INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Evaluation der Leistungsfähigkeit des Datenqualitätsmanagements mit Uniserv-Werkzeugen

Robert Reisinger

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Evaluation der Leistungsfähigkeit des Datenqualitätsmanagements mit Uniserv-Werkzeugen

Robert Reisinger

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	PD Dr. Wolfgang Hommel Dr. Latifa Si Youcef (iC Consult GmbH) Eugen Alter (iC Consult GmbH)
Abgabetermin:	28. September 2012

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 28. September 2012

. . . . . .
(Unterschrift des Kandidaten)

Abstract

Das Thema Datenqualität hat in den letzten Jahren zunehmend an Bedeutung gewonnen, was sich auch an der hohen Anzahl von Anbietern für Datenqualitätslösungen wiederspiegelt. In dieser Bachelorarbeit soll, in Zusammenarbeit mit der Firma iC Consult GmbH, das Produkt eines dieser Anbieter, nämlich der Firma Uniserv, evaluiert werden. Da eine komplette Evaluation des Systems den Rahmen dieser Arbeit sprengen würde, wird lediglich der Teilbereich der Adressvalidierung untersucht. Dabei handelt es sich um eine Prüfung und Korrektur von Adressdaten, hinsichtlich ihrer Korrektheit. Die Evaluation der Adressvalidierung erfolgt zum einen durch die Gegenüberstellung des Systems mit einem vorher festgelegten Anforderungsprofil und zum anderen durch die Gegenüberstellung mit einem Third Party Tool aus dem Hause Tolerant. Hierzu werden beide Systeme installiert und Testdaten in die Systeme eingespielt. Schließlich werden dann die Ergebnisse dieser Testläufe miteinander verglichen und somit eine Evaluation der Adressvalidierung von Uniserv erstellt.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

In den letzten 50 Jahren hat sich unser Wirtschaftssystem immer mehr von einer Industriewirtschaft in eine Art „Informationswirtschaft“ gewandelt. Nicht mehr nur das hergestellte Produkt steht im Vordergrund, sondern auch die damit zusammenhängenden Informationen über Herstellungsprozesse, Lieferketten und vor allem auch über den Kunden. Um konkurrenzfähig zu bleiben wird es zunehmend wichtiger für Unternehmen, möglichst viele Informationen zu sammeln, effizient zu gliedern, aufzubereiten und auszuwerten. Diese Informations- und Datenanhäufung führt aber, ohne das richtige Datenmanagement, fast unweigerlich zu fehlerhaften oder redundanten Daten und in Folge dessen zu schlechter Datenqualität. Laut der Ansicht einer Studie sind beispielsweise etwa 2% der Kundendatensätze innerhalb eines Monats veraltet (z.B. durch Umzug, Heirat oder Tod des Kunden)[Ech02].
Die dadurch entstehende, schlechte Datenqualität verursacht laut einer Schätzung des TDWI (The Data Warehousing Institute) aus dem Jahr 2002, bei US- Unternehmen einen finanziellen Schaden von 611 Milliarden Dollar pro Jahr. Dieser Betrag beinhaltet aber lediglich Druck-, Porto- und Personalkosten für Unterlagen, die an fehlerhafte oder doppelt vorhandene Adressen gesendet wurden. Es kann also davon ausgegangen werden, dass der tatsächliche Schaden deutlich höher anzusiedeln ist[Ech09].
Um diese Schäden zu verhindern oder zumindest zu minimieren gibt es spezielle Verfahren, die auf das Erkennen, Überwachen und Bereinigen/Korrigieren von fehlerhaften Daten ausgelegt ist. Auch die Anreicherung der Datensätze mit zusätzlichen Informationen und die Prävention vor der Eingabe (mit Eingabe ist hierbei sowohl die Eingabe von Daten per Hand als auch die automatische Dateneingabe via Datenbankenkonsolidierung gemeint) fehlerhafter Daten wird von diesen sogenannten Data Quality Management Systemen (DQMS) übernommen.

1.2 Zielsetzung

Im Rahmen dieser Bachelorarbeit soll in Kooperation mit der Firma iC Consult GmbH ein DQMS untersucht werden, das mit der Software der Firma Uniserv umgesetzt ist. Da der gesamte Umfang eines solchen Systems den Rahmen dieser Arbeit sprengen würde, werden lediglich die vier folgenden Teilbereiche betrachtet, welche noch einmal ausführlich in Kapitel 2.5 erörtert werden, wobei davon nur der Bereich der Adressvalidierung praktisch evaluiert wird. Diese DQM-Szenarien beziehen sich immer auf einen Teilbereich des gesamten DQM. So beschäftigt sich beispielsweise das Szenario Adressvalidierung ausschließlich mit der Richtigkeit von Adressdaten, ist jedoch nicht für das Erkennen von doppelten Adressen zuständig. Dies würde in den Aufgabenbereich des Dublettenchecks fallen. Im Anschluss werden nun für vier wichtige Teilmodule eines DQMS die Aufgabenbereiche und deren Umsetzung kurz erörtert.

Adressvalidierung Die Adressvalidierung prüft, korrigiert und aktualisiert postalische Komponenten. Es soll also eine fehlerhafte Adresseingabe noch vor dem Schreiben des Adressdatensatzes in die Datenbank verhindert werden oder aber falsche Datensätze aufgespürt werden. Dies wird über Ähnlichkeitssuchen in Kombination mit Referenzdatensätzen realisiert. Eine funktionierende Adressvalidierung ist notwendig, um Kosten für manuelle Adresskorrekturen (durch Sachbearbeiter) aber auch Kosten durch falsch versendete Postsendungen zu minimieren.

Dublettenprüfung Mit der Dublettenprüfung sollen redundante Datensätze erkannt und gelöscht werden. Auch dieses Modul arbeitet mit heuristischen Verfahren, die Ähnlichkeiten von Datensätzen erkennen sollen und ab einer festgelegten Übereinstimmungsquote als Dubletten erkannt werden. Dies ist eine der wichtigsten Funktionen eines DQMS, da sich Dubletten in vielen Unternehmensbereichen negativ auswirken. Sind beispielsweise in einem ERP-System für denselben Artikel mehrere Datensätze in der Artikeldatenbank vorhanden, so führt dies u.U. zu Unstimmigkeiten bei den Sachbearbeitern, die dann unter Zeitaufwand (und somit auch finanziellem Aufwand) die Unstimmigkeiten beseitigen müssen. Noch gravierender sind die Auswirkungen von Dubletten in der Kundendatenbank. Hierbei kann es z.B. im Falle einer Werbeaktion zu Mehrfachsendungen von Werbematerial an denselben Kunden kommen. Dadurch entstehen für jede unnötige Sendung auch Versand- und Druckkosten, die durch eine funktionierende Dublettenprüfung verhindert werden könnten.

Embargo-Listen (SPL) Eine Embargo-Liste oder Sanctioned-Party-List (SPL) wird eine Liste von Personen oder Organisationen genannt, die aufgrund von gesetzlichen oder politischen Vorgaben keine Leistungen erhalten dürfen. Auf dieser Liste sind beispielsweise Terroristen oder politische Führungspersönlichkeiten aus Ländern, welche mit Sanktionen belegt sind. Da eine versehentliche Leistungserbringung an diese Personen nicht nur juristische Folgen, sondern auch einen Imageschaden zur Folge haben kann, ist ein Abgleich dieser Listen mit den Kundenstammdaten notwendig.

Bankdatenvalidierung Die Bankdatenvalidierung prüft, korrigiert und aktualisiert Zahlungs- und Bankdaten. Dazu gehören neben der Zuordnung von BLZ und Bankname auch die Prüfung von IBAN, Kreditkartennummer oder SWIFT-Code. Hierbei ist eine besonders hohe Prüfungsqualität notwendig, da schon kleinste Fehler, wie z.B. Zahlendreher, zu einer Fehlüberweisung oder Fehlabbuchung führen können. Dies kann im Falle einer Lieferantenrechnung zur Folge haben, dass Mahngebühren bezahlt werden müssen, da die Fehlüberweisung erst spät erkannt wird. Aber auch bei einem Lastschriftverfahren mit dem Kunden ist es wichtig, dass fehlerhafte Zahlungsdaten schon möglichst bei der Eingabe erkannt werden. Dies verhindert zeitaufwändige Nachbesserungsarbeiten, die wieder Kosten nach sich ziehen würden.

Diese vier DQM-Szenarien sollen, im Rahmen dieser Arbeit, untersucht werden, wobei die Adressvalidierung mit der DQM-Software der Firma Uniserv implementiert und anschließend anhand eines vorher definierten Anforderungskatalogs evaluiert werden. Zusätzlich soll die Adressvalidierung der Uniserv-Software mit dem entsprechenden Modul der DQM-Software von Tolerant verglichen werden, welche sich im Moment bei einem Automobilhersteller im Einsatz befindet. Beide Softwareprodukte müssen zu diesem Zweck auf einem Testsystem

installiert und konfiguriert werden.
Die Softwarelösung von Uniserv wurde gewählt, da diese sich hauptsächlich auf den Umgang mit Kundendaten konzentriert, was sich mit den vier Teilbereichen deckt, die in dieser Arbeit behandelt werden. Uniserv ist darüber hinaus mit über 40 Jahren Erfahrung im Bereich des Data Quality Management der größte europäische Anbieter von reinen Datenqualitätslösungen, der im Jahr 2011 80% des Umsatzes in Deutschland und Frankreich erzielte, sich aber auch auf dem europäischen und dem U.S.-Markt etablieren konnte. Ein weiterer Punkt für das Produkt von Uniserv ist die hohe Kompatibilität mit allen gängigen CRM-Systemen (u.a. SAP, Siebel, PeopleSoft) und Systemplattformen (u.a. Windows, Linux/Unix, BS200)[FB11].

1.3 Struktur dieser Arbeit

Nachdem nun ein erster Überblick über die Thematik gegeben wurde, sollen in Kapitel 2 die theoretischen Konzepte vermittelt werden, die für diese Arbeit relevant sind. Hierzu zählen in erster Linie die Definitionen von Datenqualität und Datenqualitätsmanagement. Darüber hinaus werden die Ursachen und Auswirkungen von schlechter Datenqualität und die oben erwähnten DQM-Szenarien genauer betrachtet. In Kapitel 3 soll dann eine Anforderungsanalyse für ein DQMS und die Adressvalidierung erstellt werden. Diese Analyse beinhaltet sowohl allgemeine Kriterien, die eine DQM-Software erfüllen soll, sowie spezielle Anforderungen für den Bereich der Adressvalidierung. In Kapitel 4 folgt dann eine Betrachtung von aktuellen DQM-Tools, sowie eine Bewertung des aktuellen Entwicklungsstandes dieser Produkte. In Kapitel 5 werden die wichtigsten Module und Komponenten der Uniserv-Software vorgestellt. Die konkrete Umsetzung, also die Installation und Konfiguration der Software sowie die Implementierung der Adressvalidierung und deren fachliche Tests, werden in Kapitel 6 erläutert. In Kapitel 7 wird die Installation und Implementierung der Adressvalidierung mit dem Vergleichsprodukt von Tolerant erklärt. Außerdem werden auch hier die fachlichen Tests implementiert und durchgeführt. In Kapitel 8 wird die Evaluation der Testergebnisse anhand der, in Kapitel 3 erstellten Anforderungsanalyse, durchgeführt. Außerdem wird die Software von Uniserv mit dem Third Party Tool aus dem Hause Tolerant verglichen. Kapitel 9 fasst schließlich die grundlegenden Erkenntnisse dieser Arbeit zusammen.

2 Theoretische Grundlagen

2.1 Definition von Datenqualität

Für den Begriff „Datenqualität“ gibt es in der Literatur keine eindeutige Definition, da er sehr individuell und subjektiv geprägt ist. So bilden beispielsweise für Mitarbeiter der Finanzbuchhaltung vor allem die Richtigkeit und Exaktheit von Unternehmensdaten den Hauptbestandteil der Datenqualität. Für die IT-Abteilung könnten jedoch Schlüsseleindeutigkeit oder Redundanzfreiheit als Basiskriterien für gute Datenqualität gelten. Aus dieser Problematik entstand der Ansatz von Larry English, den Begriff in seine Bestandteile zu zerlegen - also „Daten“ und „Qualität“ - und diese erst separat zu betrachten und zu definieren, um sich daraus die Bezeichnung „Datenqualität“ abzuleiten. Dies ist auch der Ansatz den diese Arbeit verfolgt, weswegen im Folgenden die Einzelbegriffe definiert und daraus auf „Datenqualität“ geschlossen wird [ABEM10, S.16].

Daten Auch der Begriff der Daten ist nicht einheitlich definiert, da dieser stark mit dem Informations- und Wissensbegriff korreliert. Der Zusammenhang zwischen diesen Bezeichnungen wird mit Hilfe einer Unterteilung in syntaktische, semantische und pragmatische Ebene (Semiotik) dargestellt. Die syntaktische Ebene bezieht sich ausschließlich auf die Struktur von Zeichenfolgen, ohne den Zeichen/Zeichenfolgen eine Bedeutung zuzuweisen. Diese Folgen werden in der Semiotik auch als Daten bezeichnet. Die semantische Ebene erweitert diese Daten dahingehend, dass ihnen eine Bedeutung zugeteilt wird. Ist dies der Fall, so spricht man von Informationen. Die pragmatische Ebene erweitert den Informationsbegriff dadurch, dass der Information eine Wirkung auf den Nutzer dieser Information zugeordnet wird. In diesem Stadium spricht man von Wissen. Im Rahmen dieser Arbeit ist mit Datenqualität nicht nur die Qualität der Daten im Sinne dieser Definition gemeint, sondern auch immer die Qualität der Informationen. Sollte eine Differenzierung notwendig sein wird dies explizit angegeben werden [ABEM10, S.16-17].

Qualität Laut DIN-Norm 55350 ist „Qualität die Gesamtheit von Eigenschaften und Merkmalen eines Produktes oder einer Tätigkeit, die sich auf deren Eignung zur Erfüllung festgelegter oder vorausgesetzter Erfordernisse bezieht“. Diese Definition bezieht sich zwar auf die Fertigungs- und Dienstleistungsindustrie, kann jedoch auch auf die Qualität von Daten angewandt werden. So kann man den Auszug aus einem Datenbestand als (Daten-)Produkt ansehen, welchem man Eigenschaften und Merkmale zuordnet (siehe Datenqualität). Eine weitere Ähnlichkeit zur Fertigung physischer Produkte ist, dass auch das Datenprodukt wenn nötig angepasst, transformiert oder aufbereitet werden kann, um den vorausgesetzten Erfordernissen, also den Anforderungen des Benutzers dieses Datenprodukts, gerecht zu werden [ABEM10, S.17-19].

Datenqualität Apel, Behme, Eberlein und Merighi kommen zu dem Schluss, „dass die Qualität von Daten zum Zeitpunkt der Betrachtung sowie von dem zu diesem Zeitpunkt

Datenqualitätskriterium	Definition
Korrektheit/Fehlerfreiheit	Die Daten stimmen mit der Realität überein.
Vollständigkeit	Die Attributwerte eines Datensatzes sind mit Werten belegt, die semantisch vom Wert NULL abweichen.
Aktualität	Alle Datensätze entsprechen jeweils dem aktuellen Zustand der modellierten Welt und sind somit nicht veraltet.
Redundanzfreiheit	Innerhalb der Datensätze dürfen keine Duplikate vorkommen. Duplikate sind hierbei Datensätze, die dieselbe Entität in der realen Welt beschreiben. Diese müssen aber nicht notwendigerweise in allen Attributen übereinstimmen.
Einheitlichkeit	Die Repräsentationsstruktur einer Menge von Datensätzen ist einheitlich.
Eindeutigkeit	Ein Datensatz muss eindeutig interpretierbar sein.
Konsistenz	Die Attributwerte eines Datensatzes weisen keine logischen Widersprüche untereinander oder zu anderen Datensätzen auf.

Abbildung 2.1: Datenqualitätskriterien[ABEM10]

an die Daten gestellten Anspruchsniveau abhängt“. Um diese Qualität ermitteln zu können benötigt man, wie oben bereits erwähnt, objektive Merkmale (Datenqualitätskriterien), die „aufgrund der praktischen Erfahrung intuitiv definiert, auf Basis von Literaturrecherchen erstellt oder anhand von empirischen Untersuchungen zusammengestellt“ werden [ABEM10, S.19]. In Abbildung 2.1 sollen die Datenqualitätskriterien vorgestellt werden die für diese Arbeit besonders relevant sind. Es handelt sich dabei lediglich um eine kleine Teilmenge aus den theoretisch möglichen Qualitätskriterien.

2.2 Definition von Datenqualitätsmanagement

Der Grundgedanke des Datenqualitätsmanagement lässt sich vereinfacht in einem sogenannten Plan-Do-Check-Act-Zyklus (siehe Abbildung 2.2) darstellen. In der Planungsphase werden die Qualitätskriterien und Werte für diese Kriterien festgelegt, die erreicht werden sollen. Diese Richtlinien werden in der Ausführungsphase durch technische Umsetzungen, Personalschulungen oder anderen Techniken realisiert. In der Überprüfungsphase wird schließlich das Ergebnis evaluiert aus dem dann wiederum, in der Verbesserungsphase, Verbesserungen abgeleitet werden können. Dies führt zu Änderungen oder Neuerungen in der Planungsphase, womit sich der Zyklus schließt. Dieser Basiszyklus ist das Vorbild für komplexere Modelle wie z.B. die Total Quality Data Management Methodik nach English. Die Grundidee des Plan-Do-Check-Act-Zyklus ist jedoch auch in diesem Modell vorhanden [ABEM10, S.25]. Wichtig ist hierbei zu erkennen, dass es sich bei Datenqualitätsmanagement nicht um eine einmalige Reparatur des Datenbestandes geht, sondern um einen permanenten Prozess, der oft auch abseits der technischen Ebene durchgeführt werden muss (z.B. durch Mitarbeiterschulungen

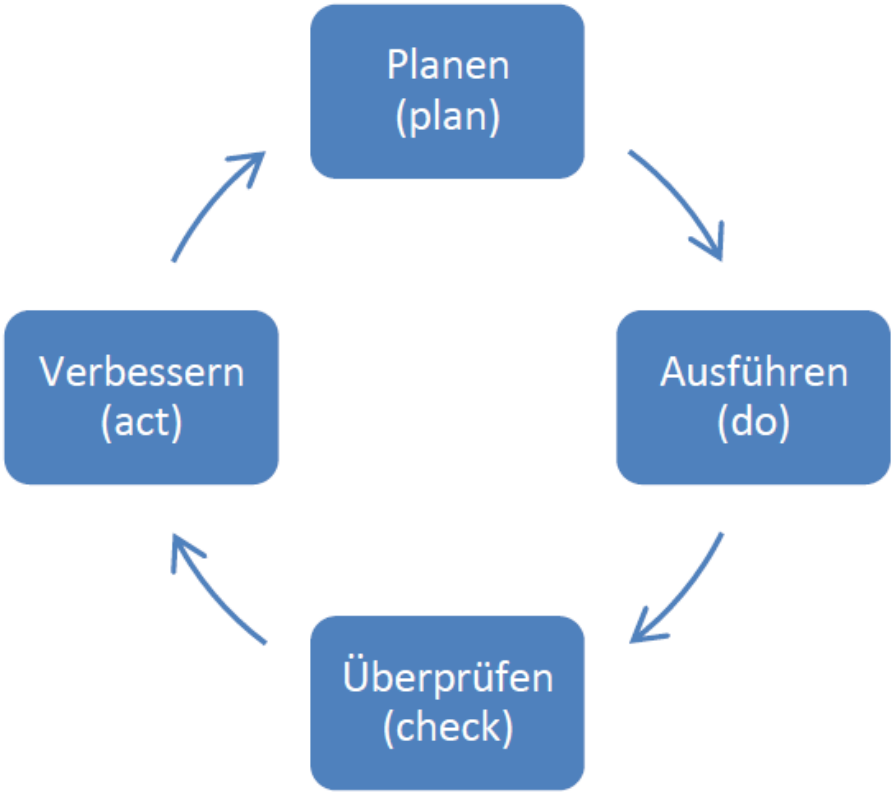

Abbildung 2.2: Plan-Do-Check-Act-Zyklus[ABEM10]

zum Thema Datenqualität) [ABEM10, S.28].

2.3 Ursachen für schlechte Datenqualität

Es gibt vielfältige Ursachen für schlechte Datenqualität. Diese können je nach Branche oder Unternehmensbereich unterschiedlich ausgeprägt sein. So sind beispielsweise bei einer Bank andere Ursachen ausgeprägt als bei einem Versandhaus oder Automobilhersteller. Im Folgenden werden die fünf wichtigsten Kriterien und deren Einfluss auf die Datenqualität erläutert.

Datenerfassung Die Datenerfassung ist oftmals die größte Fehlerquelle in Bezug auf Datenqualität. Hierzu zählt die falsche Benutzung von Eingabemasken, sowohl von internen Mitarbeitern als auch von Kunden, welche Informationen in falsche Eingabebereiche eintragen (z.B. Verwechslung von Vor- und Nachnamefeld). Natürlich sind auch Tippfehler, phonetisch ähnliche Laute (z.B. ai und ei bei Maier bzw. Meier) oder mangelhafte Nachfragen von Servicemitarbeitern (z.B. in einem Callcenter) eine potentielle Fehlerquelle. Viele dieser Fehler können aber erst durch mangelhaftes Design oder schlechte Pflichtfeldabsicherungen oder Plausibilitätsprüfungen in den Eingabemasken auftreten. Aber auch der Import von mangelhaften externen Daten wie beispielsweise eingekauften Adress- oder Kundendaten kann zu einer Verschlechterung der Datenqualität führen [ABEM10, S.35].

Prozesse Hierbei handelt es sich um die datenverarbeitenden Prozesse eines Unternehmens. Da eine solche Prozessstruktur durch nachträgliche Erweiterung oder Outsourcing schnell sehr komplex und inhomogen werden kann, kann es zu fehlerhafter Datenverarbeitung und somit zu schlechter Datenqualität kommen. Auch eventuelle Medienbrüche, also der Übergang von automatischer zu manueller Datenverarbeitung, innerhalb oder zwischen den Prozessen kann die schlechte Datenqualität verstärken [ABEM10, S.36]. Beispielsweise sollte ein Datumsfeld durch einen automatischen Prozess dahingehend geprüft werden, dass keine unrealistischen Daten eingegeben werden können. So macht das Datum 23.04.1850 für einen Kundendatensatz keinen Sinn und sollte somit durch den Prüfprozess erkannt und gemeldet werden. Tut er dies nicht oder nicht in ausreichendem Maße, so ist der Prozess fehlerhaft. Angenommen, er erkennt und meldet die Datumswerte an einen Sachbearbeiter (dies würde einen Medienbruch darstellen), welcher diese falschen Werte aus Unachtsamkeit bestätigt, so hat dies auch wieder schlechte Datenqualität zur Folge. Speziell bei Prüfprozessen für Datumswerten sollte aber darauf geachtet werden, für welchen Datenbereich diese eingesetzt werden. So würde das oben genannte Datum 23.04.1850 zwar bei einem Kunden fehlerhaft sein, jedoch könnte sich auch ein Betriebsgebäudedatensatz vorhanden sein, für den dieses Datum durchaus sinnvoll ist.

Architektur Mit (Daten-)Architektur werden die datenverarbeitenden Technologien (z.B. verschiedenen Anwendungssoftware) und der Datenfluss zwischen diesen Technologien bezeichnet. Viele dieser Programme benötigen eine eigene, spezielle Datendarstellung, wie Formatierung oder die Reihenfolge der Ein- und Ausgabeargumente. Deshalb ist oftmals eine Konvertierung der Daten notwendig, die zu Inkonsistenzen und somit zu schlechter Datenqualität führen kann. Systemerweiterungen, Systemumstellungen oder fehlerhafte Anwendungen können diesen Negativeffekt oftmals noch verstärken [ABEM10, S.36-37].

Angenommen, in einem Quellsystem existieren Kundendaten mit folgendem Schema:

Kundennummer	Vorname	Nachname	Geburtsdatum
110001	Max	Mustermann	15.01.1980
110002	Max	Mustermann	23.05.1955

Diese Datensätze werden extrahiert und in ein Zielsystem importiert, dass jedoch für einen Kundendatensatz lediglich die Felder Kundennummer, Vorname und Nachname vorsieht. Außerdem werden nur Kundennummern akzeptiert die kleiner als 100000 sind. Ist die eingetragene oder übergebene Kundennummer größer, wird der Standardwert 000000 in das Feld eingetragen. Dadurch entsteht folgende Darstellung der Datensätze:

Kundennummer	Vorname	Nachname
000000	Max	Mustermann
000000	Max	Mustermann

An dieser Stelle kann im Zielsystem nicht mehr unterschieden werden, ob es sich um zwei verschiedene Personen oder eine einzige Person handelt.

Definitionen/Datensynchronisation Damit ist die inhomogene Definition von Feldformaten desselben Feldes gemeint, die unter anderem dadurch entstehen kann, dass verschiedene Fachbereiche eines Unternehmens auch jeweils verschiedene Anwendungen benutzen, welche aber auf die gleichen Daten zugreifen. So könnte beispielsweise der Vertrieb einer Firma in der Vertriebsanwendung das Datum eines Auftrags mit dem Format Tag.Monat.Jahr abspeichern, aber das Rechnungswesen für die Abrechnung des Auftrags aus internen Gründen mit dem Datumsformat Jahr.Monat.Tag arbeiten. Dies führt zu Redundanzen, Inkonsistenzen und somit zu schlechter Datenqualität [ABEM10, S.37-38]. Geht man beispielsweise von einem Datumsformat der Form XX.XX.XX aus, so würde das Datum 02.05.12 mit dem Format Tag.Monat.Jahr den 2. Mai 2012 ergeben. Dasselbe Datum mit dem Format Jahr.Monat.Tag würde aber den 12. Mai 2002 adressieren.

