omikron (Pforzheim, Germany) www.deduplication.com	TwinFinder	C, L, N	Windows

Bemerkungen: Verwendet den lingual-mathematischen FACT-Algorithmus zum Fuzzy Pattern-Matching.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Oracle (ehemals Carleton Corp.) (Redwood Shores, CA, USA) www.oracle.com	Pure*Integrate	C, L, M	Unix, Windows, HP 9000, OS/2

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Paladyne (Lake Mary, FL, USA) www.paladyne.com	Datagration e-Business Suite	C, L, R	Windows

Bemerkungen: Identifizierung und Quantifizierung von Datenmustern mit Hilfe einer regelbasierten Engine.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Pitney Bowes docSense (Lisle, IL, USA) www.docsense.com	docSense Data Quality	C, N	IBM Mainframe, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Q-Span Information Systems (Lexington, MA, USA) www.q-span.com	DataTrust	A, C, M, P, R	Windows

Bemerkungen: Regelbasiertes System mit Feedback-Funktionalität.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
QAS Systems (London, UK) www.qas.com	QuickAddress (Pro, Batch)	C, N	Unix, OS/2, Mac, VAX, MVS, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Quess (Colorado Springs, CO, USA) www.quess.com/nodupes.html	NoDupes	C, L	Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
RuleQuest Research (St Ives, Australia) www.rulequest.com	GritBot	A, R	Unix, Windows

Bemerkungen: Verwendet Data Mining-Verfahren zum Aufspüren von Anomalien in Datenbeständen, arbeitet ausschließlich auf „flachen“ Dateien, ist anwendbar auf nominale und numerische Attribute sowie Datumsattribute (vgl. Abschnitt 5.2.3).

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Sagent (ehemals Qualitative Marketing Software) (Clearwater, FL, USA) www.sagent.com	Centrus Suite (Online, Desktop)	C, N	Windows, Unix, Mainframe

Anbieter	Produkt(e)	Kateg.	Plattform(en)
SAS Institute (Cary, NC, USA) www.sas.com	SAS/Warehouse Administrator Transformation Engine	C, M	Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Search Software America (Reading, UK) www.searchsoftware.co.uk	Data Clustering Engine	C, L, N	Unix, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Systemfabrik (Hamburg, Germany) www.systemfabrik.com	InformationIntegration	C	Windows, AIX, Solaris, HP-UX

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Taurus Software (Palo Alto, CA, USA) www.taurus.com	DataBridge, BridgeWare	C	HP 3000/9000, Solaris, DEC, AIX, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Trillium Software (Billerica, MA, USA) www.trilliumsoft.com	Trillium Software System (Converter, Parser, Geocoder, Matcher)	A, C, L, N, P	MVS, OS/390, CICS, Unix, AS/400, Windows

Bemerkungen: Kann auch in Applikationen eingebettet werden, um ad-hoc Validierungen und Bereinigungen durchzuführen, läuft unter Java.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Unitech Systems (Naperville, IL, USA) www.unitechsys.com	INstream, INsulate, ACR/Plus, ACR/XP, ACR/Data	A, C	MVS, Unix, AS/400, Windows

Anbieter	Produkt(e)	Kateg.	Plattform(en)
Vality Technology (Boston, MA, USA) www.vality.com	INTEGRITY Data Re-engineering Environment, Real Time	A, C, L, N	Unix, OS/390, OS/400, Windows

Bemerkungen: Enthält Funktionen zum probabilistischen Matching (siehe auch Abschnitt 10.2.4). Beinhaltet APIs für C, Java und Cobol.

Anbieter	Produkt(e)	Kateg.	Plattform(en)
WizSoft (Israel) www.wizsoft.com	WizRule	A, R	Windows

Bemerkungen: Verwendet Data Mining-Verfahren zum Aufspüren von Anomalien in Datenbeständen.

A.2 Abschließender Überblick

Die Tabellen A.1 und A.2 geben einen Überblick über die Zugehörigkeit der einzelnen Werkzeuge zu den zu Beginn des Kapitels definierten Kategorien.

Anbieter / Kategorie	Audit (A)	Cleansing (C)	Record Linkage (L)	Metadatenmanagement (M)	Namen/Adressen (N)	Prävention (P)	Regelentdeckung (R)
Acta Technology		•					
Applied Database Technology		•		•			
ArkiData	•	•					
Ascential Software	•	•		•			
Augustyn + Company	•	•					
The Client Server Factory		•					
Coglin Mill		•		•			
Cognos		•		•			
Computer Associates		•		•			
Compuware		•					
Cross Access		•		•			
D2K		•					
DataFlux	•	•	•		•		
Data Junction		•		•			
DataMirror		•					
DataTools		•	•		•		
Data Mentors	•	•	•		•		
DBE Software				•		•	
Dedupe Technologies		•	•				
Evoke Software	•	•	•	•		•	•
Evolutionary Technologies		•					
Firstlogic		•	•		•		
FUZZY! Informatik AG		•			•		
Gamma Research						•	
Gladstone Computer Services	•	•					

Tabelle A.1: Zuordnung von Anbietern zu Werkzeugkategorien, Teil 1

Anbieter / Kategorie	Audit (A)	Cleansing (C)	Record Linkage (L)	Metadatenmanagement (M)	Namen/Adressen (N)	Prävention (P)	Regelentdeckung (R)
Group 1 Software		•	•		•		
Harland		•	•		•		
helpIT Systems		•	•		•		
High-Tower	•					•	•
Hopewiser		•	•		•	•	
HotData	•	•			•		
Hummingbird		•		•			•
IBM		•		•			
Identex		•	•		•	•	
Informatica		•		•			
Information Builders		•					
Innovative Systems		•	•		•		
Kismet Analytical	•			•		•	
Metagenix	•	•		•		•	
Metagon Technologies		•					
Mobius	•					•	
Omikron		•	•		•		
Oracle		•	•	•			
Paladyne		•	•				•
Pitney Bowes Software Systems		•			•		
Q-Span Information Systems	•	•		•		•	•
QAS Systems		•			•		
Quess		•	•				
RuleQuest Research	•						•
Sagent		•			•		
SAS Institute		•		•			
Search Software America		•	•		•		
Systemfabrik		•					
Taurus Software		•					
Trillium Software	•	•	•		•	•	
Unitech Systems	•	•					
Vality Technology	•	•	•		•		
WizSoft	•						•

Tabelle A.2: Zuordnung von Anbietern zu Werkzeugkategorien, Teil 2

Zur Homepage der Dissertation

Glossar

Das vorliegende Glossar beschreibt die wichtigsten in dieser Arbeit verwendeten Begriffe in alphabetischer Reihenfolge und folgender Notation:

∼	steht als Kürzel für den erklärten Begriff,
↑xxx	verweist auf einen anderen Begriff des Glossars,
(xxx)	enthält die Abkürzung oder ein Synonym des Begriffs.

Die Begriffe entstammen vorwiegend aus den Themenbereichen Data Warehousing und Qualitätsmanagement (hier im Wesentlichen aus der Norm ISO 9000:2000 [Deu00a]). Ergänzend wurden Begriffe aus den Bereichen Datenbanken und Objektorientierung sowie im Verlauf dieser Arbeit eingeführte Begriffe aufgenommen.

Abstrakte Klasse: ↑Klasse, die gemeinsame ↑Merkmale ihrer ↑Subklassen festlegt, wobei von der abstrakten ↑Klasse selbst keine ↑Objekte existieren dürfen.

Aggregation:

1. Zusammenfassen von Daten mittels einer Berechnungsvorschrift.
2. Spezielle ↑Assoziation, die eine „Ganzes-Teil“-Beziehung zwischen zwei Komponenten beschreibt. Gegenüber einer ↑Komposition sind die Instanzen der als „Teil“ betrachteten Komponente jedoch nicht von der als „Ganzes“ aufgefassten Komponente abhängig.

Anforderung: Erfordernis oder Erwartung, das oder die festgelegt, üblicherweise vorausgesetzt oder verpflichtend ist.

Assoziation: Semantische Beziehung zwischen zwei oder mehreren ↑Klassen, ↑Objekten, Typen oder anderen Elementen eines ↑Schemas.

Attribut: Eigenschaft eines ↑Objektes einer ↑Klasse bzw. eines ↑Tupels einer ↑Relation.

Audit: Systematischer, unabhängiger und dokumentierter ↑Prozess zur Erlangung und objektiven Auswertung von Informationen (Aufzeichnungen, Tatsachenfeststellungen etc.) über die Erfüllung gegebener ↑Anforderungen.

Auswertungswerkzeug: Oberbegriff für diejenigen Komponenten eines ↑Data Warehouse-Systems, die der Untersuchung und Präsentation von Daten dienen. Die dabei eingesetzten Methoden reichen von einfachem Reporting bis hin zu komplexen Analysemethoden wie ↑Online Analytical Processing und ↑Data Mining.

Basisklasse: Allgemeinere ↑Klasse einer ↑Generalisierung.

Blocking: Verfahren zur Reduzierung der Laufzeitkomplexität von (probabilistischen) ↑Record Linkage-↑Prozessen. Beim ∼ wird der Suchraum nach bestimmten Kriterien in Blöcke eingeteilt, die in jeweils einem Umlauf analysiert werden.

CLIQ-Vorgehensmodell: In dieser Arbeit entwickelte Methodik zur inkrementellen ↑Datenintegration in ↑Data Warehouse-Systemen.

CLIQWorkbench: In dieser Arbeit entwickeltes Softwaresystem zur Umsetzung des ↑CLIQ-Vorgehensmodells.

Common Warehouse Metamodel (CWM): Spezifikation der OMG, die einen standardisierten Austausch von ↑Metadaten zwischen Komponenten von ↑Data Warehouse-Systemen festlegt, basierend auf der ↑Unified Modeling Language, XML und XMI (XML Metadata Interchange).

Constraint: Semantische Bedingung bzw. Restriktion.

Customer Relationship Management (CRM): Automatisierter und kontinuierlicher ↑Prozess zur Prognose des Kundenbedarfs, des Kundenverhaltens und/oder der Kundenprofitabilität. Ziele des ∼ sind die Entscheidungsunterstützung für Geschäfts- und Marketingstrategien, die Reduzierung von Kosten sowie die Gewährleistung eines optimalen Kundenservice an jedem Punkt des Kontaktes mit dem Kunden.

Data Clean(s)ing: Teil↑prozess der ↑Datenintegration, bei dem eine Bereinigung von Daten vorgenommen wird, um die ↑Qualität der Daten zu verbessern.

Data Migration: Zusammenfassender Begriff für ↑Transformationen, die der Vereinheitlichung von heterogenen Daten dienen.

Data Mining: Automatische Analyse eines großen und/oder komplexen Datenbestandes mit dem Ziel, signifikante Muster oder Trends zu entdecken, die sonst unerkannt geblieben wären. ∼ kann als Phase des ↑Knowledge Discovery in Databases aufgefasst werden.

Data Profiling: Untersuchung der Werte von ↑Attributen im Hinblick auf genutzte Wertebereiche, Mittelwert und Varianz numerischer Werte, Häufigkeit diskreter Werte, Häufigkeit von ↑Nullwerten etc.

Data Warehouse: Physische ↑Datenbank, in der sowohl ↑Schema als auch Daten integriert sind und bei der die Analyse von Daten im Vordergrund steht. Dies spiegelt sich oft in einem ↑multidimensionalen Schema wider. Häufig, aber nicht notwendigerweise findet eine ↑Historisierung von Daten statt.

Data Warehouse-System (DWS): ↑Informationssystem, das alle für das ↑Data Warehousing notwendigen Hard- und Softwarekomponenten (außer den ↑Datenquellen) umfasst.

Data Warehousing: Dynamischer, softwaregestützter ↑Prozess der Datenverarbeitung in einem ↑Data Warehouse-System, von der ↑Extraktion aus den ↑Datenquellen bis hin zur Auswertung.

Datenbank: Strukturierte Sammlung von Daten, die mit Hilfe eines ↑Datenbankmanagementsystems verwaltet werden.

Datenbankmanagementsystem (DBMS): Aus einer Speicherungs- und einer Verwaltungskomponente bestehendes Programm. Die Speicherungskomponente erlaubt es, Daten und ihre Beziehungen in einer ↑Datenbank abzulegen. Die Verwaltungskomponente stellt Funktionen und Sprachmittel zur Pflege der Daten zur Verfügung.

Datenbanksystem (DBS): Kombination eines ↑Datenbankmanagementsystems mit mindestens einer ↑Datenbank.

Datenintegration: ↑Prozess der ↑Transformation, Bereinigung und Zusammenführung von aus heterogenen ↑Datenquellen extrahierten Daten. Integrierte Daten werden im ↑Operational Data Store (bzw. ↑Zielbereich) abgelegt.

Datenintegrität: Oberbegriff für die drei Aspekte ↑Datenkonsistenz, ↑Datensicherheit und ↑Datenschutz.

Datenkonsistenz: Logische Widerspruchsfreiheit von Daten bzgl. vorgegebener Bedingungen.

Datenmodell: (Semi-)Formale Sprache zur strukturierten Beschreibung von Daten und ihren Beziehungen sowie von Operationen auf diesen Daten.

Datenobjekt: Im Kontext dieser Arbeit ist ein ∼ eine durch Projektion (SELECT ... FROM) und Selektion (WHERE ...) definierte Menge von ↑Tupeln einer ↑Relation.

Datenprodukt: Im Kontext dieser Arbeit ist ein ∼ die ↑Extension einer ↑Relation im ↑Zielbereich, d. h. eine Menge von integrierten ↑Tupeln als Ergebnis eines ↑Datenintegrations↑prozesses.

Datenqualität: Grad, in dem ein Satz inhärenter ↑Merkmale eines ↑Datenprodukts ↑Anforderungen erfüllt.

Datenqualitätsmanagement: Aufeinander abgestimmte Tätigkeiten zum Leiten und Lenken einer Organisation bzgl. ↑Datenqualität.

Datenqualitätsmanagementsystem: System zur Durchführung eines ↑Datenqualitätsmanagements.

Datenquelle: Organisationsinterne oder -externe ↑Datenbank oder andere Daten speichernde Komponente (Datei, WWW-Seite etc.), aus der Daten für das ↑Data Warehousing entnommen werden.

Datenschutz: Schutz von Personen gegen Missbrauch der sie betreffenden Daten.

Datensicherheit: Technische und organisatorische Maßnahmen zur Sicherung von Daten gegen Verlust, Beschädigung und unerlaubten Zugriff.

Domäne: Benannte Menge von Werten.

