

Aus dem Bereich Klinische Medizin
der Medizinischen Fakultät
der Universität des Saarlandes, Homburg/Saar

**Gewinnung und Analyse von
Patientendaten aus der Datenbank eines
Krankenhausinformationssystems**

**Dissertation zur Erlangung des Grades eines Doktors der
Theoretischen Medizin der Medizinischen Fakultät
der UNIVERSITÄT DES SAARLANDES**
2015

vorgelegt von: Andreas Bock
geb. am: 23.09.1977 in Saarbrücken

Zusammenfassung. Es werden elektronische Daten erzeugt, sobald ein Mensch ein Krankenhaus oder eine Arztpraxis als Patient besucht. Neben organisatorischen Daten werden auch medizinisch relevante Patientendaten erhoben und entsprechend in einem Krankenhausinformationssystem gespeichert. Die medizinischen Daten eines Patienten werden in erster Linie für dessen Behandlung relevant sein, aber es besteht darin noch weiteres Potential. Die Betrachtung und der Vergleich mehrerer Patientendaten können zu einem zusätzlichen Erkenntnisgewinn führen. Theoretisch sind in diesem Moment bereits für Studienzwecke nutzbare medizinische Daten vorhanden. Es stellt sich die Frage, ob der Zugriff auf existierende Daten, jenseits der vorgegebenen Funktionalitäten, eines Krankenhausinformationssystems möglich ist und wie man diesen konkret umsetzen kann.

Diese Arbeit behandelt die Gewinnung und Analyse von alphanumerischen Patientendaten eines Krankenhausinformationssystems des Universitätsklinikums in Homburg. Dabei werden theoretische und praktische Belange im Rahmen der Softwareentwicklung innerhalb des Systems behandelt. Es wird gezeigt, dass die Gewinnung von lokalen Patientendaten mit verfügbaren Mitteln - also ohne weitreichende Nutzung zusätzlicher Software - möglich ist. Zu einem Szenario geforderter Daten wird eine Strategie (e.g., Auswahlkriterium) entwickelt, wie man auf die gewünschte Patientenmenge und die benötigten Daten (e.g., Diagnosen) zugreifen kann.

Zusätzlich zur Bildschirmausgabe im Krankenhausinformationssystem wird für den Export eine CSV-Datei verwendet. Diese dient als Schnittstelle für externe Systeme. Es wird zudem gezeigt, dass man diese Datei zur Vorbereitung externer Analysen weiterverarbeiten kann.

Collecting and Analyzing Patient Data from the Database of a Hospital Information System

Abstract. Every time someone visits a hospital or a doctor's office, electronic data is generated. In addition to organizational data, medically relevant data is collected and saved in a hospital information system. The patient's medical data are first and foremost relevant for their treatment, but it has additional potential. Comparing the data of multiple patients can generate additional insight. In theory, data usable in research setting is already present at this stage. Accessing this existing data may be possible in hospital information systems; the problem consists of the actual practice of organizing such access.

This thesis discusses the possibility of collecting and analyzing patient data from the hospital information system of the Homburg Teaching Hospital, emphasizing both theoretical and practical concerns. The thesis concludes that such collection is possible within the existing system, without major additional software, is possible. I develop a strategy (e. g., rules of selection) for a set of data which makes it possible to access the required number of patients and the necessary data (e. g., diagnoses).

In addition to displaying these data in the hospital information system, the export routine utilizes a CSV file to export them, making it possible to connect to external systems. I will also show that files generated in this manner can be used to prepare external analyses.

Inhaltsverzeichnis

Abkürzungsverzeichnis	11
I. Einleitung	13
1. Motivation	15
1.1. Aufbau dieser Arbeit	17
1.2. Elektronische Patientendaten	19
1.2.1. Datenmodelle	19
1.2.2. Einordnung der Daten	19
1.2.2.1. Harte Patientendaten	21
1.2.2.2. Weiche Patientendaten	21
2. Studien	23
2.1. Organisation der Patientendaten	23
2.2. Verarbeitungswege	24
2.2.1. Papierbasierte Patientendaten	24
2.2.2. Elektronische Patientendaten	25
2.2.3. Vergleich der Vorgehensweisen	25
3. Krankenhausinformationssysteme	27
3.1. Beschreibung	27
3.2. Datenbanksysteme	29
3.3. Datenbankmanipulationssprache	29
3.4. Handhabungsfehler	30
3.5. Universitätsklinikum in Homburg	31
3.5.1. Programmiersprache	32
3.5.2. Subsysteme	32

3.6. Klinik für Neurologie im Krankenhaus Püttlingen	33
3.6.1. Motivation	33
3.6.2. Beschreibung	34
3.6.2.1. Beginn und Einführung	34
3.6.2.2. Dokumentation mit Textbausteinen	34
3.6.2.3. Weitere Eigenschaften	35
3.6.2.4. Studien	35
II. Methoden zur Gewinnung und Analyse von Patientendaten	37
4. Ausgangslage am Universitätsklinikum Homburg	39
4.1. Mikrobiologieszenario in eureka	39
4.2. Grundlage zur Datengewinnung aus dem SAP-System	44
4.2.1. Theoretische Vorarbeit	44
4.2.2. Praktische Umsetzung	47
5. Erweiterte Datengewinnung für das Mikrobiologieszenario	55
5.1. Informationen zur Diagnose	58
5.1.1. Hauptdiagnose	59
5.1.2. Aufnahme- und Entlassungsdiagnose	61
5.1.3. Text zu International Classification of Diseases	61
5.2. Stammdaten zum Patienten	63
5.3. Patientenbewegungen	63
5.4. Informationen zur Medikation	66
5.5. Andere Daten bzw. Informationen	70
5.6. Ergebnisausgabe	70
5.6.1. Verarbeitung der internen Tabelle	70
5.6.2. Darstellung im SAP List Viewer	73
5.6.3. Ausgabe als Datei auf dem Präsentationsserver	74
5.6.4. Ausgabe als Datei auf dem Applikationsserver	74
5.6.5. Beschreibung der Ausgabe	77
5.6.6. Vorteile der jeweiligen Ausgabe	79
5.7. Externe Datensätze	79
5.8. Datentransport	83

5.8.1. Bereitstellung auf dem Applikationsserver	84
5.8.2. Bereitstellung auf dem Hybaseserver	84
5.8.3. Abruf der Daten	84
5.9. Manuelle Abfrage in SAP	85
6. Datenschutz	87
6.1. Personenbezogene Daten im Mikrobiologieszenario	87
6.2. Maßnahmen während der Erhebung und Verarbeitung	88
6.3. Praktische Umsetzung	89
6.3.1. Patientendaten aus SAP	89
6.3.2. Patientendaten aus der Mikrobiologie	91
7. Weiterverarbeitung der gewonnenen Daten	95
7.1. Inhaltliche Vorarbeit	95
7.2. Praktische Umsetzung	97
7.2.1. Zugriff auf einzelne Werte	97
7.2.2. Zugriff auf die Medikation	101
8. Analyse der gewonnenen Daten	105
8.1. Innerhalb von SAP	105
8.1.1. Tabellenausgabe	105
8.1.2. Präsentationsgrafik	106
8.2. Daten auf dem Präsentationsserver	108
8.3. Daten auf dem Applikationsserver	112
9. Übertragbarkeit	113
 III. Ergebnisse	 115
 IV. Diskussion	 123
 Literaturverzeichnis	 129
 Abbildungsverzeichnis	 139

Tabellenverzeichnis	143
Programm-Listings	145
V. Anhang	147
A. Programmcode aus Micro11	149
B. Programmcode aus Graphstation	177
C. Programmcode aus Labanom	183
D. Programmcode aus Saptransfer	191
E. Wertezuordnung zur Clinical Document Architecture	207
E.1. Krankenhausinformationssystem	207
E.2. Antibiotika	211

Abkürzungsverzeichnis

ABAP	Advanced Business Application Programming
ALV	ABAP List Viewer SAP List Viewer
ATC	Anatomical Therapeutic Chemical
C21	SAP Produktivsystem am UKS Homburg
CDA	Clinical Document Architecture
CRF	Case Report Form
CSV	Comma Separated Values
CTMS	Clinical Trial Management System
DBMS	Datenbankmanagementsystem
DBVS	Datenbankverwaltungssystem
DML	Datenbankmanipulationssprache
DWH	Data Warehouse
ETL	Extract, Transform, Load
eureca	Enabling information re-use by linking clinical Research and Care
GCP	Good Clinical Practice
HL7	Health Level 7
HoWoS	Hospital Workflow System

Abkürzungsverzeichnis

ICD	International Classification of Diseases
IS-H	Industry Solutions Healthcare
KIS	Krankenhausinformationssystem
KK-05	Pädiatrische Onkologie und Hämatologie am UKS
MIME	Multi-Purpose Internet Mail Extensions
NFS	Network File System
PACS	Picture Archiving and Communication System
SAE	Serious Adverse Event
SQL	Structured Query Language
T21	SAP Testsystem am UKS Homburg
UdS	Universität des Saarlandes
UKS	Universitätsklinikum des Saarlandes
ZIK	Zentrum für Informations- und Kommunikationstechnik

Teil I.

Einleitung

1. Motivation

Im Zeitalter von *Big Data* werden im Gesundheitswesen mehr und mehr Daten erhoben [30]. Dies ist auch der Fall, sobald eine Person ein Krankenhaus oder eine Arztpraxis als Patient besucht [35]. Die Gründe für diese Datenerhebung sind vielfältig. Darunter fällt beispielsweise die Dokumentation der Leistungen und die Zuordnung der Krankenkasse. Einige Daten werden zum organisatorischen Ablauf benötigt, wozu schon der namentliche Aufruf des Patienten gehört. Zusätzlich werden noch medizinisch relevante Patientendaten erhoben und entsprechend hinterlegt; jeder wird die klassische Karteikarte in einer Arztpraxis kennen [76]. Zu den medizinisch relevanten Daten gehören bereits Informationen wie das Geschlecht oder das Alter. Die erhobenen Daten eines einzelnen Patienten werden in erster Linie für dessen Behandlung relevant sein, aber es besteht darin noch weiteres Potential. Die Betrachtung und der Vergleich mehrerer Patientendaten, beispielsweise auf einer Station oder innerhalb einer medizinischen Studie, können zu einem zusätzlichen Erkenntnisgewinn führen [71].

Nach [28] kann auch die Pflege am Patienten verbessert oder erleichtert werden, beispielsweise durch deutliche Kenntlichmachung von Risikofaktoren (e.g., Allergien, Ansteckungsgefahr) bei Patienten in der Stationsansicht eines Krankenhausinformationssystems (KIS), siehe Abbildung 1.1 auf der nächsten Seite und Abbildung 1.2 auf der nächsten Seite. Es ergeben sich durch die elektronische Datenverarbeitung auch Vorteile in der Abrechnung, es sei dazu als Einstieg auf [23] verwiesen.

Die weitere informationstechnische Nutzung bereits vorhandener elektronischer Patientendaten erscheint verheißungsvoll, besonders falls dadurch eine zeitintensive Neueingabe vermieden werden kann. Interessante Daten für Studienzwecke (siehe Abschnitt 2.1 auf Seite 23) können bereits erfasst sein, sie müssten nach dem Extrahieren lediglich ergänzt werden. Eine sich wiederholende manuelle schriftliche Übertragung großer Datenmengen entfällt dadurch. Es stellt sich die Frage, ob der Zugriff auf vorhandene medizinische Daten, jenseits der vorgegebenen Funktionalitäten eines Krankenhausinformationssystems, möglich ist und wie man diesen konkret umsetzen kann.

1. Motivation

Belegungen KK-05 vom 14.10.2014 13:12 mit 10 Patienten

Zim...	Bett	Gsp.	Patientenname	A	G	Geburtsdatum	PP	BA	BKat.	A	R	A	D	letzte WD	K	Wunddoku	Pfad	Berufsgruppen (Pfad)	Freitext Diagnose	...
350	350 A		Einstein Albert	35	M	14.03.1979	P	A	1B-ML				D							
350	350 B			0																
350	350 C			0																
351	351 A		Thomas Testmann	0	M	02.01.2014		V	KS										Lärmschädigungen des Innenohres	
351	351 B		Utopie Max	52	M	01.01.1962		V	NP					K						
351	351 C		April Scherz	58	M	11.11.1955		A	NP				D						Akute myeloblastische Leukämie (AML); Oh	
354	354 A			0																
354	354 B		Kanns Achim	20	M	04.02.1994		A	NP				D							
355	355 A			0																
355	355 B		Maria Kopf	7	W	07.07.2007		A	NP				A	D					Lungenentzündung	
356	356 A		Mai Anfang	2	U	11.11.2011		P	A	2B-ML			D							
357	357 A			0																
358	358 A		Netzer Günther	8	M	14.12.2005		A	NP				A	D					Torflaute	
358	358 B		Müller Kerstin	15	W	01.01.1999		V	NP				A	D					Struma	
359	359 A		PA7 Export	6	W	08.08.2008		A	NOT				D							
359	359 B			0																

Abbildung 1.1.: Ausschnitt eines Screenshots mit fiktiven Patientendaten aus dem Testsystem von Homburg in der Belegungsübersicht einer fiktiven Station. Das Dreieck mit Ausrufezeichen weist auf besondere Ansteckungsgefahr hin, während das darunterliegende blaue Symbol auf andere Risikofaktoren hinweist. Das Klicken auf dieses blaue Symbol führt zur Abbildung 1.2.

Risikoinformationen pflegen: Risikofaktoren

Name: Müller, Kerstin Geschl.: W Einr.: UKS Homburg

Geb.: 01.01.1999

Patient: 10001402 1

Risikofaktoren

M	Rsf	Bezeichnung	Bemerkung	AnlegeDat.	Angelegt von
<input type="checkbox"/>	0	Sonstiges	hasst Mozart	09.11.2010	SIMON_AR
<input type="checkbox"/>	9	Penicillin	reagiert mit Quaddeln	04.08.2004	KK-05
<input type="checkbox"/>	19	Antikoagula.	schwankende Quickwerte	09.11.2010	SIMON_AR

Abbildung 1.2.: Ausschnitt eines Screenshots aus dem Testsystem von Homburg. Dies ist eine detaillierte Ansicht für die Risikofaktoren eines fiktiven Patienten aus Abbildung 1.1.

Diese Arbeit innerhalb der Universität des Saarlandes¹ (UdS) beschäftigt sich in erster Linie mit der Gewinnung von alphanumerischen [11] Patientendaten aus dem SAP-basierten Teil des Krankenhausinformationssystems des Universitätsklinikums des Saarlandes (UKS) in Homburg² und betrachtet sich dabei ergebende Fragestellungen im Rahmen der Softwareentwicklung. Es soll dadurch auch gezeigt werden, dass die Gewinnung von lokalen Patientendaten mit vorhandenen Mitteln - also ohne weitreichende zusätzliche Nutzung anderer Software - möglich ist. Die Erklärung und Vorgehensweise im Zusammenhang mit Programmcode orientiere ich an einem besseren Verständnis für den Leser.

Bevor man die Patientendaten gewinnen kann, muss ein vorhandenes KIS zuerst untersucht werden. Wie ist es zusammengesetzt bzw. was wird im Einzelnen verwendet (e.g., relationale Datenbank [44])? Danach muss eine Strategie (e.g., Auswahlkriterium) entwickelt werden, wie man auf die gewünschte Patientenmenge (e.g., stationäre Patienten einer Station) und die benötigten Daten (e.g., Diagnosen, Medikation) zugreifen kann. Ich realisiere das in dieser Arbeit durch Programmierung innerhalb des KIS. Neben der Bildschirmausgabe wird als Schnittstelle für den Export in dieser Arbeit eine CSV³-Datei [16] verwendet, da diese Option in Krankenhausinformationssystemen weit verbreitet ist [60]. Die Kommunikationsstruktur wird dabei unter Verwendung vorhandener Mittel aufgestellt. Der Weiterverarbeitung und den Möglichkeiten zur Analyse dieser Daten werden ebenfalls Platz in dieser Arbeit eingeräumt. Dabei kann man sowohl direkt auf einem lokalen PC arbeiten, als auch die Dateien für externe Analysen (e.g., Nutzung eines Data Warehouse [68]) vorbereiten (e.g., Umwandlung in andere Formate).