Datenverfall Dieser Faktor tritt in manchen Bereichen ganz automatisch auf, da gewisse Daten nach einer gewissen Zeitspanne ihre Gültigkeit verlieren können. Darunter fallen vor allem Adress- und Telefondaten aber auch Bankdaten, Preislisten und viele weitere Bereiche sind von Datenverfall betroffen, was offensichtlich auch die Datenqualität einschränkt [ABEM10, S.38]. Wie in Kapitel 1 schon einmal erwähnt wurde, sind nach Expertenaussagen etwa 2% der Kundendatensätze innerhalb eines Monats veraltet (z.B. durch Umzug, Heirat oder Tod des Kunden) [Ech02].

2.4 Auswirkungen von schlechter Datenqualität

Als einführendes Beispiel soll ein fiktives Szenario aus dem CRM (Customer-Relationship-Management) eines Automobilherstellers dienen. Es wird von einem Kundenstamm von 16 Millionen Kunden ausgegangen, also 16 Millionen Datensätze, die aber auch fehlerhafte Daten beinhalten. Von dieser Datenmenge seien 10% der Adressdaten veraltet oder Dubletten (mehrere Datensätze die sich aber auf den gleichen Kunden beziehen) vorhanden. Der Automobilhersteller will nun eine große Werbekampagne starten und versendet an alle Kunden einen personalisierten Brief und einen Werbeprospekt mit Gesamtkosten von 2 Euro pro Sendung. Aufgrund der fehlerhaften 10% werden also 1.6 Millionen unnötige Sendungen verschickt, was zu vermeidbaren Kosten von 3.2 Millionen Euro führt. Da es sich zusätzlich um personalisierte Briefe handelt, kann auch noch ein zusätzlicher, nicht zu beziffernder, Imageschaden durch falsche Anreden oder falsch geschriebene Namen entstehen. Dies ist nur ein erfundenes Beispiel, jedoch kam es auch schon in realen Szenarien zu Kosten von bis zu 2 Milliarden US-Dollar, die durch schlechte Datenqualität entstanden sind [ABEM10, S.42]. Diese Szenarien zeigen, dass sich die Auswirkungen sehr gut in entstehenden Kosten messen lassen, weswegen nun die möglichen Kostenarten die aus schlechter Datenqualität entstehen erklärt werden. Diese Kostenarten unterteilen sich in Kosten, die durch schlechte Datenqualität verursacht werden und in Kosten zur Verbesserung oder Sicherstellung einer ausreichenden Datenqualität.

Durch schlechte Datenqualität verursachte Kosten

Direkte Kosten Diese entstehen unmittelbar aus schlechten oder fehlerhaften Daten. Wird beispielsweise erkannt, dass es für eine Bestellung bei einem Lieferanten zwei verschiedene

Bestellvolumen im System gibt, so muss verifiziert werden, welches die richtige Zahl ist, was direkte Kosten verursacht [ABEM10, S.44].

Indirekte Kosten Dies sind die Kosten, die indirekt durch schlechte Datenqualität entstanden sind (z.B. durch falsche Entscheidungen). Diese Kosten sind sehr vielfältig, weswegen nur ein kleiner Auszug stellvertretend angegeben wird.

- Verschwendung von Budgets (unnötiges/doppeltes Versenden von Werbematerial)
- Kosten durch Fehlentscheidungen (basierend auf falschen Daten)
- Imageverlust (z.B. durch falsche Anrede)
- Umsatzeinbußen (durch falsche Preiskalkulation basierend auf fehlerhaften Daten)

[ABEM10, S.44]

Kosten zur Verbesserung oder Sicherstellung einer ausreichenden Datenqualität

Präventionskosten Diese Kosten entstehen durch manuelle oder automatische Verhinderung von Fehleingaben, was natürlich zu Kosten für Mitarbeiterschulungen oder IT-Kosten führt [ABEM10, S.44].

Entdeckungskosten Entstehen meist durch spezielle Software, die gekauft oder erstellt werden muss, um automatisch fehlerhafte Daten zu erkennen und dann mit Hilfe von Mitarbeitern konfiguriert werden müssen [ABEM10, S.45].

Bereinigungskosten Diese Kosten entstehen durch die Implementierung von Updateroutinen, aber auch durch Personalaufwand, da eine vollautomatische Datenbereinigung oft nicht möglich ist [ABEM10, S.45].

2.5 Einführung in konkrete DQM-Lösungen

Um den Auswirkungen und Ursachen von schlechter Datenqualität entgegenzuwirken, gibt es eine Vielzahl von DQM-Produkten. In diesem Kapitel sollen jedoch nur diejenigen Lösungen kurz eingeführt werden, die auch im Rahmen dieser Arbeit behandelt werden, wobei die Adressvalidierung dann auch praktisch implementiert werden soll.

Adressvalidierung Die Adressvalidierung prüft, korrigiert und aktualisiert postalische Komponenten. Es soll also eine fehlerhafte Adresseingabe noch vor dem Schreiben des Adressdatensatzes in die Datenbank verhindert werden oder aber falsche Datensätze aufgespürt werden. Dies wird über Ähnlichkeitssuchen in Kombination mit Referenzdatensätzen realisiert. Die Ähnlichkeitssuche erkennt Standardabweichungen wie Groß- und Kleinschreibung (MÜNCHEN statt München), Schreibweisen nationaler Sonderzeichen (Muenchen statt München), Interpunktionsfehler (Ernst Reuter Str statt Ernst-Reuter-Str.) oder Abkürzungen (Ernst-Reuter-Straße statt Ernst-Reuter-Str.). Aber auch komplexere Ähnlichkeiten wie Zeichenfehler (Butehude satt Buxtehude), Wortdreher (Josef Franz Str. statt Franz Josef

Str.) oder aber phonetische Ähnlichkeiten (wie o und eau im Französischen). Diese Suche benötigt Referenzdaten, welche z.B. von der Deutschen Post gekauft werden können, in denen die korrekten Straßennamen, aber auch die korrekte Zuordnung von PLZ und Ort hinterlegt sind [Uni94b].

Dublettenprüfung Mit der Dublettenprüfung sollen redundante Datensätze erkannt und gelöscht werden. Auch dieses Modul arbeitet mit heuristischen Verfahren, die Ähnlichkeiten von Datensätzen erkennen sollen und ab einer festgelegten Übereinstimmungsquote als Dubletten erkannt werden. Hierbei werden alle Attributwerte eines Datensatzes berücksichtigt, die mit verschiedenen Gewichtungen in die Berechnung einfließen. So ist eine Übereinstimmung von Vorname, Nachname und Geburtsdatum für die eindeutige Identifikation einer Person aussagekräftiger als eine Übereinstimmung in Beruf, Wohnort und Gesamtumsatz. Abbildung 2.3 zeigt einen fiktiven Auszug aus einer Kundendatenbank. Die enthaltenen Datensätze werden mit Hilfe der Dublettenprüfung verglichen. In den Pfeilen ist der ermittelte Übereinstimmungsfaktor enthalten, welcher durch Ähnlichkeitssuche ermittelt wird. So hat der Faktor beim Attribut Name den Wert 1, da beide Namen exakt übereinstimmen. Das Attribut Vorname wurde jedoch nur mit dem Faktor 0,9 bewertet, da es sich beim zweiten Namen um eine Abkürzung handelt, welche jedoch von der Ähnlichkeitssuche mit hoher Wahrscheinlichkeit erkannt wurde. Sind alle Übereinstimmungsfaktoren ermittelt, werden diese noch mit den Attributgewichtungen verrechnet. Mit den Gewichtungen wird festgelegt, wie stark die einzelnen Attribute für die eindeutige Identifizierung einer Dublette verantwortlich sind. Je höher der Gewichtungsfaktor, desto wichtiger ist das Attribut für die Identifizierung. Die Summe aller Gewichtungsfaktoren müssen, mit dem Rechenverfahren, das in diesem Beispiel gewählt wurde, immer 1 ergeben. Eine Sonderstellung besitzen dabei die Schlüsselattribute, wie beispielsweise Personal_Nr. Diese werden nicht, oder nur mit sehr geringer Gewichtung in die Berechnung eingehen, da sie in der Regel eindeutig sein sollten. Die Berechnung der Übereinstimmungsquote erfolgt schließlich über die Summe aller Produkte aus Übereinstimmungsfaktor und Attributgewichtung (an dieser Stelle wären natürlich auch andere Formeln denkbar). Die errechnete Quote im Beispiel ist mit 0,98 relativ hoch, die Dublette also erkannt. Es wäre an dieser Stelle auch noch möglich die Quote in bestimmte Bereiche einzuteilen, die dann jeweils unterschiedliche Aktionen auslösen. So könnte beispielsweise eine Quote von 0.95 bis 1.00 eine automatische Löschung auslösen, von 0,70 bis 0,94 eine Benachrichtigung an einen Sachbearbeiter geschrieben werden und alle Werte unter 0,70 als Nicht-Dublette bewertet werden.

Embargo-Listen (SPL) Eine Embargo-Liste oder Sanctioned-Party-List (SPL) wird eine Liste von Personen oder Organisationen genannt, die aufgrund von gesetzlichen oder politischen Vorgaben keine Leistungen erhalten dürfen. Auf dieser Liste sind beispielsweise Terroristen oder politische Führungspersönlichkeiten aus Ländern, welche mit Sanktionen belegt sind. Da eine versehentliche Leistungserbringung an diese Personen nicht nur juristische Folgen, sondern auch einen Imageschaden zur Folge haben kann, ist ein Abgleich dieser Listen mit den Kundenstammdaten notwendig. Hierbei werden die Referenzdaten der Liste, auch wieder unter Einbeziehung von den unter Adressvalidierung genannten Ähnlichkeitskriterien, mit den Kundendaten verglichen und bei einer Übereinstimmung eine Sperre gesetzt oder eine Meldung an den zuständigen Mitarbeiter gesendet [Uni10]. Abbildung 2.4 stellt das Vorgehen einer Embargoprüfung noch einmal grafisch dar. In diesem Beispiel wird

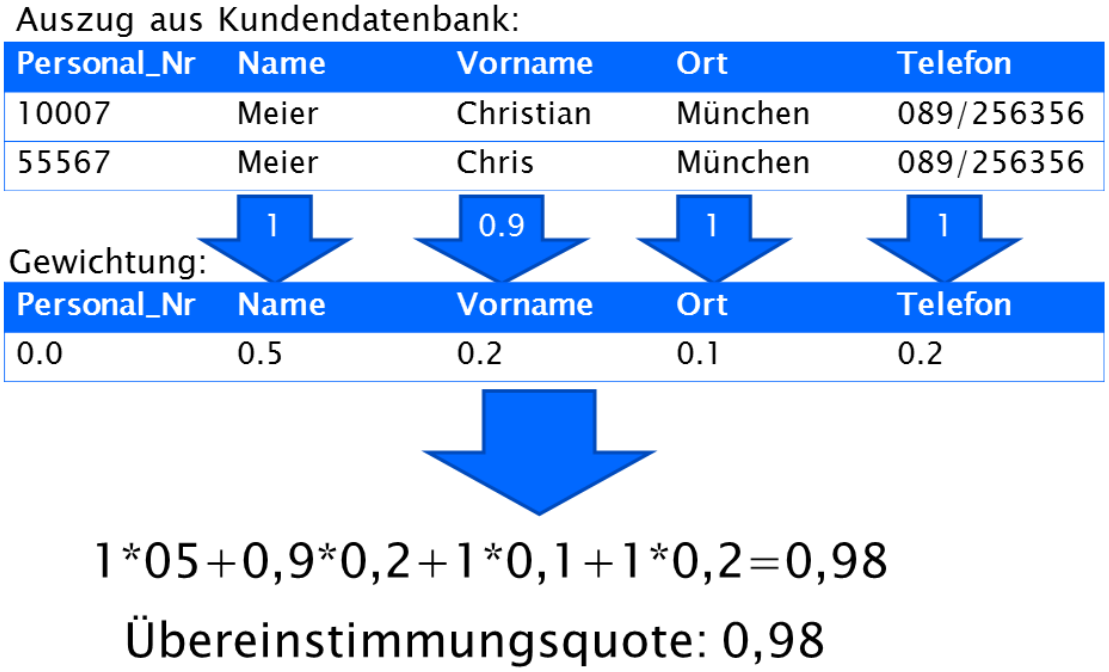

Abbildung 2.3: Beispiel: Dublettencheck

der Kunde mit dem Namen Max Embargo als sanktionierte Person erkannt. Die Übereinstimmung bei der Ähnlichkeitssuche wäre in diesem Fall bei 100 Prozent. Bei einer derart hohen Übereinstimmungsquote könnte auch relativ bedenkenlos eine automatische Sperrung des Datensatzes vorgenommen werden. Schwieriger wird die Entscheidung über die Vorgehensweise bei gefundenen Ergebnissen, deren Übereinstimmung kleiner als 100 Prozent ist. Sollte nämlich eine Person versehentlich gesperrt werden, die eigentlich nicht auf der Embargo-Liste steht, hat dies unter Umständen negative Auswirkungen auf das Image der Firma. Wird hingegen eine Person, die eigentlich auf der Embargo-Liste steht, beliefert, so sorgt dies wiederum zu schlechtem Image und/oder sogar zu juristischen Konsequenzen. Eine hohe Genauigkeit der Ähnlichkeitssuche und die richtige Auswahl der Reaktion auf die Ergebnisse dieser Suche sollten also Grundvoraussetzung für eine gute Embargoprüfung sein.

Bankdatenvalidierung Die Bankdatenvalidierung funktioniert, ähnlich wie die Adressvalidierung, durch eine Ähnlichkeitssuche und Plausibilitätsprüfung in Kombination mit Referenzdaten. Diese Datensätze enthalten nun aber Zuordnungen von BLZ und Bank, welche mit Hilfe der Ähnlichkeitssuche mit den aktuellen Bankdaten des Kunden oder Lieferanten verglichen werden. Wenn Unstimmigkeiten gefunden werden, sollen diese im besten Fall automatisch korrigiert, oder aber zumindest an einen zuständigen Mitarbeiter gemeldet werden. Abbildung 2.5 zeigt mögliche Eingaben von BLZ und Bankname(durch einen Kunden/Sachbearbeiter oder eine automatische Eingabe) und die dementsprechenden Korrekturen, die durch die Bankdatenvalidierung vorgenommen werden. Zusätzlich wird die Kontonummer anhand von etwa hundert verschiedenen Prüfziffernalgorithmen, allein in Deutschland, auf Plausibilität geprüft. Die hohe Anzahl an Berechnungsmethoden liegt daran, dass jedem Zah-

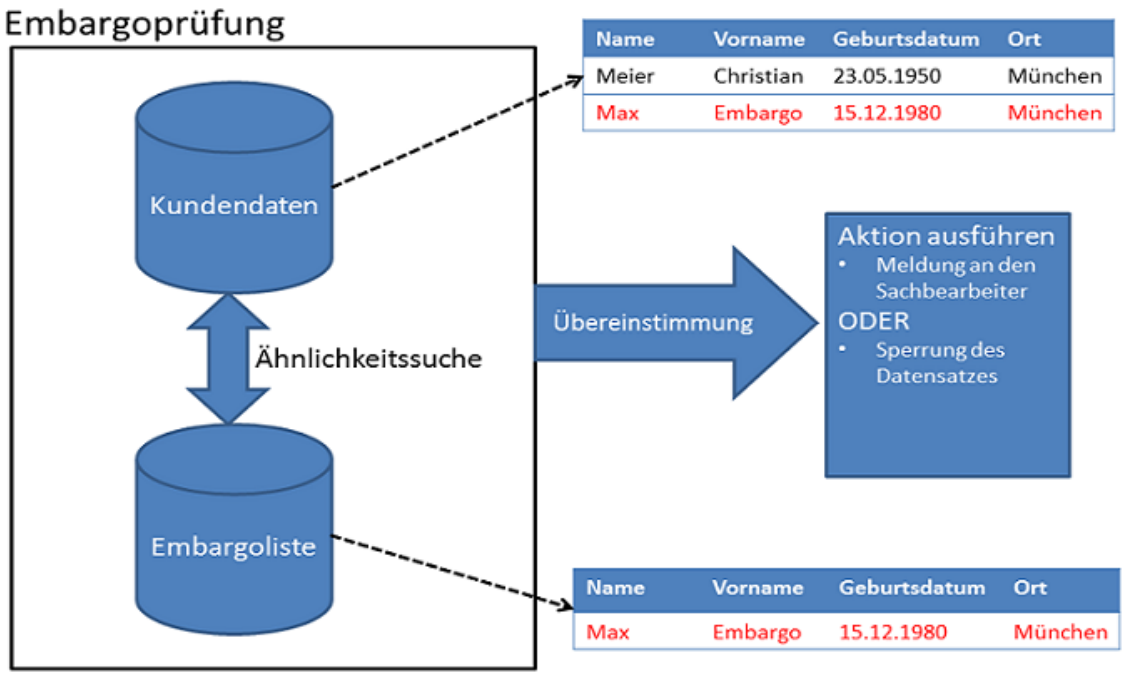

Abbildung 2.4: Beispiel: Embargoliste

lungsdienstleistern die Wahl der Prüfzifferberechnungsmethode freigestellt ist.[Bunb]. Als Beispiel wird in Abbildung 2.6 eine Prüfzifferberechnungsmethode skizziert. Die Methode wird auf eine 10-stellige Kontonummer angewendet, wobei die letzte Ziffer die Prüfziffer darstellt. Die Kontonummer wird im ersten Schritt von links nach rechts mit 9, fest definierten, Gewichtungsziffern verrechnet (siehe Abbildung 2.6, Berechnung). Die Summe aus dieser Berechnung wird dann Modulo 10 gerechnet, und dieses Ergebnis wiederum von 10 subtrahiert. Ist die errechnete Ziffer nun gleich der Prüfziffer, so ist die Kontonummer korrekt(aber natürlich nicht zwangsläufig existent). In ähnlichen, teilweise komplexeren Prüfungsverfahren, kann auch die IBAN, der SWIFT-Code oder die Kreditkartennummer validiert werden [Uni94a].

Eingabe	Ausgabe des Systems
66680013	66680013 Dresdner Bank
66650058 Sparkasse	66650085 Sparkasse
VB Pforzheim	66690000 Volksbank Pforzheim
eutsche Bnak 24 Stuttgart	60070024 Deutsche Bank 24

Abbildung 2.5: Beispiel: Bank[Uni94a]

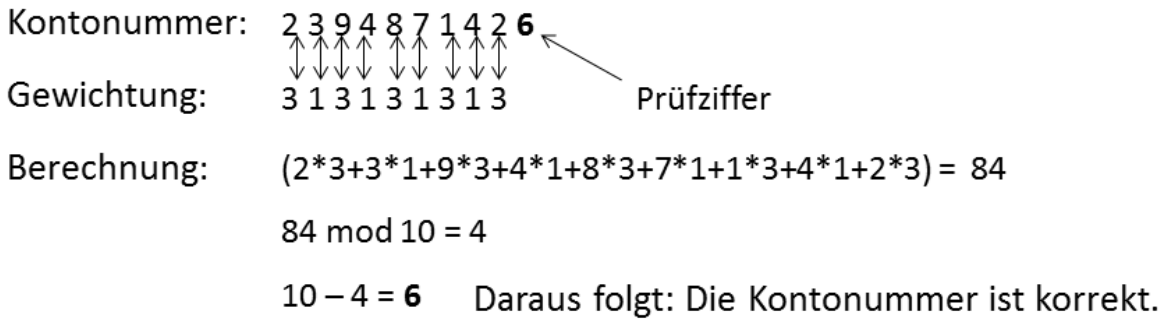

Abbildung 2.6: Prüfzifferberechnungsmethode C2 Variante 1[Buna]

3 Anforderungsanalyse eines DQMS

In diesem Kapitel sollen die Anforderungen an ein DQMS erörtert werden. Der Anforderungskatalog ist angelehnt an Anforderungsanalysen der Firma iC Consult GmbH. Die Anforderungen Unterteilen sich in verschiedene Kategorien, wobei zuerst die allgemeinen Anforderungen an ein DQMS vorgestellt werden und dann die konkreten funktionalen Anforderungen an eine Adressvalidierung. Obwohl an dieser Stelle einige Anforderungskategorien beschrieben werden, werden nur die allgemeinen funktionalen Anforderungen, ausgewählte Anforderungen aus den anderen Anforderungskategorien und die speziellen funktionalen Anforderungen für die Adressvalidierung zur späteren Evaluation herangezogen. Die Evaluation aller, hier vorgestellten Anforderungen, würde den Rahmen dieser Arbeit sprengen. Außerdem stellt der hier erarbeitete Anforderungskatalog lediglich einen kleinen Ausschnitt der möglichen Anforderungen dar, da andere Benutzer eines DQMS unter Umständen andere/mehr Anforderungen an dieses System stellen.

3.1 Allgemeine Anforderungen an ein DQMS

Allgemeine Funktionale Anforderungen

Anforderungsname	Details
Returncodes	Das System sollte über technische Returncodes verfügen, die den Status der Bearbeitung zurückgeben.
Sprachen	Das System soll, in seinen Modulen, die wichtigsten Sprachen unterstützen (u.a. Englisch, Französisch, Deutsch, Spanisch, Russisch, Chinesisch). Hiermit sind nicht die Sprachen des Interfaces gemeint, sondern die Sprachen die vom System verarbeitet werden können.
Standards	Das System sollte im besten Fall Unicode oder zumindest die wichtigsten Zeichensätze unterstützen.

Allgemeine Nicht-Funktionale Anforderungen

Anforderungsname	Details
Modi	Das System soll sowohl einen Batch-Modus für die Verarbeitung von großen Datenmengen, als auch einen Real-Time-Modus für Just-In-Time-Verarbeitungen anbieten. Der Real-Time-Modus soll vor allem in den Modulen Adressvalidierung, Dublettencheck, Embargocheck, Bankdatenvalidierung, Telefonnummervalidierung und Email-Validierung angeboten werden (Also für Validierungen von Daten deren Prüfung schon bei der Eingabe sinnvoll ist).
Performance	Das System sollte sowohl im Batch-Modus, vor allem aber auch im Real-Time-Modus eine gute Performance hinsichtlich des Datendurchsatzes erreichen.
Sicherheit	Das System soll, eine hohe Sicherheit der verarbeiteten Daten gewährleisten.

Allgemeine Technische Anforderungen

Anforderungsname	Details
Konnektivität	Das System soll über gängige Konnektoren zu Datenbanken oder ERP-Systemen verfügen.
Monitoring	Das System soll über ein integriertes Monitoring von Jobs bzw. des ganzen DQMS verfügen.
Hosting	Das System soll zentral gehostet werden können.
Hardwareanforderungen	Das System sollte, bei Standardbetrieb, möglichst wenig Hardwareleistung benötigen.
Softwareanforderungen	Das System sollte möglichst Betriebssystemunabhängig betrieben werden können und möglichst keine weitere (kostenpflichtige) Zusatzsoftware benötigen.
Multi-User-Betrieb	Das System sollte den (parallelen) Mehrbenutzerbetrieb unterstützen.

Allgemeine Migrationsanforderungen

Anforderungsname	Details
Datenkompatibilität	Das System soll alle Standarddatentypen und Datenformate verarbeiten können und dies auf einer ausreichend großen Attributanzahl.
Datentransformation	Das System soll über Datentransformationsfunktionen verfügen. Dies bezieht sich auf die Transformation von Datentypen und das damit zusammenhängende Mapping von Metadaten.
Migrationsmöglichkeiten	Die Migration von Daten soll durch das DQMS durchgeführt werden können.

3.2 Spezielle Anforderungen an die Adressvalidierung

An dieser Stelle sollen nun noch die speziellen Anforderungen an die Adressvalidierung aufgelistet werden. Auch diese Anforderungsliste ist lediglich ein Vorschlag und wird bei anderen Benutzern einer Adressvalidierung unter Umständen anders aussehen.

Anforderungsname	Details
Grundfunktionalität	Die Adressvalidierung soll richtige Datensätze als solche erkennen. Außerdem sollen falsche Datensätze erkannt und wenn möglich auch korrigiert werden. Insbesondere sollen Zahlen- oder Buchstabendreher, phonetische Fehler, Fehler in der Groß- und Kleinschreibung, Fehler bei Abkürzungen und logische Fehler (z.B. Straße ist im angegebenen Ort nicht vorhanden) erkannt werden.
PLZ-Auflösung	Die Adressvalidierung soll anhand einer PLZ den dazugehörigen Ort ermitteln können.
Ort-Auflösung	Die Adressvalidierung soll anhand eines Ortes die dazugehörige PLZ ermitteln können, oder bei einem Ort der nicht eindeutig ist, eine Liste von PLZ-Optionen zurückgeben.
HNR-Auflösung	Die Adressvalidierung soll erkennen, ob eine Hausnummer in einer Straße vorkommen kann. Sollte die Hausnummer zu groß oder zu klein sein, soll ein Intervall mit möglichen Hausnummern zurückgegeben werden.
Referenzdaten	Die Referenzdaten der Adressvalidierung sollten für mehrere Länder vorhanden sein. Darunter neben den wichtigsten europäischen Ländern auch die USA, Russland und China.
Returncodes	Die Adressvalidierung soll über Returncodes verfügen, die Aussagen über die Adressqualität der geprüften Adresse ermöglichen. Außerdem sollen Returncodes für die Qualität jedes einzelnen Attributs vorhanden sein. Auch Returncodes die die automatischen Änderungen/Korrekturen, die vorgenommen wurden, aufzeigen sollten vorhanden sein.