DWS-Manager: Administrationswerkzeug, das für die Steuerung und Überwachung der in einem ↑Data Warehouse-System stattfindenden ↑Prozesse zuständig ist.

Enterprise Application Integration (EAI): ↑Prozess der Errichtung einer IT-Infrastruktur, die es erlaubt, in einfacher (wartbarer) Weise neue oder geänderte IT-gestützte Geschäfts↑prozesse in einem Unternehmen bzw. einer Organisation zu etablieren.

Entität:

1. Individuelles Objekt der Real- oder Vorstellungswelt
2. Modellierungskonstrukt des ↑Entity/Relationship-Modells zur Beschreibung von Objekten, zu denen Informationen in einer ↑Datenbank gesammelt werden sollen.

Entity/Relationship-Modell (E/R-Modell): ↑Datenmodell zur Beschreibung von statischen Strukturen der Anwendungswelt. Das ∼ stellt hauptsächlich die drei Modellierungskonstrukte ↑Entität, ↑Assoziation und ↑Attribut bereit.

ETL-Prozess (↑Extraktion – ↑Transformation – ↑Laden): Teil↑prozess des ↑Data Warehousing, der die Datenverarbeitung von den ↑Datenquellen bis zum ↑Data Warehouse beschreibt.

Extension: Konkrete Ausprägung einer ↑Relation oder ↑Datenbank.

Extraktion: ↑Prozess der Selektion von (i. Allg. neuen oder modifizierten) Daten aus einer ↑Datenquelle und deren Bereitstellung zur ↑Datenintegration.

Fehler: Ungewünschte Abweichung vom angestrebten Verarbeitungs↑prozess, ausgelöst durch fehlerhafte Bedienung, Software- oder Hardwareprobleme.

Föderiertes Datenbanksystem (FDBS): System von möglicherweise verteilten, heterogenen ↑Datenbanksystemen, die über einen expliziten Föderierungsdienst bei weitgehender Erhaltung ihrer lokalen Autonomie kooperieren.

Fremdschlüsselattribut: ↑Attribut oder ↑Attributkombination, das/die in einer anderen ↑Relation ↑Primärschlüsselattribut ist.

Generalisierung: Modellierungskonstrukt, bei dem gleiche Eigenschaften verschiedener ↑Klassen nur einmal für eine gemeinsame ↑Basisklasse modelliert werden. Siehe auch ↑Vererbung.

Historisierung: In ↑Data Warehouse-Systemen gängige Art der Datenspeicherung, bei der ein ↑Tupel oder ↑Objekt in einer ↑Datenbank im Zuge eines Update nicht überschrieben, sondern mit einem Gültigkeitszeitraum versehen und mit dem aktualisierten ↑Tupel bzw. ↑Objekt verknüpft wird, um eine Analyse von zeitlichen Trends zu ermöglichen.

Homonymfehler: Fälschliche Zusammenführung zweier ↑Tupel oder ↑Objekte, die unterschiedliche ↑Entitäten beschreiben, in einem ↑Record Linkage-↑Prozess.

Hub-and-Spoke–Architektur: Architektur, bei der eine Speichereinheit (hier: ↑Data Warehouse oder ↑Operational Data Store bzw. ↑Zielbereich) eine zentrale Sammel- und Verteilungsfunktion einnimmt.

Identität: Eigenschaft eines ↑Objekts, die es von allen anderen ↑Objekten unterscheidet und die nicht verändert werden kann.

Index: Physische Datenstruktur zur Erhöhung der Geschwindigkeit des Zugriffs auf Daten in einer ↑Datenbank.

Informationssystem:

1. *Im weiteren Sinne:* Gesamtes informationsverarbeitendes Teilsystem einer Organisation.
2. *Im engeren Sinne:* Computergestützter Teil eines ∼s im weiteren Sinne.

Integrationsschicht: Zusammenfassende Bezeichnung für diejenigen Komponenten eines ↑Data Warehouse-Systems, die funktional zwischen den ↑Datenquellen und dem ↑Operational Data Store angesiedelt sind.

Intension: Struktur einer ↑Relation oder ↑Datenbank.

Klasse: Zusammenfassung einer ↑Klassendefinition und der Menge von ↑Objekten, die gemäß dieser Definition erzeugt wurden.

Klassendefinition: Definition der ↑Merkmale von ↑Objekten einer ↑Klasse an Hand eines Klassennamens sowie einer Menge von ↑Attributen und ↑Methoden.

Knowledge Discovery in Databases (KDD): Nicht-trivialer ↑Prozess der Extraktion von implizitem, vormals unbekanntem und möglicherweise nützlichem Wissen aus Daten, bestehend aus Phasen zur Prozessplanung, zur Datenselektion, -bereinigung und -vorverarbeitung, zur Datenreduktion und -↑transformation, zum ↑Data Mining, zur Interpretation von Ergebnissen sowie zur Konsolidierung von entdecktem Wissen.

Kohäsion: Grad der Bindung zwischen den Elementen einer ↑Softwarekomponente.

Komposition: Spezielle ↑Assoziation, die eine „Ganzes-Teil“-Beziehung zwischen zwei Komponenten beschreibt. Gegenüber einer ↑Aggregation sind die Instanzen der als „Teil“ betrachteten Komponente von der als „Ganzes“ aufgefassten Komponente abhängig.

Konformität: Erfüllung einer ↑Anforderung.

Konsolidierung: Im Kontext dieser Arbeit synonym zu ↑Datenintegration.

Konsolidierungsbereich (KB): ↑Datenbank, die im ↑CLIQ-Vorgehensmodell zwischen den ↑Transformationsbereich und den ↑Zielbereich geschaltet ist und temporär bereits integrierte, aber noch nicht abschließend qualitätsgesicherte Daten speichert.

Laden: ↑Prozess der Übertragung von konsolidierten Daten aus einem ↑Operational Data Store in das analyseorientierte ↑Schema eines ↑Data Warehouse.

Löschanweisung: ↑Tupel, das die Löschung eines ↑Tupels in einer ↑Datenquelle beschreibt. Eine in einem ↑Quelldatenpuffer eingehende ∼ weist denselben lokalen ↑Schlüssel wie das gelöschte ↑Tupel auf. Im ↑Zielbereich führt eine ∼ i. Allg. zu einer Löschung oder ↑Historisierung des zugehörigen konsolidierten ↑Tupels.

Mangel: Nichterfüllung einer ↑Anforderung in Bezug auf einen beabsichtigten oder festgelegten Gebrauch.

Mehrfachvererbung: Form der ↑Vererbung, bei der eine ↑Klasse mehr als eine direkte ↑Basisklasse besitzt.

Merging: ↑Prozess zur Verschmelzung von ↑Tupeln oder ↑Objekten, die im Rahmen eines ↑Record Linkage-↑Prozesses zusammengeführt wurden.

Merkmal: Kennzeichnende Eigenschaft.

Messmittel: Messgerät, Software, Referenzmaterial, Hilfsmittel oder eine Kombination davon, benötigt für einen ↑Messprozess.

Messprozess: Satz von Tätigkeiten zur Ermittlung eines Wertes, der ein spezifisches ↑Attribut eines Untersuchungsobjekts beschreibt.

Messung: Anwendung eines ↑Messprozesses auf ein Untersuchungsobjekt.

Metadaten: Daten über Daten und ↑Prozesse, klassifizierbar in domänenspezifische (Geschäfts-)∼ (z. B. Begriffstaxonomien) und technische ∼ (z. B. Spezifikationen von ↑Transformations↑prozessen und Zugriffsstatistiken). ∼ leisten einen entscheidenden Beitrag zum ↑Datenqualitätsmanagement.

Metadaten-Repository: ↑Datenbankmanagementsystem zur Verwaltung von ↑Metadaten.

Methode: Operation, die ein ↑Objekt ausführen kann.

Metrik: Abbildung(-svorschrift) von Untersuchungsobjekten auf Zahlen oder Symbole mit dem Ziel, ein spezifisches ↑Attribut (z. B. ein ↑Qualitätsmerkmal) zu charakterisieren.

Monitor: Werkzeug, das den ↑Prozess des ↑Monitoring unterstützt.

Monitoring: ↑Prozess innerhalb des ↑Data Warehousing, in dem die ↑Datenquellen überwacht werden, um zu extrahierende Daten zu bestimmen.

Multidimensionales Schema: ↑Schema, in dem auszuwertende Kennzahlen (z. B. Verkaufszahlen eines Unternehmens) entlang einer Menge von Dimensionen (z. B. Zeit, ↑Produkt, Filiale) strukturiert werden.

Multiplizität: Anzahl von ↑Objekten, die an einer ↑Assoziation beteiligt sein können.

Nullwert: Ausgezeichneter Wert, der in jeder ↑Domäne enthalten ist und i. Allg. für „nicht vorhanden“ oder „nicht bekannt“ steht.

Objekt: Zusammenfassung einer Datenstruktur und der darauf anwendbaren ↑Methoden zu einer Einheit. Ein ∼ besitzt eine Struktur, einen Zustand, ein Verhalten und eine ↑Identität.

Online Analytical Processing (OLAP): Interaktive, explorative Analyse von Daten auf der Basis eines ↑multidimensionalen Schemas.

Online Transaction Processing (OLTP): Arbeits↑prozess, der von den klassischen operativen, transaktionsorientierten ↑Datenbankanwendungen verfolgt wird.

Open Information Model (OIM): Ein von der Meta Data Coalition unter Führung von Microsoft entwickelter Standard, der zahlreiche ↑Schemata zur Speicherung von ↑Metadaten definiert. Zur Spezifizierung und Darstellung dieser ↑Schemata wird die ↑Unified Modeling Language eingesetzt. Eine wesentliche Eigenschaft des ∼ besteht darin, dass die vordefinierten ↑Schemata anwendungsspezifisch erweitert werden können.

Operational Data Store (ODS): Physische ↑Datenbank, die konsolidierte, feingranulare Daten speichert. Daten können nicht nur hinzugefügt, sondern auch modifiziert werden. Im Mittelpunkt steht die Bereitstellung von Daten für ein ↑Data Warehouse sowie die Auswertung von Daten auf feingranularer Ebene (z. B. offene Posten).

Operatives Datenbanksystem: ↑Datenbanksystem zur Unterstützung des ↑Online Transaction Processing.

Plan-Do-Check-Act–Zyklus (PDCA-Zyklus): Regelkreis, der Demings Idee von einer „ständigen ↑Qualitätsverbesserung“ beschreibt (Tätigkeiten werden geplant und nach bzw. während ihrer Ausführung geprüft; Prüfergebnisse führen zu Verbesserungsmaßnahmen, die wiederum in die Planung einfließen) und damit ein zentrales Konzept der ↑Total Quality Management-Philosophie darstellt.

Primärschlüsselattribut: ↑Attribut oder ↑Attributkombination, das/die eindeutig ein ↑Tupel einer ↑Relation kennzeichnet.

Produkt: Ergebnis eines ↑Prozesses.

Prozess: Satz von in Wechselbeziehung oder Wechselwirkung stehenden Tätigkeiten, der Eingaben in Ergebnisse umwandelt.

Prüfung: Bewertung der ↑Konformität durch Beobachten und Beurteilen, begleitet – soweit zutreffend – durch Messen, Testen oder Vergleichen.

QM-Handbuch: Dokument, in dem das ↑Qualitätsmanagementsystem einer Organisation festgelegt ist.

Qualifizierungsprozess: ↑Prozess zur Darlegung der Fähigkeit, festgelegte ↑Anforderungen zu erfüllen.

Qualität: Grad, in dem ein Satz inhärenter ↑Merkmale ↑Anforderungen erfüllt.

Qualitätslenkung: Teil des ↑Qualitätsmanagements, der auf die Erfüllung von ↑Qualitäts↑anforderungen gerichtet ist.

Qualitätsmanagement (QM): Aufeinander abgestimmte Tätigkeiten zum Leiten und Lenken einer Organisation bzgl. ↑Qualität.

Qualitätsmanagementplan: Dokument, das festlegt, welche Verfahren und zugehörigen Ressourcen wann und durch wen bzgl. eines spezifischen Projekts, ↑Produkts, ↑Prozesses oder Vertrages anzuwenden sind.

Qualitätsmanagementsystem (QMS): Managementsystem zum Leiten und Lenken einer Organisation bzgl. ↑Qualität.

Qualitätsmerkmal: Inhärentes ↑Merkmal eines ↑Produkts, ↑Prozesses oder Systems, auf das sich eine ↑Anforderung bezieht.

Qualitätsplanung: Teil des ↑Qualitätsmanagements, der auf das Festlegen der ↑Qualitätsziele und der notwendigen Ausführungs↑prozesse sowie der zugehörigen Ressourcen zur Erfüllung der ↑Qualitätsziele gerichtet ist.

Qualitätspolitik: Übergeordnete Absichten und Ausrichtung einer Organisation zur ↑Qualität, wie sie von der Organisationsleitung formell ausgedrückt wurden.

Qualitätssicherung (QS): Teil des ↑Qualitätsmanagements, der auf das Erzeugen von Vertrauen darauf gerichtet ist, dass ↑Qualitäts↑anforderungen erfüllt werden.

Qualitätsverbesserung: Teil des ↑Qualitätsmanagements, der auf die Erhöhung der Fähigkeit zur Erfüllung der ↑Qualitäts↑anforderungen gerichtet ist.

Qualitätsziel: Etwas bzgl. ↑Qualität Angestrebtes oder zu Erreichendes.

Quality Function Deployment (QFD): In der produzierenden Wirtschaft eingesetztes Verfahren zur Umwandlung subjektiver Kundenanforderungen in objektive technische Spezifikationen.

Quelldatenpuffer (QDP): ↑Datenbank, in der aus einer ↑Datenquelle eingehende Daten temporär gepuffert werden, bevor sie gemäß des ↑CLIQ-Vorgehensmodells vereinheitlicht und in den ↑Transformationsbereich überführt werden.

Record Linkage: Verfahren zur Identifizierung von Duplikaten in Datenbeständen, d. h. von ↑Tupeln oder ↑Objekten, die dieselbe ↑Entität der Diskurswelt beschreiben. Da in ↑Data Warehouse-Systemen ein schlüsselbasiertes $\sim$ aufgrund der Heterogenität von ↑Datenquellen nur in Ausnahmefällen möglich ist, muss i. Allg. auf ähnlichkeitsbasierte (z. B. probabilistische) $\sim$ -Verfahren zurückgegriffen werden.

Redundanz: Mehrfache Speicherung desselben Sachverhalts in einer ↑Datenbank.