1.1. Aufbau dieser Arbeit

Im weiteren Verlauf dieses Kapitels gehe ich in Abschnitt 1.2 auf Seite 19 weiter auf alphanumerische Patientendaten innerhalb eines KIS ein. Das Kapitel 2 auf Seite 23 behandelt medizinische Studien. In Kapitel 3 auf Seite 27 werden Krankenhausinformationssysteme beschrieben und ich gehe dort auf das lokale System in Homburg und ein externes Beispiel in Püttlingen ein. In Teil II auf Seite 37 befasse ich mich mit der Herangehensweise zur Datengewinnung im Rahmen dieser Doktorarbeit. Mit der Grundlage zur Gewinnung

¹<http://www.uni-saarland.de/>

²<http://www.uniklinikum-saarland.de>

³Comma-Separated-Values

von Patientendaten aus dem SAP-System des UKS beschäftige ich mich in Kapitel 4 auf Seite 39. Anschließend werden in Kapitel 5 auf Seite 55 weitere Details zur Realisierung des Datenzugriffs und -exports ausgearbeitet. Ich gehe in Kapitel 6 auf Seite 87 auf den Datenschutz ein und zeige eine Lösung innerhalb der obigen Datengewinnung. In Kapitel 7 auf Seite 95 und Kapitel 8 auf Seite 105 wird die Weiterverarbeitung und Analyse der gewonnenen Daten betrachtet. Der Übertragbarkeit des Ansatzes dieser Arbeit auf andere Krankenhäuser wird in Kapitel 9 auf Seite 113 Platz eingeräumt. Die Ergebnisse werden in Teil III auf Seite 115 zusammengefasst und anschließend in Teil IV auf Seite 123 diskutiert.

Genutzte Formate und Standards

Die Arbeit wurde unter Verwendung von \LaTeX [77] erstellt. Die verwendeten Abbildungsformate sind JPEG, PNG [89] und PDF [63]. Die Gewinnung der Daten aus dem SAP-System [59] des UKS Homburg habe ich in der Programmiersprache ABAP⁴ [46, 79] realisiert, die Weiterverarbeitung erfolgt in JAVA [4]. Die Verschiebung der Dateien innerhalb der Kommunikationsstruktur ist in Teilen mit Skriptsprachen (i.e., DOS, BASH) implementiert [73].

Die Bezeichner für vorhandene Tabellen im SAP-System des UKS stammen aus diesem System selbst. Die hier gezeigten Daten aus dem SAP-System beziehen sich immer auf das Testsystem T21 des UKS Homburg, falls nicht anders gekennzeichnet. Es werden keine wirklichen Patientendaten in dieser Arbeit gezeigt. Es handelt sich in der Regel um Fantasieeingaben, selbst im Falle einer späteren Pseudonymisierung [78]. Es sind einige Abbildungen zu sehen, welche Screenshots aus dem SAP Diktionär [90] des T21 darstellen, um die Inhalte von Tabellen schnell zu verdeutlichen. Die Daten aus diesem SAP-System werden als CSV-Dateien exportiert, welche in Ausschnitten gezeigt werden. Zur Demonstration der lokalen Betrachtung der CSV-Dateien wird in Kapitel 8 auf Seite 105 LIBREOFFICE [86] verwendet. Eine aus SAP exportierte Tabellenkalkulation in MIME⁵ HTML wird in einem Screenshot aus MICROSOFT EXCEL gezeigt. Die in meinem Programm GRAPHSTATION erzeugten Diagramme in SAP wurden als Postscript [63] gespeichert und als PDF in die Arbeit importiert. Eine weitere Nutzung von Programmen oder Standards wird an den entsprechenden Stellen genannt.

⁴Advanced Business Application Programming

⁵Multi-Purpose Internet Mail Extensions

1.2. Elektronische Patientendaten

Es stellt sich bei den elektronischen Patientendaten die Frage, welche Art von alphanumerischen Daten bei einzelnen Patienten vorkommen und wie man diese in einem Datenmodell zusammenführt. In Unterabschnitt 1.2.2 werden die Patientendaten daher in harte und weiche Daten unterteilt.

1.2.1. Datenmodelle

Datenmodelle werden nach [44] dazu verwendet, Ausschnitte aus der realen Welt in einem Datenmodell abzubilden. Dadurch wird es u.a. möglich, Daten mit den gegebenen Abhängigkeiten strukturiert abzuspeichern. Die elektronisch erfassten Patientendaten - in aller Regel in relationalen Datenbanken - werden in Systemen erfasst, welche auf einer zuvor erfolgten Datenmodellierung basieren. Die Literatur zum relationalen Modell ist zahlreich (e.g., [14]), daher an dieser Stelle lediglich ein einfaches Beispiel analog zu [44]:

Die Tabelle 1.1 auf der nächsten Seite zeigt die Darstellungen einiger Relationen jeweils als Tabelle. Ein *Patient* in einem Datenbanksystem einer Praxis sei wie in Tabelle 1.1a auf der nächsten Seite mit den Attributen *PATNR*, *NAME*, *GESCHL*, *GBDAT* abgespeichert. Es sind selbstverständlich noch weitere Attribute (e.g., Geburtsort) denkbar. In der gleichen hypothetischen Praxis gibt es wie in Tabelle 1.1c auf der nächsten Seite einzelne *Arzttermine* mit den Attributen *TERMNR*, *ZIMMER*, *ZEIT*, *ARZT*. In der Relation *hat* wird nun durch die Patientennummer und die Terminnummer zwischen den Relationen *Patient* und *Arzttermin* eine Beziehung hergestellt. Ein solches Datenmodell wird auch als *Entity-Relationship-Modell* bezeichnet. Weitere Modelle zur Datenmodellierung sind beispielsweise in der Buchreihe (e.g., [80] und Fortsetzungen) von Len Silverston zu finden.

1.2.2. Einordnung der Daten

Die sogenannten Stammdaten [53] fallen bei nahezu jedem Patienten an. Hierbei sind insbesondere Vorname, Nachname, Geschlecht, Geburtsort und Geburtstag zu nennen. Der Geburtstag gehört schon zu den harten Patientendaten.

1. Motivation

Patient			
<u>PATNR</u>	<u>NAME</u>	<u>GESCHL</u>	<u>GBDAT</u>
3452	Meier	m	2.12.1980
1341	Schmidt	w	4.6.1956
2324	Löw	m	3.2.1960

(a) Tabelle *Patient* mit den fiktiven Attributen *PATNR*, *NAME*, *GESCHL*, *GBDAT*.

hat	
<u>PATNR</u>	<u>TERMNR</u>
1341	2306
2324	2305
3452	2176

(b) Tabelle *hat* mit den fiktiven Attributen *PATNR*, *TERMNR*.

Arzttermin			
<u>TERMNR</u>	<u>ZIMMER</u>	<u>ZEIT</u>	<u>ARZT</u>
2176	3.02	9:00	Schulze
2305	4.01	11:15	Wohlfahrt
2306	4.02	10:30	Schneider

(c) Tabelle *Arzttermin* mit den fiktiven Attributen *TERMNR*, *ZIMMER*, *ZEIT*, *ARZT*.

Tabelle 1.1.: Relationen als flache Tabellen, mit unterstrichenen Schlüsselattributen. Darin enthalten ist unter anderem folgende Information: Die Patientin Schmidt (*PATNR 1341*) hat (*PATNR 1341* / *TERMNR 2306*) einen Termin (*TERMNR 2306*) zum Zeitpunkt 10:30Uhr bei Arzt Schneider in Zimmer 4.02.

1.2.2.1. Harte Patientendaten

Harte numerische Daten [5] sind per Messung feststellbar und werden objektiv [34] erhoben. Dazu gehören beispielsweise Laborwerte [92], wie die Harnsäure im Blut oder die Messung der aktuellen Körpergröße eines Patienten. Diese Daten werden meist numerisch erfasst und sind daher mit einem Rechner einfacher zu verarbeiten (e.g., Vergleich zwischen Zahlenwerten).

1.2.2.2. Weiche Patientendaten

Weiche Daten [5] beruhen auf subjektiven Beobachtungen und sind manchmal auch von der Erfahrung und Ausbildung eines Mediziners abhängig. Hierzu werden Auswahlmöglichkeiten, Klassifikationen oder freier Text verwendet. Daneben seien an dieser Stelle noch individuelle Daten erwähnt. Die gleiche Beobachtung der Reflexe eines Patienten können Ärzte, basierend auf ihrer Ausbildung, individuell verschieden darstellen und entsprechend notieren. So können die Reflexe eines Patienten im gleichen Zustand von einem Arzt als “lebhaft” und einem anderen Arzt als “sehr lebhaft” beschrieben werden [40].

Bei der Datenverarbeitung entsteht dadurch aber das Problem von schwerlich zu interpretierenden Texten, besonders im Falle unterschiedlicher Autoren. Die in Abschnitt 5.1 auf Seite 58 erhobenen *Freitexte einer Diagnose* sind ein weiteres Beispiel für weiche Patientendaten.

1. Motivation

2. Studien

In Kapitel 1 auf Seite 15 wurde der zusätzliche Mehrwert von bereits vorhandenen Patientendaten für Studienzwecke genannt. Daher gehe ich in diesem Kapitel weiter auf das Thema Studien ein. In Abschnitt 2.1 wird der organisatorische Ablauf bezüglich der Erfassung von Patientendaten in Studien skizziert. Die Aufnahme von Patienten in Studien wird ebenso betrachtet, da es mit der elektronischen Erfassung einhergehen kann. Ich erörtere in Abschnitt 2.2 auf der nächsten Seite das Potential elektronisch gewonnener medizinischer Daten gegenüber Papierdaten.

2.1. Organisation der Patientendaten

Eine medizinische Studie bedarf laut [78] einer Genehmigung, wobei die dafür zuständige Stelle (e.g., Paul-Ehrlich-Institut, Bundesinstitut für Arzneimittel und Medizinprodukte) vom thematischen Inhalt der Studie abhängt. Es ist bereits bei der Studienvorbereitung zu beachten, dass die entsprechenden Regelungen der *Good Clinical Practice* (GCP) [54] hinsichtlich des Datenmanagements Verwendung finden. Die GCP soll nach [78] durch sogenannte *Standard Operating Procedures* eingehalten werden, welche prinzipielle Arbeitsläufe definieren und somit eine Erhaltung der Datenqualität ermöglichen.

Vor dem Beginn einer Studie wird ein sogenanntes Studienprotokoll [58] erstellt. Dieses legt unter Anderem fest, in welchem Zeitraum und an welchen Orten eine Studie stattfindet. Dabei muss die Motivation für die Studie genannt und eine Kosten-Nutzen-Analyse vorgelegt werden. Außerdem sind die Art (e.g., Beobachtungsstudie, randomisierte klinische Studie) der Studie zu erläutern, sowie die vorgesehene Auswertung des Zielkriteriums (e.g., quantitativ, qualitativ) mittels statistischer Analyse [78].

Sollte ein Patient innerhalb einer teilnehmenden Klinik als Teilnehmerkandidat auserkoren worden sein, folgt bei Teilnahmeinteresse eine Aufklärung des Patienten [78]. Falls der Patient, nach dem Verstreichen einer Mindestbedenkzeit, einer Teilnahme an der Studie zustimmt, wird er nach Erfüllung weiterer organisatorischer Punkte (e.g., „Pseu-

donymisierung“) seitens des Krankenhauses bei der Studienleitung bekannt gemacht. Die Studienleitung gibt dem Krankenhaus entsprechende *Case Report Forms* (CRFs), welche nach [17] genutzt werden, um die jeweiligen Informationen über jeden Patienten innerhalb eines klinischen Versuchs aufzunehmen. Ein Erhebungsbogen für diese Studie wird vom Auftraggeber (i.e., “Sponsor” [17]) (e.g., Pharmaunternehmen) bereitgestellt und kann sowohl in schriftlicher als auch elektronische Form existieren. Im letzteren Fall spricht man von einem eCRF [3].

Das Krankenhaus führt über den Patienten eine Studienakte, sobald dieser in die Studie eintritt. Die Studienakte enthält für die Studie relevante medizinische Daten des Patienten. Diese Daten werden zusätzlich dazu verwendet werden, um die entsprechenden CRFs für den Patienten zu befüllen. Diese CRFs werden nach einer Aktualisierung zeitnah an die Studienleitung geschickt. Die Studienakte wird in der Regel weiter im Krankenhaus gelagert, selbst wenn die passenden CRFs bereits ausgefüllt und versendet wurden.

2.2. Verarbeitungswege

Es gibt für medizinische Studien wissenschaftliche Untersuchungen über den Vergleich zwischen papierbasierten und elektronischen Patienteninformationen. Die Autoren von [84] verweisen u.a. auf notwendige Qualitätskriterien der Dokumentation, um einen aussagekräftigen Vergleich anstellen zu können. Danach ist beides schwer zu vergleichen, und weitere Forschung wird als notwendig erachtet. Aufgrund der GCP ist nach [78] bei der Studiendurchführung auf die Sicherung der Daten zu achten, sowie auf die Nachvollziehbarkeit von Datenänderungen.

2.2.1. Papierbasierte Patientendaten

Die Patientendaten werden innerhalb eines Krankenhauses papierbasiert erfasst und sind dann parallel für die Studienakte zu dokumentieren. Korrekturen oder Veränderungen der Daten sind zur Bewahrung der Qualitätsanforderungen der GCP mit Datum und Unterschrift zu versehen. Die Übertragung auf den entsprechenden Erhebungsbogen der Studie erfolgt wieder papierbasiert und kann als Papierdokument [3] (e.g., Brief, Fax, Scan) an die Studienleitung geschickt werden. In der Studienleitung muss man die entsprechende papierbasierte Dokumentation wieder zusammenbringen, bevor man eine Zwischenanalyse vornimmt.

2.2.2. Elektronische Patientendaten

Im günstigsten Fall ist ein Großteil der Patientendaten im Krankenhausinformationssystem bereits erfasst. Diese Systeme erfassen zum Teil weitere Informationen, welche der Qualitätssicherung zugute kommen. So wird im SAP-System in Homburg beispielsweise bei jeder Diagnose zusätzlich Datum, Uhrzeit und der verantwortliche Mitarbeiter gespeichert. Die im KIS erfassten Daten können in das entsprechende eCRF übertragen und die restlichen Daten ergänzt werden. Das aktualisierte eCRF ist theoretisch schneller (i.e., elektronisch) an die Studienleitung übertragbar und dort mit entsprechenden Vorteilen durch Rechner zu verarbeiten. Der Sponsor wird spätestens an dieser Stelle ein klinisches Studienmanagementsystem bzw. *Clinical Trial Management System* (CTMS) [20] verwenden, um mit Hilfe dieser Daten den Verlauf einer Studie zu verfolgen.

Ein solches CTMS ist beispielsweise das System OB_TIMA¹ [13], welches an der UdS in Zusammenarbeit mit weiteren Partnern (e.g., FRAUNHOFER IBMT) kontinuierlich entwickelt wird. Laut [13] ist neben dem Managementsystem für Patientendaten ein sogenannter *Trial Builder* vorhanden. Damit können viele notwendige Dokumente (e.g., CRFs, Behandlungspläne) für eine Studie erstellt werden.

2.2.3. Vergleich der Vorgehensweisen

Ein rein papierbasierter Ansatz für Studien ist mit einem Mehraufwand verbunden, welcher für die Studien nicht zwingend einen Nutzen darstellt bzw. die Weitergabe aktueller Daten verzögern kann. Je nach Rechtslage kann eine Dokumentation in Papierform allerdings notwendig sein. Auf Papier können zudem interessante Zusatzaspekte zum Patienten vermerkt werden, die ansonsten im elektronischen Angebot noch gar nicht vorhanden sind und dort unter Umständen keinen Platz finden.