4 Betrachtung aktueller Tools im Bereich DQM

4.1 Aktuelle Tools im Bereich DQM

An dieser Stelle sollen einige der wichtigsten DQM-Tools vorgestellt werden. Da es eine große Bandbreite an DQM-Anbietern gibt, orientiert sich diese Arbeit an einer Studie der Gartner Inc. aus dem Jahr 2011. Das Ergebnis dieser Studie wird in einem sogenannten Magic Quadrant dargestellt, das die Unternehmen bzw. deren DQM-Tools in mehrere Bereiche gliedert (siehe Abb. 4.1). Nun soll kurz erörtert werden, nach welchen Kriterien die Gartner Inc. diese Bewertungsmatrix erstellt hat. Als erstes soll betrachtet werden, welche Eigenschaften ein DQM-Tool erfüllen muss, um überhaupt im Magic Quadrant aufgelistet zu werden.

4.1.1 Kriterien für die Aufnahme in den Gartner Magic Quadrant

In der folgenden Liste befinden sich die wichtigsten Kriterien für die Aufnahme in den Gartner Magic Quadrant [FB11]:

1. Es muss sich um ein eigenständiges Tool handeln. Es darf also nicht in andere Software eingebunden oder von anderer Software abhängig sein.
2. Das Tool muss mindestens über die folgenden Funktionalitäten verfügen: Standardisierung, Cleansing, Matching, Profiling und Parsing.
   a) Profiling - Es muss eine Analyse auf Attributebene (Minimumwert, Maximumwert, etc.) und eine Abhängigkeitsanalyse (tabellenübergreifende Analyse) enthalten. Die Ergebnisse der Analyse müssen tabellarisch oder grafisch dargestellt sein. Außerdem muss eine Trendanalyse umgesetzt sein.
   b) Parsing - Es müssen Funktionen zur Identifizierung und Extrahierung von Kontaktinformationen (Adresse, Email, etc.) vorhanden sein. Diese sollen für eine große Bandbreite an Datentypen funktionieren und vom Anwender konfigurierbar/erweiterbar sein.
   c) Matching - Es müssen konfigurierbare Matchingregeln/Matchingalgorithmen vorhanden sein, die für alle Datentypen und Attributanzahlen, die in einem Matchingszenario (z.B. Adressdatenanalyse, Bankdatenanalyse) erwartet werden, funktionsfähig sind.
   d) Standardisierung und Cleansing - Es müssen erweiterbare Regeln für den Umgang mit Syntax- und Semantiktransformation von Daten vorhanden sein.
3. Die Funktionalität des Tools muss für mehr als eine Sprache und mehrere Länder gewährleistet sein.
4. Das Tool muss bei mindestens 75 Kunden im Einsatz sein.

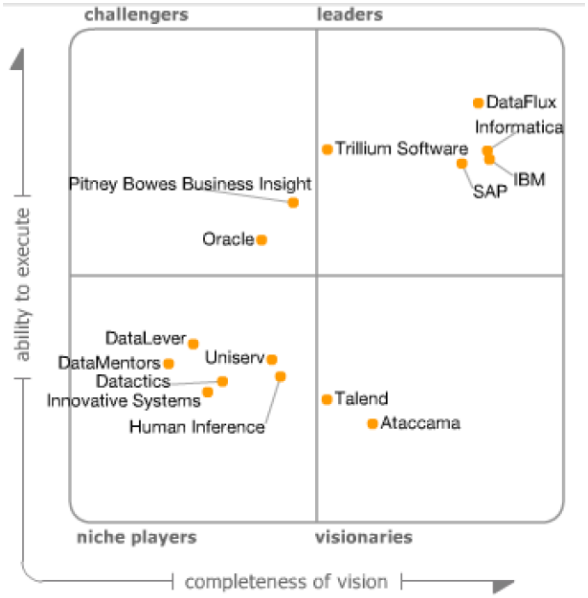

Abbildung 4.1: Magic Quadrant für Data Quality Tools[FB11]

5. Das Tool muss auf serverbasierten Architekturen und im Mehrbenutzerbetrieb lauffähig sein.

4.1.2 Bewertungskriterien im Gartner Magic Quadrant

Hat ein DQMS-Hersteller die Aufnahmekriterien erfüllt, wird er in die Matrix eingeordnet. Dies geschieht anhand der Evaluationskriterien „Ability to Execute“(Y-Achse des Quadranten) und „Completeness of Vision“ (X-Achse des Quadranten). Diese beiden Hauptkriterien sind jeweils unterteilt in verschiedene Unterkriterien, die wiederum eine Gewichtung erhalten. In den folgenden beiden Unterkapiteln sollen die Unterkriterien kurz dargestellt und erörtert werden [FB11].

Ability to Execute

Mit diesem Evaluationskriterium analysiert Gartner die Prozesse und Methoden, die es dem Anbieter ermöglichen, konkurrenzfähig und effizient zu sein und die einen positiven Effekt auf die Einnahmen, die Retention und die Reputation haben. Dies wird gemessen daran, wie erfolgreich die Anbieter ihre Vision kapitalisieren. Hierbei werden die Unterkriterien (siehe Abb. 4.2) betrachtet, die in der folgenden Auflistung erörtert werden sollen [FB11]:

- Product/Service - Bewertet wie der Anbieter die Bandbreite DQ-Funktionalität unterstützt, die vom Markt verlangt wird. Außerdem wird die allgemeine Verwendbarkeit des Tools und die Architektur in der die Funktionalitäten angeboten werden, bewertet.

- Overall Viability - Bewertet die finanzielle Stärke und die Stabilität und Stärke der Organisationsstruktur des Anbieters.

Evaluation Criteria	Weighting
Product/Service	high
Overall Viability (Business Unit, Financial, Strategy, Organization)	high
Sales Execution/Pricing	standard
Market Responsiveness and Track Record	standard
Marketing Execution	high
Customer Experience	high
Operations	no rating

Abbildung 4.2: Unterkriterien von Ability to Execute[FB11]

- Sales Execution/Pricing - Bewertet die Effektivität des Preismodells bezüglich aktueller Trends in Nachfrage und Ausgabeverhalten der Kunden. Außerdem wird die Effizienz der direkten und indirekten Vertriebswege bewertet.
- Market Responsiveness and Track Record - Bewertet die Fähigkeit des Anbieters, sich über einen längeren Zeitraum, erfolgreich an neue Nachfragen des Marktes im Bereich DQM anzupassen.
- Marketing Execution - Bewertet die Effektivität der Marketingmaßnahmen des Anbieters anhand der Marktanteile, die der Anbieter dadurch dazugewonnen hat.
- Customer Experience - Bewertet die Zufriedenheit der Kunden mit dem Anbieter in Bezug auf das Produkt, den Service, das Preis-Leistungs-Verhältnis und die allgemeine Beziehung zwischen Anbieter und Kunden.

Completeness of Vision

Mit diesem Evaluationskriterium analysiert Gartner die Anbieter in Bezug auf deren (überzeugender und logischer) Einschätzungen über den aktuellen und zukünftigen Marktwachstum, Innovationen, Kundenbedürfnisse und Konkurrenten. Die Anbieter werden anhand ihrer Fähigkeit bewertet, diese Einflusskräfte auf den Markt zu ihrem Vorteil zu nutzen. Hierbei werden die Unterkriterien (siehe Abb. 4.3) betrachtet, die in der folgenden Auflistung erörtert werden sollen [FB11]:

- Market Understanding - Bewertet den Grad, um den ein Anbieter den Markt in eine neue Richtung führt (durch Technologie, Produkte, Services) und die Fähigkeit sich an signifikante Marktänderungen anzupassen.
- Marketing Strategy - Bewerten den Grad, um den die Marketingmaßnahmen des Anbieters sich an aktuelle Trends anpasst.

Evaluation Criteria	Weighting
Market Understanding	high
Marketing Strategy	standard
Sales Strategy	standard
Offering (Product) Strategy	high
Business Model	low
Vertical/Industry Strategy	low
Innovation	high
Geographic Strategy	standard

Abbildung 4.3: Unterkriterien von Completeness of Vision[FB11]

- Sales Strategy - Bewertet die Anpassung des Verkaufsmodels des Anbieters, an die Kundenwünsche.
- Offering Strategy - Bewertet das Produkt des Anbieters daraufhin, ob es an neue Markttrends angepasst ist oder Lücken im Markt füllt.
- Business Model - Bewertet das Vorgehen des Anbieters bezüglich der Strategieumsetzung auf dem DQM-Markt. Beispielsweise die Vielfältigkeit der Vertriebskanäle, der Preisgestaltung und der Kooperation mit anderen Unternehmen.
- Vertical/Industry Strategy - Bewertet die vertikale Integration des Anbieters in den Markt. Also die Verknüpfung mit ähnlichen Märkten z.B. Hardwarezulieferer oder Integration in andere Softwareprodukte.
- Innovation - Bewertet die Innovationsfähigkeit des Anbieters, neue Technologien oder Produkte zu entwickeln, die das Potential haben den Markt zu verändern, weil sie einen Mehrwert für den Kunden bieten.
- Geographic Strategy - Bewertet die Ausbreitung des Anbieters auf Märkte außerhalb des Heimatmarktes.

4.1.3 Überblick über die Marktführer im Bereich der DQM-Tools

An dieser Stelle soll ein kurzer Überblick über die Marktführer im Bereich der DQM-Tools gegeben werden. Hierbei handelt es sich um die Firmen, die im rechten oberen Quadranten des Gartner Magic Quadrant aufgeführt sind. Die folgende Auflistung gibt eine kurze

Übersicht über die Stärken der Produkte dieser Firmen (die Reihenfolge der Auflistung ist beliebig gewählt)[FB11].

DataFlux

- Die Firma DataFlux bietet mit der „DataFlux Data Management Platform“ ein Tool, das Profiling, Matching, Cleansing und Monitoring vereint.
- Die einfach zu nutzende „Ein-Produkt-Architektur“ ist ein Hauptgrund, für den Erfolg des Tools auf dem Markt.
- Außerdem werden Data Warehousing, Master Data Management (MDM) und Datenmigration unterstützt.
- Das Tool kann mit mehr verschiedenen Datentypen umgehen, als die meisten Konkurrenzprodukte.
- Die weitere Expansion von DataFlux in Richtung MDM entspricht den aktuellen Trends des Marktes.
- Ein starker finanzieller Rückhalt, globale Marktpräsenz und ein starker Anstieg der Marktanteile, sind auf das Mutterunternehmen SAS zurückzuführen.

Informatica

- Die Firma Informatica ist ein etablierter Anbieter von DQ-Lösungen, mit starkem weltweitem Wachstum.
- Die DQ-Tools von Informatica bieten eine starke Profiling-Funktionalität (Data Explorer) aber auch Parsing-, Standardization- und Matchingverfahren (Data Quality) werden unterstützt.
- Durch Partnerschaften mit AddressDoctor und Siperian ist Informatica auf den konvergierenden Märkten von Datenintegration, Datenqualität und Master Data Management, gut positioniert.
- Informatica bietet neben Produkten für kundenbezogene Daten, auch Produkte für andere Domänen wie Produktdaten, Finanzdaten oder Gesundheitsdaten an.

IBM

- Die Produktpalette von IBM umfasst Profiling, Analysis, Parsing, Standardization und Matching, kombiniert mit einer starken Datenbanken-Abhängigkeits-Analyse.
- Die Tools können auf eine große Bandbreite von Datendomänen und Use-Cases (Datenmigration, MDM, BI) angewandt werden.
- Es ist möglich die DQ-Tools sowohl im Stand-Alone-Betrieb, als auch in Kombination mit anderen Produkten zu benutzen.
- IBM besitzt eine starke Position im DQ-Markt.

- Die DQ-Tools zeichnen sich durch eine hohe Skalierbarkeit und Konfigurierbarkeit, als auch durch leichte Handhabung und gute Performance aus.

SAP

- SAP bietet eine große Bandbreite in DQ-Funktionen und Profiling.
- Der Fokus liegt zwar auf dem Bereich der kundenbezogenen Daten, jedoch werden die Tools auch immer mehr in den Bereichen der Produkt- und Materialdaten verwendet.
- SAP besitzt einen erheblichen Marktanteil im Bereich der BI-Plattformen, aber auch einen großen Marktanteil bei den DQ-Tools. Dies erlaubt SAP die DQ-Tools in Kombination mit den BI-Plattformen zu vertreiben.
- Die gute Integrierbarkeit der DQ-Tools in andere SAP BusinessObjects-Tools ist für viele Kunden ein Hauptgrund, sich für SAP zu entscheiden.
- Eine weitere erhebliche Stärke ist die Performance der DQ-Tools.

Trillium

- Trillium bietet eine große Bandbreite von DQ-Tools (Profiling, Standardization, Matching, Parsing, Datenanreicherung) aus.
- Die Vertriebsstrategie beinhaltet einen Direktvertrieb der Produkte, aber auch den Vertrieb durch Partnerunternehmen.
- Trillium hat einen ähnlich hohen Wiedererkennungswert der Marke, wie die anderen Marktführer.
- Trillium hat, im Vergleich zu den Konkurrenten, einen besseren Real-Time-Betrieb der Tools in den Bereichen des MDM und der operativen Anwendungen.
- Eine gute Konfigurierbarkeit/Skalierbarkeit gehört auch zu den Stärken der Trillium-DQ-Tools.

Da die Gartner-Studie zwar einen sehr guten Überblick über den Markt der DQM-Tools bietet, aber der Anforderungskatalog von Gartner nicht in allen Punkten mit den Anforderungen, die in dieser Arbeit benötigt werden, übereinstimmt, sind die Tools die in dieser Arbeit verwendet werden, nicht unter den Marktführern zu finden. Dies liegt daran, dass der Fokus dieser Arbeit primär auf der Validierung personenbezogener Daten liegt. Dies ist im Anforderungsprofil von Gartner jedoch nur ein kleiner Teilbereich der technischen Anforderungen. Außerdem spielen bei der Bewertung von Gartner auch wirtschaftliche Faktoren wie Kundenanzahl, Umsatz oder Vertriebswege eine große Rolle. Diese Faktoren wurden bei der Entscheidung für Uniserv nicht berücksichtigt. In den folgenden beiden Unterkapiteln sollen nun Tolerant und Uniserv kurz vorgestellt werden, die im praktischen Teil dieser Arbeit zum Einsatz kommen.

4.2 Vorstellung von Uniserv

Uniserv erfüllt die Anforderungen von Gartner und ist im Magic Quadrant als „Niche Player“ eingestuft. Die Einstufung wird aufgrund der folgenden Stärken und Schwächen von Uniserv vorgenommen (diese Daten beziehen sich auf den Stand von Anfang 2011)[FB11]:
Stärken

- Uniserv bietet vor allem Produkte für personenbezogene Daten an, was zu einer Spezialisierung in diesem Bereich führt.
- Uniserv ist der größte europäische Anbieter von reinen DQ-Lösungen, mit mehr als 40 Jahren Erfahrung auf diesem Gebiet.
- Uniserv macht 80% des Umsatzes mit Kunden in Deutschland und Frankreich, jedoch ist eine gute Entwicklung in anderen europäischen Ländern und den U.S. zu bemerken.
- Uniserv war einer der ersten Anbieter, der seine Leistungen auch als SaaS (Software as a Service) vertrieben hat.
- Uniserv Tools werden von den meisten Kunden sowohl im Real-Time-Modus als auch im Batch-Modus betrieben, was die Stärke der Tools in beiden Bereichen unterstreicht.
- Uniserv hat seine Produktpalette stark erweitert und bietet neben Matching und Validation auch konkurrenzfähige Produkte in den Bereichen Data Quality Monitoring (Überwachung der Datenqualität) und Data Profiling (Analyse der Datenqualität).
- Durch Partnerschaften mit der Firma Taled, ist Uniserv auch in der Lage, Produkte für Datenintegration, Migration und Synchronisation anzubieten.
- Die Produkte von Uniserv lassen sich in alle relevanten CRM-Systeme (z.B. SAP, Siebel, Microsoft Dynamics) integrieren.
- Uniserv-Tools sind voll Unicode-fähig.
- Uniserv bietet, im Vergleich zu den meisten Konkurrenten, eine sehr große Kompatibilität mit einer großen Bandbreite von Betriebssystemen an (u.a. Windows, Unix/Linux, IBM i, Siemens BS2000).

Schwächen

- Uniservs starker Fokus auf personenbezogene Daten führt dazu, dass Uniserv andere Domänen, wie Finanzdatenanalyse oder Produktdatenanalyse, noch kleinere Schwächen hat und dadurch ein Wettbewerbsnachteil entsteht.
- Uniserv ist zwar eine etablierte Marke im Bereich Matching, Merging, Cleansing und Adress- und Bankdatverifikation, jedoch gibt es noch Schwachstellen in den Bereichen Monitoring.
- Außerdem werden, laut Kundenumfragen, die Bereiche Profiling, Visualisierung und Monitoring nur wenig genutzt.
- Uniservs Direktvertriebsstrategie und das Fehlen von internationalen Kooperationen, stellen einen Wettbewerbsnachteil dar.

Zusammenfassend kann bemerkt werden, dass Uniserv viele Stärken im Bereich der personenbezogenen Daten besitzt, jedoch die Gesamtproduktpalette, in Bereichen abseits der reinen DQ-Angebote kleiner ist als die der Markführern. Jedoch passt Uniserv sehr gut in das Anforderungsprofil dieser Arbeit, da der Fokus der Evaluation auf personenbezogenen Daten liegt. Da dieser Bereich von Uniserv komplett abgedeckt wird und Uniserv als etablierte Marke in diesem Feld gilt, wurden die Uniserv-Tools für diese Arbeit ausgewählt.

4.3 Vorstellung von Tolerant

Das Tool von Tolerant, das in dieser Arbeit mit Uniserv verglichen werden soll, ist nicht im Gartner Magic Quadrant aufgeführt, da der Funktionsumfang nicht den Kriterien für die Aufnahme genügt. Trotzdem soll an dieser Stelle kurz das Tool von Tolerant, unabhängig vom Magic Quadrant, erörtert werden. Die Produktpalette von Tolerant umfasst vier Bereiche: Tolerant Post, Tolerant Match, Tolerant Name, Tolerant Sanction.
Tolerant Post ist das Modul für die Adressanalyse, also die Verifikation und Korrektur von Adressdaten. Tolerant Match ist ein Dublettencheck auf Adressdaten. Tolerant Name ist eine Namensanalyse, also eine Analyse von Anrede, Vor- und Nachname. Bei Tolerant Sanction handelt es sich um einen Embargocheck.
Alle Module von Tolerant sind lauffähig auf den Betriebssysteme Windows, Linux und diverse UNIX-Derivaten. Außerdem werden Batchverarbeitungen für alle Module angeboten und Multi-Threading-Architekturen unterstützt [Tol].
Tolerant wird in dieser Arbeit verwendet, da es bei einem Kunden der Firma iC Consult GmbH bereits im Einsatz ist und aus diesem Grund Interesse an der Evaluation eines Vergleichsprodukts besteht.

5 Beschreibung der Lösungen der Firma Uniserv

5.1 Das Data Quality Service Hub (DQSH)

Das DQSH (siehe Abb.6.5) ist das Herz der Uniserv-DQ-Lösung. Dieses Hub vereint alle wichtigen Funktionalitäten unter einer Oberfläche. In diesem Kapitel sollen die wichtigsten Funktionen kurz vorgestellt werden, wobei etwas genauer auf die Bereiche eingegangen wird, die für den praktischen Teil dieser Arbeit relevant sind. Bevor aber genauer auf die einzelnen Teilbereiche eingegangen wird, soll nun vorweg der Aufbau der Oberfläche des DQSH erörtert werden.
Der mittige Bereich in Abbildung 6.5 stellt den Hauptbereich des DQSH dar. Hier werden mit Hilfe von verschiedenen Objekten die DQ-Verarbeitungen zusammengestellt. Die grafische Darstellung symbolisiert den Verlauf des Datenstroms durch das System. Im Beispiel, das eine Adressdatenvalidierung darstellt, aus Abbildung 6.5 startet der Datenstrom beim Element „tRTInput_1“. Dies ist ein Objekt das Inputdaten zur Verfügung stellt. Dann werden die Daten über ein das Mapping-Objekt „tMap_1“ auf den Input des Objekts „tUniservRT-Post_1“ gemappt. Dieses Objekt validiert die Adressen, die es als Input erhält, bewertet diese, und je nach Ergebnis werden die verarbeiteten Daten an eines der drei Output-Objekte weitergegeben (wieder über Mapping-Objekte). Das Output-Objekt „tRTOutput_6“ wird angesprochen, wenn die Adressdaten nicht verarbeitet werden konnten, oder zu fehlerhaft für eine Korrektur waren. Das Objekt „tRTOutput_7“ wird angesprochen, wenn die Analyse mehrere mögliche Zieladressen ermittelt hat. Das Objekt „tRTOutput_8“ erhält die korrigierten oder richtigen Adressen. Dieses Beispiel soll lediglich einen kurzen Überblick über den Verlauf eines DQ-Jobs geben. Eine genauere Darstellung eines Datenverlaufs und der einzelnen Objekte wird in den folgenden Unterkapiteln erklärt werden.
Auf der rechten Seite in Abbildung 6.5 ist eine Palette von Objekten dargestellt (eingeteilt in verschiedene Kategorien), die für eine DQ-Verarbeitung verwendet werden können. Einige dieser Objekte wurden im Beispiel von oben bereits erwähnt, jedoch stellen diese nur einen Bruchteil der möglichen Objekte dar, die zum Einsatz kommen können. Wegen der hohen Anzahl der Objekte, werden in dieser Arbeit nur diejenigen erklärt, die auch tatsächlich verwendet wurden.
Auf der linken Seite in Abbildung 6.5 ist das Repository. Hier sind unter anderem die Verschiedenen Uniserv-Produkte wie der Data Quality Explorer, die Data Quality Realtime Suite und die Data Integration Suite aufgelistet. In diesem Produktmodul lassen sich verschieden Jobs anlegen und verwalten. So ist der Beispieljob aus Abbildung 6.5 unter dem Reiter „Data Quality Real-Time Suite“ zu finden, da es sich um einen Real-Time-Job handelt. Die drei oben erwähnten Produkt-Module, sowie die Produkte Data Quality Monitor und Data Quality Batch Suite werden noch einmal in den folgenden Unterkapiteln vorgestellt, um einen Gesamtüberblick über das DQ-Angebot von Uniserv zu vermitteln.

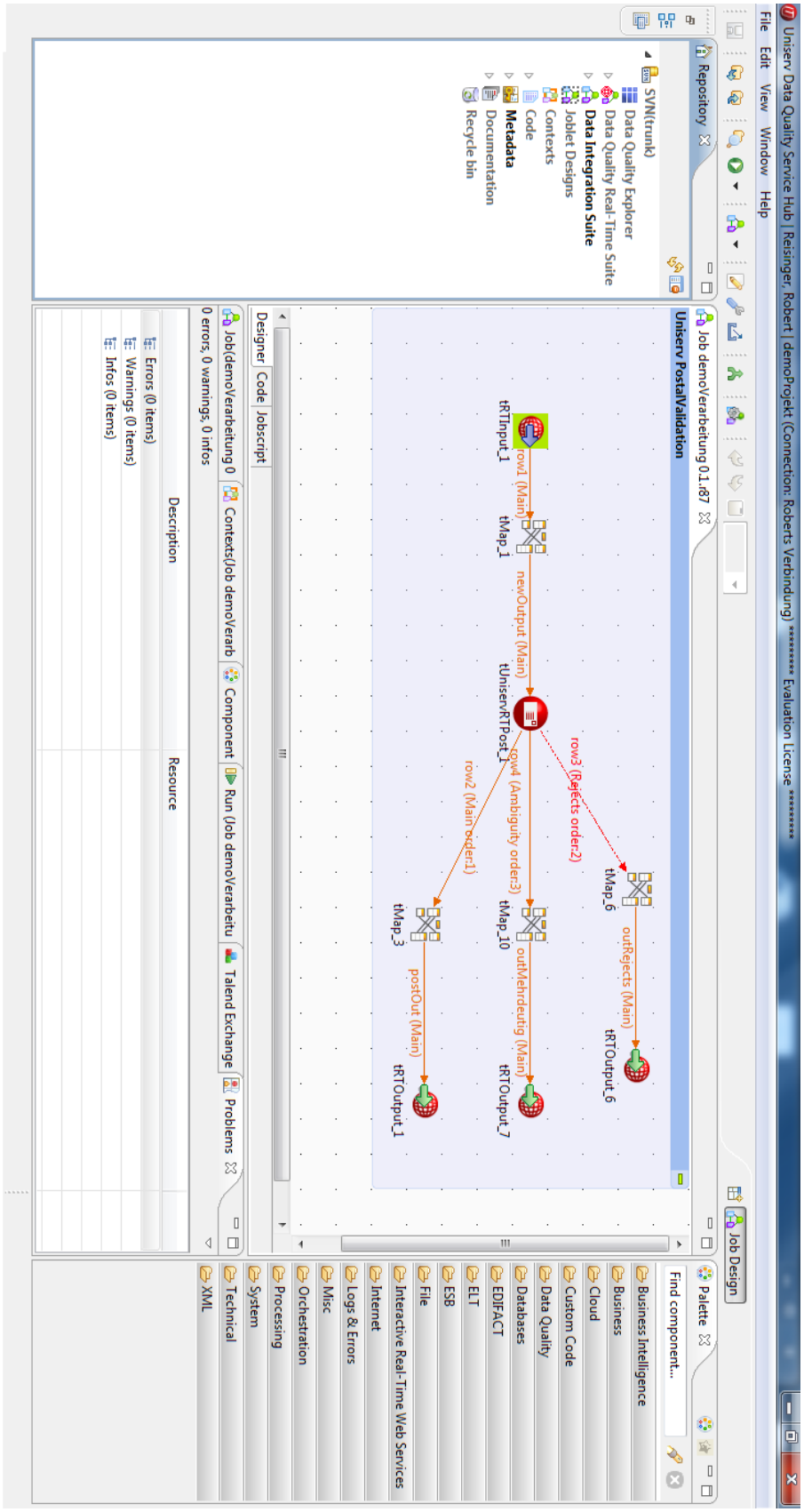

Abbildung 5.1: Uniserv Data Quality Service Hub

5.1.1 Data Quality Explorer

Der Data Quality Explorer ist ein Profiling- und Analysetool für Daten. Das Ziel des Explorers ist, den Ist-Zustand verschiedenster Daten, z.B. Finanz-, Auftrags- oder Statistikdaten, zu ermitteln, was unter anderem im Vorfeld von Migrationsprojekten oder größeren Datenqualitätsprojekten genutzt wird. Bestandteil des Profilings sind die Analyse auf Datenfeldebene (Mustererkennung, Maximum- und Minimumwerte des Feldes ermitteln, Eindeutigkeitsprüfung usw.), eine Abhängigkeitsprüfung (Bestimmung ob ein Attribut eindeutig durch andere Attribute bestimmt werden kann), eine Eindeutigkeitsprüfung (Bestimmung ob es eindeutige Kombinationen von Attributen gibt) und Join-Prüfungen (Bestimmung ob es Tabellenübergreifende Überschneidungen von Attributen gibt)[Unic].