Referentielle Integrität: Stellt sicher, dass alle Werte eines ↑Fremdschlüsselattributs in der assoziierten ↑Relation als Werte des ↑Primärschlüsselattributs vorhanden sind.

Relation: Tabellenartige Datenstruktur mit eindeutigem Namen, deren Inhalt (↑Extension) aus einer Menge von ↑Tupeln besteht.

Relational OLAP (ROLAP): Variante des ↑Online Analytical Processing, bei der Daten in einem ↑Relationalen Datenbankmanagementsystem gehalten werden.

Relationales Datenbankmanagementsystem (RDBMS): Auf dem ↑Relationenmodell basierendes ↑Datenbankmanagementsystem.

Relationenmodell: ↑Datenmodell, dass sowohl Daten als auch Beziehungen zwischen Daten in Form von ↑Relationen ausdrückt.

Schema: Beschreibt die ↑Intension einer ↑Datenbank mit den Mitteln eines ↑Datenmodells.

Schlüssel: Im Kontext dieser Arbeit ist ein $\sim$ eine Ausprägung eines ↑Primärschlüsselattributs.

Semiotik: Lehre von den Zeichen, einschließlich ihrer Syntax, Semantik und Pragmatik (Verwendung).

Sicht: Spezifischer Ausschnitt der Daten einer ↑Datenbank.

Skalierbarkeit: Ein ↑Datenbankmanagementsystem ist skalierbar, falls es bei steigenden ↑Anforderungen mit akzeptablem Aufwand derart erweitert werden kann, dass die gestiegenen ↑Anforderungen erfüllt werden.

Softwarekomponente: Kompositionseinheit, die nur vertraglich spezifizierte Schnittstellen und explizite Kontextabhängigkeiten aufweist. Eine $\sim$ kann unabhängig eingesetzt werden und unterliegt der Komposition durch dritte Parteien.

Spezialisierung: Definition einer neuen ↑Klasse als ↑Subklasse einer oder mehrerer anderer ↑Klassen (↑Basisklassen). Aufgrund der ↑Vererbung besitzt die neue ↑Klasse alle ↑Merkmale ihrer ↑Basisklassen. In ihrer ↑Klassendefinition können jedoch zusätzliche ↑Merkmale definiert oder geerbte ↑Merkmale redefiniert (überschrieben) werden.

Statistische Prozesskontrolle (SPC): Kontinuierliche begleitende Überwachung von ↑Prozessen durch Erfassung und Auswertung von qualitätsrelevanten statistischen Kennzahlen. Die $\sim$ liefert Basisdaten zur Erkennung von Schwachstellen und schafft damit die Voraussetzung für eine ständige Verbesserung von ↑Prozessen.

Subklasse: Speziellere ↑Klasse einer ↑Generalisierung.

Supply Chain Management (SCM): Steuerung von Materialien, ↑Produkten und Informationen auf ihrem Weg von Lieferanten über Hersteller bis hin zu Kunden (ggf. via Groß- und Einzelhändler) mit dem Ziel einer Kostenreduzierung bei gleichbleibender oder sogar verbesserter Erfüllung von Kunden↑anforderungen.

Surrogat: Eindeutiger, oft systemseitig vergebener Wert eines ↑Primärschlüsselattributs ohne anwendungsspezifische Semantik.

Synonymfehler: Nichterkennung der Zusammengehörigkeit zweier ↑Tupel oder ↑Objekte, die dieselbe ↑Entität beschreiben, in einem ↑Record Linkage-↑Prozess.

Total Quality Management (TQM): Organisationsumfassende Strategie, die sich an Kundenbedürfnissen ausrichtet und auf eine ständige ↑Qualitätsverbesserung abzielt.

Transaktion: Folge von Änderungsoperationen auf einer ↑Datenbank, die nur als Einheit wirksam werden kann und nach ihrer Ausführung einem konsistenten ↑Datenbankzustand hinterlässt, vorausgesetzt die ↑Datenbank war vor der Ausführung der $\sim$ konsistent.

Transformation: ↑Prozess innerhalb des ↑Data Warehousing, der Daten an Hand von bestimmten Abbildungsvorschriften umwandelt.

Transformationsbereich (TB): ↑Datenbank, die im ↑CLIQ-Vorgehensmodell zwischen ↑Quelldatenpuffer und ↑Zielbereich geschaltet ist und auf der ↑Transformationen und Prüfungen vorgenommen werden.

Tupel: Abkürzung für n-Tupel, also ein Element aus einem n-stelligen kartesischen Produkt (hier: einer ↑Relation).

Unified Modeling Language (UML): Von der OMG standardisierte Notation zur Konstruktion und Dokumentation von Modellen im Rahmen der objektorientierten Softwareentwicklung.

Vererbung: Beziehung zwischen ↑Klassen, durch die eine ↑Klasse (↑Subklasse) die ↑Merkmale einer oder mehrerer anderer ↑Klassen (↑Basisklassen) übernimmt.

Workflow Management System (WfMS): System zur Beschreibung und Steuerung von Geschäfts-↑prozessen mit dem Ziel einer erhöhten ↑Qualität der Vorgangsbearbeitung (Reduzierung von Durchlaufzeiten, Verringerung des Aufwands zur Anpassung an geänderte ↑Anforderungen etc.). Spezifizierte Arbeitsabläufe werden von einem ∼ entweder direkt ausgeführt oder in Programmcode umgesetzt. Die im Rahmen des Workflow zur Ausführung kommenden ↑Softwarekomponenten müssen über geeignete Schnittstellen verfügen, die wiederum dem ∼ in geeigneten Spezifikationen bekannt zu machen sind.

Zielbereich (ZB): ↑Datenbank, die im ↑CLIQ-Vorgehensmodell integrierte, qualitätsgesicherte, feingranulare Daten permanent speichert. Damit übernimmt der ∼ die Funktion eines ↑Operational Data Store.

Abkürzungsverzeichnis

API	Application Programming Interface
CARLOS	Cancer Registry Lower-Saxony
CLIQ	Data Cleansing mit intelligentem Qualitätsmanagement
COM	Component Object Model
CRM	Customer Relationship Management
CWM	Common Warehouse Metamodel
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
DSS	Decision Support System
DIN	Deutsches Institut für Normung
DQM	Datenqualitätsmanagement
DQMS	Datenqualitätsmanagementsystem
DTD	Document Type Definition
DTS	Data Transformation Services
DWQ	Foundations of Data Warehouse Quality
DWS	Data Warehouse-System
EAI	Enterprise Application Integration
EKN	Epidemiologisches Krebsregister Niedersachsen
ETL	Extraktion – Transformation – Laden
FDBS	Föderiertes DBS
GQM	Goal-Question-Metric
GUI	Graphical User Interface
HTML	Hypertext Markup Language
ICD	International Classification of Diseases
ICD-O	International Classification of Diseases for Oncology
ISO	International Organization for Standardization
IT	Informationstechnologie
KB	Konsolidierungsbereich
KDD	Knowledge Discovery in Databases
KRG	Krebsregistergesetz
MDC	Meta Data Coalition
MDK	Model Development Kit

MEDAS	Metadata-Based Data Auditing System
MFC	Microsoft Foundation Classes
MLC++	Machine Learning Library in C++
ODBC	Open Database Connectivity
ODS	Operational Data Store
OIM	Open Information Model
OLAP	Online Analytical Processing
OLE	Object Linking and Embedding
OLTP	Online Transaction Processing
OMG	Object Management Group
OOA	Object Oriented Analysis
OOD	Object Oriented Design
PDCA	Plan – Do – Check – Act
PDF	Portable Document Format
PSA	Persistent Staging Area
QDP	Quelldatenpuffer
QFD	Quality Function Deployment
QM	Qualitätsmanagement
QMS	Qualitätsmanagementsystem
QS	Qualitätssicherung
RDBMS	Relationales DBMS
ROLAP	Relationales OLAP
RTIM	Repository Type Information Model
SAP BW	SAP Business Warehouse
SCM	Supply Chain Management
SPC	Statistical Process Control
SQL	Structured Query Language
STL	Standard Template Library
TB	Transformationsbereich
TDQM	Total DQM
TNM	Tumor – Node – Metastasis
TQC	Total Quality Control
TQM	Total QM
UML	Unified Modeling Language
WfMS	Workflow Management System
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
YADI	Yet Another Database Interface
ZB	Zielbereich

Abbildungsverzeichnis

1.1 Entscheidungsunterstützung ohne und mit Data Warehouse 4
1.2 (a) Erfolgsbewertung und (b) ROI-Bewertung von DWS-Projekten [Met99b] 5
1.3 Erfolgsbarrieren in DWS-Projekten [Met99b] 5
1.4 Beispiele für Datenqualitätsmängel 6
1.5 Ausschnitt aus dem Information System Success Model [DM92] 7

2.1 Grobarchitektur eines DWS (strukturorientierte Sicht) 11
2.2 DWS-Referenzarchitektur (datenflussorientierte Sicht) 12

3.1 Analogie zur industriellen Fertigung 25
3.2 System-Bi-Cycle (frei nach [Ort95]) 26
3.3 Semiotisches Dreieck (frei nach [Lew90]) 27
3.4 Taxonomie von Datenqualitätsmerkmalen nach [SLW97b] (UML-Notation) 29
3.5 Taxonomie von Datenqualitätsmerkmalen nach [Hin01a] (UML-Notation) 30
3.6 Klassifikation von Datenqualitätsproblemen nach [RD00] 32

4.1 Plan-Do-Check-Act–Zyklus [Red99] 38
4.2 Modell eines prozessorientierten QMS (frei nach [Deu00b]) 39
4.3 Interne Struktur der ISO 9001:2000-Abschnitte 4 bis 8 (ohne dritte Ebene) 40
4.4 Funktionen des QM 40
4.5 Total Quality data Management nach [Eng99] 41
4.6 Methodenbasiertes DQM nach [Hel00] 41
4.7 House of Quality im DQM-Kontext 42
4.8 Gesamtnutzwert von Daten nach [Hau84] 49
4.9 Bedürfnispyramide nach [Mün95] 50

5.1 Überblick über die QUIS-Umgebung 55
5.2 Abweichungsanalyse: Modellbildung und -anwendung 56

5.3 Objekttypen im Ascential Quality Manager (frei nach [Asc01b]) 61

6.1 Stelligkeiten in einer Klassifikationshierarchie 79
6.2 Verteilung von Gewichten im probabilistischen Record Linkage 89

7.1 CLIQ-Vorgehensmodell zur inkrementellen Datenintegration in DWS 92
7.2 Task Sets und Transformationspakete 97
7.3 UML-Aktivitätsdiagramm zum CLIQ-Vorgehensmodell 128

8.1 UML-Anwendungsfalldiagramm zur CLIQWorkbench (aus [Ade01], revidiert) . . . 132
8.2 UML-Klassendiagramm der Business-Klassen (aus [Ade01], revidiert) 135

9.1 Grobarchitektur der CLIQWorkbench 138
9.2 UML-Aktivitätsdiagramm zu Anwendungsfall 3 (aus [Ade01], revidiert) 140
9.3 UML-Aktivitätsdiagramm zu Anwendungsfall 4 (aus [Ade01], revidiert) 141
9.4 UML-Aktivitätsdiagramm zu Anwendungsfall 5 (aus [Ade01], revidiert) 143
9.5 UML-Paketdiagramm zum DQM Controller 144
9.6 UML-Klassendiagramm zum Paket *JobModel* (aus [Ade01], revidiert) 144
9.7 UML-Klassendiagramm zum Paket *JobExecEngine* (aus [Ade01], revidiert) 147
9.8 UML-Klassendiagramm zum Paket *JobAnalyzeEngine* (aus [Ade01], revidiert) . . . 151
9.9 UML-Klassendiagramm zum Paket *JobControlDBEngine* (aus [Ade01], revidiert) . . 153
9.10 UML-Anwendungsfalldiagramm zum Data Migrator 154
9.11 UML-Klassendiagramm zum Data Migrator 156
9.12 Grundgerüst eines *DtPackage* 157
9.13 Einbindung von Transformationsverfahren in den Data Migrator 158
9.14 UML-Klassendiagramm zu Wrapper-Klassen im Data Migrator 159
9.15 UML-Klassendiagramm zur *MeasurementEngine* 161
9.16 UML-Klassendiagramm zu Datenstrukturen der *MeasurementEngine* 161
9.17 UML-Klassendiagramm zur *SPCEngine* 162
9.18 UML-Klassendiagramm zu Datenstrukturen der *SPCEngine* 163
9.19 UML-Klassendiagramm zur *DomainSpecificMeasurementEngine* 163
9.20 UML-Paketdiagramm zur Nachbearbeitungskomponente 164
9.21 Ausführung eines ILOG Rules-Programms [Ilo97] 164
9.22 UML-Paketdiagramm zum Data Linker 166
9.23 UML-Klassendiagramm zu Datenstrukturen des Pakets *ProbRecLink* 167

9.24 UML-Klassendiagramm zu MEDAS [Wil00] 169
9.25 UML-Klassendiagramm zu Datenstrukturen in MEDAS [Wil00] 169
9.26 UML-Klassendiagramm zum Release/Retraction Handler 171
9.27 UML-Klassendiagramm zum Paket *MrRepository* 172
9.28 Aufbau eines Quelldatenpuffers 174
9.29 Aufbau der Integrationsdatenbank 175

10.1 Schichtenarchitektur der CLIQWorkbench 177
10.2 Microsoft Repository 180
10.3 Informationsmodelle des OIM [Met99a] 181
10.4 Module der Data Transformation Services 182
10.5 Module von MLC++ 183
10.6 Schritte zur Erstellung einer ILOG Rules-basierten Applikation [Ilo97] 184

11.1 Der CARTools-Werkzeugkasten 192
11.2 Assoziationen zwischen CLIQ und den CARTools 194
11.3 Authentifizierung gegenüber (Fach-)Datenbank und Repository 200
11.4 Erstellen einer Konfiguration zur Integrationsaufgabe 201
11.5 Auswahl eines Transformationsverfahrens im Data Migrator 201
11.6 ActiveX-Skript zur Zerlegung von Datumswerten 202
11.7 Festlegung der Umrechnung von Maßeinheiten im Data Migrator 202
11.8 Planung der Statistischen Prozesskontrolle 203
11.9 Spezifizierung eines probabilistischen Record Linkage-Umlaufs 204
11.10 Spezifizierung eines Qualitätsplans 204
11.11 Spezifizierung eines gewichteten Datenobjekts für einen Qualitätsplan 205
11.12 Spezifizierung eines Indikatorplans 206
11.13 Überblick über das Protokoll der durchgeführten Integrationsaufgabe 207
11.14 X-Chart und Pareto-Chart zu den Mittelwerten im Attribut *geschlecht* 208
11.15 Überblick über die Ergebnisse der Konsistenzprüfungen 209
11.16 Interaktive Nachbearbeitung von inkonsistenten Meldungen 210
11.17 Überblick über die Ergebnisse des probabilistischen Record Linkage 211
11.18 Interaktive Aufbereitung von Matchgruppen 211
11.19 Überblick über die Ergebnisse der Datenqualitätsmessungen 212
11.20 Ausschnitt aus *MELDUNGEN_EH* 213