Bei der elektronischen Verarbeitung stellt sich, neben dem in dieser Arbeit behandelten Zugriff auf die Daten, das Problem der Übertragung der Daten zwischen Systemen [8], auch jenseits des Sicherheitsaspekts. Es muss bekannt sein, mit welchen Softwaretools man arbeitet und wie die Schnittstellen zwischen diesen Programmen definiert sind. Zur Studienleitung ist ebenfalls eine entsprechende Schnittstelle vonnöten. Die Beschränkung auf die teilnehmenden Patienten muss gesichert sein. Für die, je nach Rechtslage, zusätzliche erforderliche Papierdokumentation bedarf es einer Druckfunktionalität oder einer Option des Exports in ein ausdrückbares Format.

¹Ontology-based Trial Management System

2. Studien

Das Vorgehen in der Praxis ist je nach Krankenhaus und Studie unterschiedlich und wird möglicherweise auf die Existenz beider Erfassungen hinauslaufen. Der Auftraggeber einer Studie kann beispielsweise für Krankenhäuser eine Software (e.g., MARVIN²) zur Eingabe der Daten bzw. Generierung von eCRFs mit den eingegebenen Daten bereitstellen. Das jeweilige Krankenhaus organisiert dann selbst den vorherigen Verarbeitungsweg bis zur Eingabe der Daten. In diesem Falle werden die Daten in elektronischer Form an den Sponsor weitergegeben, unabhängig von den lokalen Lösungen der teilnehmenden Krankenhäuser.

²<http://www.xclinical.com/de/marvin/>

3. Krankenhausinformationssysteme

In diesem Kapitel beschreibe ich im Abschnitt 3.1 den Terminus Krankenhausinformationssystem. Ein KIS beinhaltet zumeist ein Datenbanksystem. In Abschnitt 3.2 auf Seite 29 werden Datenbanksysteme behandelt, gefolgt von einer entsprechenden Datenmanipulationssprache in Abschnitt 3.3 auf Seite 29. Auf das Potential einer falschen Handhabung eines KIS durch den Endnutzer gehe ich in Abschnitt 3.4 auf Seite 30 ein. Informationen zum lokalen System in Homburg gebe ich in Abschnitt 3.5 auf Seite 31. Für ein externes Beispiel gehe ich in Abschnitt 3.6 auf Seite 33 auf das System der Neurologie im Krankenhaus Püttlingen ein.

3.1. Beschreibung

Es gibt nach [75] verschiedene Generationen von Krankenhausinformationssystemen. Die erste Generation erfasst die Stammdaten und dient der Abrechnung unter gesetzlichen Vorgaben. In der zweiten Generation werden zusätzlich noch medizinische und pflegerische Daten erfasst, allerdings noch wenig orientiert am Gesamtprozess der Behandlung. Der Behandlungsprozess als solcher soll in der dritten Generation unterstützt werden, man spricht von Workflows in Medizin und Pflege. Ein Definitionsversuch für ein KIS ist nicht über eine Auflistung von Funktionalitäten zu leisten, da in Krankenhäusern meist jeweils unterschiedliche (e.g., Schwerpunkte) KIS zum Einsatz kommen. Es gibt zwar Anbieter, welche möglichst alles abdecken möchten, doch letztlich wird deren KIS im größeren Maßstab eine Produktkombination sein.

Der Autor Helmut Schlegel müht sich mit langer Vorbereitung in [75] an einem Definitionsversuch für Krankenhausinformationssysteme: "Ein Krankenhausinformationssystem ist die Gesamtheit aller Einheiten und Beteiligten, Menschen und Maschinen, die im Krankenhaus in informationsverarbeitenden Prozessen Daten erheben, verändern und auswerten, um daraus zur Entscheidungsfindung Informationen zu bilden."

Nach [29] sind Krankenhausinformationssysteme wie andere Unternehmensinforma-

tionssysteme zu sehen, d.h. sie unterstützen die Abläufe in einem Krankenhaus. Damit beschränkt sich ein KIS also nicht auf die Patientenbehandlung (e.g., Sonografie in der Inneren), sondern geht auf alle anfallenden Bereiche (e.g., Finanzbuchhaltung, Personalabteilung) ein. Hierbei ist zu beachten, dass manche Anwendungen (e.g., Materialverwaltung) in nahezu allen Untersystemen zum Einsatz kommen, wenn auch mit unterschiedlichen Schwerpunkten. Ein monolithisches Unternehmensinformationssystem stammt in der Regel von einem Hersteller oder einer bestimmten Kombinationsvariante eines Anbieters. Das Datenmodell (siehe auch Unterabschnitt 1.2.1 auf Seite 19) und die Systemarchitektur sind in diesem Fall einheitlich. Dagegen kombiniert ein heterogenes Unternehmensinformationssystem unterschiedliche Anwendungssysteme, welche miteinander kooperieren. Der Grad der Kooperationsfähigkeit der singulären Komponenten wirkt sich auf die Möglichkeiten des Systems aus.

Die nachträgliche Anbindung eines zusätzlichen Subsystems kann bei einem identischen Hersteller einfacher funktionieren, sofern eine spätere Anbindung bei den restlichen Systemen bereits vorgesehen war. Beispielsweise ist beim Unternehmensinformationssystem SAP R/3 laut [74] der Funktionsumfang schrittweise aktivierbar.

Die monolithische Lösung bringt laut [29] Vorzüge (e.g., konsistente Benutzeroberfläche, keine doppelte Datenhaltung, weniger Personal) doch erhöht sich die Abhängigkeit von einem Hersteller, da dieser die Integration von Fremdsystemen aus Verkaufsinteresse erschweren wird. Die vorgegebene Gesamtlösung kann dann breit (e.g., OP-Informationssystem, PACS¹) angelegt sein, aber kundenspezifische Anpassungen sind schwerlich zu erreichen.

Der Weg in einem Krankenhaus führt optional über einen Kernlösungsanbieter, welcher neben einer administrativen Software noch ein Auftragsmanagement anbietet. Die vorhandenen Subsysteme, welche möglicherweise für sich selbst monolithisch sind, können dann angebunden werden. Dies führt dann wieder zum Nachteil der heterogenen Systeme, und bedarf mehrfacher Datenhaltung oder Datenbanksysteme. Zur Verbindung wird inzwischen vermehrt mit Kommunikationsservern gearbeitet [29].

¹Picture Archiving and Communication System

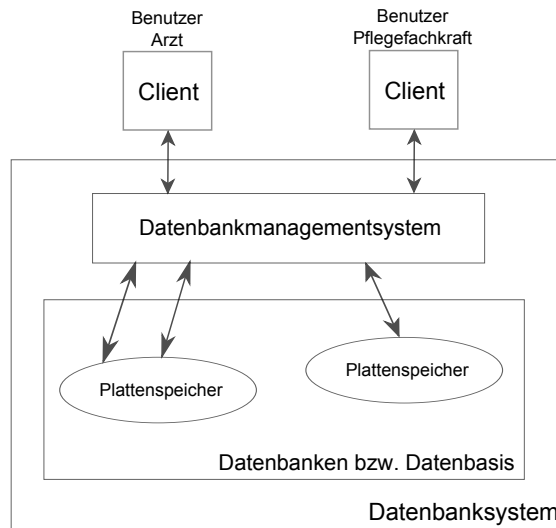


Abbildung 3.1.: Darstellung eines Datenbanksystems.

3.2. Datenbanksysteme

In der Materie der Datenbanksysteme verschwimmen die Begrifflichkeiten nach [44] des öfteren. Die vorhandenen in Beziehung stehenden Daten werden als Datenbasis bezeichnet und existieren in physischer Ebene auf einer Datenbank (e.g., ORACLE, SAP MAX-DB). Die Konsistenzkontrolle und der Zugriff auf die Daten der Datenbasis werden wiederum unter den Begriffen Datenbankverwaltungssystem (DBVS) oder Datenbankmanagementsystem (DBMS) zusammengefasst. Die kürzere Form Datenbanksystem wird nach [36] auch als Zusammenfassung beider Komponenten betrachtet. Die physische Ebene ist für den Endnutzer in der Regel gar nicht sichtbar, sondern er bekommt durch das DBMS, je nach Benutzergruppe (e.g., Krankenschwester in Augenklinik), einen entsprechenden Ausschnitt des Systems zu sehen (siehe Abbildung 3.1).

3.3. Datenbankmanipulationssprache

Die in Tabelle 1.1 auf Seite 20 gezeigten Relationen sind auch ein Beispiel für die logische Sicht des Datenbankbenutzers, welcher durch eine Datenbankmanipulationssprache (DML) die Daten abfragen und manipulieren kann. Als Beispiel für eine Abfragesprache für eine relationale Datenbank ist die Sprache SQL (Structured Query Language) [97] zu

NAME
Meier
Schmidt
Löw

(a) Select *NAME* from *Patienten*.

NAME	GESCHL	GBDAT
Schmidt	w	4.6.1956

(b) Select *NAME*, *GESCHL*, *GBDAT* from *Patienten*, *hat*, *Arzttermin* where *Patienten.PATNR=hat.PATNR* and *Arzttermin.TERMNR=hat.TERMNR* and *Arzttermin.ZEIT='10:30'*.

Tabelle 3.1.: Beispiele für SQL-Fragen zu den Relationen aus Tabelle 1.1 auf Seite 20. (a) Abfrage der Namen aller Patienten. (b) Es werden Name, Geschlecht und Geburtsdatum des Patienten abgefragt, welcher um 10:30Uhr einen Termin hat.

nennen, welche in Teil II auf Seite 37 Verwendung findet. Diese kann wie in [44] interaktiv mit einem direktem Abfrageergebnis verwendet werden oder in einer höherer Programmiersprache (e.g., ABAP in Unterabschnitt 3.5.1) eingebettet sein. In Tabelle 3.1 sind zwei Abfragebeispiele zu den Relationen aus Tabelle 1.1 auf Seite 20 zu finden. Beim ersten Beispiel werden alle Namen aus einer Tabelle abgefragt, während beim zweiten Beispiel Informationen (c.q., Name, Geschlecht, Geburtsdatum) zu dem Patienten abgefragt werden, welcher unter Verbindung mit den anderen relationalen Tabellen der Abfragebedingung (i.e., Zeit des Arzttermins) entspricht.

3.4. Handhabungsfehler

In [45] wird ein “Fall des Monats” beschrieben, welcher vor einer zu leichtfertigen Nutzung eines KIS warnen soll. Gerade die Beschleunigung des Arbeitsablaufs mit einem KIS birgt ein solches Risiko. Der Fall handelt von einem Problem mit der Patientenidentifikation in der Datenbank, während des Routinebetriebs eines Krankenhauses.

Bei einer dementen Patientin fiel nur zufällig auf, dass es sich nicht um die Patientin handelte, deren Unterlagen mitgeführt wurden. Die initiale Therapieplanung basierte zu dem Zeitpunkt bereits auf den Daten einer ehemals im gleichen Krankenhaus behandelten Patientin. Der Hauptgrund für die Verwechslung war die Tatsache, dass Name, Vorname und Geburtsdatum dieser Patientin dreimal in der Datenbank vorkamen. Die Möglichkeit der zeitsparenden Übernahme von bereits vorhandenen Patientendaten für die Generierung eines neuen Behandlungsfalls wurde fehlerhaft eingesetzt. Glücklicherweise kam es zu keinen ernsten Folgen.

Produkt	Release	Hersteller	Kurzbeschreibung des Produktes
SAP ERP	2005	sap.com	SAP ERP 6.0
SAP NETWEAVER	SAP EHP2 FOR SAP NETWEAVER 7.0	sap.com	SAP EHP2 FOR SAP NETWEAVER 7.0
EHP5 FOR SAP ERP 6.0	EHP5 FOR ERP 6.0	sap.com	EHP5 FOR SAP ERP 6.0

Abbildung 3.2.: Ausschnitt eines Screenshots aus dem Testsystem T21 des UKS Homburg zur Produktversion.

Jedem Patienten wird nach [33] am besten eine eindeutige Identifikationsnummer zugeordnet. Sollte er als eindeutiger Wiederkehrer identifiziert werden, kann er dadurch der vorherigen Identität zugeordnet werden.

Ein KIS wird letztlich von einem Menschen bedient, welcher einem Bedienungsfehler [56] unterliegen kann. Gerade in stressigen Situation fällt ein Handhabungsfehler nicht immer sofort auf. Zusätzliche Sicherheitsmaßnahmen, wie automatische Rückfragen [87] seitens des Systems, können zwar punktuell Abhilfe schaffen, aber gleichzeitig wird die Bedienung des Systems erschwert. Eine andere Option ist die Vorgabe von Pflichtfeldern [87], um den Nutzer zu einer eindeutigen Abfrage bei der Datensuche zu zwingen.

3.5. Universitätsklinikum in Homburg

Das am UKS genutzte Krankenhausinformationssystem wird seit dem Jahr 1996 mit Unterstützung des Zentrums für Informations- und Kommunikationstechnik (ZIK) betrieben [101]. In Homburg wird für das KIS der Term “Klinikinformationssystem” benutzt. Es entspricht am ehesten der Beschreibung der Lösung mit einem Kernlösungsanbieter und Auftragsmanagement in Abschnitt 3.1 auf Seite 27. Als Basis dient hierzu nach [24, 101] das Unternehmens-Informationssystem SAP R/3 [90, 94, 95] der Firma SAP, bzw. in Teilen das Nachfolgeprodukt SAP ERP² [27], siehe Abbildung 3.2. In diesem KIS ist neben weiteren Modulen (e.g., SAP FINANCIAL ACCOUNTING) auch das Modul SAP INDUSTRY SOLUTIONS HEALTHCARE (SAP IS-H) [93] aktiviert. Zudem wird das erweiternde Softwaremodul I.S.H.MED [22] der Firma SIEMENS verwendet. Diese beiden Module werden häufig in Kombination genutzt, wobei der Schwerpunkt von SAP HEALTHCARE mehr

²Enterprise-Resource-Planning

bei der Patientenadministration und den Finanzen zu sehen ist, während I.S.H.MED eine Erweiterung für die Kommunikation und die Auftragsplanung bezüglich Medizin und Pflege darstellt.

Die Systemlandschaft im SAP-Umfeld ist nach [31] meist in mehrere SAP-Systeme (e.g., Qualitätssicherungssystem, Produktivsystem) gegliedert, welche untereinander mit dem SAP-Transportwesen verbunden sind. Daher kann beispielsweise eine im Entwicklungssystem entwickelte Änderung per *Transport* in das Qualitätssicherungssystem gelangen und nach erfolgreichem Testlauf über einen *Transport* in das Produktivsystem [15] gehen. Am UKS Homburg sind für diese Arbeit hingegen das sogenannte Testsystem (T21) und das Produktivsystem (C21) relevant. Die Datenbasis für das SAP ERP ist durch eine relationale ORACLE-Datenbank [39] gegeben.

3.5.1. Programmiersprache

Zur Entwicklung im SAP-Umfeld verwende ich die bereits im SAP-System vorhandene Programmiersprache ADVANCED BUSINESS APPLICATION PROGRAMMING³ (ABAP) für NETWEAVER APPLICATION SERVER VERSION 7.02. Diese Programmiersprache wurde seit 1982 [15] von SAP kontinuierlich entwickelt, um einen Großteil der Komponenten [31] zu programmieren. Zum Bearbeiten von Quelltexten existiert in SAP-Systemen bereits eine Entwicklungsumgebung (i.e., ABAP Workbench). An dieser Stelle ist zu erwähnen, dass ABAP schon Befehle [42] zur Verfügung stellt, um per SQL (siehe Abschnitt 3.3 auf Seite 29) auf Datenbanken zugreifen zu können [79].