5.1.2 Data Quality Monitor

Der Data Quality Monitor dient zur Überwachung der Datenqualität anhand vorher definierter Regeln. Werden beispielsweise Schwellwerte oder Regeln für bestimmte Datenfelder oder in den Qualitätsparametern überschritten, reagiert der Data Quality Monitor, z.B. indem er eine E-Mail an den zuständigen Sachbearbeiter sendet. Die Monitoringabläufe können manuell oder automatisch zu einem bestimmten Zeitpunkt durchgeführt werden. Letzteres ist sinnvoll um den laufenden Betrieb nicht zu beeinträchtigen, indem man die Monitoringabläufe nachts startet. Neben der Benachrichtigungsfunktion gehört auch die Erstellung von Statistiken zur Datenqualität über einen zeitlichen Verlauf, zu den Aufgaben des Monitors[Unid].

5.1.3 Data Quality Batch Suite

Die Data Quality Batch Suite ist für die Überprüfung und, nach Möglichkeit, vollautomatischen Bereinigung von Massendaten, sowohl im Bereich der kundenbezogenen Daten, als auch in anderen Datendomänen. Beispielsweise werden fehlerhafte Daten bereinigt, die durch den Data Quality Explorer gefunden oder dem DQ Monitor gemeldet wurden. Bereinigt werden unter anderem Adressformate, Adressdaten, Dubletten, Bankdaten oder Telefonnummern. Für den nicht-kundenbezogenen Teil stellt Uniserv ein DQ-Scripting zur Ergänzung von Funktionalitäten zur Verfügung. Dieses kann, wenn nötig, natürlich aber auch für kundenbezogene Daten verwendet werden. Das Tool kann neben den DQ-Bereinigungen auch Datenkonsolidierungen durchführen. Beispielsweise wenn Kundendaten aus verschiedenen Quellen konsolidiert werden sollen[Unib].

5.1.4 Data Integration Suite

Die Data Integration Suite ist für den Zugriff auf Datenbanken, Dateien oder Applikationen zuständig. Hierzu werden eine Vielzahl von Konnektoren angeboten, die das direkte Schreiben und Lesen dieser Datenquellen ermöglichen. Neben dem Lesen und Schreiben kann die Data Integration Suite auch automatisch die Metadaten der Datenquellen ermitteln und ermöglicht so ein komfortables und leichtes Mapping der Daten zu anderen Komponenten. Auch eine Transformation der Datenstrukturen oder Datenformate kann, wenn notwendig, durch die Data Integration Suite vorgenommen werden[Unia].

5.1.5 Data Quality Real-Time Suite

Die Data Quality Real-Time Suite bietet unter anderem Funktionen zur Überprüfung von Adressformatierung, Adressdaten, Dubletten, Bankdaten oder Telefondaten. Dies scheint auf den ersten Blick, Ähnlichkeiten mit der Data Quality Batch Suite zu haben. Jedoch werden die Funktionen der Batch Suite auf große, schon vorhandene Datenmengen angewandt, wohingegen die Real-Time Suite eine sofortige Kontrolle der Daten bei der Eingabe durchführt und die Daten gegebenenfalls automatisch korrigiert oder Korrekturen vorschlägt. Die Daten werden also schon auf Validität überprüft, bevor sie überhaupt in eine Datenbank oder Zieldatei geschrieben werden. Uniserv bezeichnet diese auch als „Data Quality Firewall"[Unie]. Die Data Quality Real-Time Suite, bzw. der Teilbereich der Adressvalidierung, ist die Komponente, die in dieser Arbeit genauer evaluiert werden soll. Aus diesem Grund wird in Kapitel „Beschreibung eines Jobs in der Data Quality Real-Time Suite" ein Beispiel-Job der Real-Time Suite vorgestellt.

5.2 Beschreibung der Komponenten/Objekte zur Erstellung eines DQ-Jobs

Bevor die Jobs zur Adressvalidierung und zum Dublettencheck genauer beschrieben werden, sollen die Objekte vorgestellt werden, aus denen diese Jobs generiert werden. Einige dieser Objekte sind bereits im einleitenden Beispiel dieses Kapitels erwähnt worden, sollen nun aber noch einmal genauer betrachtet werden.

Das Objekt „tRTInput" Hierbei handelt es sich um ein Objekt, in dem das Schema der Input-Daten, für einen Real-Time-Job, festgelegt wird. Beispielsweise könnte ein Schema mit den Input-Feldern Straße, PLZ, Ort, Land definiert werden. Hierzu ist anzumerken, dass der Standarddatentyp für die Input-Felder der Typ String ist. Dies kann jedoch optional abgeändert werden. Eine Besonderheit des tRTInput-Elements ist, dass es bereits im DQSH mit Testdaten belegt werden kann. Dies ermöglicht es, einen Job noch im DQSH zu testen, ohne ihn vorher als Webservice zur Verfügung zu stellen, was die Entwicklung des Jobs stark vereinfacht.

Das Objekt „tRTOutput" Hierbei handelt es sich um ein Objekt, durch das die Output-Daten eines Jobs zurückgegeben werden. Auch dieses Objekt verfügt über ein Schema, in dem diejenigen Felder definiert werden, die Ausgegeben/Zurückgegeben werden sollen. Dies könnte beispielsweise das Schema OUT_ZIP, OUT_CITY, OUT_RES_ZIP und OUT_RES_CITY sein. Es würden also die PLZ, der Ort und die Art der Veränderungen, die auf die beiden Attribute während der Verarbeitung vorgenommen wurden, zurückgegeben werden. Das Objekt ist jedoch nur in der Lage, die Output-Werte zurückzugeben. Es ist nicht dafür gedacht die Werte beispielsweise in eine Datei oder Datenbank zu schreiben.

Das Objekt „tFileInputDelimited" Dieses Objekt ist dem „tRTInput" sehr ähnlich. Der Unterschied besteht darin, dass nicht nur ein Inputdatensatz verarbeitet werden kann, sondern eine Reihe von Inputdatensätzen, die in Form einer CSV-Datei (oder einer anderen Datei, die durch einfache Trennzeichen zwischen Attributen unterscheidet) vorliegen. Das

Objekt liest die Input-Datei Zeile für Zeile und speist die einzelnen Datensätze nacheinander in den Job ein.

Das Objekt „tFileOutputDelimited“ Hierbei handelt es sich um ein Objekt, durch das die Output-Daten eines Jobs zurückgegeben werden. Im Unterschied zu „tRTOutput“ werden die Rückgabedaten Zeile für Zeile in eine Datei geschrieben, die durch einfache Trennzeichen zwischen den Attributen unterscheidet. Einer der bekanntesten Dateitypen hierfür ist die CSV-Datei, die auch in dieser Arbeit als Ausgabedatei verwendet wird.

Das Objekt „tUniservRTPost“ Dieses Objekt beinhaltet die eigentliche Adressvalidierung. Hier werden die Adressdaten, die das Objekt als Input erhält, verarbeitet und ausgewertet. Unter anderem wird die Adresse auf Validität geprüft, fehlerhafte Adressbestandteile werden korrigiert, es werden die Werte für die Rückgabewerte oder Outputparameter errechnet, z.B. welche Qualität der gesamte Datensatz besitzt oder welche Attribute verändert wurden. Nach dieser Verarbeitung werden die Rückgabevariablen befüllt, die das Objekt dann zurückgibt. Hierbei gibt es drei mögliche Wege für die Rückgabe. Der erste Rückgabezweig ist der Main-Zweig. In diesen Zweig gibt das Objekt Rückgabewerte aus, wenn die Bearbeitung erfolgreich verlaufen ist und die Adresse als zustellbar bewertet wird. Der zweite Rückgabeweg ist der Ambiguous-Zweig. Dieser wird angesprochen, wenn die Analyse ergeben hat, dass die Eingabedaten zu mehreren möglichen Ausgaben führen können. Dies wäre z.B. der Fall, wenn die PLZ nicht angegeben wird und als Stadtname „Neustadt“ verwendet wird. Da es mehrere Neustadt in Deutschland gibt, wird in den Ambigious-Zweig eine Liste all dieser Städte zurückgegeben. Der dritte Pfad ist der Reject-Zweig. Dieser wird angesprochen, wenn anhand der Eingabedaten keine Adresse identifiziert werden konnte oder die Adresse als „nicht zustellbar“ gilt.

Das Objekt „tUniservRTMailBulk“ Dieses Objekt ist für die Erstellung der Referenzdaten zuständig, die für den Dublettencheck notwendig sind. Der Dublettencheck im Real-Time-Modus benötigt eine Adresseingabe und vergleicht diese dann mit den Referenzadressen, die schon im System vorhanden sind. Wird hierbei eine Adresse gefunden, die ein Duplikat der eingegebenen Adresse darstellt, kann dies dem Benutzer sofort mitgeteilt werden. Die dafür benötigten Referenzdaten werden durch „tUniservRTMailBulk“ generiert. Hierzu benötigt das Objekt als Input, eine Datenquelle (CSV-Datei, XML-Datei, Datenbank, Excel-Tabelle), in der die Referenzdaten hinterlegt sind. Aus diesen Daten generiert das Objekt im Anschluss die Referenzdatendatei. Dies ist notwendig, da die direkte Suche von Dubletten in den Datenquellen zu großen Performanceverlusten führen würde.

Das Objekt „tUniservRTMailSearch“ Dieses Objekt beinhaltet die Real-Time- Dublettenprüfung. Als Input wird dem Objekt ein Adressdatensatz übergeben, den es dann über spezielle Suchalgorithmen mit den Referenzdaten vergleicht. Als Output besitzt das Objekt zwei verschiedene Pfade. Der erste Pfad ist der Main-Pfad, der angesprochen wird, sobald der Eingabedatensatz als Dublette identifiziert ist. Der zweite Pfad ist der Reject-Pfad, der die Outputdaten erhält wenn keine Dublette gefunden wurde. Diese Darstellung kann zu Verwirrungen führen, da der an sich positive Fall, dass keine Dublette entdeckt wurde, in den Reject-Pfad gelangt.

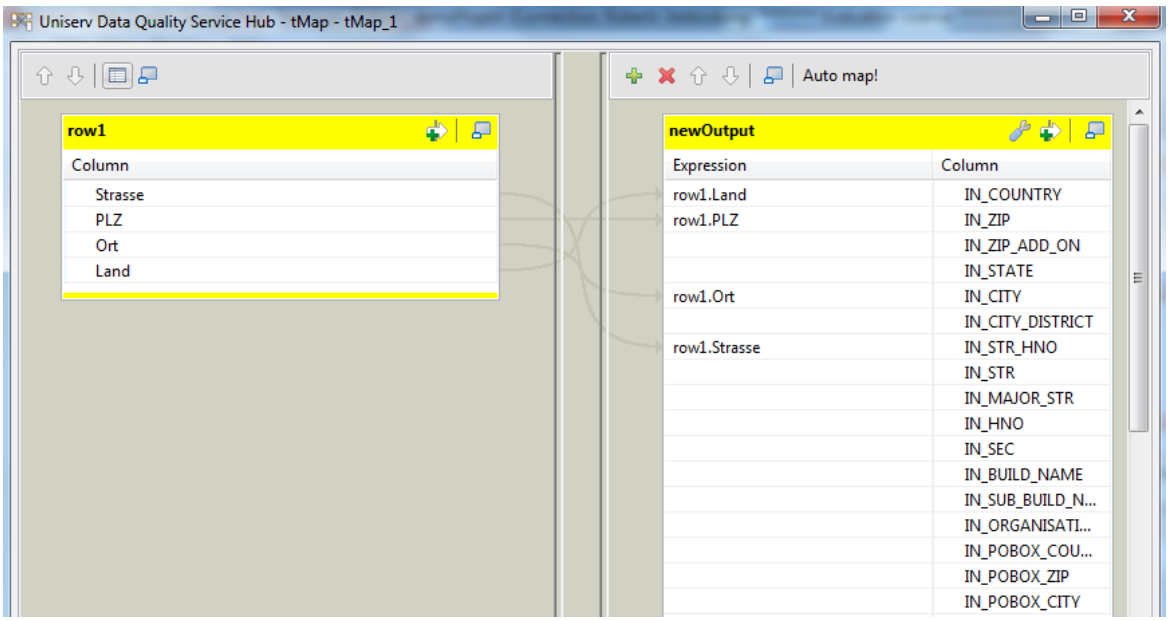

Abbildung 5.2: Mapping in Uniserv

Das Objekt „tMap“ Das Objekt „tMap“ ist die Verknüpfung zwischen Eingabeobjekten, Ausgabeobjekten und Verarbeitungsobjekten. Es dient dazu, das Outputschema eines oder mehrerer Objekte auf das Inputschema eines oder mehrerer Objekte zu mappen. Ein Map-Objekt hat somit n Eingänge und n Ausgänge, wobei n>0 ist. Beispielsweise besitzt das Objekt „tUniservRTPost“ eine ganze Reihe an Input-Variablen, durch das Mapping können jedoch ganz gezielt die gewünschten Variablen befüllt werden, die das „tRTInput“-Objekt zur Verfügung stellt. In Abbildung 5.2 ist das Mapping eines Input-Objekts auf einen „tUniservRTPost“-Objekt zu sehen. Man sieht, dass das „tUniservRTPost“-Objekt noch weitere Felder besitzt, die jedoch nicht befüllt werden.

5.3 Beschreibung eines Jobs in der Data Quality Realt-Time Suite

Als erstes wird im DQSH der Job erstellt, wie in Abbildung 5.3. Als Inputdaten in „tRTInput_1“ werden Straße, PLZ, Ort und Land festgelegt. Diese werden auf die entsprechenden Inputfelder von „tUniservRTPost_1“ gemappt wie in Abbildung 5.2. Wie man sieht, gibt es bei „tUniservRTPost_1“ nun nur einen Ausgang, den Main-Zweig. Die anderen beiden Ausgänge werden in diesem Beispiel nicht behandelt, was zur Folge hat, dass eine mehrdeutige oder nicht korrigierbare Eingabeadresse keine Ausgabe erzeugt. Die Outputfelder von „tUniservRTPost_1“, die ausgegeben werden sollen, werden schließlich noch auf das Objekt „tRTOutput_1“ gemappt wie in Abbildung 5.4. Aus Darstellbarkeitsgründen kann nur die rechte Seite des Mappings angezeigt werden. Die Variablen in der Tabellenspalte „Expression“ sind die Output-Variablen von „tUniservRTPost_1“ und die Variablen in der Tabellenspalte „Column“ sind die Input-Variablen von „tRTOutput_1“. Die Input-Variablen „tRTOutput_1“ sind zwar mit dem Präfix „OUT“ gekennzeichnet, da diese ausgegeben werden sollen, jedoch handelt es sich in Bezug auf das Mapping um Input-Variablen. Nun soll der Job noch getestet werden, bevor dieser als Webservice zur Verfügung gestellt wird. Also

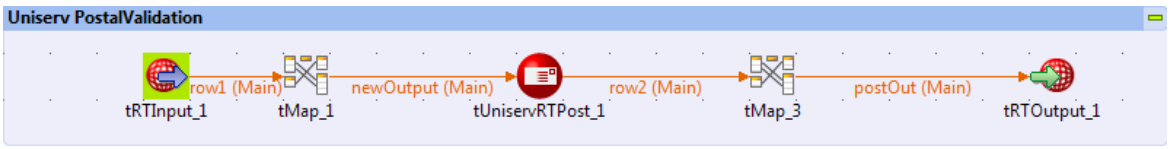

Abbildung 5.3: Beispieljob für Adressvaliderung in Uniserv

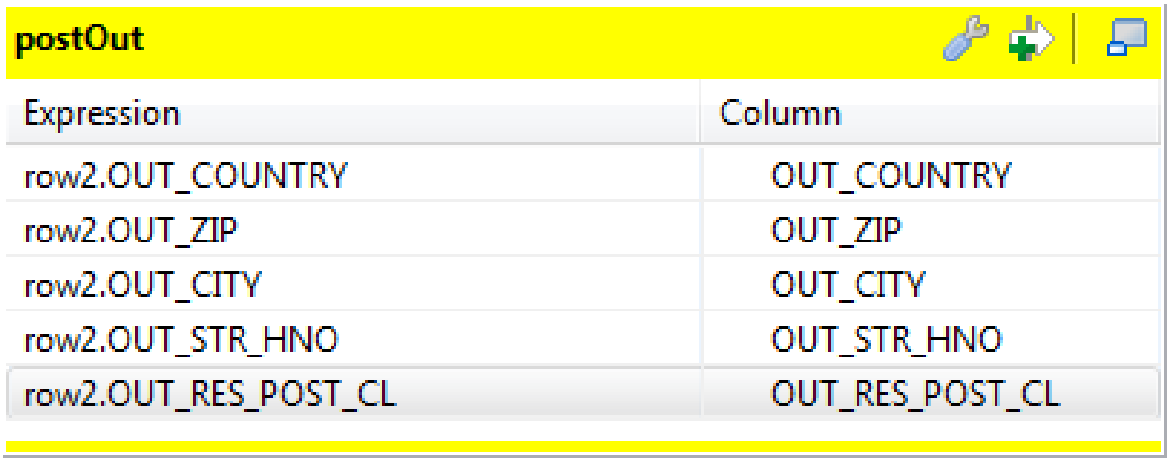

Abbildung 5.4: Mapping von „tUniservRTPost_1“ und „tRTOutput_1“

wird das Objekt „tRTInput_1“ mit folgenden Testdaten befüllt:

- Straße: Lantspergerstraße 143
- PLZ: 80336
- Ort: München
- Land: DE

Nun wird erwartet, das als Output die korrigierte Adresse ausgegeben wird und zusätzlich noch der Inhalt der Variable OUT_RES_POST_CL, die die Adresse hinsichtlich ihrer Zustellbarkeit mit Werten von 1 bis 5 bewertet. Nach dem Durchlauf des Tests erhält man dann die korrekte Ausgabe in Abbildung 5.5, wobei die Adresse mit 3, also als „noch zustellbar“, bewertet wurde. Nachdem man sich, auf diesem Weg, von der gewünschten Funktionalität überzeugt hat, kann der Job als Webservice zur Verfügung gestellt werden. Dies funktioniert, indem man einen Rechtsklick auf den Job ausführt und dann die Option „Deploy as Webservice“ wählt. Hier kann man nun noch angeben, unter welcher URL der Webservice verfügbar sein soll.

#1. tRTOutput_1

key	value
OUT_COUNTRY	DE
OUT_ZIP	80339
OUT_CITY	München
OUT_STR_HNO	Landsberger Str. 143
OUT_RES_POST_CL	3

Abbildung 5.5: Rückgabewerte einer Adressvalidierung in Uniserv

6 Realisierung der Adressvalidierung der Firma Uniserv

6.1 Installation der Software

6.1.1 Installationsvoraussetzungen

Die Installation der Uniserv-Software erfordert im Wesentlichen zwei Komponenten. Die erste ist der Lizenzserver, der die Lizenzdateien für die Hauptinstallation verwaltet, aber auch für die Lizenzverwaltung im laufenden Betrieb der Software zuständig ist. Die zweite Komponente ist der eigentliche Server, auf dem die jeweiligen Module für Bankdatenvalidierung, Adressvalidierung, Dublettencheck usw. installiert werden. Außerdem werden auf diesem Server die notwendigen Referenzdaten hinterlegt. Somit ist der Server das Herz der Software, an den die Such- oder Vergleichsanfragen gesendet, dort dann verarbeitet und schließlich zurückgegeben werden. Neben der normalen Installation von Lizenzserver und Server gibt es auch noch die Option, einen sogenannten High Available License Service (HAL) aufzubauen (siehe Abb. 6.1). Dieser ermöglicht es, mehrere Lizenzserver parallel zu betreiben, um eine höhere Ausfallsicherheit zu gewährleisten[Unif]. Zusätzlich zu den beiden Servern können auch noch Clients installiert werden, die unter anderem für die Steuerung des Batchbetriebs der Software zum Einsatz kommen. Eine Übersicht über den Aufbau einer Standardsystemarchitektur enthält Abbildung 6.2. Ein großer Vorteil der Uniserv-Software liegt darin, dass alle erwähnten Komponenten (Lizenzserver, Server, Client) auf vielen verschiedenen Systemplattformen lauffähig sind (u.a. Windows, Linux/Unix, BS200). Als Betriebssystem des Testsystems, das in dieser Arbeit verwendet wird, wurde auf Anraten der Firma Uniserv Windows 7 benutzt. Im folgenden Kapitel wird der Installationsvorgang dieses Testsystems erörtert.

6.1.2 Installation der Testumgebung

Als Hardware für die Testumgebung dient ein Rechner mit einer Taktfrequenz von 2.27GHz (Dual-Core) und 4GB Hauptspeicher. Als Betriebssystem wurde die 64-Bit-Version von Windows 7 Home Premium gewählt. Im Folgenden wird die Installation der Uniserv-Software Schritt für Schritt in der Reihenfolge dargestellt, in der sie vorgenommen werden müssen. Vor dem Start der eigentlichen Installation sollte sichergestellt werden, dass Oracle/Sun Java der Version 6 auf dem System installiert ist, da diese als Voraussetzung für die Uniserv-Software benötigt wird. Eine Besonderheit bei der Installation des Testsystems ist, dass Lizenzserver und Server nicht auf verschiedenen Rechnern installiert werden, sondern beide auf einem System betrieben werden.