11.21 Ausschnitt aus *MELDUNGEN_ES* 214

12.1 Datenfluss im SAP Business Warehouse 2.0B 220

12.2 Aktualisierung von ODS-Objekten im SAP BW 2.0B 221

12.3 Ansatz zur Integration der CLIQWorkbench in das SAP BW 2.0B 222

13.1 Rückblick auf die Kapitelstruktur dieser Arbeit 228

Zur Homepage der Dissertation

Tabellenverzeichnis

2.1 Gegenüberstellung von operativer Datenbank und Data Warehouse [Her01a] 13
2.2 Zuordnung von lokalen zu globalen Schlüsseln 17

3.1 Gegenüberstellung von Ansätzen zur Merkmalsklassifizierung 29

4.1 Reaktionen auf Datenqualitätsmängel in Abhängigkeit ihrer Fehlerklasse 48

5.1 Forschungsarbeiten zum DQM 64

6.1 Zuordnung von Granularitätsebenen zu Datenqualitätsmerkmalen 71

10.1 Umfang der Implementierung der CLIQWorkbench (Stand: Dezember 2001) 188

11.1 Relationenschema von *MELDUNGEN* 197
11.2 Attributbezogene Metadaten zu *MELDUNGEN* 198

12.1 Gegenüberstellung der Basiskonzepte von SAP BW 2.0B und CLIQWorkbench . . . 221
12.2 Vorläufige Auswahl von CLIQWorkbench-Modulen für das SAP BW 222

A.1 Zuordnung von Anbietern zu Werkzeugkategorien, Teil 1 245
A.2 Zuordnung von Anbietern zu Werkzeugkategorien, Teil 2 246

Literaturverzeichnis

[ABH+99] H.-J. Appelrath, M. Beyer, H. Hinrichs, J. Kieschke, K. Panienski, M. Rohde, A. Scharnofske, W. Thoben, I. Wellmann, F. Wietek und L. Zachewitz. *CARLOS (Cancer Registry Lower-Saxony): Tätigkeitsbericht für den Zeitraum 1.1.-31.12.1998.* Technischer Bericht, OFFIS, Oldenburg, 1999.

[Ade01] T. Aden. *Entwicklung eines ISO 9000-konformen Steuerungsmoduls zum Datenqualitätsmanagement in Data Warehouse-Systemen.* Diplomarbeit, Universität Oldenburg, 2001.

[AFH+97] H.-J. Appelrath, J. Friebe, H. Hinrichs, V. Kamp, J. Rettig, W. Thoben und F. Wietek. Softwarewerkzeuge für (epidemiologische) Krebsregister. In M. P. Baur, R. Fimmers und M. Blettner, Hrsg., *Medizinische Informatik, Biometrie und Epidemiologie (GMDS'96), Bonn*, S. 470–474, MMV Medizin Verlag, München, 1997.

[Aka90] Y. Akao, Hrsg. *Quality Function Deployment: Integrating Customer Requirements into Product Design.* Productivity Press, 1990.

[AL99] D. Awalt und B. K. Lawton. Introduction to Data Transformation Services. *SQL Server Magazine*, 1(1):34–36, 1999.

[Alh98] S. S. Alhir. *UML in a Nutshell.* O'Reilly, London, 1998.

[AM97] S. Anahory und D. Murray, Hrsg. *Data Warehouse – Planung, Implementierung und Administration.* Addison-Wesley, 1997.

[Apa01] Apache. *http://xml.apache.org/xerces-c/index.html*, 2001.

[Asc01a] Ascential Software. *Quality Manager – Methodology and Application Guide, Version 4.8.1*, 2001.

[Asc01b] Ascential Software. *Quality Manager – User's Guide, Version 4.8.1*, 2001.

[Atz00] P. Atzeni. *Database Systems: Concepts, Languages and Architectures.* McGraw-Hill, London, 2000.

[BA96] R. J. Brachman und T. Anand. The Process of Knowledge Discovery in Databases: A Human-Centered Approach. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth und R. Uthurusamy, Hrsg., *Advances in Knowledge Discovery and Data Mining*, S. 37–57. AAAI/MIT Press, Cambridge, MA, 1996.

[Bal95] H. Balzert. *Methoden der objektorientierten Systemanalyse.* BI Wissenschaftsverlag, 1995.

[BBC+99] P. A. Bernstein, T. Bergstraesser, J. Carlson, S. Pal, P. Sanders und D. Shutt. Microsoft Repository Version 2 and the Open Information Model. *Information Systems*, 24(2):71–98, 1999.

[BFG98] P. L. Bowen, D. A. Fuhrer und F. M. Guess. Continuously Improving Data Quality in Persistent Databases. *Data Quality*, 4(1), 1998.

[BFKM85] L. Brownston, R. Farrell, E. Kant und N. Martin. *Programming Expert Systems in OPS5*. Addison-Wesley, Reading, MA, 1985.

[BG01] A. Bauer und H. Günzel, Hrsg. *Data Warehouse Systeme – Architektur, Entwicklung, Anwendung*. dpunkt, Heidelberg, 2001.

[BHM98] N. Bissantz, J. Hagedorn und P. Mertens. Data Mining. In H. Mucksch und W. Behme, Hrsg., *Das Data Warehouse-Konzept*, S. 437–446, Gabler, Wiesbaden, 1998.

[BHS+97] P. A. Bernstein, B. Harry, P. Sanders, D. Shutt und J. Zander. The Microsoft Repository. In *Proc. of the 23rd Intl. Conf. on Very Large Databases (VLDB), Athen*, S. 3–12, 1997.

[BJRR98] G. Booch, I. Jacobson, J. Rumbaugh und J. Rumbaugh. *The Unified Modeling Language User Guide*. Addison-Wesley, 1998.

[BKL00] S. Busse, R.-D. Kutsche und U. Leser. Strategies for the Conceptual Design of Federated Information Systems. In *Proc. of the 3rd Intl. Workshop on Engineering Federated Information Systems (EFIS'2000), Dublin*, S. 23–32, 2000.

[BM86] F. Bry und R. Manthey. Checking Consistency of Database Constraints: A Logical Basis. In *Proc. of the 12th Intl. Conf. on Very Large Data Bases (VLDB'86), Kyoto*, S. 13–20, 1986.

[BMP01] A. Behrend, R. Manthey und B. Pieper. An Amateur's Introduction to Constraints and Integrity Checking in SQL. In A. Heuer, F. Leymann und D. Priebe, Hrsg., *Datenbanksysteme in Büro, Technik und Wissenschaft, 9. GI-Fachtagung BTW 2001, Oldenburg*, S. 405–423, Springer, Berlin, März 2001.

[Bra96] M. H. Brackett. *The Data Warehouse Challenge – Taming Data Chaos.* Wiley, New York, 1996.

[Bro83] R. G. Brown. *Introduction to Random Signal Analysis and Kalman Filtering.* Wiley, New York, 1983.

[BRS95] J. Becker, M. Rosemann und R. Schütte. Grundsätze ordnungsmäßiger Modellierung. *Wirtschaftsinformatik*, 37(5):435–445, Oktober 1995.

[BS84] B. G. Buchanan und E. H. Shortliffe, Hrsg. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, MA, 1984.

[BS97a] A. Berson und S. J. Smith. *Data Mining and OLAP*. McGraw-Hill, 1997.

[BS97b] H. Brenner und I. Schmidtmann. Auswirkungen von Record Linkage Fehlern auf die Nutzung von Krankheitsregistern: Methodische Grundlagen. In R. Muche, G. Büchle, D. Harder und W. Gaus, Hrsg., *Medizinische Informatik, Biometrie und Epidemiologie (GMDS'97)*, S. 486–489, MMV Medizin Verlag, München, 1997.

[BSZ94] H. Brenner, C. Stegmaier und H. Ziegler. Estimating Completeness of Cancer Registration in Saarland/Germany with Capture-Recapture Methods. *European Journal of Cancer*, 30A(11):1659–1663, 1994.

[BT99] D. P. Ballou und G. K. Tayi. Enhancing Data Quality in Data Warehouse Environments. *Communications of the ACM*, 42(1):73–78, 1999.

[BW84] V. R. Basili und D. M. Weiss. A Method for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering,* 10(6):728–738, 1984.

[BWPT98] D. Ballou, R. Y. Wang, H. Pazer und G. K. Tayi. Modeling Information Manufacturing Systems to Determine Information Product Quality. *Management Science,* 44(4):462–484, 1998.

[CB01] G. Casella und R. L. Berger. *Statistical Inference*. Duxbury Press, 2001.

[CBS98] T. Connolly, C. Begg und A. Strachan. *Database Systems.* Addison-Wesley, 2. Auflage, 1998.

[CD97] S. Chaudhuri und U. Dayal. An Overview of Data Warehousing and OLAP-Technology. *SIGMOD Record,* 26(1):65–74, 1997.

[CDN88] J. P. Chin, V. A. Diehl und K. L. Norman. Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface. In *Proc. of CHI'88 (Human Factors in Computing Systems)*, S. 213–218, ACM Press, New York, 1988.

[CGL+99] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi und R. Rosati. A Principled Approach to Data Integration and Reconciliation in Data Warehousing. In S. Gatziu, M. Jeusfeld, M. Staudt und Y. Vassiliou, Hrsg., *Proc. of the Intl. Workshop on Design and Management of Data Warehouses (DMDW'99), Heidelberg*, S. 16/1–16/11, 1999.

[Con97] S. Conrad. *Föderierte Datenbanksysteme: Konzepte der Datenintegration.* Springer, Berlin, 1997.

[CRI00] CRISP-DM Consortium. *CRISP-DM 1.0 – Step-by-Step Data Mining Guide.* http://www.crisp-dm.org, 2000.

[CY91a] P. Coad und E. Yourdon. *Object-Oriented Analysis.* Prentice-Hall, 1991.

[CY91b] P. Coad und E. Yourdon. *Object-Oriented Design*. Prentice-Hall, 1991.

[Dat99] Data Mart Consulting. *Einsatzfelder von Technologien und Anwendungen im Data Warehouse-Umfeld bei verschiedenen betriebswirtschaftlichen Aufgabenstellungen*. Technischer Bericht, Data Mart Consulting/TU Darmstadt, 1999.

[DeM82] T. DeMarco. *Controlling Software Projects.* Yourdon Press, New York, 1982.

[Dem00] W. E. Deming. *Out of the crisis.* MIT Press, 2000.

[Deu94] Deutscher Bundestag. Gesetz über Krebsregister (Krebsregistergesetz, KRG). *Bundesgesetzblatt,* 79:3351–3355, 1994.

[Deu95] Deutsches Institut für Medizinische Dokumentation und Information (DIMDI). *ICD-10: Internationale statistische Klassifikation der Krankheiten und verwandter Gesundheitsprobleme, 10. Revision, Band I: Systematisches Verzeichnis.* ecomed, Landsberg, 1995.

[Deu00a] Deutsches Institut für Normung. *DIN EN ISO 9000: Qualitätsmanagementsysteme – Grundlagen und Begriffe*. Beuth, Berlin, 2000.

[Deu00b] Deutsches Institut für Normung. *DIN EN ISO 9001: Qualitätsmanagementsysteme – Anforderungen*. Beuth, Berlin, 2000.

[DG98] G. Dodge und T. Gorman. *Oracle8 Data Warehousing – A Hands-On Guide to Designing, Building, and Managing Oracle Data Warehouses.* Wiley, New York, 1998.

[DJ98] A. M. Disney und P. M. Johnson. Investigating Data Quality Problems in the PSP. In *Proc. of the 6th ACM SIGSOFT Intl. Symposium on Foundations of Software Engineering, Orlando, FL*, S. 143–152, 1998.

[DJK00] T. Dasu, T. Johnson und E. Koutsofios. Hunting Data Glitches in Massive Time Series Data. In B. D. Klein und D. F. Rossin, Hrsg., *Proc. of the 5th Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 190–199, 2000.

[DM92] W. H. DeLone und E. R. McLean. Information Systems Success: The Quest for the Dependent Variable. *Information Systems Research*, 3(1):60–95, 1992.

[Dör00] J. Dörsam. *Datenvalidierung und Fehlerbehandlung bei der Datenübernahme in ein Data Warehouse am Beispiel des SAP Business Information Warehouse.* Diplomarbeit, Berufsakademie Mannheim, 2000.

[Dub02] Dublin Core Metadata Initiative. *http://dublincore.org*, 2002.

[EE98] G. Eddon und H. Eddon. *Inside Distributed COM*. Microsoft Press, München, 1998.

[EM00] A. Eisenberg und J. Melton. SQL Standardization: The Next Steps. *ACM SIGMOD Record*, 29(1):63–67, 2000.

[Eng99] L. P. English. *Improving Data Warehouse and Business Information Quality.* Wiley, New York, 1999.

[Epp01] M. J. Eppler. A Generic Framework for Information Quality in Knowledge-Intensive Processes. In E. M. Pierce und R. Katz-Haas, Hrsg., *Proc. of the 6th Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 329–346, 2001.

[EW00] M. J. Eppler und D. Wittig. Conceptualizing Information Quality: A Review of Information Quality Frameworks from the Last Ten Years. In B. D. Klein und D. F. Rossin, Hrsg., *Proc. of the 5th Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 83–96, 2000.

[Fei91] A. V. Feigenbaum. *Total Quality Control*. McGraw-Hill, Singapur, 1991.

[Fer99] M. Fernandez Lopez. Overview of Methodologies for Building Ontologies. In *Proc. of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods, Stockholm*, S. 4/1–4/13, 1999.

[Fis92] J. Fischer. *Datenmanagement: Datenbanken und betriebliche Datenmodellierung.* Oldenbourg, München, 1992.

[FP97] N. E. Fenton und S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. Intl. Thomson Computer Press, London, 1997.

[FPS96] U. M. Fayyad, G. Piatetsky-Shapiro und P. Smyth. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In *Proc. of the 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, S. 82–88. AAAI Press, 1996.

[Fri96] J. Friebe. *Yet Another Database Interface (YADI)*. Technischer Bericht, OFFIS, Oldenburg, 1996.