3.5.2. Subsysteme

Es kommen in den unterschiedlichen Kliniken und Abteilungen des UKS verschiedene Subsysteme zum Einsatz, zum Teil sind diese autonom [25]. Informationen werden mit dem Kommunikationsserver GLOVERLEAF [49] der HEALTH-COMM GmbH über eine Patientenschnittstelle ausgetauscht. In der Praxis hat sich im Laufe dieser Arbeit herausgestellt, dass man letztlich bei einigen Subsystemen (e.g., Mikrobiologie, Stand 21.1.2014) mit obigem Kommunikationsserver nicht direkt auf die strukturierten Daten der monolithischen Systeme zugreifen kann. Allerdings werden die Systeme und der Datenaustausch seitens des ZIK kontinuierlich verbessert.

³Früher wurde sie "Allgemeinen-Berichts-Aufbereitungs-Prozessor" genannt.

3.6. Klinik für Neurologie im Krankenhaus Püttlingen

In dieser Arbeit erweitere ich u.a. ein bereits vorhandenes Krankenhausinformationssystem (siehe Abschnitt 3.5 auf Seite 31), um die darin enthaltenen medizinische Daten zur weiteren Verarbeitung extrahieren zu können. Zu einem ähnlichen Zweck (c.q., Generierung von Arztbriefen) hat man in der Neurologie des Krankenhauses Püttlingen ein komplett neues KIS eingeführt. Damit wurde allerdings vor Ort ein individueller Standard geschaffen, welcher parallel zu den restlichen Systemen im Krankenhaus Püttlingen existieren muss. Hinsichtlich des Systems in der Klinik für Neurologie beruht der Inhalt der folgenden Unterabschnitte im Wesentlichen auf Informationen aus [40].

3.6.1. Motivation

Man bemühte man sich vor Ort schon seit Jahren um eine Verbesserung der elektronischen Erfassung von Patienteninformationen, verbunden mit der inhaltlichen Verwertung [40]. Die bloße Papiergenerierung sollte nicht fortgesetzt werden. Das Hauptaugenmerk lag hierbei unter anderem auf der Verwendung der elektronischen Patientendaten für Arztbriefe. Der Arzt erstellte bis dahin den Arztbrief meist mit Hilfe einer sprachlichen Aufzeichnung, welche dann von einer Sekretärin getippt wurde. Ein Verfahren, welches heute in Teilen vielerorts noch Anwendung findet [50].

Die spätere Verwendung von freien Textfeldern in einem KIS resultierte darin, dass die verwendeten Textfelder letztlich weiche Patientendaten enthielten, welche sehr individuell waren. Arztbriefe wurden nicht mit Hilfe von Vorgaben erstellt, sondern waren individuelle Momentaufnahmen. Es sollte daher Werkzeuge geben, um sich bei der Eingabe von Diagnosen ausschließlich in einem vorgegebenen Rahmen zu bewegen. Die Individualität sollte vorgegeben werden, damit es nur noch zu leichten Schwankungen im Informationswert kommen kann.

Das Ziel dabei war es, nach Erfahrungen in der Vergangenheit, durch gezieltere informationstechnische Arbeit mit neuen Patientendaten vorausschauend zu arbeiten. Es sollten keine Informationen mehr in Papierbergen verloren gehen und die Nutzung schriftlicher Unterlagen reduziert werden. Mit verbesserten Arztbriefen sollte auch die Epikrise für den Hausarzt verbessert werden.

Zur gezielten Analyse der Möglichkeit einer Realisierung eines solchen neuen KIS,

konsultierte Herr Dr. med. Dipl. Math. Helmut Jäger⁴⁵ einige Experten (e.g., Herr Prof. Dr. Güttler von der Hochschule für Technik und Wirtschaft des Saarlandes (HTW Saar), Herr Prof. Dr. Dr. h.c. mult. Wahlster vom DFKI⁶). Das Spin-Off-Unternehmen GIMTEC⁷ der HTW Saar hat daraufhin eine entsprechende Software entwickelt.

3.6.2. Beschreibung

In der Neurologie des Krankenhauses Püttlingen wird seit mehr als 10 Jahren das System HoWoS der Firma GIMTEC eingesetzt [40]. HoWoS steht für Hospital Workflow System. Bisher wurden laut [40] positive Erfahrungen mit dieser Software gemacht, und sie wird in der Neurologie Püttlingen flächendeckend eingesetzt.

3.6.2.1. Beginn und Einführung

Es wurde zuerst evaluiert, was in der Neurologie Püttlingen benötigt wird und wie die Arbeitsprozesse in der Praxis aussehen. Für Layoutfragen innerhalb von HoWoS wurden seitens GIMTEC Vorschläge gemacht.

Das System wurde unter der Ärzteschaft in der Neurologie nach einer Einarbeitungszeit insgesamt gut aufgenommen. Andere Systeme (e.g., Labor, Radiologie) konnten eingebunden werden, allerdings waren dort Anpassungen notwendig. Die Eingabe der Medikation wurde ebenfalls eingearbeitet, um für die Arztbriefe verfügbar zu sein.

3.6.2.2. Dokumentation mit Textbausteinen

Wie auch in anderen KIS ist beispielsweise der Verweis auf ein EEG⁸ möglich; im System wird der Arzt aber nun gezwungen, dieses auch im gegebenen Rahmen zu beurteilen [40]. Für die Interpretation wird zwar ein Text generiert, doch dieser entsteht daraus, dass der Bearbeiter aus für ein EEG passenden Haupttextbausteinen auswählt und diese an einzelnen Textstellen mit entsprechender Auswahl oder harten Patientendaten anpasst. Mit Hilfe der Systemdatenbank gibt es auswählbare Vorgaben für die einzelnen Stellen. Wegen der Textbausteine existieren alleine für EEGs ca. 20000 Kombinationsmöglichkeiten an Beschreibungen. In der Vergangenheit bestand dagegen noch die Gefahr, dass eine

⁴Chefarzt der Neurologischen Abteilung im Knappschafts Krankenhaus Püttlingen

⁵Ärztlicher Direktor

⁶Deutsches Forschungszentrum für Künstliche Intelligenz

⁷Gesellschaft für Informations- und Managementsysteme mbH

⁸Elektroenzephalogramm

genauere Bewertung nicht mehr stattfand. Die Auswahl der Textbausteine bzw. Haupttextbausteine hat den weiteren Vorteil, dass durch bestimmte Kombinationen zusätzliche visuelle Informationswerte erzeugt werden, beispielsweise durch eine Textfärbung, welche auf den wissenschaftlichen oder pathologischen Zweig verweisen. An einzelnen Stellen ist ein Mischung von Freitext in Verbindung mit Pflichttextbausteinen möglich.

Im Ergebnis führt das dazu, dass nach allen Eingaben ein Arztbrief innerhalb von Sekunden generierbar ist. Kontrollmöglichkeiten sind auch vorhanden; so kann man sich beispielsweise anzeigen lassen, welche EEGs noch offen sind. Die generelle Idee der Freitextgenerierung aus kodierten Informationen führt laut [65] zu guten Resultaten.

3.6.2.3. Weitere Eigenschaften

In der klassischen schriftlichen Krankenakte war Platz für eine persönliche Notiz zum Patienten [76], denn immerhin möchte sich der Patient nicht als Patientenummer mit einer beliebigen Anzahl an Eingabewerten fühlen. In der Neurologie Püttlingen gibt es dafür optionale Zusatzattribute bei medizinischen Informationen, welche dann wieder vom Hausarzt eingeordnet werden.

Es gibt einen Patientenkalender, in dem für den Arzt alle Maßnahmen einsehbar sind. In HoWoS erfolgt neben der Pflegedokumentation auch die Abrechnung. Es werden außerdem nur für den Benutzer relevante Daten angezeigt. Die Pflege und Erweiterung des Systems erfolgt durch *Customizing* [21], die Software ist also anpassbar. Ein "Netzarzt", damit sind hauptsächlich Hausärzte gemeint, kann mit einer entsprechenden Berechtigung Daten von seinen Patienten einsehen.

3.6.2.4. Studien

Für Studien werden einzelne Studienformular-Module angelegt, welche letztlich jeweilig angepasst werden. Es ergibt sich daraus ein Fragebogen bzw. CRF, welcher schon mit den bereits vorhandenen Patientendaten bereitgestellt und im Einzelfall lediglich ergänzt werden muss. Die Aufbereitung der Patientendaten für Studien ist dadurch vereinfacht, dass alle Vorgänge im gleichen System erfolgen. In der Neurologie werden die Studiendaten jedes halbe Jahr übertragen. Die Namen und weitere persönliche Daten der Patienten sind davon nicht betroffen.

Teil II.

Methoden zur Gewinnung und Analyse von Patientendaten

4. Ausgangslage am Universitätsklinikum Homburg

In Teil II erläutere ich die Gewinnung und Analyse von Patientendaten¹ der Kinderklinik am Universitätsklinikum Homburg. Die Auswahl der gesuchten Werte aus dem Krankenhausinformationssystem (c.q., SAP) entspricht dem Mikrobiologieszenario des EU-Projektes EURECA²³ [91]. Das KIS in Homburg betrachte ich bereits in Abschnitt 3.5 auf Seite 31. Weitere Informationen zu dem Szenario des EU-Projektes sind in Abschnitt 4.1 enthalten. Anschließend wird das Selektionskriterium für die Patientendaten in Abschnitt 4.2 auf Seite 44 näher betrachtet, welches die Grundlage zur weiteren Datengewinnung in Kapitel 5 auf Seite 55 darstellt.

4.1. Mikrobiologieszenario in eureca

Innerhalb des EU-Projektes EURECA, an welchem die UdS teilnimmt, werden Daten aus dem Krankenhausinformationssystem in Homburg benötigt. Das Projekt soll dazu beitragen, die Lücke zwischen einem Krankenhausinformationssystem und externen klinischen Analysemethoden der entsprechenden Daten zu schließen. Das sogenannte Mikrobiologieszenario dient vor Ort zur Bereitstellung von Daten aus dem KIS und der Mikrobiologie, um später mithilfe externer Programme Statistiken über resistente und multiresistente Erreger in Zusammenhang mit einer Station erstellen zu können.

Dieser Absatz basiert auf der Beschreibung in [52]. Während der Behandlung (e.g., Chemotherapie) eines Patienten kann es zu einer Infektion kommen, welche dann ebenfalls berücksichtigt werden muss. Es soll möglichst nicht zu einem schweren unerwünschten Ereignis kommen, einem sogenannten *Serious Adverse Event* (SAE) [78]. Im Falle

¹Es werden in den Beispielen ausschließlich Testdaten ohne realen Personenbezug verwendet.

²<http://eurecaproject.eu>

³Enabling information re-use by linking clinical Research and Care

einer Infektion mit einem resistenten oder multiresistenten Erreger ist die Behandlung mit passenden Medikamenten ein wichtiger Zeitfaktor. Durch das Mikrobiologieszenario soll man in einem Krankenhaus von bisherigen Erfahrungen profitieren können, indem man Daten über den Patienten, die Erreger und die eingenommenen Antibiotika kontinuierlich erhebt. Diese Informationen sollen kombiniert in einer statistischen Analyse zu einer verbesserten Behandlung beitragen. Das belegte Zimmer bzw. Bett eines Patienten sind relevant, um eine eventuelle Verbreitung der Erreger zu erkennen. Nach dem Zusammenbringen dieser Daten auf einem Studienserver sollen diese für weitere Analysezwecke zur Verfügung stehen.

In existierenden Krankenhausinformationssystemen sind diese Daten allerdings nicht zwingend an der gleichen Stelle gespeichert oder in direkten Zusammenhang zu bringen. Am UKS befinden sich die relevanten Stammdaten (e.g., Alter) und Informationen zum Fall (e.g., Diagnosen) verteilt im SAP System. An der KK-05 wird die Medikamentengabe inzwischen ebenfalls im SAP System hinterlegt. Andere Daten (e.g., Probenort, Informationen zum Erreger) befinden sich in den Systemen der Mikrobiologie.

Die derzeit angestrebte Version wird sich für Testzwecke auf die Patienten einer Station (c.q., Pädiatrische Onkologie und Hämatologie) beschränken. In diesem Teil der Arbeit wird hauptsächlich der Anteil der Datengewinnung aus dem SAP betrachtet und in Teilen auch die anschließende Übertragung (siehe Abschnitt 5.8 auf Seite 83) und die Möglichkeit der Weiterverarbeitung (siehe Kapitel 7 auf Seite 95) zur Vorbereitung externer Analysen.

In Abbildung 4.1 auf der nächsten Seite ist eine Übersicht zum Datenerhebungsteil und Transport der Dateien des Mikrobiologieszenarios bis zum Studienserver zu sehen. Der Export über das Network-File-System (NFS) des UKS erfolgt sowohl für die erstellten Dateien aus dem SAP-System als auch die Dateien aus der Mikrobiologie. In dem EU-Projekt kommen diese Daten zur weiteren Verarbeitung auf einem Studienserver zusammen. Dort werden u.a. daraus CDA-Dokumente (Clinical Document Architecture) [12] generiert, um an späterer Stelle des Projektes im HEALTH LEVEL 7 Standard (HL7) [9] arbeiten zu können. Weitere Details werden in Kapitel 7 auf Seite 95 erläutert. Maßnahmen zur Umsetzung des Datenschutzes der Daten aus dem SAP und aus der Mikrobiologie werden in Kapitel 6 auf Seite 87 behandelt. In Tabelle 4.1 auf Seite 42 und Tabelle 4.2 auf Seite 43 sind die gesuchten Werte aus dem SAP für das Mikrobiologieszenario aufgelistet (Stand 11.10.2014).

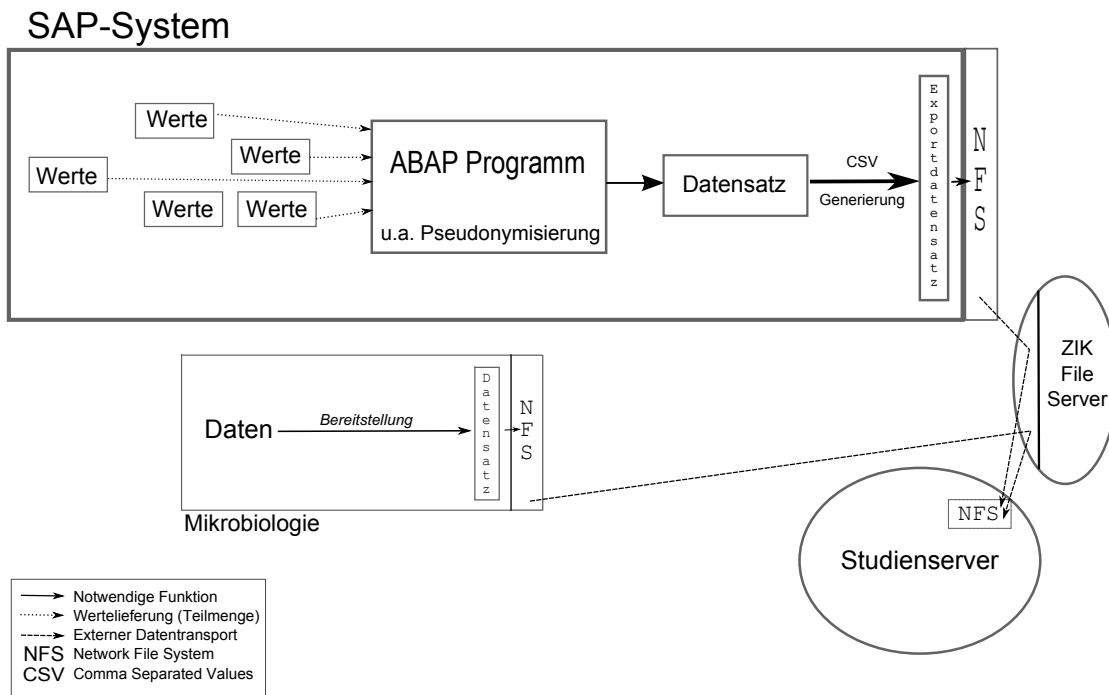


Abbildung 4.1.: Datenerhebung und Transport im Mikrobiologieszenario bis zum Studienserver.