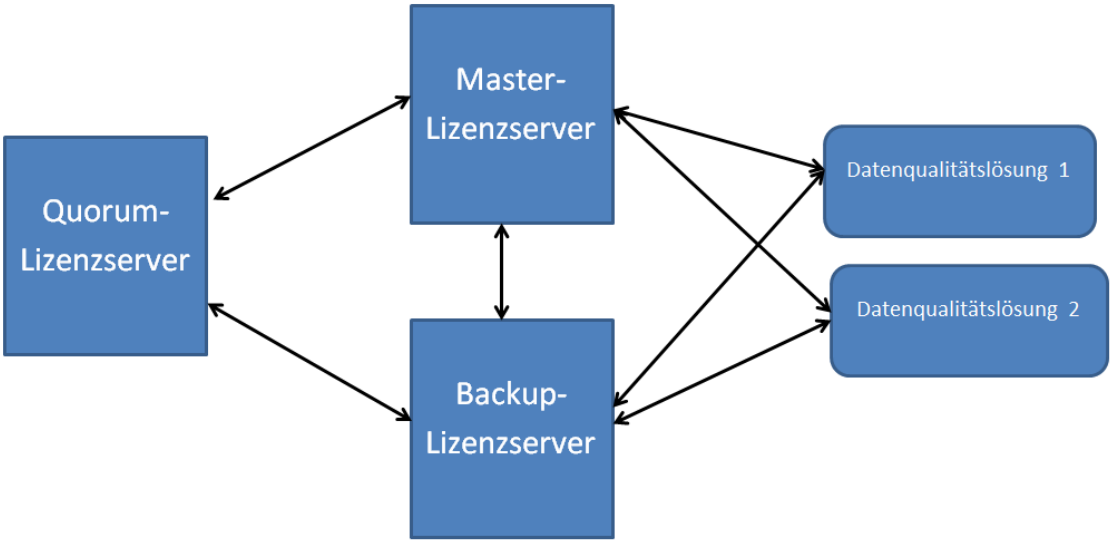

Abbildung 6.1: HAL-Verbund von Uniserv[Unif]

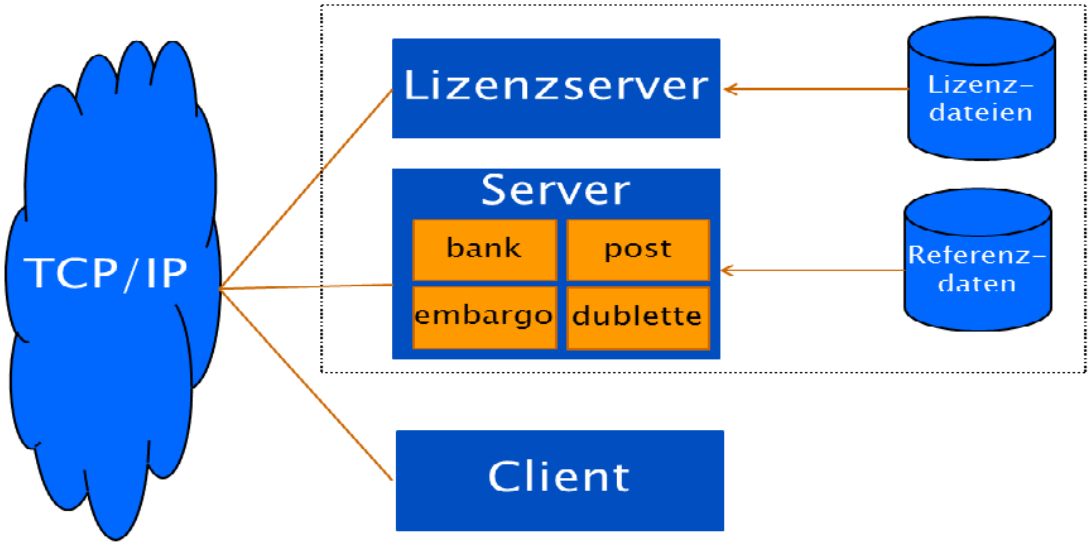

Abbildung 6.2: Architektur der Software von Uniserv[Uni94b]

Installation des Lizenzservers

Als erste Komponente wird der Lizenzserver installiert, da ohne diesen keine Lizenzvalidierung für die Serverinstallation möglich ist. Bei der Lizenzserverinstallation werden folgende Schritte durchgeführt:

1. Start der Setup-Datei bzw. des Installations-Wizards für die Lizenzserverinstallation.
2. Angeben des Zielorts, an den der Lizenzserver installiert werden soll. Im Fall des Testsystems wird der Lizenzserver unter D:\Uniserv\installiert.
3. Angeben, ob eine Normalinstallation oder eine HAL-Installation durchgeführt werden soll. Hier wird die Normalinstallation gewählt, da eine HAL-Installation bei einer Testumgebung mit nur einem Rechner keinen Sinn ergeben würde.
4. Angeben eines Passwortes für den Remote-Zugriff auf den Lizenzserver.
5. Angeben des TCP-Ports für den lokalen Lizenzserver. Hier wird der Default-Port 6800 benutzt, da dieser noch nicht durch einen anderen Dienst auf dem System benutzt wird.
6. Fertigstellen der Installation [Uni11a].
7. Auf die Lizenzierungswebseite von Uniserv gehen und dort den erhaltenen Aktivierungscode der Form XXXXX-XXXXX-XXXXX-XXXXX eingeben. Zusätzlich dazu muss noch die Datei „lic-serv-config.xml“ hochgeladen werden, in der Hard- und Softwareinformationen zum System stehen, auf dem der Lizenzserver installiert ist. Diese Datei wird bei der Installation des Lizenzservers automatisch generiert und dient zur Bindung der Lizenzdateien an möglichst eindeutige und fälschungssichere Eigenschaften des Systems wie Computername oder IP des Systems. Sind diese Konfigurationsdatei und der Aktivierungscode eingegeben, kann man die Lizenzdateien herunterladen, für die man eine Freischaltung besitzt.
8. Kopieren der Lizenzdateien in den Ordner:
   D:\Uniserv\UniservLicService\licenseServer\licenses\.
9. Neustart des Lizenzservers durchführen. Also den Service „Uniserv License Service“ neustarten.

[Uni11b]

Installation des Servers Nach der Installation des Lizenzservers kann mit der Installation des Servers fortgefahren werden, was in den folgenden Schritten durchgeführt wird:

1. Überprüfen, ob der Lizenzserver bzw. dessen Service läuft. Sollte dies nicht der Fall sein muss der Lizenzserver gestartet werden.
2. Start der Setup-Datei bzw. des Installations-Wizards für die Serverinstallation.
3. Angeben des Zielorts an den der Server installiert werden soll. Wichtig ist hierbei, dass der Pfad nicht der selbe Pfad ist, unter dem auch schon der Lizenzserver installiert wurde. Deshalb wurde im Testsystem der Pfad D:\Uniserv64\gewählt.

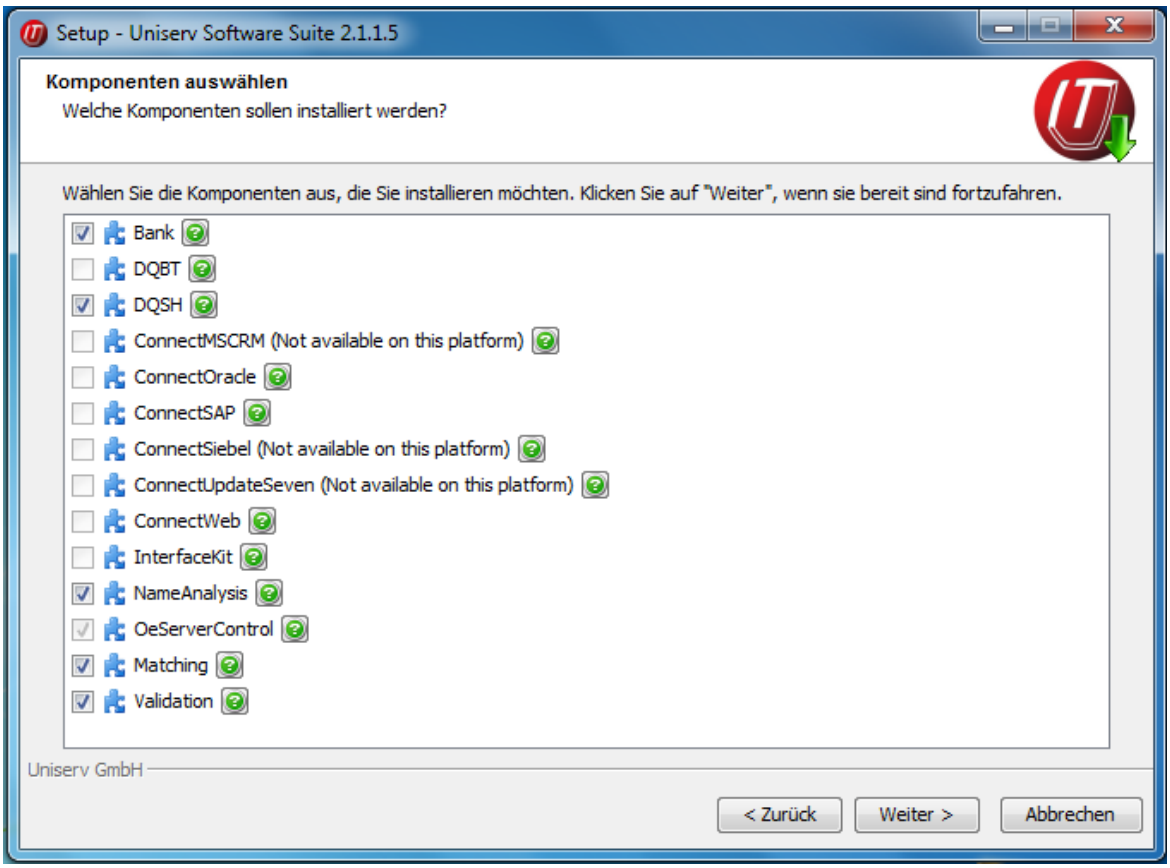

Abbildung 6.3: Installationsmaske mit Komponentenwahl von Uniserv

4. Die Setup-Routine stellt nun fest, dass eine lokale Instanz des Lizenzservers vorhanden ist und fragt, ob man diesen Lizenzserver benutzen möchte. Dies wird im Falle des Testsystems mit ja beantwortet. Verneint man diese Frage, müsste man die IP-Adresse und den Port des Lizenzservers eingeben, was bei einer lokalen Instanz des Lizenzservers jedoch überflüssig ist.

5. Es wird anhand der Lizenzen ermittelt, für welche Module man eine Installationsberechtigung besitzt. Die Komponenten, für die dies der Fall ist, werden in einer Liste dargestellt. Komponenten die aufgrund von mangelnden Lizenzen nicht installiert werden dürfen, werden automatisch ausgegraut und können nicht gewählt werden. Unter den restlichen Komponenten kann man nun auswählen, welche installiert werden sollen (siehe Abb. 6.3). Bei Bank handelt es sich offensichtlich um die Bankvalidierung, die Adressvalidierung ist im Modul Validation enthalten und der Dublettencheck und die Embargo-Liste werden mit dem Modul Matching installiert. Außerdem wird mit dem Modul DQSH (Data Quality Service Hub) noch ein Tool zur Erstellung und Verwaltung von Testcases zur Verfügung gestellt [Uni11b].

Installation des Apache HTTP Server und des Apache Tomcat 6.0 Servlet/JSP Container Im Anschluss an die Serverinstallation wird automatisch auch ein Apache HTTP Server und ein Apache Tomcat 6.0 installiert. Die Installation für das Testsystem wurde mit den Standardeinstellungen durchgeführt. Die Installation dieser beiden Komponenten ist notwendig, um die Uniserv-Software via Web-Browser konfigurieren zu können. Eine genauere Übersicht über das Zusammenspiel von Webserver, Tomcat und Uniserv-Server bzw. auch über die Konfiguration gibt es in Kapitel 6.2.

Installation der Referenzdaten Nach dem Abschluss der Serverinstallation müssen nun noch die Referenzdaten für die Adressvalidierung installiert werden. Hierfür werden die mitgelieferten Adressdaten via einer Setup-Routine automatisch in das installierte Uniserv-System integriert. Hierzu sind auch keine weiteren Einstellungen notwendig, da das Setup den Zielort automatisch erkennt und dort die Referenzdaten installiert.

6.2 Konfiguration der Software

6.2.1 Technische Umsetzung der Konfiguration

Da die ein großer Teil der Konfiguration über eine Webanwendung gemacht wird, soll an dieser Stelle kurz das Zusammenspiel zwischen dem Browser, dem Webserver und dem Tomcat erklärt werden. Es wird davon ausgegangen, dass die URL der Webanwendung für die Konfiguration schon im Browser geöffnet ist. Grafisch veranschaulicht wird das Vorgehen in Abbildung 6.4.

1. In der Anwendung können Einstellungen für das Uniserv-System angegeben werden. Sobald diese abgesendet werden, werden sie via HTTP/HTTPS an den Apache HTTP Server gesendet.

2. Der HTTP Server sendet die Anfragedaten via TCP (Transmission Control Protocol) und AJP (Apache Java Protocol) an den Apache Tomcat.

3. Der Tomcat bearbeitet mit der dementsprechenden Java-Applikation die Anfrage auf dem Server, z.B. das Schreiben der empfangenen Konfigurationsdaten in eine Datei/-Datenbank. Zusätzlich wird ein Output (Rückgabedaten) generiert.

4. Der Output wird via TCP und AJP an den HTTP Server gesendet.

5. Der HTTP Server gibt den Output an den Browser weiter.

6.2.2 Konfiguration der Testumgebung

Der Kern der Uniserv-Testumgebung ist das DQSH(Data Quality Service Hub). Um dieses Hub nach der Installation benutzen zu können, sind noch einige Einstellungen notwendig, die über eine Webanwendung konfiguriert werden. In der folgenden Auflistung werden die notwendigen Schritte kurz erörtert:

1. Aufruf der Konfigurationsanwendung über:
„http://localhost:8080/com.uniserv.dqsh.administrator/“.

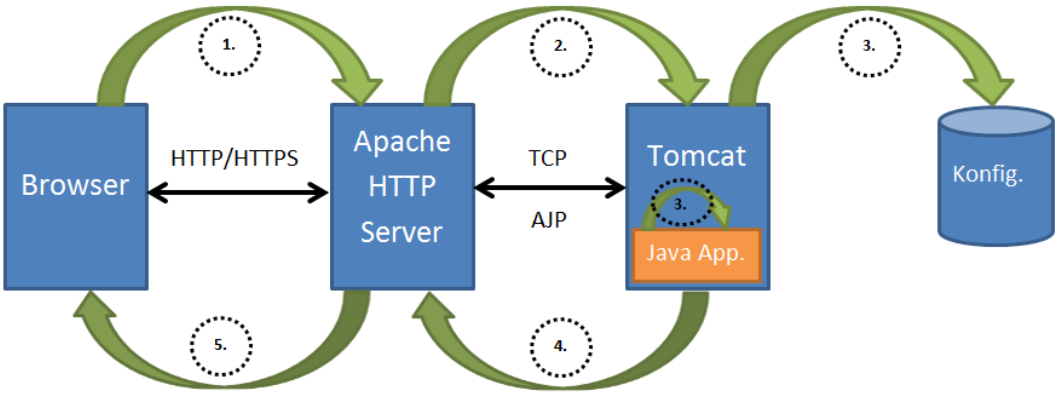

Abbildung 6.4: Kommunikation zwischen Browser, Apache HTTP Server und Tomcat

2. Nach der Eingabe des Benutzernamens und des Passworts, gelangt man in die Oberfläche in Abbildung 6.5. Diese Oberfläche dient zur Administration des Uniserv-Systems.

3. Als nächstes muss unter dem Menüpunkt „Users“ ein Benutzer angelegt werden. Hierzu werden folgende Angaben benötigt:
   - Login - Hier wird der Login Name hinterlegt. In der Testumgebung ist dies die E-Mail-Adresse des Benutzers.
   - First Name - Hier wird der Vorname des Benutzers eingegeben.
   - Last Name - Hier wird der Nachname des Benutzers eingegeben.
   - Password - Hier muss der Benutzer sein Anmeldepasswort hinterlegen.
   - Svn login - Hier wird der Nutzername der Svn-Verwaltung angegeben. Als Standard wird hier der Benutzer „uniserv“ gesetzt.
   - Svn password - Hier wird das Passwort für die Svn-Verwaltung gesetzt.
   - Role - Unter diesem Menüpunkt wird ein Profil hinterlegt, das dem Benutzer seine Rolle bzw. seine Berechtigungen zuweist. Hierbei gibt es die Wahlmöglichkeit zwischen Administrator, Viewer, Operation manager und Designer. Für den Nutzer des Testsystems werden hier alle Rollen aktiviert. Dies erlaubt dem Benutzer einen vollen und uneingeschränkten Zugriff auf das System.
   - Active - Über diese Checkbox kann das Profil aktiviert und deaktiviert werden. Für den User des Testsystems wird das Profil natürlich aktiviert.
   - Data Integration Suite - Über diese Checkbox wird dem Nutzer die Berechtigung für die Data Integration Suite erteilt. Diese wird für den Testnutzer aktiviert.
   - Data Quality Explorer - Über diese Checkbox wird dem Nutzer die Berechtigung für den Data Quality Explorer erteilt. Diese wird für den Testnutzer aktiviert.
   - Data Quality Monitor - Über diese Checkbox wird dem Nutzer die Berechtigung für den Data Quality Monitor erteilt. Diese wird für den Testnutzer deaktiviert, da diese Option nicht durch die Testlizenzen abgedeckt ist.

- Data Quality Real Time Suite - Über diese Checkbox wird dem Nutzer die Berechtigung für die Data Quality Real Time Suite erteilt. Dies wird für den Testnutzer aktiviert.

4. Sind alle Einstellungen vorgenommen, wird der Benutzer über Betätigen des Save-Buttons angelegt.

5. Als nächstes muss nun über den Menüpunkt „Projects“ ein Projekt erstellt werden. Hierzu werden folgenden Angaben benötigt:
   - Label - Hier wird der Name des Projektes hinterlegt. In der Testumgebung wurde der Name „demoProjekt“ gewählt.
   - Language - Hier kann die Programmiersprache festgelegt werden, in der dieses Projekt realisiert wird. Im Fall der Testumgebung ist dies Java.
   - Active - Diese Checkbox muss aktiviert werden, um das Projekt zu aktivieren.
   - Description - Hier kann optional eine Projektbeschreibung hinterlegt werden.
   - Author - Hier kann der Designer des Projekts hinterlegt werden.
   - Url - Hier wird die Url hinterlegt, unter der die Webservices des Projekts aufgerufen werden können. In der Testumgebung ist dies: „http://localhost/svn/repotis/DEMOPROJEKT“.
   - Login - Hier wird der Svn-Login abgefragt. Also wird „uniserv“ eingegeben.
   - Password - Hier wird das Passwort des Benutzers abgefragt.
   - Svn commit mode - Hier wird hinterlegt, wie neue Svn-Versionen commited werden. In der Testumgebung wird dies auf „Automatic“ gesetzt.
   - Svn lock mode - Hier kann definiert werden wie die Svn-Sperren für ein Projekt gesetzt werden (nur wichtig bei mehreren Projektmitarbeitern). Hier wird also auch die Option „Automatic“ gewählt.
   - Svn user log - Mit dieser Checkbox kann man die Erstellung einer Logdatei für den Svn-Betrieb aktivieren. Dies wird für die Testumgebung nicht aktiviert.

6. Sind alle Einstellungen vorgenommen, wird das Projekt über Betätigen des Save-Buttons angelegt.

7. Als letzten Schritt muss man nun noch die User, die an einem Projekt arbeiten sollen, dem Projekt zuweisen. Dies wird unter dem Menüpunkt „Projects authorizations“ durchgeführt.

8. In diesem Menü sind in einer Tabelle alle angelegten User aufgelistet und in einer zweiten Tabelle alle Projekte. Die Zuweisung erfolgt dadurch, dass der User per Drag and Drop auf das entsprechende Projekt gezogen wird. Im Fall der Testumgebung wird der angelegte User „robert.reisinger@campus.lmu.de“ auf das Projekt „demoProjekt“ gezogen.

9. Nun, da alle notwendigen Einstellungen vorgenommen wurden, kann man sich aus der Administrationsapplikation ausloggen.

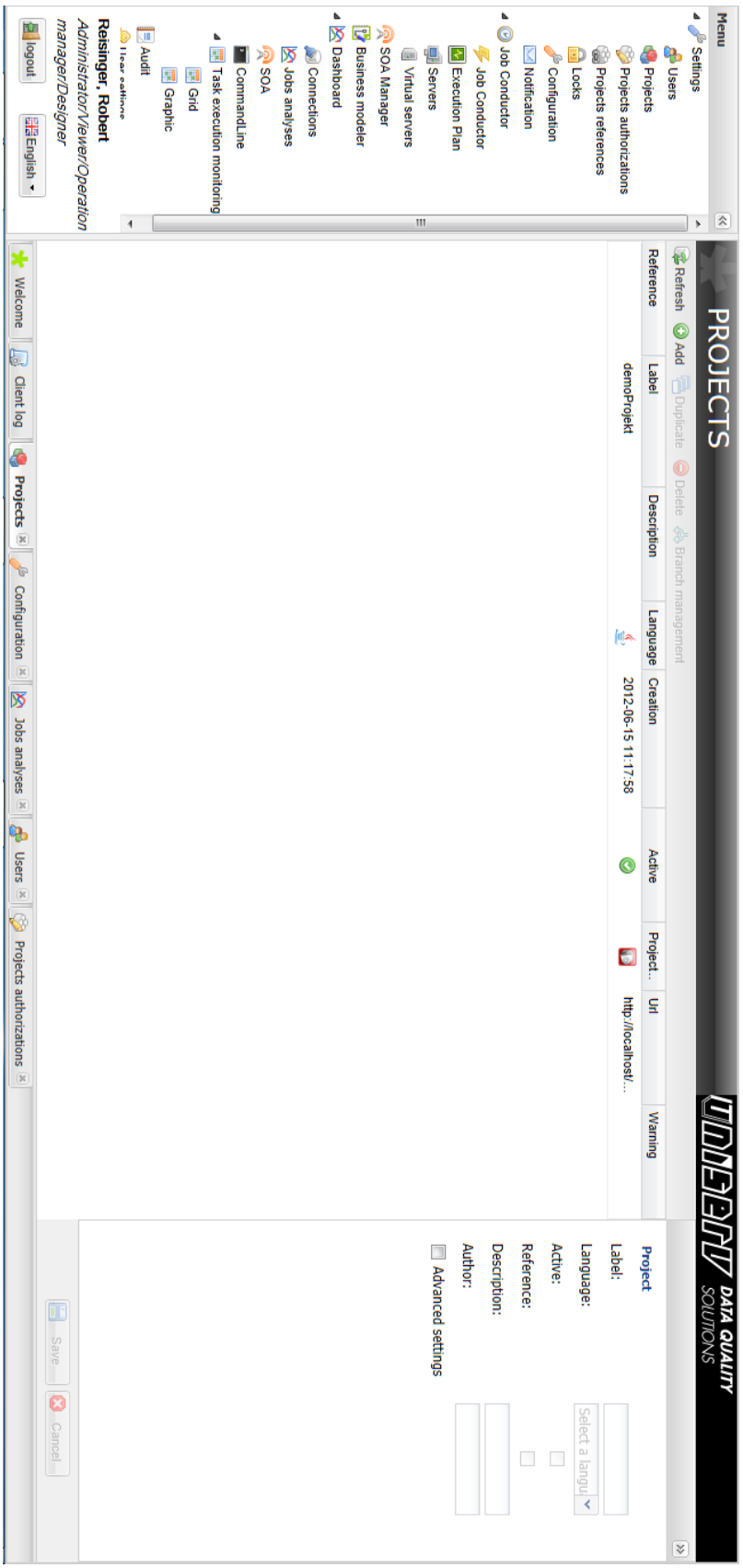

Abbildung 6.5: Uniserv Administration

6.3 Definition des Prüfszenarios für die Adressvalidierung

6.3.1 Beschreibung der Testdaten

Die Basisversion der Testdaten wurde von der Firma iC Consult GmbH zur Verfügung gestellt. Dabei handelt es sich um eine Excel Datei mit Adressdaten von Lieferanten, die die folgende Form aufweisen:

Attributbezeichnung	Beispiel
LIEF_NR	100382
ZA	10
NAME_1	BENTELER AUTOMOBILTECHNIK
NAME_2	GMBH
NAME_3	WERK 20 WEIDENAU
VEBLIZ_STAT	SP
STREET	EDISONSTR. 39
COUNTRY	D
ZIPCODE	90431
TOWN	NUERNBERG
POSTFACH	14 64
ISOCOUNTRYCODE	DE
ADDR_COUNTRY	DE
ADDR_REGION	BAYERN
ADDR_POSTCODE	90431
ADDR_MUNICIPALITY	Nürnberg
ADDR_SETTLEMENT	Nürnberg
ADDR_STREETNAME	Edisonstrasse
ADDR_STREETBASENAME	EDISON
ADDR_STREETTYPE	STRASSE
ADDR_HOUSENUMBER	39
VALIDATION_INFO	1 HouseNo, Street, City, Postalcode ok and Houseno checked

Dabei handelt es sich bei den letzten 10 Attributen um die erwarteten Rückgabewerte und bei den restlichen Attributen um die Eingabewerte. Im nächsten Schritt wurden nun die Attribute ausgewählt, die für die Evaluation der Adressvalidierung herangezogen werden sollen. Hierbei handelt es sich um die Eingabeattribute STREET, ZIPCODE, TOWN und ISOCOUNTRYCODE. Zusätzlich wurde zur besseren Auswertung und Identifikation der Datensätze noch ein Attribut ID hinzugefügt. Außerdem beinhaltet die Excel Datei Testdaten für verschiedene Länder. Da jedoch nur deutsche Adressdaten validiert werden sollen (aufgrund der Referenzdaten, die zur Verfügung stehen), werden nur deutsche Adressdaten ausgewählt. Diese 2066 Testdatensätze werden in eine CSV-Datei exportiert und sind somit bereit zur Auswertung. Ein Beispieltestdatensatz in der CSV-Datei wäre: „1;DE;WALDNIELER STR. 151;41068;MOENCHENGLADBACH“. Zu beachten ist hierbei, dass alle Testdaten in Großbuchstaben vorliegen, was bei der Implementierung der Adressvalidierung eine Rolle spielt. So behandelt Tolerant, im Gegensatz zu Uniserv, eine Umformung von „WALDNIELER STR.“ in „Waldnieler Str.“ nicht als Korrektur der Adresse. Die Änderung von „MUEN-

CHEN“ in „München“ wird jedoch von beiden Systemen als Änderung gewertet.
Als letzten Schritt bei der Präparation der Testdaten, muss nun noch das Attribut VALIDATION_INFO, für alle deutschen Datensätze, ausgewertet werden. Dieses beinhaltet den erwarteten Rückgabewert eines Datensatzes. Für die Auswertung ist es wichtig, die Anzahl der jeweiligen Rückgabewerte für alle Datensätze zu bekommen. Hierzu wird in Excel das Auftreten jedes möglichen Rückgabewertes gezählt. Hierbei tritt die folgende Verteilung der Rückgabewerte auf:

Rückgabewert	Count(*)
1 HouseNo, Street, City, Postalcode ok and Houseno checked	1180
1 HouseNo, Street, City, Postalcode ok	45
1 HouseNo, Street and City ok, Postalcode matches in the first 2 digits	32
1 Houseno, Street and Postalcode ok, City does not match	198
1 HouseNo and City ok, special streetname check for ES,IT,PT	0
2 City ok, special streetname check for ES,IT,PT	0
2 Street and City ok, Postalcode matches in the first 2 digits	2
2 Street and Postalcode ok, City does not match	40
2 Street, City, Postalcode ok	163
3 City ok	41
3 Postalcode and City ok	317
97 did not pass any check	42
98 Geocoding failed; no entry found	6
99 Geocoding not available for country	0
Keine Adresse	0

Diese Rückgabewerte dienen als Richtlinie dafür, welche Ergebnisse bei der Auswertung durch Tolerant und Uniserv erwartet werden. Sie werden jedoch nicht direkt mit den Auswertungen der beiden Testsysteme verglichen, da die Referenzdaten bei der obigen Auswertung älter waren als die der Testsysteme von Tolerant und Uniserv.