[FS69] I. P. Fellegi und A. B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[Gar98] S. R. Gardner. Building the Data Warehouse. *CACM*, 41(9):52–60, 1998.

[GBMS99] C. H. Goh, S. Bressan, S. Madnick und M. Siegel. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. *ACM Transactions on Information Systems*, 17(3):270, 1999.

[GD00a] M. S. Gendron und M. J. D'Onofrio. Data Quality in the Healthcare Industry: An Exploratory Study. In *Proc. of the 4^{th} World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'2000), Orlando, FL*, 2000.

[GD00b] M. S. Gendron und M. J. D'Onofrio. An Exploratory Study Investigating Data Quality in the Healthcare Industry: What are the Implications for Data Warehousing? In *Proc. of Americas Conf. on Information Systems (AMCIS'2000), Long Beach, CA*, 2000.

[GFS+01] H. Galhardas, D. Florescu, D. Shasha, E. Simon und C.-A. Saita. Declarative Data Cleaning: Language, Model, and Algorithms. In *Proc. of the 27^{th} VLDB Conf., Rom*, 2001.

[GFSS00] H. Galhardas, D. Florescu, D. Shasha und E. Simon. AJAX: An Extensible Data Cleaning Tool. In *Proc. of the 2000 ACM SIGMOD Conf.*, S. 590, 2000.

[GGC97] P. Gluchowski, R. Gabriel und P. Chamoni. *Management-Support-Systeme: Computergestützte Informationssysteme für Führungskräfte und Entscheidungsträger*. Springer, Berlin, 1997.

[GH93] A. Griffin und J. R. Hauser. The Voice of the Customer. *Marketing Science*, 12(1):1–27, 1993.

[GH01a] U. Grimmer und H. Hinrichs. Datenqualitätsmanagement mit Data Mining-Unterstützung. In K. Hildebrand, Hrsg., *Business Intelligence (Praxis der Wirtschaftsinformatik), HMD 222*, S. 70–80. dpunkt, 2001.

[GH01b] U. Grimmer und H. Hinrichs. A Methodological Approach to Data Quality Management Supported by Data Mining. In E. M. Pierce und R. Katz-Haas, Hrsg., *Proc. of the 6^{th} Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 217–232, 2001.

[GHJV01] E. Gamma, R. Helm, R. Johnson und J. Vlissides. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, 2001.

[GJJ+98] M. Gebhardt, M. Jarke, M. A. Jeusfeld, C. Quix und S. Sklorz. Tools for Data Warehouse Quality. In *Proc. of the 10^{th} IEEE Intl. Conf. on Scientific and Statistical Database Management (SSDBM'98), Capri*, S. 229–232, 1998.

[GL95] M. Gertz und U. W. Lipeck. 'Temporal' Integrity Constraints in Temporal Databases. In J. Clifford und A. Tuzhilin, Hrsg., *Recent Advances in Temporal Databases – Proc. of the Intl. Workshop on Temporal Databases*, S. 77–92, Springer, Berlin, September 1995.

[Gün00] H. Günzel. Data Warehouse – eine Basis für Data Mining? *Datenbank-Rundbrief*, 25:6–10, 2000.

[Gro76] S. Grotz-Martin. *Informations-Qualität und Informations-Akzeptanz in Entscheidungsprozessen.* Dissertation, Universität Saarbrücken, 1976.

[Gru95] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. *Intl. Journal of Human and Computer Studies*, 43(5/6):907–928, 1995.

[GS98] M. Gertz und I. Schmitt. Data Integration Techniques Based on Data Quality Aspects. In I. Schmitt, C. Türker, E. Hildebrandt und M. Höding, Hrsg., *Proc. 3. Workshop Föderierte Datenbanken, Magdeburg*, S. 1–19, Shaker, Aachen, 1998.

[HA01] H. Hinrichs und T. Aden. An ISO 9001:2000 Compliant Quality Management System for Data Integration in Data Warehouse Systems. In D. Theodoratos, J. Hammer, M. Jeusfeld und M. Staudt, Hrsg., *Proc. of the Intl. Workshop on Design and Management of Data Warehouses (DMDW 2001), Interlaken*, S. 1/1–1/12, 2001.

[Har99] A. Harren. *Konzeptionelles Data Warehouse-Design.* Diplomarbeit, Universität Oldenburg, 1999.

[Has99] W. Hasselbring. On Defining Computer Science Terminology. *Communications of the ACM*, 42(2):88–91, 1999.

[Has00a] N. Hashmi, Hrsg. *Business Information Warehouse for SAP*. Prima Tech, Roseville, CA, 2000.

[Has00b] W. Hasselbring. The Role of Standards for Interoperating Information Systems. In K. Jakobs, Hrsg., *Information Technology Standards and Standardization: A Global Perspective*, S. 116–130, Idea Group Publishing, Hershey, PA, 2000.

[Has02] W. Hasselbring. Component-Based Software Engineering. *Handbook of Software Engineering and Knowledge Engineering Vol. 2*, 2002. Erscheint voraussichtlich Anfang 2002.

[Hau84] P. Hauke. *Informationsverarbeitungsprozesse und Informationsbewertung.* GBI-Verlag, München, 1984.

[HC88] J. R. Hauser und D. Clausing. The House of Quality. *Harvard Business Review*, 3:63–73, 1988.

[Hea02] Health Level Seven. *http://www.hl7.org*, 2002.

[HEK99] J. Hartung, B. Elpelt und K.-H. Klösener. *Statistik: Lehr- und Handbuch der angewandten Statistik.* Oldenbourg, München, 12. Auflage, 1999.

[Hel00] M. Helfert. Maßnahmen und Konzepte zur Sicherung der Datenqualität. In R. Jung und R. Winter, Hrsg., *Data Warehousing Strategie – Erfahrungen, Methoden, Visionen*, S. 61–77, Springer, Berlin, 2000.

[Her01a] O. Herden. *Eine Entwurfsmethodik für Data Warehouses.* Dissertation, Universität Oldenburg, 2001.

[Her01b] O. Herden. Parameterised Patterns for Conceptual Modelling of Data Warehouses. In J. Barzdins und A. Caplinskas, Hrsg., *Databases and Information Systems – 4^{th} Intl. Baltic Workshop DB&IS 2000, Vilnius, Lithuania, Selected Papers*, S. 97–108, Kluwer Academic Publishers, Dordrecht, 2001.

[HGG01] J. Hipp, U. Güntzer und U. Grimmer. Data Quality Mining – Making a Virtue of Necessity. In *Proc. of the 6^{th} ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD 2001), Santa Barbara*, S. 52–57, 2001.

[HGW+95] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio und Y. Zhuge. The Stanford Data Warehousing Project. *IEEE Data Engineering Bulletin*, Juni 1995.

[HH99] A. Harren und O. Herden. MML und mUML – Sprache und Werkzeug zur Unterstützung des konzeptionellen Data Warehouse-Designs. In *2. Workshop Data Mining und Data Warehousing als Grundlage moderner entscheidungsunterstützender Systeme (DMDW'99), Magdeburg*, S. 57–67, 1999.

[Hin99a] H. Hinrichs. Intelligente Datenbereinigung in epidemiologischen Registern. In A. P. Buchmann, Hrsg., *Datenbanksysteme in Büro, Technik und Wissenschaft, 8. GI-Fachtagung BTW'99, Freiburg im Breisgau*, S. 291–306, Springer, Berlin, 1999.

[Hin99b] H. Hinrichs. Metadata-based Quality Management of Warehouse Data. In K. Richta, Hrsg., *Proc. of the 19th Conf. on Current Trends in Databases and Information Systems (DATASEM'99), Brno*, S. 239–248, Masaryk University Press, Brno, 1999.

[Hin00a] H. Hinrichs. CLIQ – Intelligent Data Quality Management. In A. Caplinskas, Hrsg., *Proc. of the 4th IEEE Intl. Baltic Workshop on Databases and Information Systems, Vilnius, Lithuania, Vol. 2*, S. 296–297, Technika Press, Vilnius, 2000.

[Hin00b] H. Hinrichs. CLIQ – Intelligent Data Quality Management. In A. Hinze und B.-J. Hommes, Hrsg., *Proc. of the 7th Doctoral Consortium CAiSE*00, Stockholm*, S. 25–36. FU Berlin, 2000.

[Hin00c] H. Hinrichs. Quality Assurance of Warehouse Data Using Statistical Process Control. In B. Lings und K. Jeffery, Hrsg., *British National Conf. on Databases BNCOD 2000 Short Papers, Exeter, Research Report No 393*, S. 3–7. University of Exeter, 2000.

[Hin01a] H. Hinrichs. Datenqualitätsmanagement in Data Warehouse-Umgebungen. In A. Heuer, F. Leymann und D. Priebe, Hrsg., *Datenbanksysteme in Büro, Technik und Wissenschaft, 9. GI-Fachtagung BTW 2001, Oldenburg*, S. 187–206, Springer, Berlin, März 2001.

[Hin01b] H. Hinrichs. Statistical Quality Control of Warehouse Data. In J. Barzdins und A. Caplinskas, Hrsg., *Databases and Information Systems – 4th Intl. Baltic Workshop DB&IS 2000, Vilnius, Lithuania, Selected Papers*, S. 69–84, Kluwer Academic Publishers, Dordrecht, 2001.

[HL98] J. Hill und S. Laufer. *Data Transformation: Key to Information Sharing.* Gartner Group Strategic Analysis Report, 1998.

[HLW98] K.-T. Huang, Y. W. Lee und R. Y. Wang. *Quality Information and Knowledge Management.* Prentice-Hall, 1998.

[HM01] M. Helfert und E. von Maur. A Strategy for Managing Data Quality in Data Warehouse Systems. In E. M. Pierce und R. Katz-Haas, Hrsg., *Proc. of the 6th Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 62–76, 2001.

[Hor88] E. Horvitz. Reasoning under Varying and Uncertain Resource Constraints. In *Proc. of the 7th National Conf. on Artificial Intelligence (AAAI'88), Minneapolis*, S. 111–116. Morgan Kaufmann, 1988.

[HP99] H. Hinrichs und K. Panienski. Experiences with Knowledge-Based Data Cleansing at the Epidemiological Cancer Registry of Lower-Saxony. In F. Puppe, Hrsg., *XPS-99: Knowledge-Based Systems - Survey and Future Directions, 5th German Conf. on Knowledge-Based Systems, Würzburg, LNAI 1570*, S. 218–226, Springer, Berlin, 1999.

[HS95] M. A. Hernandez und S. J. Stolfo. The Merge/Purge Problem for Large Databases. In *Proc. of the 1995 ACM SIGMOD Conf.*, S. 127–138, 1995.

[HS00] A. Heuer und G. Saake. *Datenbanken – Konzepte und Sprachen.* International Thomson Publishing, Bonn, 2. Auflage, 2000.

[Hul97] R. Hull. Managing Semantic Heterogeneity in Databases: A Theoretical Perspective. In *Proc. of the 16th ACM Symposium on Principles of Database Systems (ACM PODS-1997)*, S. 51–61, ACM Press, Tucson, 1997.

[HW00] H. Hinrichs und T. Wilkens. Metadata-Based Data Auditing. In N. F. F. Ebecken und C. A. Brebbia, Hrsg., *Data Mining II (Proc. of the 2nd Intl. Conf. on Data Mining, Cambridge, UK)*, S. 141–150, WIT Press, Southampton, 2000.

[HZ95] W.-C. Hou und Z. Zhang. Enhancing Database Correctness: A Statistical Approach. *ACM SIGMOD Record*, 24(2):223–232, 1995.

[HZ96] R. Hull und G. Zhou. A Framework for Supporting Data Integration Using the Materialized and Virtual Approaches. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data, Montreal*, S. 481–492, 1996.

[Ilo97] Ilog. *ILOG Rules for C++ 5.0 – User's Manual*, 1997.

[Ilo02] Ilog. *http://www.ilog.com/products/rules*, 2002.

[Imh98] C. Imhoff. Ensuring Data Quality through Data Stewardship. *DM Review, http://www.dmreview.com*, April 1998.

[Inm96] W. H. Inmon. *Building the Data Warehouse.* Wiley, New York, 1996.

[Inm00] W. H. Inmon. ODS Types. *DM Review, http://www.dmreview.com*, 2000.

[Int97] International Union Against Cancer (UICC). *TNM – Klassifikation maligner Tumoren.* Springer, 5. Auflage, 1997.

[Jar89] M. A. Jaro. Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[JBS97] S. Jablonski, M. Böhm und W. Schulze. *Workflow-Management – Entwicklung von Anwendungen und Systemen.* dpunkt, Heidelberg, 1997.

[JE01] B. Jin und S. M. Embury. Non-Intrusive Assessment of Organisational Data Quality. In E. M. Pierce und R. Katz-Haas, Hrsg., *Proc. of the 6th Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 398–411, 2001.

[JJNS98] M. A. Jeusfeld, M. Jarke, H. W. Nissen und M. Staudt. ConceptBase – Managing Conceptual Models about Information Systems. In P. Bernus, K. Mertins und G. Schmidt, Hrsg., *Handbook on Architectures of Information Systems*, S. 265–285. Springer, 1998.

[JJQ99] M. A. Jeusfeld, M. Jarke und C. Quix. Qualitätsanalyse im Data Warehousing. *Informatik/Informatique (Journal of the Swiss Information Processing Societies) No. 3*, S. 9–13, Juni 1999.

[JJQV99] M. Jarke, M. A. Jeusfeld, C. Quix und P. Vassiliadis. Architecture and Quality in Data Warehouses: An Extended Repository Approach. *Information Systems*, 24(3):229–253, 1999.

[JLVV00] M. Jarke, M. Lenzerini, Y. Vassiliou und P. Vassiliadis. *Fundamentals of Data Warehouses.* Springer, Berlin, 2000.

[Joh00] P. L. Johnson. *ISO 9000 – The Year 2000 and Beyond.* McGraw-Hill, New York, 2000.

[Jon99] R. M. Jones. *Introduction to MFC Programming with Visual C++.* Microsoft Technology Series, 1999.

[JPM+91] O. M. Jensen, D. M. Parkin, R. MacLennan, C. S. Muir und R. G. Skeet, Hrsg. *Cancer Registration: Principles and Methods (IARC Scientific Publications, Vol. 95).* International Agency for Research on Cancer (IARC), Lyon, 1991.

[JQJ98] M. A. Jeusfeld, C. Quix und M. Jarke. Design and Analysis of Quality Information for Data Warehouses. In *Proc. of the 17th Intl. Conf. on Conceptual Modeling (ER'98), Singapur*, S. 349–362, 1998.