Patientennummer
Name
Vorname
Geschlecht
Geburtsdatum
Einrichtung (Kürzel)
[Einrichtung] Kurzname der Einrichtung
[Einrichtung] Name der Einrichtung
[Einrichtung] Länderschlüssel
[Einrichtung] Postleitzahl
[Einrichtung] Ort
[Einrichtung] Straße und Hausnummer
[Einrichtung] Institutskennzeichen
[Einrichtung] Telefonnummer
Fallnummer
[Hauptdiagnose] Laufende Nummer Diagnose
[Hauptdiagnose] Schlüsselung einer Diagnose
[Hauptdiagnose] Identifikationskürzel für Diagnosekatalog
[Hauptdiagnose] Freitext einer Diagnose
[Hauptdiagnose] Datum, an dem der Satz hinzugefügt wurde
[Hauptdiagnose] Uhrzeit, zu der der Satz hinzugefügt wurde
[Hauptdiagnose] Text der Diagnose
[Aufnahmediagnose] Laufende Nummer Diagnose
[Aufnahmediagnose] Schlüsselung einer Diagnose
[Aufnahmediagnose] Identifikationskürzel für Diagnosekatalog
[Aufnahmediagnose] Freitext einer Diagnose
[Aufnahmediagnose] Datum, an dem der Satz hinzugefügt wurde
[Aufnahmediagnose] Uhrzeit, zu der der Satz hinzugefügt wurde
[Aufnahmediagnose] Text der Diagnose
[Entlassungsdiagnose] Laufende Nummer Diagnose
[Entlassungsdiagnose] Schlüsselung einer Diagnose
[Entlassungsdiagnose] Identifikationskürzel für Diagnosekatalog
[Entlassungsdiagnose] Freitext einer Diagnose
[Entlassungsdiagnose] Datum, an dem der Satz hinzugefügt wurde
[Entlassungsdiagnose] Uhrzeit, zu der der Satz hinzugefügt wurde
[Entlassungsdiagnose] Text der Diagnose

Tabelle 4.1.: Gesuchte Werte für das Mikrobiologieszenario (Stand 10.11.2014) Teil 1.
Der zweite Teil ist in Tabelle 4.2 auf der nächsten Seite zu finden.

Laufende Nummer einer Bewegung
Datum der Bewegung
Uhrzeit der Bewegung
Bewegungsendedatum
Bewegungsendezeit
Stationäre OrgEinheit, die einem Fall zugewiesen wird
BauId eines Zimmers
BauId eines Bettenstellplatzes
[Aufnahmebewegung] Laufende Nummer einer Bewegung
[Aufnahmebewegung] Datum der Bewegung
[Aufnahmebewegung] Uhrzeit der Bewegung
[Aufnahmebewegung] Bewegungsgrund - 1. und 2. Stelle
[Entlassungsbewegung] Laufende Nummer einer Bewegung
[Entlassungsbewegung] Datum der Bewegung
[Entlassungsbewegung] Uhrzeit der Bewegung
Materialnummer in KH für Materialverbrauch
Materialkurztext
(Angeforderte) Menge des Materials
Mengeneinheit
Datum der Verabreichung eines Materials
Uhrzeit der Verabreichung eines Materials
Substanzgruppe (Text)
Verabreichungsart (po. bzw. iv.)
Warengruppe
ATC-Code
Wirkstoff 1 (Hauptwirkstoff)
[Wirkstoff 1] Menge eines Wirkstoffs in einem Material
[Wirkstoff 1] Mengeneinheit eines Wirkstoffs in einem Material
Wirkstoff 2
[Wirkstoff 2] Menge eines Wirkstoffs in einem Material
[Wirkstoff 2] Mengeneinheit eines Wirkstoffs in einem Material
Wirkstoff 3
[Wirkstoff 3] Menge eines Wirkstoffs in einem Material
[Wirkstoff 3] Mengeneinheit eines Wirkstoffs in einem Material
Laufende Nummer der Materialanforderung

Tabelle 4.2.: Gesuchte Werte für das Mikrobiologieszenario (Stand 10.11.2014) Teil 2. Der zweite Teil ist in Tabelle 4.1 auf der vorherigen Seite zu finden. Der *ATC-Code* (Anatomisch-Therapeutisch-Chemisches Klassifikationssystem) [38] ist eine Klassifikation für Arzneistoffe.

4.2. Grundlage zur Datengewinnung aus dem SAP-System

Bei dem Bedarf nach Patientendaten innerhalb eines Programms stellt sich zuerst die Frage, welche Daten benötigt werden. Wie grenzt man diese Daten ein und welchen Weg beschreitet man, um diese abzufragen? Man möchte Daten von bestimmten Patienten haben und die zu verarbeitenden Daten möglichst auf diese Patienten reduzieren. Daher wird ein passendes Selektionskriterium für die Datenbankabfragen benötigt.

Die zu erfassenden Patientendaten für das Mikrobiologieszenario wurden folgendermaßen eingegrenzt: Relevant sind die Patienten, welche am aktuellen Tag stationär in der Pädiatrischen Onkologie und Hämatologie (KK-05) behandelt werden. Verlegte oder bereits entlassene Patienten sind nicht relevant.

4.2.1. Theoretische Vorarbeit

Nach [6] werden ausführbare ABAP-Programme in der Regel ABAP-Report genannt. Bei Anpassungsbedarf für die Abfrage sind diese mit einem Selektionsbild (e.g., Abbildung 5.5 auf Seite 86) ausgestattet, um nach Eingabe der Parameter für das Selektionskriterium mit der Datenverarbeitung zu beginnen. Es ist möglich, die Kriterien schon in der Programmierung komplett festzulegen und so die Userinteraktion zu vermeiden. Im Rahmen des Mikrobiologieszenarios möchte man die Daten von Patienten haben, welche zum aktuellen Stichtag stationär auf der KK-05 sind.

Die Schnittstellen zu den im Folgenden genannten Diktionär-Tabellen in SAP werden beispielsweise in [32] geschildert. Ein erster Zugriff auf die Standardtabelle NPAT (Stammdaten Patient, siehe Abbildung 4.2 auf der nächsten Seite) würde trotz ihrer 136 Felder die Daten nicht eingrenzen, weil dort beispielsweise weder das aktuelle Zimmer noch die belegte Station hinterlegt sind. Die Tabelle NFAL (IS-H: Fälle, siehe Abbildung 4.3 auf Seite 46) mit Schlüsselfeld *FALNR* (Fallnummer) ist mit 89 Feldern schon etwas kleiner, aber sie hat deutlich mehr Einträge als NPAT. Nahezu alle Patienten aus NPAT werden einen Fall haben, doch viele Patienten werden mehrere Fälle haben, da sie das Krankenhaus im Laufe ihres Lebens wiederholt besucht haben. Die Daten zum belegten Zimmer und der zugehörigen Station sind hier nicht vorhanden. Allerdings hat man hier mit der im Fall hinterlegten *PATNR* (Patientennummer) immerhin schon eine Verbindung zu den Stammdaten der Patienten in NPAT.

Man sieht in NFAL allerdings auch das Feld *BEWLF* mit Datenelement *LLFDBEW*,

4.2. Grundlage zur Datengewinnung aus dem SAP-System

Transp. Tabelle: NPAT tiv
 Kurzbeschreibung: IS-H: Stammdaten Patient (allgemein)

Eigenschaften | Auslieferung und Pflege | **Felder** | Eingabehilfe/-prüfung | Währungs-/Mengenfelder

Suchhilfe | Eingebauter Typ | 1 / 136

Feld	Key	Ini...	Datenelement	Datentyp	Länge	DezS...	Kurzbeschreibung	Gruppe
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3		0 Mandant	
PATNR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	PATNR	CHAR	10		0 IS-H: Patientenummer	
EINRI	<input type="checkbox"/>	<input type="checkbox"/>	EINRI	CHAR	4		0 IS-H: Einrichtung	
GSCHL	<input type="checkbox"/>	<input type="checkbox"/>	GSCHL	CHAR	1		0 IS-H: Geschlechtskennzeichen - intern	
NNAME	<input type="checkbox"/>	<input type="checkbox"/>	NNAME_PAT	CHAR	30		0 IS-H: Nachname Patient	
NNAMS	<input type="checkbox"/>	<input type="checkbox"/>	NNAMES_PAT	CHAR	80		0 IS-H: standardisierter Nachname	
VNAME	<input type="checkbox"/>	<input type="checkbox"/>	VNAME_PAT	CHAR	30		0 IS-H: Vorname Patient	
VNAMS	<input type="checkbox"/>	<input type="checkbox"/>	VNAMES_PAT	CHAR	80		0 IS-H: standardisierter Vorname	
TITEL	<input type="checkbox"/>	<input type="checkbox"/>	RI_TITEL	CHAR	15		0 IS-H: Titel	
NAMZU	<input type="checkbox"/>	<input type="checkbox"/>	RI_NAMZU	CHAR	15		0 IS-H: Namenszusatz	
VORSW	<input type="checkbox"/>	<input type="checkbox"/>	RI_VORSW	CHAR	15		0 IS-H: Vorsatzwort	
NAME2	<input type="checkbox"/>	<input type="checkbox"/>	NAME2_PAT	CHAR	30		0 IS-H: Pseudonym, Ordensname	
GBDAT	<input type="checkbox"/>	<input type="checkbox"/>	RI_GBDAI	DATS	8		0 IS-H: Geburtsdatum	
GENAM	<input type="checkbox"/>	<input type="checkbox"/>	GENAM	CHAR	30		0 IS-H: Geburtsname	
GENAS	<input type="checkbox"/>	<input type="checkbox"/>	GENAMS_PAT	CHAR	80		0 IS-H: standardisierter Geburtsname	
GLAND	<input type="checkbox"/>	<input type="checkbox"/>	GLAND	CHAR	3		0 IS-H: Geburtsland	
TODKZ	<input type="checkbox"/>	<input type="checkbox"/>	PAT_TOD	CHAR	1		0 IS-H: Kennzeichen Patient verstorben	
TODDT	<input type="checkbox"/>	<input type="checkbox"/>	TODDV	DATS	8		0 IS-H: Todesdatum von	
TODZT	<input type="checkbox"/>	<input type="checkbox"/>	TODZV	TIMS	6		0 IS-H: Todesuhrzeit von	
TODDB	<input type="checkbox"/>	<input type="checkbox"/>	TODDB	DATS	8		0 ISH: Todesdatum_bis	
TODZB	<input type="checkbox"/>	<input type="checkbox"/>	TODZB	TIMS	6		0 ISH: Todesuhrzeit_bis	
TODUR	<input type="checkbox"/>	<input type="checkbox"/>	TODUR	CHAR	3		0 IS-H: Todesursache	
ANRED	<input type="checkbox"/>	<input type="checkbox"/>	RI_ANRDS	CHAR	2		0 IS-H: Anredeschlüssel	
FAMST	<input type="checkbox"/>	<input type="checkbox"/>	RI_FAMST	CHAR	1		0 IS-H: Familienstand	
KONFE	<input type="checkbox"/>	<input type="checkbox"/>	RI_KONFE	CHAR	2		0 IS-H: Konfession	
NATIO	<input type="checkbox"/>	<input type="checkbox"/>	RI_NATIO	CHAR	3		0 IS-H: Staatsangehörigkeit	
SPRAS	<input type="checkbox"/>	<input type="checkbox"/>	SPRAS_PAT	LANG	1		0 IS-H: Sprache Patient	
LAND	<input type="checkbox"/>	<input type="checkbox"/>	LAND_PAT	CHAR	3		0 IS-H: Land des Patientenwohnortes	
PSTLZ	<input type="checkbox"/>	<input type="checkbox"/>	PSTLZ_PAT	CHAR	10		0 IS-H: Postleitzahl Patient	

Abbildung 4.2.: Screenshot eines Teils der Tabelle NPAT aus dem SAP Diktionär des T21.

4. Ausgangslage am Universitätsklinikum Homburg

Transp. Tabelle: NFAL
 Kurzbeschreibung: IS-H: Fälle

Eigenschaften | Auslieferung und Pflege | **Felder** | Eingabehilfe/-prüfung | Währungs-/Mengenfelder

Suchhilfe | Eingebauter Typ | 3 / 89

Feld	Key	Ini...	Datenelement	Datentyp	Länge	DezS...	Kurzbeschreibung	Gruppe
FALNR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FALNR	CHAR	10		0 IS-H: Fallnummer	FALNR
FALLAR	<input type="checkbox"/>	<input type="checkbox"/>	FALLART	CHAR	1		0 IS-H: Fallart	
PATNR	<input type="checkbox"/>	<input type="checkbox"/>	PATNR	CHAR	10		0 IS-H: Patientenummer	PATNR
BEKAT	<input type="checkbox"/>	<input type="checkbox"/>	BEKAT	CHAR	6		0 IS-H: Behandlungskategorie	
ABRKZ	<input type="checkbox"/>	<input type="checkbox"/>	ABRKZ_FALL	CHAR	1		0 IS-H: Abrechnungskennzeichen eines Falles	
SICHV	<input type="checkbox"/>	<input type="checkbox"/>	SICHV	CHAR	1		0 IS-H: Kennzeichen für Sicherheitsverwahrung	
EINZG	<input type="checkbox"/>	<input type="checkbox"/>	EINZGEBIET	CHAR	9		0 IS-H: Einzugsgebiet	
KZTXI	<input type="checkbox"/>	<input type="checkbox"/>	TEXT_NFAL	CHAR	30		0 IS-H: Bemerkung zum Fall	
KUALF	<input type="checkbox"/>	<input type="checkbox"/>	LLFDKUA	CHAR	10		0 IS-H: Letzte vergebene Nummer Kostenübernahmeantrag	
BEWLF	<input type="checkbox"/>	<input type="checkbox"/>	LLFDBEW	NUMC	5		0 IS-H: Letzte vergebene Bewegungsnummer	
DGNLF	<input type="checkbox"/>	<input type="checkbox"/>	LLFDDIA	NUMC	3		0 IS-H: Obsolet	
PGRLF	<input type="checkbox"/>	<input type="checkbox"/>	LLFDPGF	NUMC	3		0 ISH: Letzte Laufende Nummer für Patientengruppe (NPGF)	
INFKZ	<input type="checkbox"/>	<input type="checkbox"/>	INFKZ	CHAR	1		0 Infektionskennzeichen	
STATU	<input type="checkbox"/>	<input type="checkbox"/>	FALLSTATUS	CHAR	1		0 IS-H: Fallstatus	
FZIFF	<input type="checkbox"/>	<input type="checkbox"/>	FZIFF_FALL	CHAR	1		0 IS-H: Prüfziffer Fall	
NOTAN	<input type="checkbox"/>	<input type="checkbox"/>	NOTAUF	CHAR	1		0 IS-H: Notaufnahmekennzeichen	
KRZAN	<input type="checkbox"/>	<input type="checkbox"/>	KURZAUF	CHAR	1		0 IS-H: Kurzaufnahmekennzeichen	
ENDAI	<input type="checkbox"/>	<input type="checkbox"/>	ENDAI	DATS	8		0 IS-H: Datum der Entbindung	
ENTIM	<input type="checkbox"/>	<input type="checkbox"/>	ENTZEIT	TIMS	6		0 IS-H: Zeit der Entbindung	
FGTYP	<input type="checkbox"/>	<input type="checkbox"/>	GESTYP	CHAR	1		0 IS-H: Entbindungstyp	
KZKOM	<input type="checkbox"/>	<input type="checkbox"/>	ISH_KZKOM	CHAR	1		0 IS-H: Kennzeichen, daß es zu Komplikationen gekommen ist	
KOMTX	<input type="checkbox"/>	<input type="checkbox"/>	TEXT_ENTB	CHAR	50		0 IS-H: Bemerkung zur Entbindung	
KOLTX	<input type="checkbox"/>	<input type="checkbox"/>	RI_LGTXI	CHAR	1		0 IS-H: Kennzeichen, ob Langtext vorhanden ist	
ARBUN	<input type="checkbox"/>	<input type="checkbox"/>	ARBUN	DATS	8		0 IS-H: Arbeitsfähigkeits-Bis-Datum	
ENDDI	<input type="checkbox"/>	<input type="checkbox"/>	ENDDI_FAL	DATS	8		0 IS-H: Endedatum des Falles	
FSPER	<input type="checkbox"/>	<input type="checkbox"/>	FSPER	CHAR	1		0 IS-H: Fakturasperre eines Falles	
ERDAI	<input type="checkbox"/>	<input type="checkbox"/>	RI_ERDAI	DATS	8		0 IS-H: Datum, an dem der Satz hinzugefügt wurde	
ERUSR	<input type="checkbox"/>	<input type="checkbox"/>	RI_ERNAM	CHAR	12		0 IS-H: Name des Sachbearbeiters, der den Satz hinzugefügt hat	
UPDAI	<input type="checkbox"/>	<input type="checkbox"/>	RI_UPDAI	DATS	8		0 IS-H: Änderungsdatum	

Abbildung 4.3.: Screenshot eines Teils der Tabelle NFAL aus dem SAP Diktionär des T21 mit Schlüsselfeld *FALNR* (Fallnummer). Die Tabelle enthält mit *PATNR* auch die dem Fall zugehörige Patientenummer.