6.3.2 Klassifikation der Rückgabedaten

Um die Ergebnisse der Adressvalidierung von Uniserv und Tolerant vergleichbar zu machen, ist es in einem ersten Schritt notwendig, die möglichen Rückgabedaten unabhängig von beiden Systemen zu klassifizieren. Diesen Klassen werden dann, in einem späteren Schritt, die tatsächlichen Rückgabewerte der beiden Systeme zugewiesen. Somit ist ein quantitativer Vergleich beider Systeme auf dieser Klassen-Ebene möglich. In der folgenden Auflistung werden die Ergebnisklassen für die Adressvalidierung vorgestellt:

- Klasse 1 - Diese Klasse enthält alle sicher gefundenen Adressen, unabhängig davon ob kleinere Veränderungen vorgenommen wurde oder nicht.
  - A - Adressqualität = 100%
  - B - Adressqualität <100% aber die Adresse wird noch sicher gefunden und gilt als zustellbar.
- Klasse 2 - Diese Klasse enthält alle Datensätze mit fehlerhaftem Orte, fehlerhafter PLZ oder Ort-PLZ-Kombinationen.

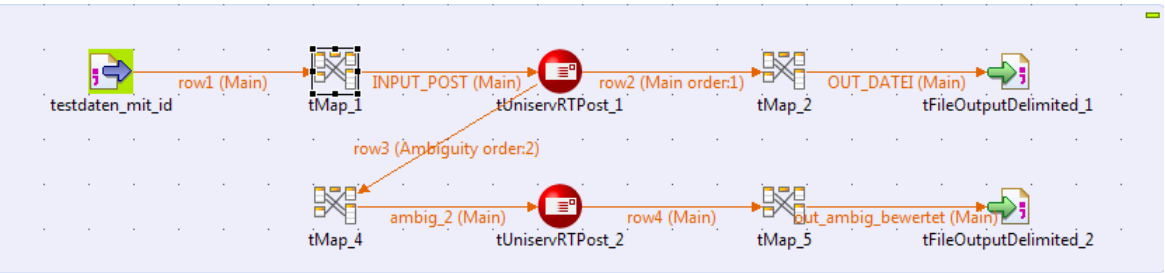

Abbildung 6.6: Jobdesign für die Testdatenvalidierung mit der Data Integration Suite

  - A - Ort und PLZ fehlen oder sind so fehlerhaft, dass eine Korrektur nicht möglich war.
  - B - Die PLZ ist fehlerfrei; der Ort fehlt oder ist so fehlerhaft, dass eine Korrektur nicht möglich ist.
  - C - Ein eindeutiger Ort ist fehlerfrei oder mit leichten/korrigierbaren Fehlern; PLZ fehlt oder ist nicht korrigierbar.
  - D - Ein mehrdeutiger Ort ist fehlerfrei oder mit leichten/korrigierbaren Fehlern; PLZ fehlt oder ist nicht korrigierbar.

- Klasse 3 - Diese Klasse enthält alle Datensätze mit fehlerhafter Straße oder Hausnummer oder Straßen-Hausnummer-Kombination.
  - A - Hausnummer und Straße fehlen oder sind nicht korrigierbar.
  - B - Die Hausnummer fehlt; die Straße ist richtig oder mit leichten/korrigierbaren Fehlern.
  - C - Die Straße fehlt oder ist nicht korrigierbar; die Hausnummer ist richtig.
  - D - Die Straße ist richtig oder mit leichten/korrigierbaren Fehlern; die Hausnummer ist in der Straße nicht vorhanden.
  - E - Die Straße ist richtig oder mit leichten/korrigierbaren Fehlern aber mehrdeutig; die Hausnummer ist richtig.

- Klasse 4 - Diese Klasse enthält alle Datensätze die sowohl in Klasse 2 als auch in Klasse 3 eingeteilt werden.

6.4 Implementierung der Prüfszenarien für die Adressvalidierung

6.4.1 Beschreibung der Implementierung über die Data Integration Suite

Als erster Schritt der Implementierung muss der Job für die Adressvalidierung im Data Quality Service Hub erstellt werden (siehe Abb.6.6). Die Erstellung dieses Jobs soll nun Schritt für Schritt erörtert werden:

Anlegen der Metadaten Als erstes muss das Objekt „testdaten_mit_id“ angelegt werden. Hierbei handelt es sich um einen „tFileOutputDelimited“ der in Kapitel 5 erörtert wurde. Um dieses Objekt anzulegen, klickt man im Menü auf der linken Seite des DQSH (siehe Abb.

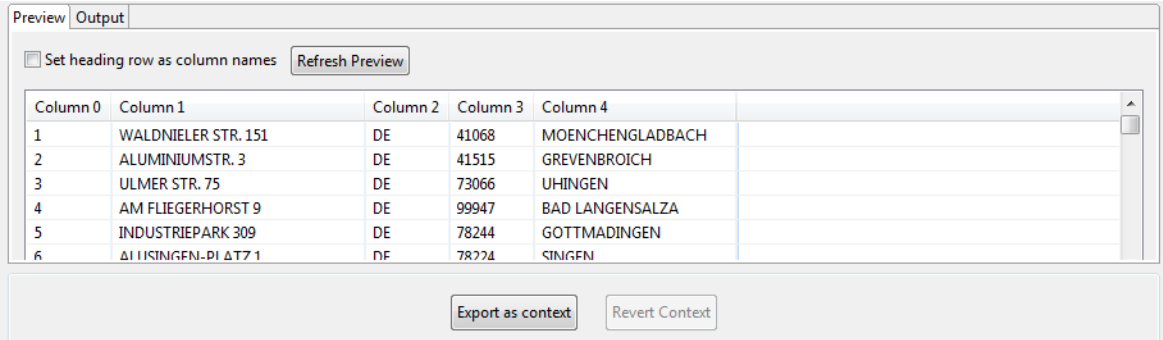

Abbildung 6.7: Anzeige von einigen Testdatensätzen

5.1) auf den Unterpunkt „Metadaten“ und dort dann auf den Unterpunkt „File delimited“. Hier wählt man nun die Option „Create file delimited“ aus. In der folgenden Maske wird der Name des Objekts abgefragt, in diesem Fall wurde „testdaten_mit_id“ gewählt. Als nächstes wird der Dateipfad der Inputdatei abgefragt(in diesem Fall: „D:\testdaten.csv“). Das DQSH erkennt nun automatisch die verwendeten Trennzeichen in dieser Datei und zeigt exemplarisch einige Datensätze an (siehe Abb. 6.7). In der letzten Maske kann man schließlich die Spaltenbezeichnungen und den Datentyp der Eingabedaten festlegen. Wie in Abbildung 6.7 zu sehen ist, sind die Spalten mit „Column 0“„Column 1“ usw. gekennzeichnet. Dies wird abgeändert und die Spalten/Attribute erhalten die Namen (von links nach rechts) „IN_ID“, „IN_STR_HNO“, „IN_COUNTRY“, „IN_ZIP“ und „IN_CITY“. Das DQSH versucht außerdem die Datentypen der einzelnen Attribute zu ermitteln. Für die ID und die PLZ wird Integer erkannt und vorgeschlagen. Jedoch ist es für die weitere Verarbeitung sinnvoll, diese Datentypen in String zu ändern.

Erstellen des Jobaufbaus Als nächstes werden die einzelnen Komponenten (2x tUniservRTPost, 4x tMap, 2x tFileOutputDelimited und die eben angelegte Input-Datei), wie in Abbildung 6.6 angeordnet und miteinander verknüpft.

Mapping von „tMap1“ Im dritten Schritt wird das erste Mapping in der Komponente „tMap1“ konfiguriert. Hierbei werden die Input-Attribute, wie in der folgenden Tabelle dargestellt, auf die Input-Attribute des Objekts „tUniservRTPost_1“ abgebildet:

testdate_mit_id	tUniservRTPost_1	Beispieldaten
IN_STR_HNO	IN_STR_HNO	WALDNIELER STR. 151
IN_COUNTRY	IN_COUNTRY	DE
IN_ZIP	IN_ZIP	41068
IN_CITY	IN_CITY	MOENCHENGLADBACH
IN_ID	ORG_IN_ID	1
IN_STR_HNO	ORG_IN_STR_HNO	WALDNIELER STR. 151
IN_CITY	ORG_IN_CITY	MOENCHENGLADBACH

Bei den ersten vier Attributen handelt es sich um Attribute, die zur Auswertung der Adresse benutzt werden. Das Attribut „IN_ID“ wird, ohne verarbeitet zu werden, an die Variable „ORG_IN_ID“ durchgereicht. Auch die Input-Attribute „IN_STR_HNO“ und „IN_CITY“

werden zusätzlich noch einmal in die Variablen „ORG_IN_STR_HNO“ bzw. „ORG_IN_CITY“ gemappt, welche auch nicht verarbeitet werden, sondern die postalische Prüfung unverändert wieder verlassen. Dies hat den Sinn, dass diese Variablen, in unbearbeiteter Form, im Schritt „Mapping von tMap2“ noch einmal benötigt werden.

Konfiguration von „tUniservRTPost_1 “ Wie in Kapitel 5 schon erwähnt wurde, besitzt das Objekt „tUniservRTPost_1 “ drei verschiedene Ausgänge (Main-Pfad, Ambiguous-Pfad und Reject-Pfad). In diesem Job ist es jedoch sinnvoller für die spätere Auswertung, den Main-Pfad und den Reject-Pfad zusammenzufassen. Dies geschieht durch das Setzen eines Hakens bei der Option „Use rejects“ im Objekt „tUniservRTPost_1 “. Dadurch werden nun sowohl die zustellbaren Adressen, als auch die fehlerhaften oder unzustellbaren Adressen in den Main-Pfad ausgegeben. Lediglich die mehrdeutigen Adressen werden in einen eigenen Pfad ausgelagert, da einen mehrdeutige Adresse, die in „tUniservRTPost_1 “ verarbeitet wird, mehrere Ausgabeadressen generiert (nämlich alle möglichen Adressen, die diese mehrdeutige Adresse annehmen könnte).

Mapping von „tMap2“ Als nächstes wird das Mapping in der Komponente „tMap2“ vorgenommen. Dieses Objekt erhält als Inputwerte die Main-Ausgabe des Objekts „tUniservRTPost_1 “ und mappt diese Ausgabeattribute auf die Attribute von „tFileOutputDelimited_1“. Dieser schreibt die Datensätze dann wiederum in eine Ausgabedatei, welche als Basis für die Auswertung der Testdaten dient. Das Mapping wird wie in Abbildung 6.8 realisiert. Hierbei gibt es jedoch einige Besonderheiten zu beachten. Auf der rechten Seite der Grafik sieht man die Attribute, wie sie (mit Strichpunkt getrennt) datensatzweise in die Ausgabedatei geschrieben werden (die Reihenfolge wurde, zur besseren Darstellbarkeit, verändert). Auf der linken Seite sieht man die Ausgabeattribute von „tUniservRTPost_1 “. Die weiß hinterlegten Attribute werden, wie auch schon zuvor, von der linken Seite auf die rechte Seite gemappt. Dabei handelt es sich um die Returncodes die im folgenden Kapitel genauer erörtert werden, sowie um die Attribute „OUT_MVAL_CITY“ und „OUT_MVAL_STREET“, die die Qualität von Stadt und Straße in einem Intervall von 0 bis 100 angeben. Außerdem wird das Attribut „IN_ID“ ausgegeben, in dem die ID des Datensatzes vermerkt ist. Die Attribute „OUT_COUNTRY“ und „OUT_ZIP“ enthalten das Rückgabeland und die Rückgabepostleitzahl.
Etwas komplexer wird das Mapping bei den markierten Attributen. Wie schon erwähnt wurde bewertet Uniserv ein Attribut als geändert, wenn lediglich die Großbuchstaben in Kleinbuchstaben geändert werden. Tolerant hingegen bewertet in diesem Fall das Attribut als nicht geändert. Um beide Systeme vergleichbar zu machen, werden in Uniserv zwei zusätzliche Ausgabevariablen definiert, nämlich „STREET_CORRECTED“ und „CITY_CORRECTED“. Diese Variablen werden mit einem Boolean-Wert beliefert, den die Verarbeitung in der Mitte von Abbildung 6.8 zurückgibt. Bei dieser Verarbeitung handelt es sich um einen Zwischenschritt zwischen dem Output von „tUniservRTPost_1 “ und dem Input von „tFileOutputDelimited_1“. Wie in der Abbildung zu sehen ist, werden der ersten Zeile der Verarbeitung die Attribute „OUT_STR_HNO“ und „ORG_IN_STR_HNO“ übergeben. In der Verarbeitung selbst wird dann die folgende Codezeile verarbeitet:

```
1 StringHandling.UPCASE(row2.OUT_STR_HNO).equals(row2.ORG_IN_STR_HNO) ? true : false
```

Die korrigierte Straße in „OUT_STR_HNO“ wird wieder in Großbuchstaben transformiert und dann mit dem ursprünglichen Input-Wert des Straßenfeldes verglichen, das unverändert

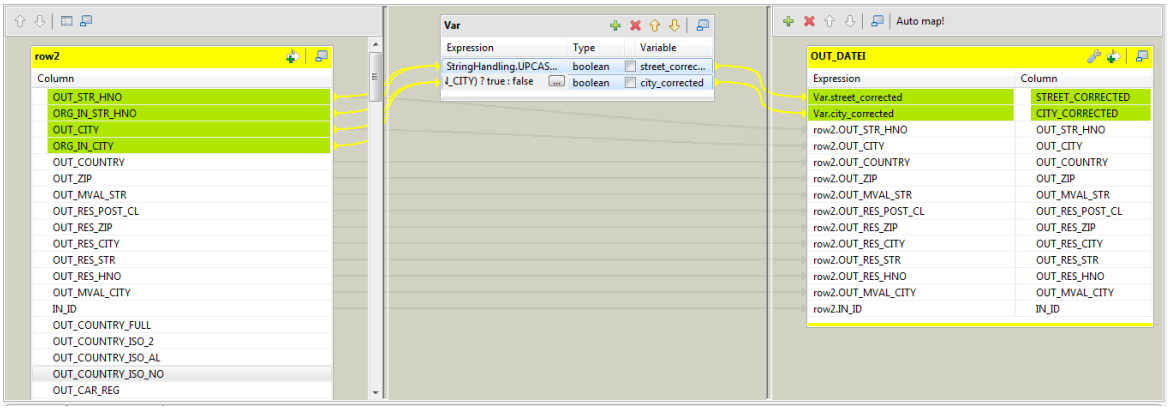

Abbildung 6.8: Mapping im Objekt „tMap2“

im Attribut „ORG_IN_STR_HNO“ zwischengespeichert wurde. Sind beide Attribute gleich, wurde abgesehen von der Änderung von Groß- in Kleinbuchstaben keine weitere Korrektur vorgenommen und es wird „true“ zurückgegeben. Ansonsten wurde eine andere Änderung vorgenommen und er Rückgabewert ist „false“. Dieser Boolsche Wert wird dann dem Ausgabeattribut „STREET_CORRECTED“ übergeben und kann somit mit dem Rückgabewert von Tolerant verglichen werden. Analog wird bei den Attributen „OUT_CITY“ und „ORG_IN_CITY“ vorgegangen.

Mapping von „tMap4“ und „tMap5“ Das Objekt „tMap4“ ist für das Mapping des Ambiguous-Pfads von „tUniservRTPost_1“ und dem Input von „tUniservRTPost_2“ zuständig. Dieser Schritt ist notwendig, da die Ausgabeattribute des Ambiguous-Zweigs noch nicht alle Returncodes zur Verfügung stellen, die für die Evaluation notwendig sind. Die gesamte Bandbreite an Returncodes/Returnattributen ist lediglich im Main-Zweig verfügbar. Deswegen werden, wie in Abbildung 6.9, die Ausgabedaten von „tUniservRTPost_1 “ als neue Eingabedaten für „tUniservRTPost_2“ verwendet, um dort dann im Main-Zweig die entsprechenden Returncodes zu bekommen. Hierbei ist zu bemerken, dass „tUniservRTPost_1“ für einen mehrdeutigen Eingabedatensatz, mehrere Ausgabedatensätze/Lösungsvorschläge erstellt, die dann von „tUniservRTPost_2“ verarbeitet werden. Die Ausgabe von „tUniservRTPost_2 “ wird dann wiederum über „tMap5“ an das Ausgabeobjekt „tFileOutputDelimited_2“ gesendet um dort in einen Datei geschrieben zu werden. Dieser Vorgang bzw. die Attribute die in die Datei geschrieben werden, wurden unter „Mapping von tMap2“ bereits behandelt.

6.4.2 Beschreibung der Returncodes von Uniserv

Die Software von Uniserv stellt eine Vielzahl von Returncodes zur Verfügung. Im Folgenden sollen jedoch nur diejenigen vorgestellt werden, die für die Durchführung der Evaluation benutzt werden. Die Variable, die die Gesamtqualität eines Adressdatensatzes bewertet, ist OUT_RES_POST_CL. Diese kann mit den folgenden Werten belegt sein:

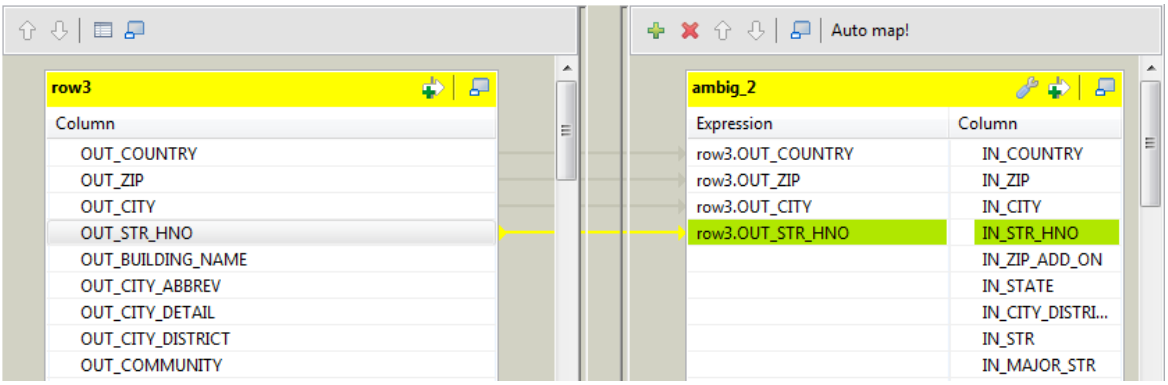

Abbildung 6.9: Mapping im Objekt „tMap4“

Wert	Details
1	Adresse sicher gefunden, unverändert
2	Adresse sicher gefunden, verändert
3	Adresse unsicher gefunden, verändert, gilt aber als zustellbar
5	Adresse nicht gefunden

Zusätzlich dazu gibt es Variablen, die die Qualität einzelner Adresskomponenten bewerten. Hierbei stehen u.a. folgende Variablen zur Verfügung:

Variable	Details
OUT_RES_ZIP	Bewertung der PLZ
OUT_RES_CITY	Bewertung des Stadtnamens
OUT_RES_STR	Bewertung des Straßennamens
OUT_RES_HNO	Bewertung der Hausnummer

Diese Variablen können jeweils verschiedene Werte annehmen, wobei einige Werte für alle Variablen gleich sind und andere nur für spezielle Variablen zulässig sind. In der folgenden Tabelle werden die allgemeingültigen Werte aufgeführt:

Wert	Abkürzung	Details
1	OK	Eingabe war richtig
2	NI	Keine Eingabe vorhanden
4	C	Eingabe wurde geändert (nicht bei OUT_RES_HNO)
5	SC	Die Eingabe wurde umgeschlüsselt
6	NF	Die Eingabe wurde nicht gefunden

Die folgenden Tabellen zeigen die speziellen Werte der einzelnen Variablen:

OUT_RES_ZIP:

Wert	Abkürzung	Details
3	PI	Nur Teile der PLZ wurden eingegeben
11	DF	Default PLZ wurde verwendet (kleinste PLZ des Ortes)

OUT_RES_STR:

Wert	Abkürzung	Details
7	NC	Eingabe nicht geprüft
9	NP	Eingabe nicht geprüft wegen Postfachangabe im Straßenfeld

OUT_RES_HNO:

Wert	Abkürzung	Details
7	NC	Eingabe nicht geprüft

6.5 Auswertung der Ergebnisse

6.5.1 Vorbereitung der Auswertung

Der Uniserv Data Integration-Job liefert als Ergebnis 2 Dateien. Zum einen die Datei „outmain.csv“, die die Ausgabe des Main-Pfades enthält und zum anderen die Datei „outambig.csv“, die die Ausgabe des Ambiguous-Zweigs enthält. Der Aufbau der beiden Dateien ist allerdings identisch. Somit ergibt sich die folgende Ausgabe:

Attributname	Beispiel
IN_ID	1
OUT_COUNTRY	DE
OUT_ZIP	41068
OUT_CITY	Mönchengladbach
OUT_STR_HNO	Waldnieler Str. 151
OUT_MVAL_STR	100
OUT_RES_POST_CL	2
OUT_RES_ZIP	OK
OUT_RES_CITY	C
OUT_RES_STR	C
OUT_RES_HNO	OK
OUT_MVAL_CITY	100
STEET_CORRECTED	true
CITY_CORRECTED	false

Um alle Daten auswerten zu können, ist es nun notwendig, beide Ausgabedateien zu konsolidieren und gleichzeitig die Vergleichbarkeit mit Tolerant zu gewährleisten. Die Datei „outmain.csv“ enthält 2049 Datensätze und die Datei „outambig.csv“ enthält 84 Datensätze. Dies ergibt in der Summe 2133 Datensätze. Es wurden aber lediglich 2066 Inputdatensätze verwendet. Die Differenz entsteht dadurch, dass die Datei „outambig.csv“ für einen Input-Datensatz mindestens 2 Output-Datensätze (Lösungsvorschläge für den mehrdeutigen Input-Datensatz) enthält. Bei Tolerant werden insgesamt 2066 Datensätze zurückgegeben, was einen Vergleich zwischen beiden Systemen schwieriger gestaltet. Das Problem ist, dass die mehrdeutigen Ergebnisse (also 2133-2066 = 67 Datensätze bei Uniserv und 2066-2066 = 0 Datensätze bei Tolerant) immer eine hohe Adressqualität besitzen, da es sich um Lösungsvorschläge des jeweiligen Systems handelt. Aus diesem Grund wird bei der Evaluation in

dieser Arbeit, bei mehrdeutigen Adressen, immer der erste Adressvorschlag ausgewählt und der Rest verworfen. Für die Konsolidierung der beiden Ausgabedateien „outmain.csv“ und „outambig.csv“ heißt dies, dass aus der Datei „outambig.csv“ immer der erste Datensatz einer Vorschlagsmenge (= die Menge an Adressvorschlägen, die durch einen Input-Datensatz generiert werden) ausgewählt wird, und in die Datei „outmain.csv“ eingefügt wird. Nach dieser Bearbeitung enthält die Datei „outmain.csv“ nun 2066 Datensätze (was der Anzahl der Input-Datensätze entspricht) und ist nun bereit zur Auswertung.

6.5.2 Durchführung der Auswertung

Für die Auswertung wird die Datei „outmain.csv“, die nun 2066 Datensätze umfasst, in Excel importiert. Hierbei ist darauf zu achten, dass das Datenformat aller importierten Spalten auf das Format „Text“ eingestellt ist, da es ansonsten zu fehlerhafter Darstellung der Hausnummern kommen kann. Da das Hausnummer-Attribut nicht nur einzelne Nummern, sondern auch Intervalle von Hausnummern enthält, ist es ansonsten möglich, dass das Hausnummerintervall „1-12“ als Datum interpretiert wird.

Auswertung des Returncodes „OUT_RES_POST_CL“ Wie im vorangegangenen Kapitel vorgestellt wurde, handelt es sich bei diesem Returncode um die Bewertung der Zustellbarkeit einer Sendung an diese Adresse. Interessant ist hierbei die Anzahl des Auftretens der einzelnen Werte dieses Returncodes. Deshalb wird in der Spalte „OUT_RES_POST_CL“ in der Datei „outmain.csv“ gezählt wie oft die jeweiligen Werte auftreten. Dies führt zu folgendem Ergebnis:

OUT_RES_POST_CL	COUNT(*)
1	17
2	1792
3	233
4	0
5	24
Summe	2066

Interessant an diesen Daten ist, dass es nur 17 Datensätze mit Klassifikation 1 („sicher gefunden, unverändert“) gibt aber 1792 mit Klassifikation 2 („sicher gefunden, verändert“). Dies liegt daran, dass in die internen Berechnungen für diesen Wert, die Attribute „OUT_RES_CITY“ und „OUT_RES_STREET“ herangezogen werden, welche aufgrund der Großbuchstabenproblematik immer eine Veränderung anzeigen. Die 17 Datensätze mit Klassifikation 1, sind diejenigen Datensätze, die aus der Ambiguous-Datei eingefügt wurde.