[JR00] R. Jung und F. Rowohl. Vergleichende Analyse der Standardisierungsbestrebungen im Metadatenmanagement: Neue Metamodelle der MDC und OMG. In R. Jung und R. Winter, Hrsg., *Data Warehousing 2000. Methoden, Anwendungen, Strategien,* S. 113–133, Physica, Heidelberg, 2000.

[Jur93] J.-M. Juran. *Handbuch der Qualitätsplanung.* Verlag Moderne Industrie, 1993.

[JV97] M. Jarke und Y. Vassiliou. Data Warehouse Quality: A Review of the DWQ Project. In *Proc. of the 1997 Conf. on Information Quality, MIT, Cambridge, MA*, S. 98–112, 1997.

[Kao94] P. Kaomea. *Valuation of Data Quality: A Decision Analysis Approach*. Technischer Bericht, Massachusetts Institute of Technology, Cambridge, MA, 1994.

[Kem99] H.-G. Kemper. *Architektur und Gestaltung von Management-Unterstützungssystemen*. Teubner, 1999.

[KKPP98] D. Kaplan, R. Krishnan, R. Padman und J. Peters. Assessing Data Quality in Accounting Information Systems. *Communications of the ACM*, 41(2):72–78, 1998.

[Kra91] U. Krakau. *Eine Untersuchung zu Qualitätsmerkmalen und Qualitätsmaßen von Informationen.* Diplomarbeit, Universität Konstanz, 1991.

[KRRT98] R. Kimball, L. Reeves, M. Ross und W. Thornthwaite. *The Data Warehouse Lifecycle Toolkit.* Wiley, New York, 1998.

[KS96] V. Kashyap und A. P. Sheth. Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach. *VLDB Journal*, 5(4):276–304, 1996.

[KS99] V. Kamp und L. Sitzmann. MOA: A Hybrid Multidimensional Data Model for Interactive Data Exploration Based on Object-Relational Analysis Units. In *Proc. of the 5th Intl. Conf. on Information Systems Analysis and Synthesis (ISAS'99), Vol. 5*, S. 195–202. Intl. Institute of Informatics and Systemics (IIIS), 1999.

[KSD97] R. Kohavi, D. Sommerfield und J. Dougherty. Data Mining using MLC++ – A Machine Learning Library in C++. *International Journal of Artificial Intelligence Tools*, 6(4):537–566, 1997.

[KSW97] B. Kahn, D. M. Strong und R. Y. Wang. A Model for Delivering Quality Information as Product and Service. In *Proc. of the 1997 Conf. on Information Quality, MIT, Cambridge, MA*, S. 80–94, 1997.

[Kuh91] R. Kuhlen. Zur Theorie informationeller Mehrwerte. In *Wissensbasierte Informationssysteme und Informationsmanagement – Proc. 2. Intl. Symposium für Informationswissenschaft (ISI'91),* S. 26–39. Universitätsverlag Konstanz, 1991.

[Kur99] A. Kurz. *Data Warehousing – Enabling Technology*. MITP-Verlag, Bonn, 1999.

[Lew90] T. Lewandowski. *Linguistisches Wörterbuch*. Quelle & Meyer, Heidelberg, 1990.

[LJ99] P. Lehmann und J. Jaszewski. Business Terms as a Critical Success Factor for Data Warehousing. In S. Gatziu, M. Jeusfeld, M. Staudt und Y. Vassiliou, Hrsg., *Proc. of the Intl. Workshop on Design and Management of Data Warehouses (DMDW'99), Heidelberg*, S. 7/1–7/5, 1999.

[LLL00] M. L. Lee, T. W. Ling und W. L. Low. IntelliClean: A Knowledge-Based Intelligent Data Cleaner. In *Proc. of the 6th Intl. Conf. on Knowledge Discovery and Data Mining*, S. 290–294, 2000.

[LLLK99] M. L. Lee, H. Lu, T. W. Ling und Y. T. Ko. Cleansing Data for Mining and Warehousing. In *Proc. of the 10th Intl. Conf. on Database and Expert Systems Applications (DEXA'99), Florenz*, S. 751–760, 1999.

[LRO96] A. Y. Levy, A. Rajaraman und J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proc. of the 22nd Intl. Conf. on Very Large Databases (VLDB), Bombay*, S. 251–262, VLDB Endowment, Saratoga, CA, 1996.

[LZW+97] W. J. Labio, Y. Zhuge, J. L. Wiener, H. Gupta, H. Garcia-Molina und J. Widom. The WHIPS Prototype for Data Warehouse Creation and Maintenance. In *Proc. of the 1997 ACM SIGMOD Conf., Tuscon, Arizona*, S. 557–559, 1997.

[Mad93] S. E. Madnick. The Voice of the Customer: Innovative and Useful Research Directions. In *Proc. of the 19th VLDB Conference, Dublin*, S. 701–704, 1993.

[Mal99a] J. Maloney. *Distributed COM Application Development Using Visual C++ 6.0*. Prentice-Hall, 1999.

[Mal99b] C. Malorny. *TQM umsetzen.* Schäffer-Poeschel, Stuttgart, 1999.

[Man90] R. Manthey. Satisfiability of Integrity Constraints: Reflections on a Neglected Problem. In *Proc. of the 2nd Workshop on Foundations of Models and Languages for Data and Objects, Aigen, Österreich*, S. 169–179, September 1990.

[Mar98] W. Martin. *Data Warehousing, Data Mining, OLAP*. International Thomson Publishing, Bonn, 1998.

[MB98] H. Mucksch und W. Behme. *Das Data Warehouse-Konzept.* Gabler, Wiesbaden, 1998.

[Met99a] Meta Data Coalition. *Open Information Model, Version 1.1 (Proposal)*, 1999.

[Met99b] Meta Group. *Data Warehouse Scorecard*, 1999.

[MG94] F. P. Miville und R. von Gustke. Was ist Qualität, und wie sollte man Qualität verstehen, um erfolgreiches Qualitätsmanagement zu betreiben? *HMD – Theorie und Praxis der Wirtschaftsinformatik*, 175:8–19, 1994.

[MH01] J. Meister und H. Hinrichs. *CLIQ Data Migrator*. Technischer Bericht, OFFIS, 2001.

[Mic91] Z. Michalewicz, Hrsg. *Statistical and Scientific Databases.* Ellis Horwood Series in Computers and Their Applications, 1991.

[MKIS96] E. Mena, V. Kashyap, A. Illarramendi und A. P. Sheth. Managing Multiple Information Sources through Ontologies: Relationship between Vocabulary Heterogeneity and Loss of Information. In *Proc. of KRDB'96 (Knowledge Representation Meets Databases), Budapest*, S. 50–52, 1996.

[MKR+00] S. Mantel, B. Knobloch, T. Rüffer, M. Schissler, K. Schmitz, O. K. Ferstl und E. J. Sinz. *Analyse der Integrationspotenziale von Kommunikationsplattformen für verteilte Anwendungssysteme.* Universität Bamberg, FORWIN-Bericht FWN-2000-009, 2000.

[MM00a] J. I. Maletic und A. Marcus. Data Cleansing: Beyond Integrity Analysis. In B. D. Klein und D. F. Rossin, Hrsg., *Proc. of the 5th Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 200–209, 2000.

[MM00b] J. I. Maletic und A. Marcus. *Technical Report CS-00-02 Automated Identification of Errors in Data Sets.* University of Memphis, 2000.

[Mün95] B. Münzer. *Aspekte der Datenqualität – Analyse vorhandener Ansätze und weitergehende Systematik des Qualitätsbegriffes.* Diplomarbeit, Universität Konstanz, 1995.

[Mon00] R. Monson-Haefel. *Enterprise JavaBeans.* O'Reilly, 2000.

[MS94] D. L. Moody und G. Shanks. What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models. In P. Loucopoulos, Hrsg., *Proc. of the 13th Intl. Conf. on the Entity Relationship Approach (ER'94)*, S. 94–111, Springer, Manchester, Dezember 1994.

[MS98] D. R. Musser und A. Saini. *STL Tutorial and Reference Guide : C++ Programming with the Standard Template Library*. Addison-Wesley, 6. Auflage, 1998.

[MSR99] R. Müller, T. Stöhr und E. Rahm. An Integrative and Uniform Model for Metadata Management in Data Warehousing Environments. In S. Gatziu, M. Jeusfeld, M. Staudt und Y. Vassiliou, Hrsg., *Proc. of the Intl. Workshop on Design and Management of Data Warehouses (DMDW'99), Heidelberg*, S. 12/1–12/16, 1999.

[MSS01] A. Mädche, S. Staab und R. Studer. Ontologien. *Wirtschaftsinformatik*, 43(4):393–395, 2001.

[Nal01] P. Nallet. *OLE DB – Programmierung mit Consumer Templates.* Addison-Wesley, 2001.

[Nea90] R.-E. Neapolitan. *Probabilistic Reasoning in Expert Systems.* Wiley, New York, 1990.

[Nee01] M. P. Neely. A Proposed Framework for the Analysis of Source Data in a Data Warehouse. In E. M. Pierce und R. Katz-Haas, Hrsg., *Proc. of the 6th Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 33–43, 2001.

[NLF99] F. Naumann, U. Leser und J. C. Freytag. Quality-driven Integration of Heterogeneous Information Sources. In *Proc. of the 1999 Intl. Conf. on Very Large Databases (VLDB'99), Edinburgh*, S. 447–458, 1999.

[NR99] F. Naumann und C. Rolker. Do Metadata Models meet IQ Requirements? In *Proc. of the 4th Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 99–114, 1999.

[NR00] F. Naumann und C. Rolker. Assessment Methods for Information Quality Criteria. In B. D. Klein und D. F. Rossin, Hrsg., *Proc. of the 5th Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 148–162, 2000.

[Obj00] Object Management Group. *Common Warehouse Metamodel (CWM) Specification*, 2000.

[Obj01] Object Management Group. *Unified Modeling Language Specification Version 1.4*, 2001.

[Oes97] B. Oesterreich. *Objektorientierte Softwareentwicklung mit der Unified Modeling Language*. Oldenbourg, Wien, 1997.

[Ope02] Open Applications Group. *http://www.openapplications.org*, 2002.

[Orr98] K. Orr. Data Quality and Systems Theory. *Communications of the ACM*, 41(2):66–71, 1998.

[Ort95] E. Ortner. *Informationsmanagement I, Vorlesungsskript.* Universität Konstanz, 1995.

[OV99] M. T. Oszu und P. Valduriez. *Principles of Distributed Database Systems.* Prentice-Hall, Englewood Cliffs, NJ, 2. Auflage, 1999.

[PAG96] Y. Papakonstantinou, S. Abiteboul und H. Garcia-Molina. Object Fusion in Mediator Systems. In *Proc. of the 22nd Intl. Conf. on Very Large Databases (VLDB), Bombay*, S. 413–424, VLDB Endowment, Saratoga, CA, 1996.

[Pat93] B. Patterson. The Need for Data Quality. In *Proc. of the 19th VLDB Conference, Dublin*, S. 709, 1993.

[Pfe74] J. Pfestorf. *Kriterien für die Bewertung betriebswirtschaftlicher Informationen.* Dissertation, Freie Universität Berlin, Fachbereich Wirtschaftswissenschaft, 1974.

[Pij94] G. J. van der Pijl. Quality of Information and the Goals and Targets of the Organization. In *Proc. of the 1994 ACM SIGCPR Conference, Alexandria, USA*, S. 165–172, ACM Press, New York, 1994.

[Pok99] J. Pokorny. To the Stars through Dimensions and Facts. In *Proc. of the 3rd Intl. Conf. on Business Information Systems (BIS'99), Posen*, S. 135–147, Springer, London, 1999.

[Pos75] H.-J. Postel. Probleme beim Aufbau eines Informationssystems für Sicherheitsbehörden (II). *Datenverarbeitung in Steuer, Wirtschaft und Recht, 2/1975*, S. 55–61, 1975.

[PR97] V. Poe und L. Reeves. *Aufbau eines Data Warehouse.* Prentice-Hall, 1997.

[PS94] B.-U. Pagel und H.-W. Six. *Software Engineering.* Addison-Wesley, Bonn, 1994.

[PT87] H. Prade und C. Testemale. Representation of Soft Constraints and Fuzzy Attribute Values by means of Possibility Distributions in Databases. In J. Bezdek, Hrsg., *Analysis of Fuzzy Information: Vol. II, Artificial Intelligence and Decision Systems*, S. 213–229. CRC Press, Boca Raton, 1987.

[Qui93] R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, Los Altos, CA, 1993.

[Qui00] R. Quinlan. *GritBot – An Informal Tutorial.* http://www.rulequest.com/gritbot-unix.html, 2000.

[RD00] E. Rahm und H. H. Do. Data Cleaning: Problems and Current Approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*, 23(4):3–13, 2000.

[Red96] T. C. Redman. *Data Quality for the Information Age.* Artech House, Boston, 1996.

[Red98] T. C. Redman. The Impact of Poor Data Quality on the Typical Enterprise. *Communications of the ACM*, 41(2):79–82, 1998.

[Red99] G. Redeker. *Grundlagen des Qualitätsmanagements, Vorlesungsskript.* Institut für Qualitätssicherung, Universität Hannover, 1999.

[RG94] M. Reingruber und W. W. Gregory. *The Data Modeling Handbook: A Best-Practice Approach to Building Quality Data Models.* Wiley, 1994.

[RH00] V. Raman und J. M. Hellerstein. *Potters Wheel: An Interactive Framework for Data Cleaning and Transformation.* University of California at Berkeley, 2000.

[Rij79] C. J. Van Rijsbergen. *Information Retrieval.* Butterworth, London, 1979.

[RJB98] J. Rumbaugh, I. Jacobson und G. Booch. *The Unified Modeling Language Reference Manual.* Addison-Wesley, 1998.

[RN95] S. Russell und P. Norvig. *Artificial Intelligence – A Modern Approach*. Prentice-Hall, 1995.

[Rot96] J. Rothenberg. Metadata to Support Data Quality and Longevity. In *Proc. of the IEEE Metadata Conference, Silver Spring, MD*, April 1996.

[RW99] M. Rohde und F. Wietek. *Das Datenschema für das Epidemiologische Krebsregister Niedersachsen.* Technischer Bericht, OFFIS, Oldenburg, 1999.

[SAP97] SAP AG. *Business Information Warehouse – Technologie*. Technischer Bericht, SAP AG, Walldorf, 1997.