Tabelle-Feld	Bedeutung
<i>nbew-einri</i>	Einrichtung (Kürzel)
<i>nbew-falnr</i>	Fallnummer
<i>nbew-ldnr</i>	Laufende Nummer einer Bewegung
<i>nbew-orgpf</i>	Stationäre OrgEinheit, die einem Fall zugewiesen wird
<i>nbew-zimnr</i>	BauId eines Zimmers
<i>nbew-bett</i>	BauId eines Bettenstellplatzes

Tabelle 4.3.: Werte für die Typzeile der zu befüllenden Tabelle für das Programm `selectoptions_demo1`. Die entsprechenden Werte in der Diktionär-Tabelle sind in Abbildung 4.4 auf der nächsten Seite und Abbildung 4.5 auf Seite 49 zu sehen.

welches die letzte vergebene Bewegungsnummer darstellt. Diese “Bewegungen” betrachte ich im folgenden genauer. Sollte ein Patient im Krankenhaus aufgenommen worden sein und sich im Rahmen einer ambulanten oder stationären Aufnahme bzw. Verlegung befinden [55], so erhält er eine zum Fall fortlaufende Bewegungsnummer. Die Verlegung auf eine andere Station ist ebenfalls eine Patientenbewegung. Hierbei werden Zeit und Datum zum Anfang der Bewegung erfasst, ebenso wie Zeit und Datum am Ende der Bewegung. Die Patientenbewegungen aller Patienten werden in der Tabelle NBEW (Bewegung zum Fall, siehe Abbildung 4.4 auf der nächsten Seite und Abbildung 4.5 auf Seite 49) erfasst. Es bietet sich daher an, für das Szenario nach den aktuellen Bewegungen auf oder zur KK-05 zu suchen.

4.2.2. Praktische Umsetzung

Mit der vorherigen Beschreibung in Unterabschnitt 4.2.1 auf Seite 44 und den genannten Patienten in Abschnitt 4.2 auf Seite 44 kann man ein erstes Selektionskriterium für eine Abfrage in ABAP entwickeln, um die entsprechenden Daten zu erfassen. An dieser Stelle liegt zur Verdeutlichung der Fokus auf der Eingrenzung der Daten auf die relevanten Bewegungen der Testpatienten des T21, so dass vorerst lediglich einige Datenfelder aus NBEW in eine interne Tabelle [79] gefüllt werden. Die weitere Erfassung von Daten zu der eingegrenzten Menge wird in Kapitel 5 auf Seite 55 behandelt.

Die interne Tabelle in ABAP soll aus Zeilen (i.e., Struktur in ABAP) bestehen, wie sie in Tabelle 4.3 beschrieben sind [61]. Dazu muss in ABAP zuerst dieser Typ [79] definiert werden, um anschließend daraus eine Tabelle zu erstellen. Eine Instanz einer solchen

4. Ausgangslage am Universitätsklinikum Homburg

Transp. Tabelle: NBEW
 Kurzbeschreibung: IS-H: Bewegungen zum Fall

Eigenschaften | Auslieferung und Pflege | **Felder** | Eingabehilfe/-prüfung | Währungs-/Mengenfelder

Suchhilfe | Eingebauter Typ | 1 / 114

Feld	Key	Ini...	Datenelement	Datentyp	Länge	DezS...	Kurzbeschreibung	Gruppe
MANDI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDI	CLNT	3		0 Mandant	
EINRI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	EINRI	CHAR	4		0 IS-H: Einrichtung	EINRI
FALNR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FALNR	CHAR	10		0 IS-H: Fallnummer	FALNR
LFDNR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	LFDNR	NUMC	5		0 IS-H: Laufende Nummer einer Bewegung	LFDNR
BEWTY	<input type="checkbox"/>	<input type="checkbox"/>	BEWTY	CHAR	1		0 IS-H: Bewegungstyp	
BWART	<input type="checkbox"/>	<input type="checkbox"/>	RI_BWART	CHAR	2		0 IS-H: Bewegungsart	
BWIDI	<input type="checkbox"/>	<input type="checkbox"/>	BWIDI	DATS	8		0 IS-H: Datum der Bewegung	
BWIZT	<input type="checkbox"/>	<input type="checkbox"/>	BWIZT	TIMS	6		0 IS-H: Uhrzeit der Bewegung	
PLANB	<input type="checkbox"/>	<input type="checkbox"/>	PLANB	CHAR	1		0 IS-H: Plankennzeichen Datum der Bewegung	
STATU	<input type="checkbox"/>	<input type="checkbox"/>	ISH_ABSTIN	CHAR	2		0 IS-H: Interner Status eines ambulanten Besuchs	
BWPD1	<input type="checkbox"/>	<input type="checkbox"/>	BWPD1	DATS	8		0 IS-H: Plan-Datum der Bewegung	
BWPZ1	<input type="checkbox"/>	<input type="checkbox"/>	BWPZ1	TIMS	6		0 IS-H: Plan-Zeit der Bewegung	
BWED1	<input type="checkbox"/>	<input type="checkbox"/>	BWED1	DATS	8		0 IS-H: Bewegungsenddatum, Beginndatum der Folgebewegung	
BWEZ1	<input type="checkbox"/>	<input type="checkbox"/>	BWEZ1	TIMS	6		0 IS-H: Bewegungsendzeit, Beginnzeit der Folgebewegung	
PLANE	<input type="checkbox"/>	<input type="checkbox"/>	PLANE	CHAR	1		0 IS-H: Plankennzeichen des Enddatums einer Bewegung	
LFDREF	<input type="checkbox"/>	<input type="checkbox"/>	LFDREF	NUMC	5		0 IS-H: Laufende Nummer der Folgebewegung	
TELENR	<input type="checkbox"/>	<input type="checkbox"/>	TELE_INI	CHAR	16		0 IS-H: Telefonnummer d. Patienten in der Einrichtung	
TVKZ	<input type="checkbox"/>	<input type="checkbox"/>	TVKZ	CHAR	1		0 IS-H: Kennzeichen für TV	
KZTXI	<input type="checkbox"/>	<input type="checkbox"/>	TEXT_BEW	CHAR	50		0 IS-H: Bemerkung zur Bewegung eines Falles	
LGTXI	<input type="checkbox"/>	<input type="checkbox"/>	RI_LGTXI	CHAR	1		0 IS-H: Kennzeichen, ob Langtext vorhanden ist	
NOTKZ	<input type="checkbox"/>	<input type="checkbox"/>	RI_NOTKZ	CHAR	1		0 IS-H: Notfalkennzeichen	
UNFKZ	<input type="checkbox"/>	<input type="checkbox"/>	UNFART	CHAR	3		0 IS-H: Unfallart	
UNFNR	<input type="checkbox"/>	<input type="checkbox"/>	UNFNR	CHAR	12		0 IS-H: Aktenzeichen / Bearbeitungsnummer für Unfall	
UNFOR	<input type="checkbox"/>	<input type="checkbox"/>	ORT01_UNF	CHAR	25		0 IS-H: Ort des Unfalls	
UNFZT	<input type="checkbox"/>	<input type="checkbox"/>	UNFZT	TIMS	6		0 IS-H: Uhrzeit des Unfalls	
UNFD1	<input type="checkbox"/>	<input type="checkbox"/>	UNFD1	DATS	8		0 IS-H: Unfalldatum	

Abbildung 4.4.: Screenshot eines Teils der Tabelle NBEW aus dem SAP Diktionär des T21. Markiert sind die Felder *EINRI*, *FALNR* und *LFDNR* aus Tabelle 4.3 auf der vorherigen Seite.

4.2. Grundlage zur Datengewinnung aus dem SAP-System

Transp. Tabelle: NBEW
 Kurzbeschreibung: IS-H: Bewegungen zum Fall

Eigenschaften | Auslieferung und Pflege | **Felder** | Eingabehilfe/-prüfung | Währungs-/Mengenfelder

Suchhilfe | Eingebauter Typ | 25 / 114

Feld	Key	Inl...	Datenelement	Datentyp	Länge	DezS...	Kurzbeschreibung	Gruppe
UNFZT	<input type="checkbox"/>		UNFZT	TIMS	6		0 IS-H: Uhrzeit des Unfalls	
UNFDI	<input type="checkbox"/>		UNFDI	DATS	8		0 IS-H: Unfalldatum	
UNFTX	<input type="checkbox"/>		TEXT_UNF	CHAR	50		0 IS-H: Bemerkung zum Unfall	
UNLTX	<input type="checkbox"/>		RI_LGTXI	CHAR	1		0 IS-H: Kennzeichen, ob Langtext vorhanden ist	
WLFRI	<input type="checkbox"/>		WLFRI	CHAR	2		0 IS-H: Priorität für die Vormerkungs-/Wartelistenverwaltung	
AUFDS	<input type="checkbox"/>		AUFDT_LAST	DATS	8		0 IS-H: spätestes Aufnahme datum	
ORGFA	<input type="checkbox"/>		NZUWFA	CHAR	8		0 IS-H: OrgEinheit, die einem Fall fachl. zugewiesen wird	
ORGPF	<input type="checkbox"/>		NZUWPF	CHAR	8		0 IS-H: OrgEinheit, die einem Fall zugewiesen wird	ORGPF
ZIMMR	<input type="checkbox"/>		ISH_ZIMMID	CHAR	8		0 IS-H: BauId eines Zimmers	ZIMMR
BETT	<input type="checkbox"/>		BETTID	CHAR	8		0 IS-H: BauId eines Bettenstellplatzes	BETT
PLANR	<input type="checkbox"/>		ISH_PLANR	CHAR	1		0 IS-H: Plankennzeichen für räumliche Zuweisung	
DAUER	<input type="checkbox"/>		ISH_DAUER	NUMC	5		0 IS-H: Voraussichtl. Aufenthalts- bzw. Behandlungsdauer	
EXTKH	<input type="checkbox"/>		EXTKH	CHAR	10		0 IS-H: Identifikation eines externen Krankenhauses	
ORGAU	<input type="checkbox"/>		ORGAU	CHAR	8		0 IS-H: OrgId der Aufnahme stelle	
EZUST	<input type="checkbox"/>		EZUST	CHAR	2		0 IS-H: Entlassungszustand	
TODUR	<input type="checkbox"/>		TODUR	CHAR	3		0 IS-H: Todesursache	
ARBUN	<input type="checkbox"/>		ARBUN	DATS	8		0 IS-H: Arbeitsunfähigkeits-Bis-Datum	
ABWGN	<input type="checkbox"/>		ISH_ABWGN	CHAR	1		0 IS-H: Kennzeichen, ob Abwesenheit genehmigt ('X')	
ERDAI	<input type="checkbox"/>		RI_ERDAI	DATS	8		0 IS-H: Datum, an dem der Satz hinzugefügt wurde	
ERUSR	<input type="checkbox"/>		RI_ERNAM	CHAR	12		0 IS-H: Name des Sachbearbeiters, der den Satz hinzugefügt hat	
UPDAT	<input type="checkbox"/>		RI_UPDAT	DATS	8		0 IS-H: Änderungsdatum	
UFUSR	<input type="checkbox"/>		RI_UFNAM	CHAR	12		0 IS-H: Name des ändernden Sachbearbeiters	
STORN	<input type="checkbox"/>		RI_STORN	CHAR	1		0 IS-H: Stornokennzeichen	
STUSR	<input type="checkbox"/>		STORN_USER	CHAR	12		0 IS-H: Name des stornierenden Sachbearbeiters	
STDAT	<input type="checkbox"/>		STORN_DAT	DATS	8		0 IS-H: Stornierungsdatum	
UNFRN	<input type="checkbox"/>		ISH_RETINR	CHAR	15		0 IS-H: Nummer des Rettungsdienstes bei Unfall	

Abbildung 4.5.: Screenshot eines Teils der Tabelle NBEW aus dem SAP Diktionär des T21. Markiert sind die Felder *ORGPF*, *ZIMMR* und *BETT* aus Tabelle 4.3 auf Seite 47.

Listing 4.1: Definition und Deklaration einer Struktur und einer internen Tabelle.

```
1 REPORT  zabo_selectoptions_demo1.
2 * Beschreibung eines (Zeilen)Typs
3 TYPES: BEGIN OF gty_micro ,
4         einri TYPE nbew-einri ,
5         falnr TYPE nbew-falnr ,
6         lfdnr TYPE nbew-lfdnr ,
7         orgpf TYPE nbew-orgpf ,
8         zimmr TYPE nbew-zimmr ,
9         bett  TYPE nbew-bett ,
10 END OF gty_micro .
11 DATA :
12 * Tabelleninstanz aus dem oben beschriebenen Typ
13         gt_micro TYPE TABLE OF gty_micro ,
14 * Instanz des obigen Typs
15         gs_micro TYPE gty_micro .
```

Zeile wird in den späteren Kapiteln noch benötigt werden, um die Werte innerhalb des Programms zu übertragen. In Listing 4.1 ist der entsprechende Programmcode zu sehen. Es wird folgendermaßen vorgegangen, um die Patienten für das Selektionskriterium zu erhalten: Es erfolgt eine Abfrage auf die Tabelle NBEW, um alle für diese Einrichtung und Station hinterlegten Bewegungen zu erhalten, welche zum Stichtag relevant sind. Da NBEW auch die zur Bewegung passende Fallnummer enthält, wird man damit später in Kapitel 5 auf Seite 55 den zum Fall zugehörigen Patienten und weitere Daten erhalten.