Auswertung des Returncodes „OUT_RES_ZIP“ Dies ist der Returncode, der die Art der Auswertung und Änderung am PLZ-Feld zurückgibt. Auch hier ist die Auswertung relativ einfach, da die einzelnen Werte des Attributs „OUT_RES_ZIP“ lediglich gezählt und dann aufsummiert werden müssen. Dies führt zu folgendem Ergebnis (es werden nur die Werte in der Tabelle dargestellt, die mindestens einmal vorkommen):

OUT_RES_ZIP	COUNT(*)
OK	1958
C	9
DF	1
SC	97
LEER	1
Summe	2066

Interessant hierbei ist, dass 1 Attributwert leer ist, das Attribut also keinen Rückgabewert enthält. Hierbei handelt es sich um einen Datensatz der nicht verarbeitet werden konnte und somit auch keine Returncodes erzeugt hat. Ansonsten waren 1958 Postleitzahlen in Ordnung, 97 mussten umgeschlüsselt werden, 9 wurden korrigiert und bei einer PLZ wurde eine Default-PLZ verwendet.

Auswertung des Returncodes „OUT_RES_HNO“ Hierbei handelt es sich um den Returncode, der die Auswertung und Änderung an der Hausnummer zurückgibt. Auch hier muss, wie schon bei den vorangegangenen Auswertungen lediglich gezählt werden, was zu folgendem Ergebnis führt:

OUT_RES_HNO	COUNT(*)
OK	1957
SC	4
NF	7
NI	90
NC	7
LEER	1
Summe	2066

Auch hier ist ein Attributwert leer. Hierbei handelt es sich um den selben Datensatz wie auch schon bei der Auswertung von „OUT_RES_ZIP“. Ansonsten war bei 97 Adressen (NF+NI) kein Input für die Hausnummer vorhanden und 7 Hausnummern konnten nicht geprüft werden. 1957 Hausnummern waren in Ordnung und 4 wurden umgeschlüsselt.

Auswertung des Returncodes „OUT_RES_CITY“ Bei diesem Returncode ist die Auswertung etwas komplexer. Zählt man die einzelnen Werte von „OUT_RES_CITY“ so erhält man:

OUT_RES_CITY	COUNT(*)
OK	17
C	2038
NF	1
SC	10
Summe	2066

Das hohe Auftreten von „C“ erklärt sich wieder durch die Großbuchstabenproblematik. Eine Lösung hierfür wurde jedoch beim Jobdesign vorgenommen, indem eine zusätzliche Variable „STREET_CORRECTED“ eingefügt wurde, die auf „true“ gesetzt wurde, wenn lediglich die Großbuchstaben in Kleinbuchstaben transformiert wurden und die auf „false“ gesetzt wurde, wenn andere Änderungen vorgenommen wurden. Wenn man nun alle Da-

tensätze zählt, die bei „STREET_CORRECTED“ auf „false“ gesetzt sind, erhält man noch nicht die genaue Anzahl aller Datensätze die korrigiert („C“) wurden. Dies ist der Fall, weil die Datensätze die unter „SC“ fallen, auch umgeschlüsselt wurden und somit auch ein „false“ im Attribut „STREET_CORRECTED“ enthalten. Um dies zu verhindern, werden alle Datensätze gezählt, die ein „C“ als Wert von „OUT_RES_CITY“ enthalten und deren Wert von „STREET_CORRECTED“ auf „false“ steht. Somit erhält man die Anzahl aller Datensätze die wirklich korrigiert wurden. Um die Anzahl der Datensätze zu finden in denen die Straße in Ordnung ist, werden die Datensätze gezählt die den Wert „OK“ in „OUT_RES_CITY“ haben und dazu werden die Datensätze addiert die „C“ enthalten und deren Wert bei „STREET_CORRECTED“ auf „true“ steht. Dies verhindert, dass der Datensatz mit dem Wert „NF“ mitgezählt wird, der bei „STREET_CORRECTED“ auch auf „true“ gesetzt ist, da die Eingabe leer ist. Die Werte „NF“ und „SC“ hingegen, können wieder ganz normal gezählt werden. Somit ergibt sich dann die folgende endgültige Auswertung:

OUT_RES_CITY	COUNT(*)
OK	1489
C	566
NF	1
SC	10
Summe	2066

Auswertung des Returncodes „OUT_RES_STREET“ Die Auswertung des Returncodes „OUT_RES_STREET“ erfolgt ganz analog zur Auswertung von „OUT_RES_CITY“, da auch hier der Wert „C“ nach der gleichen Methode wie oben, in „OK“ und „C“ aufgesplittet werden muss ohne die anderen Datensätze fälschlicherweise mitzuzählen. Somit ergibt sich folgende Auswertung:

OUT_RES_CITY	COUNT(*)
OK	1213
C	821
NF	23
NC	3
SC	5
LEER	1
Summe	2066

7 Realisierung der Adressvalidierung der Firma Tolerant

7.1 Installation der Software

Als Voraussetzung für die Installation wird das Java Runtime Environment(JRE) und Java Development Kit(JDK) benötigt(mindestens in der Version 6.18 [JRE] bzw. 1.6.0.18 [JDK]). Die Testumgebung wird auf dem selben System installiert, auf dem auch schon das Uniserv-Tool installiert wurde. In der folgenden Aufzählung soll nun schrittweise der Installationsprozess erklärt werden:

1. Download der Installationsdaten („InstallPOST1.22.9992_20120731_win_x86_64.zip“) und der Referenzdaten für Deutschland („DB5_DEU5BI_120401.zip“).

2. Entpacken der Installationsdaten.

3. Starten der Setup-Batch-Datei „install_win_x86_64.bat“.

4. Das Setup extrahiert nun automatisch die relevanten Installationsdaten und fordert zur Eingabe des Installationspfades auf. Hier wird der Pfad: „D:\TolerantPost“ eingegeben.

5. Anschließend werden noch einige Einstellungen für die Installation abgefragt, die folgendermaßen beantwortet werden (Die Einstellungen orientieren sich nach einer Installationsanleitung der Firma iC Consult GmbH):
   - Install Batch - N (Der Batchmodus ist für die Testumgebung nicht relevant, da sich die Evaluation auf den Real-Time-Modus beschränkt.)
   - Install Service - Y (Installation des Real-Time-Service)
   - Register Service - N (Dies wird aufgrund der Installationsanleitung der Firma iC Consult GmbH gewählt.)
   - Service Port: 9002 (Dies ist der Port des Real-Time-Service)
   - Install GUI - Y
   - An dieser Stelle wird gefragt, auf welche Art die GUI installiert werden soll. Die erste Variante besteht darin, die GUI als Web-Applikation zu installieren, auf die der User via Web-Browser zugreifen kann. Die zweite Variante ist die Installation einer Standalone-GUI. Für diese Installation wird die erste Variante ausgewählt.

6. Bestätigung der Eingaben und Start der Installation.

7. Kopieren der Datei „postserviceconfig.xml“ in den Pfad: „D:\TolerantPost\config“. In dieser Datei, die von iC Consult GmbH zur Verfügung gestellt wurde, ist unter anderem der Lizenzschlüssel hinterlegt.

8. Kopieren der Datei „synonyms.ini“ in den Pfad: „D:\TolerantPost\config“. In dieser Datei werden Synonyme hinterlegt, die bei der Verarbeitung von Anfragen an Tolerant ersetzt werden können (z.B. 9—DEUTSCHLAND—Deutschland würde den Übergabewert DEUTSCHLAND in den Wert Deutschland transformieren). Auch diese Datei wurde von ic Consult GmbH zur Verfügung gestellt.

9. Neustarten des Dienstes „Tolerant Post Service“.

10. Durchführen eines Funktionstests, indem man „http://localhost:9002/postService/admin/V1/plainPing“ aufruft. Als Rückgabewert sollte man nun die aktuelle Tolerant-Versionsnummer bekommen.

7.2 Implementierung der Prüfszenarien für die Adressvalidierung

7.2.1 Beschreibung des Skripts für die Anfragen an den Webservice

Das Skript soll die Testdatensätze einzeln aus einer Datei lesen und den gelesenen Datensatz dann an den Webservice von Tolerant schicken. Anschließend sollen die Rückgabewerte des Webservice in eine CSV-Datei eingelesen werden. Hierzu ist anzumerken, dass die Testdaten für Tolerant sich lediglich dadurch von den Testdaten für Uniserv unterscheiden, dass keine ID vorhanden ist, da diese bei Tolerant nicht benötigt wird. Ansonsten sind die Testdaten identisch. Im folgenden Listing ist der Hauptteil des Codes dargestellt der für die Realisierung der Anforderungen notwendig ist:

```
  private static String project      = "TypeAhead";
  private static String profile      = "TypeAhead-AV";
  private static String url          = "http://localhost:9002";
4 private static int    maxResNumber = 1;
  public static void main(String argv[]) throws APIException
  {
    String[] inFieldNames = { "street", "country", "postalcode", "city" };
    String[] outFieldNames = { "street", "number", "country", "postalcode", "city", "score", "quality", "ReturnCode",
9    "ShortReport", "ElementInputStatus" };
   try {
    FileOutputStream fos = new FileOutputStream(new File("D:\\out_tolerant.csv"));
          BufferedReader br = new BufferedReader (new FileReader ("D:\\testdatenTOL.csv"));
                String line = null;
14              String[] attribute = null;
                while (br.ready ()) {
                    line = br.readLine ();
                    attribute = line.split(";");
                    PostConnection connection = new PostConnection(project, profile, url, maxResNumber);
19                  PostQuery query = new PostQuery(inFieldNames, attribute);
                    try
                    {
                      connection.check(query);
                      if (query.getNrResults() > 0)
24                    {
                          for(int k = 0; k < outFieldNames.length; k++)
                          {
                            fos.write(query.getResultValue(0,outFieldNames[k]).getBytes());
                            if(k+1 < outFieldNames.length){
29                              fos.write(";".getBytes());
                            }
                          }
                          fos.write(System.getProperty( "line.separator" ).getBytes());

34                    }
                    }
```

Listing 7.1: Skript für Anfragen an den Webservice

In Zeile 1-3 werden die Konstanten festgelegt, die später für den Verbindungsaufbau zum Webservice benutzt werden. In Zeile 4 wird die Konstante „maxResNumber“ auf 1 gesetzt. Dies ist notwendig um auf eine Anfrage genau eine Antwort zu erhalten. Somit wird auch bei einer mehrdeutigen Inputadresse lediglich die erste Antwort bzw. der erste Resultatdatensatz

zurückgegeben, was für die bessere Vergleichbarkeit mit Uniserv notwendig ist. Somit liefert das Skript für 2066 Inputdatensätze genau 2066 Outputdatensätze. In Zeile 7 werden die Attributnamen für die Inputdaten definiert und in Zeile 8 wird das Selbe für die Outputdaten gemacht. In Zeile 11 und 12 werden die Objekte zum Ein- bzw. Auslesen einer Datei definiert. In Zeile 13 und 14 werden dann noch zwei Variablen definiert, die für die spätere Verarbeitung der Eingabedaten benötigt werden. In Zeile 15 wird eine While-Schleife begonnen, die erst wieder verlassen wird, sobald „br“ nicht mehr bereit zum Lesen ist(wenn also das Ende der Datei erreicht wurde). In Zeile 16 wird nun ein Datensatz gelesen und in „line“ geschrieben. In Zeile 17 wird die Variable „line“ anhand des Separators „;“ gesplittet und die einzelnen Attribute werden dann in das Array „attribute“ geschrieben. In Zeile 17 erfolgt der Verbindungsaufbau mit den Konstanten die in Zeile 1-4 festgelegt wurden. Der Ausschnitt der Request-URL „http://localhost:9002/postService/post/V1/TypeAhead/TypeAhead-AV“ zeigt, an welchen Stellen des Request die Konstanten „project = TypeAhead“„profile = TypeAhead-AV“ und „url = http://localhost:9002“ eingesetzt werden. In Zeile 18 wird der Query erstellt, der an den Webservice gesendet wird. Dieser wird konstruiert aus den Arrays „inFieldNames“(enthält Attributbezeichnungen) und „attribute“ (enthält die Attributwerte). In Zeile 22 wird der Query an den Webservice gesendet. In Zeile 27 und 29 werden die einzelnen Rückgabeattribute inklusive des Trennzeichens „;“ in die Ausgabedatei geschrieben. Sind alle Rückgabeattribute in die Datei geschrieben (Ende der for-Schleife) wird in Zeile 32 noch der das Zeichen in die Datei geschrieben, das die einzelnen Datensätze voneinander trennt. Ist das Skript fertig, stehen in der Ausgabedatei 2066 Datensätze, die fertig für die Auswertung sind.

7.2.2 Beschreibung der Returncodes von Tolerant

In Tolerant gibt es zwei verschiedene Modi für die Adressvalidierung. Zum einen gibt es den Interactive Mode, in dem die meisten/wichtigsten Adressfelder übergeben und ausgewertet werden. Zum anderen gibt es den Fast Completition Mode, der Just-In-Time Komplettierungsvorschläge anbieten kann. Der Fast Completition Mode wartet also nicht bis der Datensatz komplett eingegeben wurde, sondern überprüft die Eingabe schon während diese vorgenommen wird. Da der Interactive Mode auch im Uniserv-Testsystem realisiert ist, werden die Returncodes des Interactive Modes zum Vergleich mit Uniserv herangezogen. Die Variable, die die Qualität eines ganzen Adressdatensatzes in Tolerant beschreibt, ist das Score Value oder auch die Mail Ability Score. Diese kann 6 verschiedene Werte annehmen, welche in folgender Tabelle dargestellt sind:

Score Value	Details
5	All elements are valid
4	Quite reliable address
3	Unreliable address, but delivery possible
2	Very unreliable address
1	Corrupted address
0	Delivery impossible

Für eine detailliertere Auswertung wird das Short Report Value benutzt. Dieses bewertet die einzelnen Elemente eines Adressdatensatzes auf deren Qualität. Beim Short Report Value handelt es sich um einen String der Länge 8. Wobei jede Position innerhalb des Strings ein

Element des Datensatzes repräsentiert. Die genaue Zuordnung ist in der folgenden Tabelle dargestellt:

Position	Element
1	post.Country
2	post.PostalCode
3	post.city
4	post.SubCity
5	post.Street
6	post.Number
7	post.DeliveryService
8	post.Organization

Jede dieser Positionen kann dann wiederum einen der folgenden Werte annehmen:

Wert	Details
-	Not considered/analysed
*	The status of this field could not be identified
0	The value was ok and remains unchanged
1	The syntax/notation has been scaled/standardized
2	The syntax has been corrected
4	The input value was ambiguous und could not be corrected
5	The input value is wrong identified
6	The input values are missing but the output could be otherwise identified/validated
7	The input values and the output could not be otherwise identified
A	The field was too short

So würde beispielsweise das Short Report Value „050-00–“ bedeuten, dass das Land, die Stadt, die Straße und die Hausnummer korrekt waren, dass die PLZ falsch war und dass die restlichen Felder nicht analysiert wurden.

7.3 Auswertung der Ergebnisse

Die Auswertung der Ergebnisse von Tolerant erfolgt, wie bei den Ergebnissen von Uniserv, über eine Excel-Tabelle. Auch hier werden die Daten der Ausgabedatei „out_tolerant.csv“ mit der gleichen Vorgehensweise wie bei den Uniserv-Daten in Excel importiert. Anschließend kann sofort mit der Auswertung begonnen werden. Die Details zu den einzelnen Returncodes wurden bereits im Kapitel „Beschreibung der Returncodes von Tolerant“ erörtert.

Auswertung des „Score Value“ Das „Score Value“ bewertet die Zustellbarkeit einer Sendung an diese Adresse. Für die Auswertung werden die Werte des Attributs „Score Value“ gezählt und nach den Werten gruppiert die auftreten. Daraus folgt die folgende Darstellung:

Score Value	COUNT(*)
5	1577
4	356
3	27
2	100
1	6
Summe	2066

Auswertung des „Short Reports“ für PLZ Der „Short Report“ ist ein 8-Bit-String wobei jedes Bit einen eigenen Attributwert repräsentiert. Die Details dazu sind in Kapitel „Beschreibung der Returncodes von Tolerant“ erörtert worden. Für die Auswertung des PLZ-Feldes, wird das zweite Bit aus dem „Short Report“ extrahiert und gezählt, wie oft ein spezieller Code an dieser Bit-Stelle vorkommt. Somit ergibt sich folgende Auswertung:

Short Report (PLZ)	COUNT(*)
0	1929
2	137
Summe	2066

Auswertung des „Short Reports“ für Stadt Auch hier wird für die Auswertung genauso vorgegangen wie beim „Short Reports“ für PLZ. Jedoch wird nun das dritte Bit des „Short Reports“ extrahiert und ausgewertet, was zu folgendem Ergebnis führt:

Short Report (Stadt)	COUNT(*)
0	1457
1	537
2	72
Summe	2066

Auswertung des „Short Reports“ für Straße Die Auswertung ist auch bei hier die Selbe wie bei den Vorgängern, jedoch wird nun das 5. Bit extrahiert und ausgewertet, was zu folgendem Ergebnis führt:

Short Report (Straße)	COUNT(*)
-	5
0	1526
1	417
2	107
5	5
6	4
7	1
*	1
Summe	2066

Auswertung des „Short Reports“ für Hausnummer Auch hierbei handelt es sich um die gleich Auswertung wie zuvor, nur dass das 6. Bit analysiert wird, was zu folgendem Ergebnis führt:

Short Report (Hausnummer)	COUNT(*)
-	7
0	1839
2	103
6	91
*	26
Summe	2066

8 Evaluation der Ergebnisse und Gegenüberstellung mit der Software von Tolerant

8.1 Gegenüberstellung von Uniserv mit dem Anforderungskatalog

In diesem Unterkapitel erfolgt nun eine Gegenüberstellung des Systems von Uniserv mit der Anforderungsanalyse aus Kapitel 3. Es werden jedoch lediglich diejenigen Anforderungen evaluiert, die auch im Rahmen dieser Arbeit getestet werden konnten. Die folgende Auflistung enthält den Anforderungsnamen und das erzielte Ergebnis von Uniserv. Eine detaillierte Beschreibung der einzelnen Anforderungen ist in Kapitel 3 vorgenommen worden.

Allgemeine Technische Anforderungen

Anforderungsname	Ergebnis der Auswertung
Konnektivität	Dies wurde zwar nicht explizit überprüft, jedoch bietet das DQSH viele Konnektoren zu Datenbanken, Dateiformaten und auch ERP-Systemen. Da diese jedoch nicht genauer untersucht wurden, wird dieser Punkt auch als noch offen angesehen.
Monitoring	Nicht geprüft.
Hosting	Das System von Uniserv kann zentral gehostet werden. Jedoch wurde dies nicht genauer untersucht und somit gilt dieser Punkt als noch offen.
Hardwareanforderungen	Nicht geprüft.
Softwareanforderungen	Für die Installation und den Betrieb von Uniserv unter Windows war keine kostenpflichtige Zusatzsoftware erforderlich. Außerdem unterstützt Uniserv eine Reihe anderer Betriebssysteme, beispielsweise Linux/Unix oder BS200. Die Unterstützung von anderen Betriebssystemen außer Windows wurde jedoch auch nicht untersucht. Weswegen auch dieser Punkt als offen anzusehen ist.
Multi-User-Betrieb	Nicht geprüft.

Allgemeine Funktionale Anforderungen

Anforderungsname	Ergebnis der Auswertung
Returncodes	Uniserv verfügt über technische Returncodes, die den Status der Bearbeitung zurückgeben. Dieser Punkt ist somit voll erfüllt.
Sprachen	In dieser Arbeit wurden zwar nur deutsche Adressdaten bearbeitet, jedoch ist aus den Handbüchern von Uniserv ersichtlich, dass auch viele andere Sprachen wie Englisch, Französisch, Italienisch, Spanisch, Russisch etc. unterstützt werden. Somit ist auch dieser Punkt voll erfüllt.
Standards	Uniserv unterstützt sowohl Unicode, als auch die wichtigsten anderen Standardzeichensätze. Somit ist auch dieser Punkt voll erfüllt.

Allgemeine Nicht-Funktionale Anforderungen

Anforderungsname	Ergebnis der Auswertung
Modi	Uniserv unterstützt sowohl einen Batch-Modus für die Verarbeitung von großen Datenmengen, als auch einen Real-Time-Modus für Just-In-Time-Verarbeitungen. Der Real-Time-Modus unterstützt unter anderem die Module Adressvalidierung, Dublettencheck, Embargocheck, Bankdatenvalidierung, Telefonnummervalidierung und Email-Validierung. Somit ist dieser Punkt voll erfüllt.
Performance	Nicht geprüft.
Sicherheit	Nicht geprüft.

Allgemeine Technische Anforderungen

Anforderungsname	Ergebnis der Auswertung
Konnektivität	Dies wurde zwar nicht explizit überprüft, jedoch bietet das DQSH viele Konnektoren zu Datenbanken, Dateiformaten und auch ERP-Systemen. Da diese jedoch nicht genauer untersucht wurden, wird dieser Punkt auch als noch offen angesehen.
Monitoring	Nicht geprüft.
Hosting	Das System von Uniserv kann zentral gehostet werden. Jedoch wurde dies nicht genauer untersucht und somit gilt dieser Punkt als noch offen.
Hardwareanforderungen	Nicht geprüft.
Softwareanforderungen	Für die Installation und den Betrieb von Uniserv unter Windows war keine kostenpflichtige Zusatzsoftware erforderlich. Außerdem unterstützt Uniserv eine Reihe anderer Betriebssysteme, beispielsweise Linux/Unix oder BS200. Die Unterstützung von anderen Betriebssystemen außer Windows wurde jedoch auch nicht untersucht. Weswegen auch dieser Punkt als offen anzusehen ist.
Multi-User-Betrieb	Nicht geprüft.

Allgemeine Migrationsanforderungen

Anforderungsname	Ergebnis der Auswertung
Datenkompatibilität	Das System unterstützt eine große Anzahl an Standarddatentypen und Datenformate. Im Rahmen dieser Arbeit wurde jedoch lediglich die Verarbeitung von CSV-Dateien und Excel-Tabellen überprüft.
Datentransformation	Uniserv verfügt über Datentransformationsfunktionen und eine Metadatenverwaltung. Es ermöglicht das Mapping von Metadaten und dabei auch die Transformation von Datentypen. Dieser Punkt ist somit voll erfüllt.
Migrationsmöglichkeiten	Nicht geprüft.

Spezielle Anforderungen an die Adressvalidierung

Anforderungsname	Ergebnis der Auswertung
Grundfunktionalität	Die Grundfunktionalitäten der Adressvalidierung werden von Uniserv voll erfüllt. Dies geht aus der Auswertung der Testdaten, als auch aus durchgeführten Tests mit einzelnen Datensätzen hervor.
PLZ-Auflösung	Die Adressvalidierung von Uniserv verfügt über eine PLZ-Auflösung. Auch dies geht aus der Testdatenauswertung hervor. Somit ist dieser Punkt voll erfüllt.
Ort-Auflösung	Die Adressvalidierung von Uniserv verfügt über eine Ort-Auflösung. Auch dies geht aus der Testdatenauswertung hervor. Somit ist dieser Punkt voll erfüllt.
HNR-Auflösung	Die Adressvalidierung von Uniserv verfügt über eine Hausnummerauflösung. Auch dies geht aus der Testdatenauswertung hervor. Auch die Rückgabe eines Hausnummerintervalls, mit möglichen Hausnummern für eine PLZ ist unterstützt. Somit ist dieser Punkt voll erfüllt.
Referenzdaten	Die Referenzdaten der Adressvalidierung sind sowohl für die meisten europäischen Länder vorhanden, als auch für die USA, Russland und China. Somit ist dieser Punkt voll erfüllt.
Returncodes	Das System von Uniserv bietet Returncodes an, mit denen die automatischen Änderungen, die Vorgenommen wurden, angezeigt werden können. Außerdem gibt es Returncodes, die die Adressqualität eines Datensatzes zurückgeben. Auch die Returncodes für die Datenqualität einzelner Attribute sind vorhanden. Somit ist dieser Punkt voll erfüllt.

Allgemein lässt sich sagen, dass Uniserv in keinem der Punkte, die überprüft wurden, Schwächen aufweist. Dies war jedoch auch zu erwarten, da gravierende Mängel in einem der getesteten Punkte wohl auch die Aufnahme von Uniserv in den Gartner Magic Quadrant verhindert hätte.

8.2 Gegenüberstellung der Ergebnisse mit der Software von Tolerant

8.2.1 Gegenüberstellung der Returncodes beider Systeme

Für die Evaluation der Ergebnisse ist es notwendig, die Returncodes beider Systeme so gut wie möglich aufeinander abzubilden. Dies wird durch die nun folgenden Mapping-Tabellen vorgenommen.

Mapping von Score Value und OUT_RES_POST_CL

In der folgenden Tabelle werden die Werte von OUT_RES_POST_CL den Werten des Score Value zugeordnet:

Tolerant	Details	Uniserv	Details
5	All elements are valid	1	sicher gefunden, unverändert
4	Quite reliable adress	2	sicher gefunden, verändert
3	Unreliable adress, but delivery possible	3	unsicher gefunden, verändert, gilt als zustellbar
2	Very unreliable adress	5	nicht gefunden
1	Corrupted adress	5	nicht gefunden
0	Delivery impossible	5	nicht gefunden

Mapping von Short Report Value und OUT_RES_*

Da es sich beim Short Report Value um einen 8-bit String handelt, bei dem jedes Bit einen Adressbestandteil repräsentiert, muss zuerst eine Zuordnung dieser Bits auf die entsprechende Ausgabevariable von Uniserv vorgenommen werden. Hierzu dient das Mapping in der folgenden Tabelle:

Adressbestandteil	Short Report Value Bit	Uniserv
Land	1	NULL
PLZ	2	OUT_RES_ZIP
Stadt	3	OUT_RES_CITY
Stadtteil	4	OUT_RES_CITY_DISTRICT
Strasse	5	OUT_RES_STR
Hausnummer	6	OUT_RES_HNO
Zustelldienst	7	NULL
Organisation	8	OUT_RES_ORGANISATION

In der folgenden Tabelle wird nun der Wert eines Short Report Value Bits auf die dementsprechenden Werte der OUT_RES_-Variablen vorgenommen. Die Werte, die nicht zugeordnet werden können, werden aus Gründen der Übersichtlichkeit weggelassen.