[SB97] I. Schmidtmann und H. Brenner. Auswirkungen von Record Linkage Fehlern auf die Nutzung von Krankheitsregistern: Empirische Beispiele. In R. Muche, G. Büchle, D. Harder und W. Gaus, Hrsg., *Medizinische Informatik, Biometrie und Epidemiologie (GMDS'97)*, S. 490–493, MMV Medizin Verlag, München, 1997.

[SC99] G. Shanks und B. Corbitt. Understanding Data Quality: Social and Cultural Aspects. In *Proc. of the 10^{th} Australasian Conf. on Information Systems*, S. 785–797, 1999.

[Sha99] G. Shanks. Semiotic Approach to Understanding Representation in Information Systems. In *Proc. of the 1999 Information Systems Foundations Workshop, Macquarie University, Sydney*, 1999.

[She31] W. A. Shewhart. *Economic Control of Quality of Manufactured Product.* D. Van Nostrand, New York, 1931.

[SL90] A. P. Sheth und J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[SLW97a] D. M. Strong, Y. W. Lee und R. Y. Wang. 10 Potholes in the Road to Information Quality. *Computer (IEEE)*, 30(8):38–46, 1997.

[SLW97b] D. M. Strong, Y. W. Lee und R. Y. Wang. Data Quality in Context. *Communications of the ACM*, 40(5):103–110, 1997.

[Sno90] R. T. Snodgrass. Temporal Databases – Status and Research Directions. *ACM SIGMOD Record*, 19(4):83–89, 1990.

[Sno95] R. T. Snodgrass, Hrsg. *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, Dordrecht, 1995.

[Som00] I. Sommerville. *Software Engineering.* Addison-Wesley, 6. Auflage, 2000.

[SR96] L. Seligman und A. Rosenthal. A Metadata Resource to Promote Data Integration. In *Proc. of the IEEE Metadata Conference, Silver Spring, MD*, April 1996.

[SR98] M. Sonnenschein und G. Reents. *Softwaretechnik, Vorlesungsskript.* Universität Oldenburg, 1998.

[SSL01] A. Seemann, B. Schmalzridt und P. Lehmann, Hrsg. *SAP Business Information Warehouse.* Galileo Press, Bonn, 2001.

[Sta92] R. Stamper. Signs, Organisations, Norms and Information Systems. In *Proc. of the 3^{rd} Australasian Conf. on Information Systems*, 1992.

[Sta99] Standish Group. *Migrate Headaches*, Februar 1999.

[Sta00] M. Stal. Reich der Mitte: Die Komponententechnologien COM+, EJB und CORBA Components. *OBJEKTspektrum*, 2000(3), 2000.

[SVV01] M. Staudt, A. Vaduva und T. Vetterli. *The Role of Metadata for Data Warehousing.* Department of Information Technology, Universität Zürich, Technischer Bericht 99.06., September 2001.

[Szy98] C. Szyperski, Hrsg. *Component Software*. Addison-Wesley, Harlow, UK, 1998.

[Tap99] H. Tapken. *Anfrageverarbeitung und -optimierung im MOA-Kontext*. Diplomarbeit, Universität Oldenburg, 1999.

[TB98] G. K. Tayi und D. P. Ballou. Examining Data Quality. *Communications of the ACM*, 41(2):54–57, 1998.

[TB99] D. Theodoratos und M. Bouzeghoub. Data Currency Quality Factors in Data Warehouse Design. In S. Gatziu, M. Jeusfeld, M. Staudt und Y. Vassiliou, Hrsg., *Proc. of the Intl. Workshop on Design and Management of Data Warehouses (DMDW'99), Heidelberg*, S. 15/1–15/16, 1999.

[Tha94] B. Thalheim. *A Survey on Database Constraints.* Technischer Bericht, Reihe Informatik I-8/1994, Fakultät für Mathematik, Naturwissenschaften und Informatik, Cottbus, 1994.

[TKS01] A. Tsois, N. Karayiannidis und T. Sellis. MAC: Conceptual data modeling for OLAP. In D. Theodoratos, J. Hammer, M. Jeusfeld und M. Staudt, Hrsg., *Proc. of the Intl. Workshop on Design and Management of Data Warehouses (DMDW 2001), Interlaken*, S. 5/1–5/11, 2001.

[TR00] T. Teschke und J. Ritter. Towards a Foundation of Component-Oriented Software Reference Models. In G. Butler und S. Jarzabek, Hrsg., *Generative and Component-Based Software Engineering, 2^{nd} Intl. Symposium (GCSE 2000), Erfurt, Revised Papers, LNCS 2177*, S. 70–84, 2000.

[TS97] D. Theodoratos und T. K. Sellis. Data Warehouse Configuration. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos und M. A. Jeusfeld, Hrsg., *Proc. of the 23^{rd} Intl. Conf. on Very Large Databases (VLDB), Athen*, S. 126–135. Morgan Kaufmann, 1997.

[Uni00] Unicode Consortium. *The Unicode Standard Version 3.0.* Addison-Wesley, 2000.

[VA96] M. W. W. Vermeer und P. M. G. Apers. The Role of Integrity Constraints in Database Interoperation. In *Proc. of the 22^{nd} Intl. Conf. on Very Large Databases (VLDB), Bombay*, S. 425–435, VLDB Endowment, Saratoga, CA, 1996.

[Val00] Vality. *The Integrity Data Reengineering Environment – Integrity Real Time User Guide for C, Version 3.6*, 2000.

[Val02] Vality. *http://www.vality.com*, 2002.

[VEE+00] V. S. Verykios, M. G. Elfeky, A. K. Elmagarmid, M. Cochinwala und S. Dalal. On the Accuracy and Completeness of the Record Matching Process. In B. D. Klein und D. F. Rossin, Hrsg., *Proc. of the 5^{th} Intl. Conf. on Information Quality, MIT, Cambridge, MA*, S. 54–69, 2000.

[VGD99] A. Vavouras, S. Gatziu und K. R. Dittrich. The SIRIUS Approach for Refreshing Data Warehouses Incrementally. In A. P. Buchmann, Hrsg., *Datenbanksysteme in Büro, Technik und Wissenschaft, 8. GI-Fachtagung BTW'99, Freiburg im Breisgau*, S. 80–96, Springer, Berlin, 1999.

[VVS00] T. Vetterli, A. Vaduva und M. Staudt. Metadata Standards for Data Warehousing: Open Information Model vs. Common Warehouse Metamodel. *SIGMOD Record*, 29(3):68–75, 2000.

[Wag01] K. W. Wagner. *PQM – Prozessorientiertes Qualitätsmanagement (Leitfaden zur Umsetzung der ISO 9001:2000).* Hanser, München, 2001.

[Wan98] R. Y. Wang. A Product Perspective on Total Data Quality Management. *Communications of the ACM*, 41(2):59–65, 1998.

[WB97] M.-C. Wu und A. P. Buchmann. Research Issues in Data Warehousing. In K. R. Dittrich und A. Geppert, Hrsg., *Datenbanksysteme in Büro, Technik und Wissenschaft, 7. GI-Fachtagung BTW'97, Ulm*, S. 61–82, Springer, Berlin, März 1997.

[Web99] R. Weber. *Information Systems Control and Audit*. Prentice-Hall, Upper Saddle River, NJ, 1999.

[Wes00] P. Westerman. *Data Warehousing: Using the Wal-Mart Model*. Morgan Kaufmann, 2000.

[WH98] D. Wells und M. Hope. *Ovum Evaluates: Data Warehousing Tools and Strategies*. Ovum, London, 1998.

[Wid95] J. Widom. Research Problems in Data Warehousing. In *Proc. of the 4^{th} Intl. Conf. on Information and Knowledge Management (CIKM'95)*, 1995.

[Wie98] J. H. Wieken. Meta-Daten für Data Marts und Data Warehouses. In H. Mucksch und W. Behme, Hrsg., *Das Data Warehouse-Konzept*, S. 275–315, Gabler, Wiesbaden, 1998.

[Wie00] F. Wietek. *Intelligente Analyse multidimensionaler Daten in einer visuellen Programmierumgebung und deren Anwendung in der Krebsepidemiologie*. Dissertation, Universität Oldenburg, Fachbereich Informatik, http://docserver.bis.uni-oldenburg.de/publikationen/dissertation/2000/wieint00/wieint00.html, März 2000.

[Wil00] T. Wilkens. *Entwicklung eines generischen Werkzeugs zum Data Auditing*. Diplomarbeit, Universität Oldenburg, 2000.

[Win99] W. E. Winkler. *The State of Record Linkage and Current Research Problems*. Technischer Bericht, U. S. Bureau of the Census, 1999.

[WK97] E. Woods und E. Kyral. *Ovum Evaluates: Data Mining*. Ovum, New York, 1997.

[Woo99] C. Wood. *OLE DB and ODBC Developer's Guide*. IDG Books Worldwide, 1999.

[Wor90] World Health Organization. *ICO-O – International Classification of Diseases for Oncology*. WHO, Genf, 1990.

[WRK92] R. Y. Wang, M. P. Reddy und H. B. Kon. *Toward Quality Data: An Attribute-Based Approach*. Technischer Bericht, Massachusetts Institute of Technology, Cambridge, MA, 1992.

[WSF95] R. Y. Wang, V. C. Storey und C. P. Firth. A Framework for Analysis of Data Quality Research. *Knowledge and Data Engineering*, 7(4):623–640, 1995.

[WSG94] R. Y. Wang, D. M. Strong und L. M. Guarascio. *Beyond Accuracy: What Data Quality Means to Data Consumers*. Technischer Bericht, Massachusetts Institute of Technology, Cambridge, MA, 1994.

[WW96] Y. Wand und R. Y. Wang. Anchoring Data Quality Dimensions in Ontological Foundations. *Communications of the ACM*, 39(11):86–95, 1996.

[ZS99] T. Zurek und M. Sinnwell. Data Warehousing Has More Colours Than Just Black & White. In *Proc. of the Industrial Session on Data Warehousing – Experiences, Challenges, and Directions, VLDB'99, Edinburgh*, 1999.

Zur Homepage der Dissertation

Index

Abstandsmaß, 72
ActiveX, 158, 159, 181
    Template Library, 157
Aggregation, 21, 247
AJAX, 51
Aktivitätsdiagramme, 131
AnalysisProcess, 135
Anforderungen, 247
    Kunden-, 25
    Operative, 129
    Qualitäts-, 130
    Realisierungs-, 131
    Technische, 130
    Wartungs-, 130
Anwendungsfälle, 131, 139
Ascential Quality Manager, 57
Assoziation, 247
ATL, *siehe* ActiveX Template Library
Attribut, 247
    Beschreibendes, 173
    Dimensionales, 173
    Fakt-, 173
Audit, 60, 115, 247
Ausreißer, 56
Auswertungswerkzeuge, 21, 247

Bedürfnispyramide, 49
Befunddatum, 80
Begriffstaxonomie, 23
Blocking, 88, 124, 247
Blockvariablen, 88, 107, 124

CARLOS, 193
CARTools, 193, 194
    CARAMEL, 193
    CARELIS, 193
    CARESS, 193
    CARTRUST, 193
CLIQ, 91
    -*Process*, 146, 218
    -*ProcessConfiguration*, 146, 218
    -*ProcessInfo*, 146
    -*ProcessResult*, 146, 218
    -Vorgehensmodell, 91, 131, 133, 134, 138, 139, 142, 145, 146, 148, 154, 160, 165, 168, 173, 178, 194, 200, 206, 222, 223, 228, 248
    -Workbench, 129, 130, 134, 137–139, 144, 146, 150, 171, 172, 177, 180, 185, 187, 217–219, 228, 248
        Console, 171, 178
ColumnSet, 170
COM, *siehe* Component Object Model
Common Warehouse Metamodel, 179, 230, 248
Component Object Model, 155, 180
ConceptBase, 63
Constraint, 93, 248
CRM, *siehe* Customer Relationship Management
Customer, 134, 145
Customer Relationship Management, 3, 227, 248
Cut-Off, 88, 107
CWM, *siehe* Common Warehouse Metamodel

DaimlerChrysler, 55, 223
Data
    Auditing, 235
    Auditor, 139, 168, 178, 184, 186
    Cleaning, *siehe* Data Cleansing
    Cleansing, 47, 235, 248
    Dictionary, 22
    Lineage, 51
    Linker, 139, 165, 178, 182, 186, 221
    Migration, 16, 248
    Migrator, 139, 154, 178, 182, 185
    Mining, 5, 18, 20, 21, 45, 55, 87, 116, 138, 168, 183, 230, 248
    Profiling, 45, 248
    Quality Mining, 55
    Warehouse, 3, 12, 119, 120, 248
    Warehouse-System, 3, 91, 130, 193, 227, 248
    Warehousing, 5, 11, 248
Database/Repository Interface, 171, 178, 186
DataObject, 134, 162
DataPumpTask, 156, 158
DataSource, 134

DataSphere, 63
Daten
-freigabe, 118
-integration, 4, 91, 130, 131, 178, 227, 248
-integrationsprozess, 22
-integrität, 249
-konsistenz, 249
-modell, 249
-objekt, 100, 113, 249
-produkt, 26, 41, 69, 91, 133, 249
-quelle, 14, 15, 132, 249
-rücknahme, 119
-schutz, 249
-sicherheit, 249
Datenbank, 248
-managementsystem, 248
Relationales, 253
-system, 248
Föderiertes, 4, 250
Operatives, 252
Datenqualität, 6, 26, 249
-sanalyse, 45, 46
-sindikator, 89, 112
-slenkung, 44, 115
-smanagement, 8, 39, 129, 227, 229, 235, 249
Methodenbasiert, 40
-smanagementsystem, 39, 130, 249
-smerkmale, 23, 28, 41, 227, 229
-smetriken, 44
-smängel, 32, 46
-splanung, 40
-ssicherung, 50
-sverbesserung, 46, 228
Symptomorientierte, 116
Ursachenorientierte, 117
DBInterface, 161, 172
DCN-Rate, 90
DCO-Rate, 89
Decision Support-System, 4
Dekorierer, 158
Deskriptive Statistik, 55
DEST_TABLES, 174
DiagramEngine, 150
Dimension, 21
DomainSpecificMeasurementEngine, 163, 219
Dominanzbeziehung, 114
Domäne, 249
Domänenwissen, 74
DQM, *siehe* Datenqualitätsmanagement
-Plan, 41
Controller, 139, 155, 178, 186, 217
DQMAttributeDescriptor, 162, 219
DQMDataDescriptor, 162
DQMS, *siehe* Datenqualitätsmanagementsystem
DQMTupleInconsistency, 164
Drill Through, 14
DSS, *siehe* Decision Support-System
DtActiveXWrapper, 159
DtCopyWrapper, 159
DtDTSPackageDecorator, 157
DtIntegrityWrapper, 159
DtPackage, 156
DTS, *siehe* Microsoft Data Transformation Services
DtTaskSet, 156
DtTransDefaultWrapper, 159
DtTransformWrapper, 158
DtUnitConverterWrapper, 159
Duplicate Elimination Sorted-Neighbourhood Method, 78
DWQ, *siehe* Foundations of Data Warehouse Quality
DWS, *siehe* Data Warehouse-System
-Administrator, 22
-Manager, 21, 249