Zur Abfrage der Tabelle NBEW und für die Übertragung zu der Tabelle aus Listing 4.1 benötigt man zusätzliche Informationen, weil die Datenbank mit zahlreichen Patientenbewegungen gefüllt ist, welche in diesem Moment nicht von Bedeutung (e.g., Entlassungen) sind. Das Feld *Einrichtung* in NBEW kann mit 'UKSH' bedient werden und als stationäre Organisationseinheit ist die 'KK-05' von Interesse. Das aktuelle Datum erhält man mit Hilfe der Systeminformation *sy-datum*. In Listing 4.2 auf der nächsten Seite ist die Datenabfrage zu finden. Es werden die gesuchten Werte aus NBEW abgefragt und in die entsprechenden Felder der Zieltabelle übertragen. Dabei sind zuerst die Zeilen

Listing 4.2: Datenabfrage aus NBEW zur Befüllung der internen Tabelle aus Listing 4.1 auf der vorherigen Seite.

```
1 DATA so_einri LIKE nbew-einri value 'UKSH'.
2 DATA so_orgpf LIKE gs_micro-orgpf value 'KK-05'.
3 START-OF-SELECTION.
4 SELECT * FROM nbew INTO CORRESPONDING FIELDS OF TABLE
      gt_micro
5 WHERE einri = so_einri
6 AND orgpf = so_orgpf
7 *Bewegungsende nach oder am heutigen Tag
8 AND bwedt GE sy-datum
9 *Bewegungsanfang vor oder am heutigen Tag
10 AND bwidt LE sy-datum
11 *Aufnahme (1) oder Verlegung (3) oder Abwesenheitsende (7)
      zur orgpf
12 AND ( bewty = '1' OR bewty = '3' OR bewty = '7' )
13 AND storn = space "Keine Stornierungskennzeichnung
14 AND plane = space. "Keine Planungskennzeichnung
15 SORT gt_micro DESCENDING BY falnr.
16 DELETE ADJACENT DUPLICATES FROM gt_micro COMPARING falnr.
```

4. Ausgangslage am Universitätsklinikum Homburg

Report ZABO_selectoptions_demo1pur

Einr.	Fall	LfdNr.	Pfl. OE	Zimmer	Bett
UKSH	17779	1	KK-05	9-0358	9-0358A
UKSH	28251	1	KK-05	9-0350	9-0350A
UKSH	28562	1	KK-05	9-0355	9-0355B
UKSH	28760	1	KK-05	9-0354	9-0354B
UKSH	30245	1	KK-05	9-0351	9-0351C
UKSH	30647	1	KK-05	9-0356	9-0356A
UKSH	32159	1	KK-05	9-0359	9-0359A
UKSH	2206	17	KK-05	9-0358	9-0358B
UKSH	21959	4	KK-05	9-0351	9-0351B
UKSH	40145	2	KK-05	9-0351	9-0351A

Report ZABO_selectoptions_demo1pur

Einr.	Fall	LfdNr.	Pfl. OE	Zimmer	Bett
UKSH	28251	1	KK-05	9-0350	9-0350A
UKSH	40145	2	KK-05	9-0351	9-0351A
UKSH	21959	4	KK-05	9-0351	9-0351B
UKSH	30245	1	KK-05	9-0351	9-0351C
UKSH	28760	1	KK-05	9-0354	9-0354B
UKSH	28562	1	KK-05	9-0355	9-0355B
UKSH	30647	1	KK-05	9-0356	9-0356A
UKSH	17779	1	KK-05	9-0358	9-0358A
UKSH	2206	17	KK-05	9-0358	9-0358B
UKSH	32159	1	KK-05	9-0359	9-0359A

(a) Ergebnis der Abfrage mit fiktiven Patienten. (b) Ergebnis sortiert nach Zimmer und Bett, siehe Unterabschnitt 5.6.1 auf Seite 70.

Abbildung 4.6.: Darstellung der Abfrage aus Listing 4.2 auf der vorherigen Seite mit Hilfe des ABAP-List-Viewers in Listing 4.3 auf der nächsten Seite.

von Interesse, welche der Einrichtung und der gesuchten Organisationseinheit entsprechen. Zu Bewegungen in NBEW gibt es ein Anfangsdatum und eine Anfangsuhrzeit, sowie ein Datum und eine Uhrzeit für das Ende der Bewegung. Solange die Bewegung nicht beendet ist, ist der Wert des Enddatums der '31.12.9999'. Die gesuchten Patientenbewegungen dürfen vor dem aktuellen Datum nicht beendet (e.g., inzwischen auf andere Station verlegt) worden sein und müssen auch vor dem aktuellen Datum beginnen (e.g., kein Eintrag für den nächsten Tag). Es sind die Bewegungstypen relevant, welche eine Aufnahme oder Verlegung zu oder innerhalb der Station darstellen. Sollte ein Patient für eine Weile abwesend (e.g., Behördenbesuch) sein, entstehen Bewegungen zum Abwesenheitsbeginn und zum Abwesenheitsende, welche eine vorherige Bewegung (e.g., Aufnahme) gleichzeitig beenden. Daher wird zusätzlich nach Bewegungen mit dem Bewegungstyp Abwesenheitsende gesucht. Stornierte oder lediglich geplante Bewegungen sollen nicht abgefragt werden.

Theoretisch sind mehrfache Einträge zu einer Fallnummer in der internen Tabelle `gt_micro` möglich, falls ein Patient beispielsweise einen Ausflug macht und nach seiner Rückkehr am gleichen Tag entlassen wird. Diese Mehreinträge werden vor der weiteren Verarbeitung aus `gt_micro` gelöscht, siehe Unterabschnitt 5.6.1 auf Seite 70.

Listing 4.3: ABAP-List-Viewer zur Anzeige einer internen Tabelle innerhalb von SAP.
Ein Beispielresultat ist in Abbildung 4.6a auf der vorherigen Seite zu sehen.

```
1 *...
2 *Im Deklarationsteil
3 DATA: go_alv_grid TYPE REF TO cl_salv_table,
4 columns TYPE REF TO cl_salv_columns_list,
5 lo_display TYPE REF TO cl_salv_display_settings.
6 *...
7 *...
8 *Darstellung der Tabelle im SAP List Viewer
9 cl_salv_table=>factory( "Objektorientierte Version des ALV
10     IMPORTING      r_salv_table   = go_alv_grid
11     CHANGING       t_table        = gt_micro      ).
12 columns = go_alv_grid->get_columns( ).
13 columns->set_optimize( abap_true ).
14 lo_display = go_alv_grid->get_display_settings( ).
15 lo_display->set_stripped_pattern( abap_true ).
16 go_alv_grid->display( ).
```

Eine Visualisierung des Ergebnisses zur oben erklärten Abfrage im Testsystem des UKS ist in Abbildung 4.6a auf der vorherigen Seite zu sehen. Dies ist eine Darstellung mit Hilfe des ABAP-List-Viewers⁴ (ALV) [15] in seiner objektorientierten [43] Variante. Die entsprechend notwendigen Einträge im Deklarationsteil und der Programmcode selbst sind in Listing 4.3 zu finden. Mit Hilfe der gefundenen Bewegungsnummern und den damit bekannten Fallnummern, können nun in Kapitel 5 auf Seite 55 weitere Patientendaten erhoben werden.

⁴Die Bezeichnung SAP-List-Viewer findet auch Verwendung.

5. Erweiterte Datengewinnung für das Mikrobiologieszenario

Mit der Abfrage der relevanten Patientenbewegungen aus der Tabelle NBEW habe ich in Kapitel 4 auf Seite 39 die Grundlage für die Datenabfrage in diesem Szenario erklärt. Im folgenden werden, mithilfe der Fallnummer, dem Fall entsprechende Diagnoseinformationen in Abschnitt 5.1 auf Seite 58 abgefragt und in Abschnitt 5.2 auf Seite 63 um die Patientenstammdaten erweitert. Zudem werden die Bewegungen zur Aufnahme und Entlassung in Abschnitt 5.3 auf Seite 63 erhoben. Darauf folgt in Abschnitt 5.4 auf Seite 66 die komplexe Erfassung und Gewinnung der Medikamentengabe im jeweiligen Fall. Im Anschluss daran werden in Abschnitt 5.5 auf Seite 70 noch die restlichen Werte zur Einrichtung befüllt. Die weiteren Aspekte zu anderen Daten und Datentransport werden in Abschnitt 5.7 auf Seite 79 und Abschnitt 5.8 auf Seite 83 behandelt. In Abschnitt 5.9 auf Seite 85 beschreibe ich eine Verbesserung bei der Parametereingabe für manuelle Abfragen in SAP.

Zu Beginn von Programmcode ist in diesem Kapitel immer die entsprechende Ergänzung im Deklarationsteil vorangestellt, um den Deklarationsteil nicht jedes Mal neu abzubilden. Eine erweiterte Version des kompletten Programm ist in Anhang A auf Seite 149 zu finden. Die Vorgehensweise zur Datenabfrage bzw. der Programmcode in diesem Kapitel orientiert sich an einem besseren Verständnis für den Leser.

Wie in Abschnitt 4.1 auf Seite 39 beschrieben, sollen die gesuchten Patientendaten aus dem SAP-System exportiert werden können. Dies ist mit einer analog zu Abschnitt 4.2 auf Seite 44 verwendeten internen Tabelle möglich, weil man diese innerhalb von SAP mit Hilfe von ABAP entsprechend weiterverarbeiten kann. Daher kann man sie später als CSV-Datei exportieren. Dieses Vorgehen erläutere ich in Abschnitt 5.6 auf Seite 70 genauer.

Zuvor muss diese interne Tabelle noch um die in den Tabellen 4.1 auf Seite 42 und 4.2 auf Seite 43 geschilderten Werte erweitert werden, damit sie in den folgenden Abschnitten

entsprechend befüllt werden kann. Dazu wird ein neuer Strukturtyp für die Instanz der internen Tabelle definiert. Dies erfolgt bereits komplett in Listing 5.1, um in den folgenden Abschnitten einen einheitlichen Zeilentyp zu benutzen. Dabei wird noch Platz für ein Pseudonym aus Kapitel 6 auf Seite 87 gelassen.

Der definierte Strukturtyp für die Zeilen der neuen interne Tabelle setzt sich aus Feldern der Tabellen NPAT, NFAL, NBEW, TN01 (Einrichtungen), NDIA (Diagnosen), NKDI (Diagnosekataloge), NMATV (Erbrachte Materialien zum Fall), Struktur ISH_MM_CONSUMP_TXT (Texte zum fallbezogenen Materialverbrauch), T023T¹ (Bezeichnungen zu Warengruppen) und MARA (Allgemeine Materialdaten) zusammen. Eine Übersicht ist in Tabelle 5.1 auf Seite 59 zu sehen. Durch die Übernahme von Typen aus den Tabellen entsteht noch ein weiterer Vorteil: Da es sich bis auf eine Ausnahme um Standardtabellen in SAP handelt, ist eine einfachere Übertragbarkeit (siehe Kapitel 9 auf Seite 113) für andere Einrichtungen mit SAP HEALTHCARE gegeben.

Listing 5.1: Definition des Strukturtyps zur späteren Deklaration einer internen Tabelle in SAP für das Mikrobiologieszenario. Die ungekürzte Definition ist in Anhang A auf Seite 149 zu finden.

```
1  TYPES: BEGIN OF gty_micro ,
2  pseud TYPE c LENGTH 40, "Platz für ein Pseudonym"
3  patnr TYPE npat-patnr, "Patientennummer"
4  gschl TYPE npat-gschl, "Geschlecht"
5  gbdat TYPE npat-gbdat, "Geburtsdatum"
6  einri TYPE nfal-einri, "Einrichtung (Kürzel)"
7  *Informationen zur Einrichtung
8  einkb TYPE tn01-einkb, "Kurzname der Einrichtung"
9  einbz TYPE tn01-einbz, "Name der Einrichtung"
10 land TYPE tn01-land, "Länderschlüssel"
11 pstlz TYPE tn01-pstlz, "Postleitzahl"
12 ort TYPE tn01-ort, "Ort"
13 stras TYPE tn01-stras, "Straße und Hausnr"
14 instnr TYPE tn01-instnr, "Institutskennzeichen"
15 telf1 TYPE tn01-telf1, "Telefonnummer"
16 *Fallnummer
```

¹Keine Standardtabelle

17 falnr TYPE nfal-falnr,
18 **KH Hauptdiagnose [Hd] oder Fachabteilungshauptdiagnose*
19 hfdialfdnr TYPE ndia-ldnr, "[Hd] Laufende Nummer Diagnose
20 hfdkat1 TYPE ndia-dkat1, "[Hd] Schlüsselung einer Diagnose
21 hfdkey1 TYPE ndia-dkey1, "[Hd] Identifikationskürzel
Katalog
22 hfditxt TYPE ndia-ditxt, "[Hd] Freitext einer Diagnose
23 hferdat TYPE ndia-erdat, "[Hd] Datum
24 hfertim TYPE ndia-ertim, "[Hd] Uhrzeit
25 hfdtext1 TYPE nkdi-dtext1, "[Hd] Text der Diagnose
26 **Aufnahmediagnose [Ad]*
27 adm_dialfdnr TYPE ndia-ldnr, "[Ad] Laufende Nummer
Diagnose
28 *...
29 **Entlassungsdiagnose [Ed]*
30 dis_dialfdnr TYPE ndia-ldnr, "[Ed] Laufende Nummer
Diagnose
31 **Aktuelle Bewegung*
32 lfdnr TYPE nbew-ldnr, "Laufende Nummer einer Bewegung
33 bwidt TYPE nbew-bwidt, "Datum der Bewegung
34 bwizt TYPE nbew-bwizt, "Uhrzeit der Bewegung
35 bwedt TYPE nbew-bwedt, "Bewegungsendedatum
36 bwezt TYPE nbew-bwezt, "Bewegungsendezeit
37 **Stationäre Org.Einheit, die einem Fall zugewiesen wird*
38 orgpf TYPE nbew-orgpf, "Stationäre OrgEinheit
39 zimmr TYPE nbew-zimmr, "BauId eines Zimmers
40 bett TYPE nbew-bett, "BauId eines Bettenstellplatzes
41 **Admission Bewegung [Ab]*
42 admldnr TYPE nbew-ldnr, "[Ab] Laufende Nummer
43 *...
44 admbwgr1 TYPE nbew-bwgr1, "[Ab] Bewegungsgrund - 1.Stelle
45 **Entlassung [Eb]*
46 disldnr TYPE nbew-ldnr, "[Eb] Laufende Nummer

```
47 *...
48 *Medikation
49 matnr TYPE nmatv-matnr, "Materialnummer im KH
50 maktx TYPE ish_mm_consump_txt-maktx, "Materialkurztext
51 menge TYPE nmatv-menge, "(Angeforderte) Menge des
    Materials
52 meins TYPE nmatv-meins, "Mengeneinheit
53 *Verabreichung eines Materials
54 consdt TYPE nmatv-consdt, "Datum
55 constm TYPE nmatv-constm, "Uhrzeit
56 wgbez TYPE t023t-wgbez, "Warengruppenbezeichnung
57 verabrant TYPE c LENGTH 2, "Verabreichungsart, po oder iv
58 matkl TYPE mara-matkl, "Warengruppe
59 ish_atc TYPE mara-ish_atc, "ATC-Code
60 *Wirkstoffe
61 ish_agent1 TYPE mara-ish_agent1, "Wirkstoff1 (
    Hauptwirkstoff)
62 ish_ag1quant TYPE mara-ish_ag1quant, "[W1] Menge
63 ish_ag1unit TYPE mara-ish_ag1unit, "[W1] Mengeneinheit
64 ish_agent2 TYPE mara-ish_agent2, "Wirkstoff2
65 *...
66 nname TYPE npat-nname, "Name des Patienten
67 vname TYPE npat-vname, "Vorname des Patienten
68 *Laufende Nummer der Materialanforderung
69 lnrlm TYPE nmatv-lnrlm,
70 END OF gty_micro.
```

5.1. Informationen zur Diagnose

Die erste Befüllung der internen Tabelle `gt_micro` mit Daten aus NBEW erfolgt analog zu Kapitel 4 auf Seite 39. Zu beachten ist hier allerdings die Tatsache, dass die neue interne Tabelle mit dem Zeilentyp aus Listing 5.1 auf Seite 56 mehr Einträge (e.g., Uhrzeit der Bewegung) aus NBEW enthält, welche entsprechend befüllt werden. Es werden mit

Tabelle	Bedeutung
NPAT	Stammdaten Patient
NFAL	Fälle
NBEW	Bewegungen zum Fall
TN01	Einrichtungen
NDIA	Diagnosen
NKDI	Diagnosekataloge
NMATV	Erbrachte Materialien zum Fall
ISH_MM_CONSUMP_TXT (Struktur)	Texte zum fallbez. Materialverbrauch
T023T	Bezeichnungen zu Warengruppen
MARA	Allgemeine Materialdaten

Tabelle 5.1.: Bedeutung der Tabellen aus Listing 5.1 auf Seite 56.

diesem Vorgang ausschließlich die Tabellenspalten mit Werten befüllt, deren Bezeichner identisch (e.g., *bwizt*) zu NBEW ist. Mit Hilfe dieser Werte aus NBEW erhält man nun weitere Informationen (e.g., Diagnosen). Die Diagnosen sind in der Tabelle NDIA im ICD²-Code [96] abgelegt.