Tolerant	Details	Uniserv	Details
*/-	Field-Status could not be identified / Not analysed	NC/LEER	nicht geprüft
0	The value was ok	OK	Eingabe richtig
1/2	The syntax has been scaled/-corrected	C/DF/SC	Änderung vorgenommen
5	The input value is wrong identified	NF	nicht gefunden
6/7	The input values are missing but the output could (not) be otherwise identified	NI	keine Eingabe

8.2.2 Gegenüberstellung der Returncodes anhand der Mapping-Vorschriften

In diesem Kapitel werden die Returncodes von Tolerant und Uniserv anhand der Mapping-Vorschriften verglichen. Es werden also die ermittelten Ergebnisse aus den Kapiteln 6.5 (Uniserv-Ergebnisse) und 7.3 (Tolerant-Ergebnisse) gegenübergestellt. Die Details zu den einzelnen Werten der Returncodes sind in Kapitel 6.4.2 (Uniserv-Returncodes), Kapitel 7.2.2 (Tolerant-Returncodes) oder zusammengefasst in Kapitel 8.2.1 zu finden.

Gegenüberstellung von Score Value und OUT_RES_POST_CL

Score Value	COUNT(*)	OUT_RES_POST_CL	COUNT(*)
5	1577	1	17
4	356	2	1792
3	27	3	233
2	100	5	(24)
1	6	5	24
Summe	2066	Summe	2066

Auffällig an der Gegenüberstellung der beiden Returncodes ist, dass sich vor allem die ersten beiden Returncodes (5 und 4 bei Score Value bzw. 1 und 2 bei OUT_RES_POST_CL) stark unterscheiden. Dies liegt an der schon erwähnten Großbuchstabenproblematik. Da Uniserv die Änderung von Großbuchstaben in Kleinbuchstaben als eine Änderung wertet, Tolerant dies jedoch nicht macht, unterscheiden sich die Werte innerhalb dieser beiden Returncodes sehr stark. Die 17 Datensätze die von Uniserv mit 1 („sicher gefunden, unverändert“) bewertet werden, sind lediglich die 17 mehrdeutigen Datensätze die gefunden wurden. Bei den Rückgabewerten 2 bzw. 5 sind die 24 Datensätze bei OUT_RES_POST_CL in Klammern gesetzt, da diese für die Summe nicht mitgezählt werden. Dies hat den Grund, dass die Score Values 2 und 1 beide auf den OUT_RES_POST_CL-Wert von 5 gemappt werden, jedoch nur einmal gezählt werden sollen.

Gegenüberstellung von Short Report(PLZ) und OUT_RES_ZIP

Short Report(PLZ)	COUNT(*)	OUT_RES_ZIP	COUNT(*)
0	1929	OK	1958
2	137	C,DF,SC	107
*/-	0	LEER	1
Summe	2066	Summe	2066

Diese beiden Returncodes sind sehr ähnlich. Uniserv hat lediglich 29 PLZ mehr als richtig identifiziert. Dies kann unter anderem an den Referenzdaten bzw. der Referenzdatenaktualität liegen. Ist beispielsweise die neu eingeführte PLZ 33333 bei den Tolerant-Referenzdaten noch nicht vorhanden, so wird diese natürlich auch nicht als OK gewertet. Kommt nun diese PLZ mehrfach in den Testdaten vor, kann es zu einer Differenz von 29 Datensätzen kommen obwohl nur eine PLZ nicht in den Referenzdaten von Tolerant vorhanden ist.

Gegenüberstellung von Short Report(Stadt) und OUT_RES_CITY

Short Report(Stadt)	COUNT(*)	OUT_RES_CITY	COUNT(*)
0	1457	OK	1489
1/2	609	C,SC	576
7	0	NF	1
Summe	2066	Summe	2066

Auch diese Returncodes sind in der Verteilung sehr ähnlich. Der Unterschied von 32 Datensätzen bei den Returncodes 0 und OK ist auf den Unterschied von 29 Datensätzen bei den PLZ-Returncodes zurückzuführen. Somit verbleiben nur noch 3 Datensätze, die von Uniserv als OK gewertet wurden und von Tolerant nicht.

Gegenüberstellung von Short Report(Straße) und OUT_RES_STR

Short Report(Straße)	COUNT(*)	OUT_RES_STR	COUNT(*)
0	1526	OK	1213
1/2	524	C,SC	826
5	5	NF	23
6/7	5	NI	0
*/-	6	NC,LEER	4
Summe	2066	Summe	2066

Der große Unterschied bei den ersten Returncodes liegt unter anderem daran, dass Uniserv die Korrektur eines Leerzeichens am Ende eines Straßennamens als tatsächliche Korrektur ansieht, Tolerant dies jedoch nicht als solche Bewertet. So wird beispielsweise die Straße „HERBERT-KNEITZ-STR.32“ bei der kein Leerzeichen zwischen der Hausnummer und der Straße vorkommt, von Tolerant als 0 (also OK) gekennzeichnet und von Uniserv als korrigiert (C). Die Ausgabe ist jedoch in beiden Systemen die gleiche, nämlich „Herbert-Kneitz-Str. 32“.

Gegenüberstellung von Short Report(HNR) und OUT_RES_HNO

Short Report(HNO)	COUNT(*)	OUT_RES_HNO	COUNT(*)
0	1839	OK	1957
2	103	SC	4
5	0	NF	7
6/7	91	NI	90
*/-	33	NC,LEER	8
Summe	2066	Summe	2066

Bei der Hausnummer nimmt nun Tolerant mehr Syntaxkorrekturen vor, als Uniserv (Returncodes 2 bzw. SC). Der Grund hierfür ist jedoch nicht ganz ersichtlich. Beispielsweise wird die Hausnummer (hier aus den Input-Daten) „AN DER TALLE 27-31“ von Tolerant als 0, also OK, gewertet. Die Hausnummer „DAIMLERSTR. 49-53“ wird jedoch als 2 gewertet. Die Syntax der Adresse wurde also laut Tolerant korrigiert. Bei beiden Beispielen ist genau ein Leerschritt zwischen der Hausnummer und der Straße und jeweils auch kein Leerschritt hinter der Hausnummer. Auch der Punkt bei der „DAIMLERSTR. 49-53“ spielt keine Rolle (wie an anderen Datensätzen zu sehen ist). Die Ausgabe von Tolerant ist „Daimlerstr. 49-53“ bzw. „An der Talle 27-31“ womit sichergestellt wäre, dass auch sonst keine Korrekturen an der Hausnummer vorgenommen wurden. Zu dieser Problematik kann also leider keine Erklärung gegeben werden, jedoch erklärt es den großen Unterschied in den Returncodes 2 bzw. SC.

8.2.3 Gegenüberstellung von Uniserv und Tolerant anhand der Klassifikation der Rückgabedaten

In Kapitel 6.3.2 wurden Ergebnisklassen definiert, in die die Datensätze beider Systeme nun eingeteilt werden sollen. Die Klasse 1 beinhaltet alle Adressdatensätze, die nach der Bearbeitung als zustellbar bewertet werden. Klasse 2 enthält alle Datensätze, die wegen Fehlern in PLZ bzw. Ort nicht zustellbar sind. Klasse 3 enthält alle Datensätze, die wegen Fehlern im Straßennamen oder der Hausnummer nicht zustellbar sind. Klasse 4 enthält alle Datensätze, die sowohl die Kriterien für Klasse 2 als auch für Klasse 3 erfüllen. Im Unterschied zur Klassifikation in Kapitel 6.3.2, werden mehrdeutige Orte bzw. mehrdeutige Straßen nicht in die Ergebnisklassen 2 bzw. 3 eingeordnet, sondern in die Klasse 1. Dies wird gemacht, da bei der Auswertung der Testdaten die Variante gewählt wurde, immer den ersten Rückgabedatensatz einer mehrdeutigen Adresse auszuwählen. Dieser Rückgabedatensatz ist dann natürlich korrekt und somit in Klasse 1 einzustufen. In den folgenden Abschnitten wird nun erklärt, nach welchem Schema die Ergebnisdaten den Klassen zugeordnet werden.

Klasse 1 Klasse 1 ist im Gegensatz zu den anderen Klassen relativ leicht zu ermitteln. Inhalt der Klasse sollen alle zustellbaren Adressdatensätze sein. Diese bekommt man in Uniserv, indem man alle Datensätze zählt die bei OUT_RES_POST_CL die Werte 1 (sicher gefunden, unverändert), 2 (sicher gefunden, verändert) oder 3 (unsicher gefunden, verändert, gilt als zustellbar) besitzen. Diese Rückgabewerte werden nur vergeben wenn die Adresse zustellbar ist. Bei Tolerant ist es die selbe Vorgehensweise, nur dass jeder Datensatz gezählt wird, für den das Score Value 5 (all elements are valid), 4 (quite reliable adress) oder 3(unreliable adress, but delivery possible) ist. Das Ergebnis dieser Auswertung ist, dass Uniserv 2042

Klasse 1 Datensätze und Tolerant 1960 Klasse 1 Datensätze liefert.

Klasse 2 Für die Klassifizierung der Uniserv-Ausgabedatensätze in Klasse 2 werden alle Datensätze gesucht, deren OUT_RES_POST_CL-Werte nicht 1, 2 oder 3 sind. Aus dieser Menge werden dann die Datensätze gezählt, die bei OUT_RES_CITY oder bei OUT_RES_ZIP oder bei beiden die Werte „NC“, „NF“, „NI“ oder „LEER“ besitzen. Diese Menge enthält nun aber auch die Datensätze, die auch in Klasse 4 vorkommen, da nicht geprüft wurde, welche Werte OUT_RES_STR und OUT_RES_HNO besitzen. Also werden von der ermittelten Menge noch die Datensätze abgezogen, die bei OUT_RES_STR und OUT_RES_HNO nicht die Werte „NC“, „NF“, „NI“ oder „LEER“ besitzen. Nach der Auswertung ergibt sich, dass die Klasse 2 bei Uniserv 9 Datensätze enthält, die aber alle in Klasse 4 eingestuft werden. Somit ergibt sich für die Klasse 2 ein Wert von 0 Datensätzen.
Bei Tolerant ist die Ermittlung der Klasse 2 sehr einfach, da in den Short Reports von PLZ und Stadt nur die Werte 0 (the value was ok), 1(the syntax has been scaled) und 2 (the syntax has been corrected) vorkommen. Es wurde also bei keinem Datensatz eine Wert für PLZ und Stadt ermittelt, der eine Einordnung in die Klasse 2 rechtfertigen würde. Somit werden auch bei Tolerant 0 Datensätze der Klasse 2 zugeordnet.

Klasse 3 Die Verarbeitung der Datensätze für die Einstufung in Klasse 3 erfolgt bei den Uniserv-Daten analog zu der Einstufung von Klasse 2. Für die Klassifizierung der Uniserv-Ausgabedatensätze in Klasse 3 werden alle Datensätze gesucht, deren OUT_RES_POST_CL-Werte nicht 1, 2 oder 3 sind. Aus dieser Menge werden dann die Datensätze gezählt, die bei OUT_RES_STR oder bei OUT_RES_HNO oder bei beiden die Werte „NC“, „NF“, „NI“ oder „LEER“ besitzen. Diese Menge enthält nun aber auch die Datensätze, die auch in Klasse 4 vorkommen, da nicht geprüft wurde, welche Werte OUT_RES_ZIP und OUT_RES_CITY besitzen. Also werden von der ermittelten Menge noch die Datensätze abgezogen, die bei OUT_RES_ZIP und OUT_RES_CITY nicht die Werte „NC“, „NF“, „NI“ oder „LEER“ besitzen. Nach der Auswertung ergibt sich, dass die Klasse 2 bei Uniserv 24 Datensätze enthält, wovon aber einige noch in Klasse 4 eingestuft werden. Somit ergibt sich für die Klasse 2 ein Wert von 15 Datensätzen.
Bei Tolerant ist die Ermittlung der Klasse 3 sehr einfach, da in Klasse 2 keine Datensätze gefunden wurden, muss die Klasse 4 auch 0 Elemente enthalten, da diese die Schnittmenge aus Klasse 2 und Klasse 3 darstellt. Somit entfallen auf die Klasse 3 alle restlichen Datensätze, die nicht in Klasse 1, Klasse 2 oder Klasse 4 sind. Somit ergibt sich, dass 106 Datensätze in Klasse 3 einzuordnen sind.

Klasse 4 Klasse 4 besitzt bei Tolerant, wie eben gezeigt wurde 0 Elemente. Bei Uniserv werden alle Datensätze gesucht, deren OUT_RES_POST_CL-Werte nicht 1, 2 oder 3 ist und die die Werte „NC“, „NF“, „NI“ oder „LEER“ bei OUT_RES_STR oder OUT_RES_HNO (oder beides) haben. Geschnitten wird diese Menge mit der Menge aller Datensätze, deren OUT_RES_POST_CL-Werte nicht 1, 2 oder 3 ist und die die Werte „NC“, „NF“, „NI“ oder „LEER“ bei OUT_RES_CITY oder OUT_RES_ZIP (oder beides) haben. Ist dies Ausgeführt, erhält man bei den Uniserv-Daten 9 Datensätze in der Klasse 4.

Übersicht über alle Klassen

	Uniserv	Tolerant
Klasse 1	2042	1960
Klasse 2	0	0
Klasse 3	15	106
Klasse 4	9	0

Die Übersicht zeigt, dass Uniserv von den 2066 Datensätzen, die getestet wurden 2042 als zustellbar erkennt oder die Datensätze so korrigieren kann, dass eine Zustellung möglich ist. Tolerant schafft dies nur bei 1960 Datensätzen. An den Werten für Klasse 2 ist zu erkennen, dass Tolerant bei keinem Datensatz einen nicht korrigierbaren Fehler bei PLZ und Ort gefunden hat. Bei Uniserv sind es immerhin 9 Datensätze, die unter anderem wegen Fehlern in PLZ und Ort nicht korrigiert werden konnten. Dies ergibt sich daraus, dass bei Uniserv 9 Datensätze in Klasse 4 vorhanden sind. Da die Klasse 4 die Schnittmenge aus Klasse 3 und Klasse 2 ist, bedeutet dies, dass 9 Datensätze von Uniserv noch der Klasse 2 zuzuordnen sind. Die Klasse 3 umfasst die fehlerhaften Datensätze hinsichtlich ihrer Straße und Hausnummer. Hier scheint Tolerant erheblich fehleranfälliger zu sein als Uniserv. Dies ist ein Hinweis auf mangelnde Aktualität oder schlechtere Qualität der Referenzdaten. Insgesamt ergibt die Auswertung ein besseres Ergebnis für Uniserv als für Tolerant. Dies resultiert aus dem Grund, dass Uniserv bei 2066 Datensätzen 86 Datensätze mehr in den Status „Zustellbar“ transformieren kann, als Tolerant. Uniserv ist also in der Lage 4,16% der Datensätze besser zu verwerten als Tolerant, was sich bei einer größeren Datenmenge durchaus bemerkbar machen würde. In Abbildung 8.1 ist das Ergebnis der Evaluation noch einmal grafisch dargestellt.

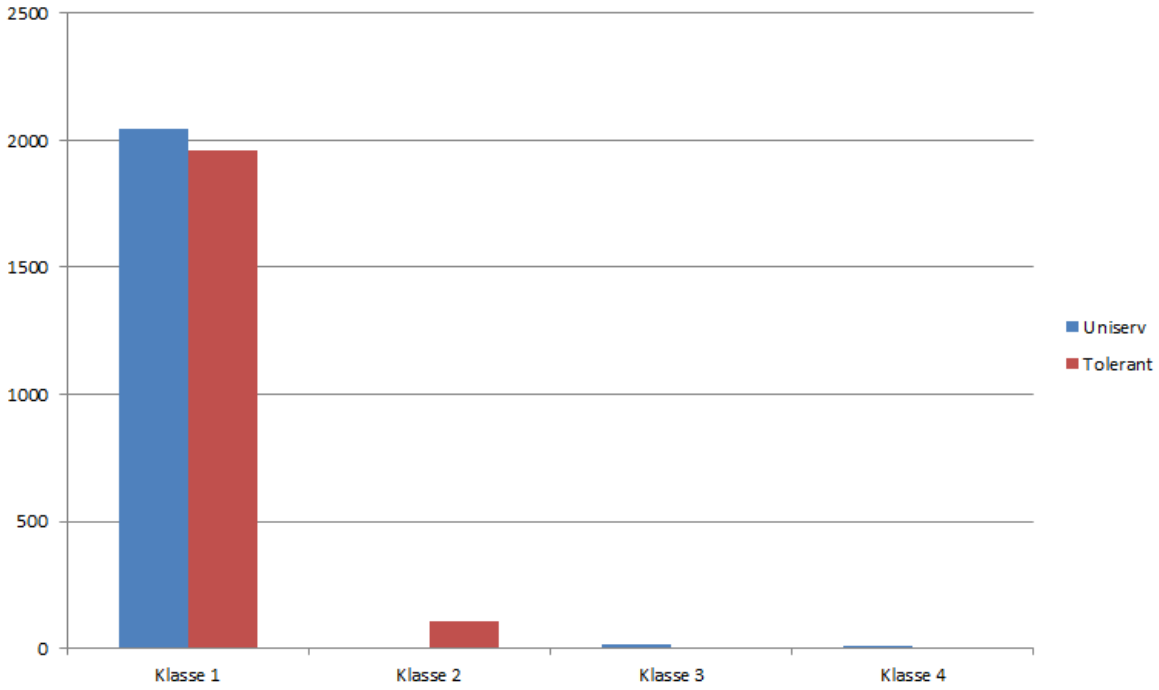

Abbildung 8.1: Auswertung der Testdaten anhand der Ergebnisklassen

9 Zusammenfassung und Ausblick

9.1 Zusammenfassung

In dieser Arbeit wurde die Leistungsfähigkeit des Datenqualitätsmanagements (DQMS) von Uniserv evaluiert. Dies beinhaltet, neben der theoretischen Betrachtung der Thematik, die Erstellung eines Anforderungsprofils an ein DQMS, anhand dessen die Leistungsfähigkeit des Systems beurteilt werden kann. Neben den allgemeinen Anforderungen an ein DQMS, behandelt diese Arbeit vor allem den Teilaspekt der Adressvalidierung. Diese wurde in einem Testsystem implementiert um anhand der Auswertung von Testdatensätzen einen quantitativen Vergleich mit dem Third Party Tool aus dem Hause Tolerant zu erstellen. Der Vergleich mit der Adressvalidierung von Tolerant wurde gewählt, da diese Arbeit in Kooperation mit der Firma iC Consult GmbH erstellt wurde, die bei einem Kunden die Adressvalidierung von Tolerant betreibt. Um die Adressvalidierungen von Tolerant und Uniserv vergleichen zu können, mussten beide Systeme in einem ersten Schritt installiert und konfiguriert werden. Außerdem musste eine Klassifizierung der Testdaten und ein Mapping der Returncodes beider Systeme vorgenommen werden, um eine möglichst hohe Vergleichbarkeit zu erreichen. Schließlich wurden die Tests durchgeführt und die Ergebnisse beider Systeme ausgewertet und gegenübergestellt. Zusammenfassend ist zu bemerken, dass unter den Testbedingungen das System von Uniserv ein, um ca. 4% besseres Ergebnis erzielt als Tolerant, gemessen an der Anzahl der Adressen, die als „zustellbar“ gewertet werden. Dies spricht für eine Empfehlung der Adressvalidierung von Uniserv. Jedoch muss auch bemerkt werden, dass die Tests mit der Standardkonfiguration beider Systeme durchgeführt wurden. Dies bedeutet, dass sowohl bei Uniserv, als auch bei Tolerant eine Reihe von Konfigurationen möglich sind, die das Ergebnis erheblich beeinflussen können. So würde beispielsweise eine Veränderung an der Variable, die bewertet ob eine Adresse als „zustellbar“ gilt zur Folge haben, dass mehr oder weniger Datensätze als „zustellbar“ bewertet werden. Trotzdem bieten die Ergebnisse dieser Arbeit einen guten Überblick über die starke Leistungsfähigkeit der Adressvalidierung bzw. des gesamten DQMS von Uniserv. Dies resultiert zum einen aus dem Vergleich mit Tolerant, den Uniserv für sich entscheiden konnte, zum anderen aber auch aus der Erfüllung aller Anforderungen, die im Rahmen dieser Arbeit an eine Adressvalidierung gestellt wurden. Auch die generellen Anforderungen an ein DQMS, die bewertet werden konnten, wurden von Uniserv erfüllt. Darunter fallen unter anderem funktionale Anforderungen, wie die Unterstützung mehrerer Sprachen, die analysiert werden können, oder die Kompatibilität mit diversen Standards, z.B. Unicode. Aber auch eine hohe Konnektivität und Datenkompatibilität oder Funktionen zur Datentransformation werden von Uniserv unterstützt. Um jedoch ein noch genaueres Bild von der Leistungsfähigkeit des DQMS von Uniserv zu bekommen, sind noch einige Punkte offen, die im folgenden Kapitel kurz angesprochen werden.

9.2 Ausblick

Aufgrund der hohen Komplexität eines DQMS, konnte im Rahmen dieser Arbeit lediglich der Teilbereich der Adressvalidierung genauere untersucht werden. Jedoch gibt es neben der Adressvalidierung noch einige interessante Bereiche eines DQMS. Darunter fallen z.B. der Dublettencheck, der für die Erkennung von mehrfach vorhandenen Datensätzen eingesetzt wird, die Embargoprüfung, die für die Erkennung staatlich sanktionierter Personen/-Organisationen im Datenbestand benutzt wird, oder die Bankdatenvalidierung, die eine algorithmische Überprüfung von Kontonummern, Bankleitzahlen oder Kreditkartennummern ermöglicht. Zusätzlich könnten auch noch die Telefondatenanalyse oder die Email-Analyse aber auch nicht-personenbezogene Analysen wie die Finanzdatenanalyse untersucht werden. Auch technischere Fragestellungen, wie beispielsweise die Datenmigration via Uniserv oder die Performance des Systems bei der Analyse großer Datenmengen im Batch-Modus könnten untersucht werden. Darüber hinaus beinhaltet auch die Adressvalidierung an sich noch offene Punkte. So könnten beispielsweise Adressen aus anderen europäischen Ländern, oder wichtigen außereuropäische Länder wie den USA, Russland oder China getestet werden. Besonders die letzten beiden Länder stellen schon alleine wegen der unterschiedlichen Buchstaben/Zeichen eine Herausforderung an eine Adressvalidierung dar.

Abbildungsverzeichnis

Literaturverzeichnis

[ABEM10] Apel, Detlef, Wolfgang Behme, Rüdiger Eberlein und Christian Merighi: *Datenqualität erfolgreich steuern - Praxislösungen für Business-Intelligence-Projekte*. Carl Hanser Verlag, München, 2010.

[Buna] Bundesbank, Deutsche: *Prüfzifferberechnung*. `http://www.bundesbank.de/Navigation/DE/Kerngeschaeftsfelder/Unbarer_Zahlungsverkehr/Pruefzifferberechnung/pruefzifferberechnung.html`.

[Bunb] Bundesbank, Deutsche: *Prüfzifferberechnungsmethoden zur Prüfung von Kontonummern auf ihre Richtigkeit*. `http://www.bundesbank.de/Redaktion/DE/Downloads/Kerngeschaeftsfelder/Unbarer_Zahlungsverkehr/pruefzifferberechnungsmethoden.pdf`.

[Ech02] Echerson, Wayne W.: *Data Warehousing Special Report: Data quality and the bottom line*. Technischer Bericht, 101communicationsLLC, 2002.

[Ech09] Echerson, Wayne W.: *Who Ensures Clean, Consistent Data?* Technischer Bericht, tdwi, 2009.

[FB11] Friedman, Ted und Andreas Bitterer: *Magic Quadrant for Data Quality Tools*. Technischer Bericht, Gartner, 2011.

[Tol] Tolerant: *Produkte*. `http://www.tolerant-software.de`.

[Unia] Uniserv: *Data Integration Suite*. `http://www.uniserv.com/de/products/data-integration-services.php`.

[Unib] Uniserv: *Data Quality Batch Suite*. `http://www.uniserv.com/de/products/data-quality-solutions/data-cleansing-data-quality.php`.

[Unic] Uniserv: *Data Quality Explorer*. `http://www.uniserv.com/de/products/data-quality-solutions/data-profiling-data-quality.php`.

[Unid] Uniserv: *Data Quality Monitor*. `http://www.uniserv.com/de/products/data-quality-solutions/data-monitoring-data-quality.php`.

[Unie] Uniserv: *Data Quality Real-Time Services*. `http://www.uniserv.com/de/products/data-quality-solutions/real-time-data-quality.php`.

[Unif] Uniserv: *Der High Availability License Service (HAL)*.

[Uni94a] Uniserv: *bank - Online Komponente (Release 1.6)*, 1994.

[Uni94b] Uniserv: *post - Online Komponente (Release 2.0)*, 1994.

[Uni10] UNISERV: *embargoCheck - Online Komponente (Release 1.1)*, 2010.

[Uni11a] UNISERV: *Uniserv License Service - Installer (Release 1.1)*, 2011.

[Uni11b] UNISERV: *Uniserv Software Suite - Installer (Release 1.1)*, 2011.