Editierabstand, 53
Eindeutigkeit, 31, 82
Einheitlichkeit, 31, 70, 86, 96
EKN, *siehe* Epidemiologisches Krebsregister Niedersachsen
EM-Algorithmus, 107
Enterprise Application Integration, 249
Entity/Relationship-Modell, 249
Entität, 249
Entscheidungsregeln, 56
Epidemiologisches Krebsregister Niedersachsen, 191, 192, 228
Ereigniskanal, 186
Erfassungsfehler, 34
Erfüllungsgrad, 26, 31, 41
ETL
-Prozess, 14, 23, 182, 249
-Werkzeug, 14, 23, 51, 86, 159, 182, 235
Extension, 249
Extraktion, 15, 249
-skomponente, 15
Anfragegesteuert, 16
Ereignisgesteuert, 16
Periodisch, 16
Sofort, 16

Fachrelationen, 173
Feedback, 135
Fehler, 32, 250
Foundations of Data Warehouse Quality, 63
Fragile Base Class Problem, 185
Fremdschlüsselattribut, 250
Frequenzanalyse, 89

Garbage in, garbage out, 7, 227
Gateway, 16
Genauigkeit, 30, 79, 82, 83
Generalisierung, 250
Geschäftsprozessoptimierung, 7
Geschäftsregeln, 22, 59
Gesetz über Krebsregister, 191
Glaubwürdigkeit, 30
Goal-Question-Metric–Ansatz, 44, 63
GQM, *siehe* Goal-Question-Metric–Ansatz
Granularitätsebenen, 69
GritBot, 56

HiQiQ, 63
Historisierung, 16, 119, 250
Homonymfehler, 18, 89, 165, 250
House of Quality, 43
Hub-and-Spoke–Architektur, 14, 250
HV-Anteil, 90

Identität, 250
IDispatch, 157
IDTSDataPumpTransform, 157
ILOG Rules, 164, 178, 184, 195
    -Agenda, 165
    -Arbeitsspeicher, 165
    -Kontext, 164
    -Programm, 164
InconsReview, 164, 165
InconsReviewDB, 164, 165
Index, 250
IndicatorPlan, 162
Indikatorplan, 112, 142
InfoCube, 220
Information, 27
    -ssystem, 250
    Retrieval, 70, 229
Inkrementelle Updates, 106, 154, 172
Integration
    -sabschnitte, 122
    -saufgabe, 133, 139
    -sdatenbank, 172, 174
    -sschicht, 14, 91, 250
    -sschritte, 206
Integrität
    Referentielle, 253
IntelliClean, 63
Intension, 250
Interpretierbarkeit, 30
Interschema Knowledge Networks, 63
ISO 9000:2000, 38
ISO 9001, 38, 91, 119, 228, 229

Job, 134, 145
    -*Analyze*, 150
    -*AnalyzeDB*, 154
    -*AnalyzeEngine*, 150, 217
    -*Configuration*, 134, 145
    -*ControlDB*, 153
    -*ControlDBEngine*, 153, 218
    -*Exec*, 146
    -*ExecDB*, 153
    -*ExecEngine*, 146, 217
    -*Model*, 144
    -*PlanEngine*, 145, 217, 218
    -*RunInfo*, 148
    -*Step*, 148
    -*StepController*, 148
    -*StepResult*, 149
    -*Thread*, 149

Kaizen, 38
Kalmanfilter, 87
KDD, *siehe* Knowledge Discovery in Databases
KeyIntegrityMeasurementEngine, 165
Klasse, 250
    -ndefinition, 250
    Abstrakte, 247
    Basis-, 247
    Business-, 131, 134, 144
    Sub-, 254
Knowledge Discovery in Databases, 5, 250
Kohäsion, 185, 250
Komponentenbasierte Softwareentwicklung, 185
Komposition, 250
Konfigurierungsschritte, 200
Konformität, 251
Konkurrenzbeziehung, 117
Konsistenz, 30, 74, 82, 83, 87, 116
    -regeln, 74
Konsolidierung, 251
    -sbereich, 92, 110, 111, 118, 120, 121, 126, 160, 166, 251
Kontrollgrenzen, 74, 100
Kontrollkarte, 101

Kontrollnummern, 193
Korrektheit, 30, 71, 86
Kosten
-arten, 48
-faktoren, 8
KRG, *siehe* Gesetz über Krebsregister
Kunde, 132
Kundenfeedback, 119, 133

Ladekomponente, 21
Laden, 251
Offline-, 21
Online-, 21
Löschanweisung, 121, 251

m-Wahrscheinlichkeit, 107
M/I-Index, 90
Managementinformationssystem, 4
Mangel, 251
MatchResult, 167
Matchvariablen, 88, 107
MdsDataAuditingControl, 169
MdsDataMiningManager, 169
MdsDateiInteraktionManager, 169
MdsDbInteraktionManager, 169
MdsDomMdManager, 170
MdsQuellMdManager, 170
Measurement Engines, 139, 178, 186, 221
MeasurementEngine, 161–163
MEDAS, *siehe* Data Auditor
-AnalysisProcess, 170
-AnalysisResult, 170
-Cluster, 170
-ClusterSet, 170
-DecisionTree, 170
-IntervalRepresentative, 170
-NearestNeighborSet, 170
-Rule, 170
-RuleSet, 170
-TargetAttribute, 170
Mehrfachvererbung, 251
Merging, 54, 109, 124, 166, 182, 193, 195, 223, 251
Merkmal, 251
Messbarkeit, 26
Messmittel, 251
Messprozess, 251
Messung, 251
Metadaten, 22, 251
-Reengineering, 137
-Repository, 19, 22, 130, 155, 251
Domänenspezifische, 22
Technische, 22
Metarelationen, 173
Methode, 251
Metriken, 69, 251
MFC, *siehe* Microsoft Foundation Classes
Microsoft
Data Transformation Services, 155, 178, 181
Excel, 150, 178
Foundation Classes, 178
Internet Explorer, 178, 206
Repository, 178, 179
SQL Server, 177, 179
MLC++, 169, 178, 183
Monitor, 14, 251
Monitoring, 15, 251
Log-basiert, 15
Replikationsbasiert, 15
Snapshot-basiert, 15
Trigger-basiert, 15
Zeitstempelbasiert, 15
MrDbmPackage, 171
MrDbmRepository, 171
MrRepository, 156, 161, 171, 218
Multi Source-Probleme, 32
Multidimensionales
Datenmodell, 13
Schema, 251
Multiplizität, 251
Mängel, 32

Neighbourhood Hash Joins, 78
Normierung, 69
Nullwert, 252
-semantik, 18
Nutzungsauflagen, 116
Nützlichkeit, 30

Objekt, 252
Objektorientierte Analyse, 131
Objektorientierter Entwurf, 131, 138
ODBC, 16, 178
ODS, *siehe* Operational Data Store
OIM, *siehe* Open Information Model
OLAP, *siehe* Online Analytical Processing
Relational, 4, 253
OLE DB, 16, 155, 181
OLTP, *siehe* Online Transaction Processing
Online Analytical Processing, 4, 21, 252
Online Transaction Processing, 3, 126, 252
OOA, *siehe* Objektorientierte Analyse
OOD, *siehe* Objektorientierter Entwurf

Open Information Model, 180, 252
Data Transformations Model, 155, 180, 181
Database and Warehousing Model, 171
Relational Database Schema Model, 171, 180
Repository Type Information Model, 179
Operational Data Store, 14, 16, 220, 252
OPS-5, 103, 164, 184
Oracle 8i, 177
Organisation
-sleitung, 131, 195
-sstrukturen, 22

Pakete, 137
Parallelisierbarkeit, 122
Pareto-Chart, 101
PDCA, *siehe* Plan-Do-Check-Act–Zyklus
Persistent Staging Area, 219
PerspectiveWall, 164, 168
Phonetischer Code, 19
Plan-Do-Check-Act–Zyklus, 38, 62, 252
Potter's Wheel A-B-C, 63
Pragmatik, 27
Primärschlüsselattribut, 252
Process, 134
ProcessConfiguration, 135
ProcessResult, 135
Produkt, 252
ProtocolInformation, 135
Prozess, 252
-orientiertes QMS, 38
-redesign, 47
-verbesserung, 47
Prüfung, 252
PSA, *siehe* Persistent Staging Area

QDP_SRC_TABLES, 174
QFD, *siehe* Quality Function Deployment
QMS, *siehe* Qualitätsmanagementsystem
Qualifizierungsprozess, 252
Quality
-*Measurement*, 162
MeasurementPlan, 162
-*Plan*, 162
Function Deployment, 42, 132, 230, 253
of Service, 28
Qualität, 6, 252
-sindikator, 195
-slenkung, 252
-smanagement, 8, 37, 252
-Handbuch, 252
-smanagementphilosophien, 38
-smanagementplan, 252
-smanagementsystem, 38, 253
-smerkmal, 253
-smessung, 111
-splan, 112, 117, 142
-splanung, 253
-spolitik, 253
-ssicherung, 253
-sverbesserung, 253
-sziel, 40, 253
Quelldatenpuffer, 92, 95, 121, 123, 140, 154, 173, 182, 197, 253
Quellsystem, 15
QUIS, 55, 223

RecLink, 166
-*Conf*, 167
-*Cover*, 167
-*Exec*, 167
-*ExecDB*, 167, 218
-*Pass*, 167
-*Review*, 166, 168, 182
-*ReviewDB*, 168, 218
Key-, 166
Prob-, 166
Record Linkage, 78, 87, 124, 182, 195, 223, 235, 253
Probabilistisches, 78, 88, 106, 193
Schlüsselbasiertes, 106
RecordInfo, 167
Redundanz, 253
-freiheit, 31, 77, 87
Referentielle Integrität, 31, 84, 95
Registerstelle, 192, 193
Relation, 253
-enmodell, 253
Release/Retraction Handler, 139, 170, 178, 186
Relevanz, 31, 70
Requirement, 134
ROLAP, *siehe* OLAP, Relational

SAP Business Warehouse, 219
Schema, 253
-integration, 17, 137
Schlüssel, 253
-eindeutigkeit, 31, 83
-integrität, 30, 95
Schneeflockenschema, 4, 13
SCM, *siehe* Supply Chain Management
Semantik, 27

Semiotik, 27, 253
Semiotisches Dreieck, 27
Sicht, 254
Single Source-Probleme, 32
Skalenniveau, 70
Skalierbarkeit, 254
Snowflake Schema, *siehe* Schneeflockenschema
Softwarekomponente, 185, 254
SOURCES, 176
SPC, *siehe* Statistische Prozesskontrolle
-*Engine*, 162
-*ParetoChartEntry*, 162
-Plan, 100
-*Plan*, 162
-*XChartEntry*, 162
Spezialisierung, 254
SRC_TABLES, 176
Staging Area, 14
Star Schema, *siehe* Sternschema
Statistische Inferenz, 115
Statistische Prozesskontrolle, 73, 99, 123, 142, 195, 223, 254
Statusattribute
Externe, 175
Interne, 175
Stelligkeit, 79
Sternschema, 4, 13
Supply Chain Management, 3, 227, 254
Surrogat, 94, 254
Synonymfehler, 18, 89, 165, 254
Syntaktik, 27
System-Bi-Cycle, 26
Systemnutzer, 132, 196
SystemUser, 134, 150

Task Set, 97, 99, 156
TDQM, *siehe* Total Data Quality Management
Total Data Quality Management, 8, 40, 62
Total Quality
Control, 38
data Management, 40
Management, 8, 38, 57, 60, 62, 254
TQC, *siehe* Total Quality Control
TQM, *siehe* Total Quality Management
TransactionManager, 150, 218
TransactionManagerDB, 154
Transaktion, 254
Transferabschnitte, 122
Transformation, 254
-sbereich, 14, 15, 92, 96, 100, 105, 110, 121, 154, 160, 166, 182, 254
-skomponente, 16
-soperatoren, 52
-spaket, 97, 98, 155, 156, 181
TransitionManager, 150, 218
Transitiver Abschluss, 109
Tupel, 254
Typ I-Fehler, 45

u-Wahrscheinlichkeit, 107
UML, *siehe* Unified Modeling Language
UNICODE, 19
Unified Modeling Language, 131, 180, 254
UnitConversion, 159, 178
Update
-Anweisungen, 104
-Häufigkeit, 80
User Satisfaction, 70

Vality Integrity, 159, 167, 178, 182
Verbesserungsmaßnahmen, 86
Vererbung, 254
Verständlichkeit, 31, 70
Vertrauensstelle, 192, 193
Vollständigkeit, 31, 76, 82, 83, 87, 116

WfMS, *siehe* Workflow Management System
Wiederverwendung, 186
Wissen, 27
Workflow Management System, 23, 143, 254
Wrapper, 92, 183

X-Chart, 101
Xerces, 152, 178
XMLEngine, 152
XMLRunInfo, 152

YADI, 169, 178

Zeitnähe, 31, 80
Zertifizierung, 50
Zielbereich, 92, 105, 111, 115, 118, 120, 121, 126, 160, 199, 255
Zufallsstichproben, 115
Zuverlässigkeit, 30

Zur Homepage der Dissertation

Lebenslauf

Persönliche Daten

Name:	Holger Hinrichs
Geburtsdatum/-ort:	18. Mai 1971, Westerstede
Staatsangehörigkeit:	deutsch

Schulbildung

1977 – 1981	Besuch der Brakenhoff-Schule in Westerstede (Grundschule)
1981 – 1983	Besuch der Robert-Dannemann-Schule in Westerstede (Orientierungsstufe)
1983 – 1990	Besuch des Gymnasiums Westerstede
5/1990	Erlangung der Allgemeinen Hochschulreife (Abitur)

Hochschulstudium

1990 – 1996	Studium der Informatik an der Carl von Ossietzky Universität Oldenburg
1/1996	Erlangung des akademischen Grads Diplom-Informatiker

Berufstätigkeit

1996 – 2002	Wissenschaftlicher Mitarbeiter am Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme (OFFIS), Oldenburg