5.1.1. Hauptdiagnose

Zunächst werden die Informationen zur Krankenhausauptdiagnose abgefragt. Da unter Umständen im laufenden Betrieb eine Diagnose noch nicht als solche gekennzeichnet wurde, greift das Programm bei Ermangelung derselben auf die Fachabteilungshauptdiagnose zu. Die Diagnosen werden im lokalen System mit entsprechenden Kennzeichen versehen, um auf die Art der Diagnose (e.g., Operationsdiagnose) hinzuweisen. Dazu werden die entsprechenden Kennzeichen in der Tabelle NDIA mit Ja (i.e., 'X') oder Nein (i.e., SPACE) markiert.

Da in jeder Zeile, der weiter zu befüllenden internen Tabelle, eine Bewegung mit einem entsprechenden Fall steht, wird zum jeweiligen Fall die entsprechende Information zur Hauptdiagnose aus der Tabelle NDIA gesucht. Der entsprechende Text der Diagnose zum ICD-Code ist in einer anderen Tabelle (i.e., NKDI) hinterlegt, weswegen dieser Schritt erst später in Unterabschnitt 5.1.3 auf Seite 61 erfolgt.

Die Suche nach dem entsprechenden Eintrag in NDIA erfolgt mithilfe der Fallnummer. Hierbei ist zu beachten, dass man nicht versehentlich auf eine stornierte Diagnose

²International Classification of Diseases

zugreifen möchte. Sollte die Abfrage nach der Krankenhaushauptdiagnose nicht erfolgreich (c.q., *sy-subrc*³ ungleich 0) sein, wird auf die passende Fachabteilungshauptdiagnose zugegriffen. Sollte beides nicht vorhanden sein, werden die Werte nicht befüllt. Die entsprechende Abfrage ist in Listing 5.2 zu finden.

Listing 5.2: Abfrage zur Hauptdiagnose. Es wird für jeden Eintrag der zu befüllenden internen Tabelle die darin enthaltene Fallnummer verwendet, um die Information zur Diagnose aus NDIA zu gewinnen. Zuerst wird nach einer Krankenhaushauptdiagnose gesucht und beim Fehlen derselben die Fachabteilungshauptdiagnose. Danach werden die entsprechenden Werte auf die interne Tabelle übertragen.

```
1  *Deklarationsteil
2  DATA: t_ndia LIKE ndia.
3  *...
4  LOOP AT gt_micro INTO gs_micro.
5  SELECT SINGLE * FROM ndia INTO t_ndia
6  WHERE einri = so_einri
7  AND falnr = gs_micro-falnr "Fallnummer"
8  AND storn = space "Nicht storniert"
9  AND khdia = 'X'. "Kennzeichnung als Krankhaushauptdiagnose"
10 IF sy-subrc = 0. "SELECT war erfolgreich"
11 *Übertragung der neuen Werte in die Struktur
12  MOVE t_ndia-lfdnr TO gs_micro-hfdialfdnr.
13  MOVE t_ndia-dkat1 TO gs_micro-hfdkat1.
14  MOVE t_ndia-dkey1 TO gs_micro-hfdkey1.
15  MOVE t_ndia-ditxt TO gs_micro-hfditxt.
16  MOVE t_ndia-erdat TO gs_micro-hferdat.
17  MOVE t_ndia-ertim TO gs_micro-hfertim.
18 *Aktualisierung der zu befüllenden Tabelle mit der
    Struktur
19  MODIFY gt_micro FROM gs_micro.
20 ELSE. "SELECT war nicht erfolgreich"
21  SELECT SINGLE * FROM ndia INTO t_ndia
```

³Rückgabewert der SELECT-Anweisung

```
22 WHERE einri = so_einri
23 AND falnr = gs_micro-falnr
24 AND storn = space
25 AND fhdia = 'X'. "Fachabteilungshauptdiagnose
26 IF sy-subrc = 0. "SELECT war erfolgreich
27     MOVE t_ndia-ldnr TO gs_micro-hfdialfdnr.
28     MOVE t_ndia-dkat1 TO gs_micro-hfdkat1.
29     MOVE t_ndia-dkey1 TO gs_micro-hfdkey1.
30     MOVE t_ndia-ditxt TO gs_micro-hfditxt.
31     MOVE t_ndia-erdat TO gs_micro-hferdat.
32     MOVE t_ndia-ertim TO gs_micro-hfertim.
33     MODIFY gt_micro FROM gs_micro.
34 ENDIF.
35 ENDIF.
```

5.1.2. Aufnahme- und Entlassungsdiagnose

Die Information aus Unterabschnitt 5.1.1 auf Seite 59 existiert in NDIA ebenso zur Aufnahme- und Entlassungsdiagnose. Die dazu notwendige Abfrage in Listing 5.3 auf der nächsten Seite ist ähnlich der Abfrage in Listing 5.2 auf der vorherigen Seite, daher an dieser Stelle lediglich die Abfrage zur Aufnahmediagnose. Bei der Entlassungsdiagnose wird nach einem anderen Kennzeichen (i.e., *endia* (Zeile 6)) gesucht und dementsprechend das Ergebnis der Abfrage auf die dafür vorgesehenen Werte in der Struktur *gs_micro* übertragen.

5.1.3. Text zu International Classification of Diseases

Neben der Kodierung der ICD-10 Diagnosen sind auch die entsprechenden Bezeichnungen [47] im SAP hinterlegt. Diese aus Tabelle NKDI abzufragen ist der nächste Schritt. Dazu wird jede Zeile der zu befüllenden internen Tabelle durchgegangen und zu jedem Katalog (i.e., *dkat*) mit Schlüssel der Diagnose (i.e., *dkey*) die entsprechend hinterlegte Bezeichnung in NKDI gesucht. Zwar werden lokal nur die deutschsprachigen Bezeichnungen sicher gepflegt, dennoch sollte bei der Abfrage die explizite Nennung der Sprache verwendet werden. Da es hier ausschließlich um einen Wert (c.q., *dtext1*) aus NKDI geht,

Listing 5.3: Abfrage zur Aufnahmediagnose.

```
1  *Aufnahmediagnose
2  SELECT SINGLE * FROM ndia INTO t_ndia
3  WHERE einri = so_einri
4  AND falnr = gs_micro-falnr
5  AND storn = space
6  AND afdia = 'X'.
7  IF sy-subrc = 0.
8    MOVE t_ndia-lfdnr TO gs_micro-adm_dialfdnr.
9    MOVE t_ndia-dkat1 TO gs_micro-adm_dkat1.
10   MOVE t_ndia-dkey1 TO gs_micro-adm_dkey1.
11   MOVE t_ndia-ditxt TO gs_micro-adm_ditxt.
12   MOVE t_ndia-erdat TO gs_micro-adm_erdat.
13   MOVE t_ndia-ertim TO gs_micro-adm_ertim.
14   MODIFY gt_micro FROM gs_micro.
15  ENDIF.
```

wird direkt auf diesen zugegriffen. Der Code für diese Beschreibung ist in Listing 5.4 auf der nächsten Seite zu finden.

5.2. Stammdaten zum Patienten

Mit den bereits zuvor gewonnenen Fallnummern kann man über die Tabelle NFAL auch die den Fällen entsprechenden Patientennummern finden und diese in der internen Tabelle ergänzen, siehe dazu den Programmcode in Listing 5.5 auf Seite 65.

Es sei an dieser Stelle erwähnt, dass man nicht jedes Mal alle Zeilen aus der zu befüllenden Tabelle mit einem *Loop*⁴ [42] neu durchgehen muss. Dies geschieht hier zur besseren Übersicht. Mit der geladenen Zeile aus der internen Tabelle kann man nach der Abfrage einer Diagnose anschließend die obige Abfrage der Patientnummer starten. Daher ist bei dem im Anhang A auf Seite 149 befindlichen Code des kompletten Programms diese Gewinnungsoperation schon im gleichen *Loop* enthalten, in welchem auch die Diagnosen abgefragt werden.

Die Patientnummer stellt neben der Fallnummer eine weitere Identifizierungsmöglichkeit von Patienten dar. Ich beschäftige mich daher in Kapitel 6 auf Seite 87 mit einer bereits in diesem Programm möglichen Pseudonymisierung der Daten.

Mithilfe der oben erhaltenen Patientnummer lassen sich aus der Tabelle NPAT die gesuchten Name, Vorname, Geburtsdatum und Geschlecht gewinnen, wie in Listing 5.6 auf Seite 65 zu sehen ist.

5.3. Patientenbewegungen

Bisher wurde mit den gewonnenen Daten (e.g., Fallnummer) zur aktuellen Bewegung auf weitere Tabellen zugegriffen. Zur Gewinnung von Bewegungsinformationen hinsichtlich der Aufnahme und Entlassung des Patienten wird die zuvor aus NBEW gewonnene Fallnummer verwendet, um damit wiederum die weiteren zur Fallnummer passenden Bewegungen aus NBEW zu erhalten. Die Daten zur aktuellen Bewegung werden bereits mit der in Kapitel 4 auf Seite 39 erläuterten Abfrage eingebunden.

Zur Verdeutlichung weiterer Möglichkeiten, frage ich an dieser Stelle in Listing 5.7 auf Seite 67 für die Aufnahme auch die ersten Stellen des Bewegungsgrunds (e.g., '05'

⁴Programmschleife

Listing 5.4: Suche nach den Bezeichnungen zu ICD-Codes, welche in den Abfragen in Listing 5.2 auf Seite 60 und 5.3 auf Seite 62 gewonnen wurden.

```
1  *Einmaliges Durchgehen der internen Tabelle
2  LOOP AT gt_micro INTO gs_micro.
3  *Hauptdiagnose
4  SELECT dtext1 FROM nkdi INTO gs_micro-hfdtext1
5  WHERE spras = 'DE'
6  AND dkat = gs_micro-hfdkat1
7  AND dkey = gs_micro-hfdkey1.
8  ENDSELECT.
9  IF sy-subrc = 0. "Erfolgreich?
10  MODIFY gt_micro FROM gs_micro. "Update der Tabelle
11  ENDIF.
12  *Aufnahmediagnose
13  SELECT dtext1 FROM nkdi INTO gs_micro-adm_dtext1
14  WHERE spras = 'DE'
15  AND dkat = gs_micro-adm_dkat1
16  AND dkey = gs_micro-adm_dkey1.
17  ENDSELECT.
18  IF sy-subrc = 0.
19  MODIFY gt_micro FROM gs_micro.
20  ENDIF.
21  *Entlassungsdiagnose
22  SELECT dtext1 FROM nkdi INTO gs_micro-dis_dtext1
23  WHERE spras = 'DE'
24  AND dkat = gs_micro-dis_dkat1
25  AND dkey = gs_micro-dis_dkey1.
26  ENDSELECT.
27  IF sy-subrc = 0.
28  MODIFY gt_micro FROM gs_micro.
29  ENDIF.
30  ENDLOOP.
```


Listing 5.5: Gewinn der Patientennummer zum Fall über die Tabelle NFAL.

```
1  *Deklarationsteil
2  DATA: t_fall LIKE nfal.
3  *...
4  LOOP AT gt_micro INTO gs_micro.
5  SELECT SINGLE * FROM nfal INTO t_fall
6  WHERE falnr = gs_micro-falnr.
7  IF sy-subrc = 0.
8    MOVE t_fall-patnr TO gs_micro-patnr.
9    MODIFY gt_micro FROM gs_micro.
10 ENDIF.
11 ENDLOOP.
```

Listing 5.6: Gewinnung von personenbezogenen Daten.

```
1  *Deklarationsteil
2  DATA: t_rednpat LIKE npat.
3  *...
4  LOOP AT gt_micro INTO gs_micro.
5  SELECT SINGLE * FROM npat INTO t_rednpat
6  WHERE patnr = gs_micro-patnr.
7  IF sy-subrc = 0.
8    MOVE t_rednpat-gschl TO gs_micro-gschl.
9    MOVE t_rednpat-gbdat TO gs_micro-gbdat.
10   MOVE t_rednpat-nname TO gs_micro-nname.
11   MOVE t_rednpat-vname TO gs_micro-vname.
12   MODIFY gt_micro FROM gs_micro.
13 ENDIF.
14 ENDLOOP.
```

bedeutet Entbindung) ab.

5.4. Informationen zur Medikation

Bei der Auflistung der gesuchten Werte zur Medikation in Tabelle 4.2 auf Seite 43 und der Erstellung des Zeilentyps in Listing 5.1 auf Seite 56 stellt sich die Frage, was bei mehreren Medikamenten bzw. Medikamentengaben passiert. Dazu verwende ich jeweils die bisher gewonnene Zeile als Ausgangsbasis und für jede weitere Medikamentengabe füge ich diese Zeile mit den entsprechenden Informationen (e.g., Verabreichungsart) hinzu. Diese zu einem Patienten jeweils mehrzeiligen Daten sind wiederverwendbar, wie ich in Kapitel 7 auf Seite 95 erkläre. Die neu geschaffenen Zeilen in der neuen internen Tabelle `gt_final` können natürlich auch weiterhin mit den oben gezeigten Operationen befüllt werden, wie in diesem Abschnitt ebenfalls gezeigt wird. Ich rate dazu, möglichst viele Operationen (e.g., Patientenstammdaten) zur Informationsgewinnung schon vor der Erweiterung mit der Medikation zu erledigen. Ansonsten können, aufgrund entstandener Mehrzeiligkeit pro Fall, sich wiederholende redundante Abfragen entstehen.

Der erste Schritt ist hier die Einbindung der NMATV, aus welcher der Materialverbrauch zum Fall gelesen wird. An dieser Stelle wird die bisherige interne Tabelle (i.e., `gt_micro`) als Ausgangspunkt verwendet und die neue interne Tabelle (i.e., `gt_final`) gleichen Typs nach und nach jeweils pro Fall befüllt. Es wird für jede Zeile aus `gt_micro` durch Abfrage (i.e., Einrichtung und Fallnummer) auf NMATV eine temporäre ebenbürtige interne Tabelle (i.e., `gt_nmatv`) befüllt, welche den Materialverbrauch zu diesem Fall darstellt. Anschließend werden mit Hilfe der `gt_nmatv` wiederum die passenden Zeileneinträge zusammen mit der jeweiligen Zeile (i.e., `gs_micro`) aus `gt_micro` erzeugt und sie in die neue Tabelle (i.e., `gt_final`) eingetragen. Sollten sich keine passenden Einträge (e.g., keine Medikamentengabe) zum Fall in NMATV befinden, wird lediglich die ursprüngliche Zeile (i.e., `gs_micro`) aus `gt_micro` verwendet.

Für den gesuchten Materialkurztext (c.q., Produktname) wird ein Funktionsaufruf (i.e., Funktion `ISH_MAT_CONSUMP_INFO`⁵) auf die Struktur `RNMCONS` (Anlegen/Ändern fallbezogener Materialverbrauch) verwendet. Auf diese Weise erhält man in Listing 5.8 für jeden Materialverbrauch jeweils Produktnamen, Materialnummer, Menge, Mengeneinheit, Datum, Uhrzeit und Nummer der Materialanforderung.

⁵Dokumentation dazu aus SAP-T21: Texte, Infos zu einem fallbezogenen Materialverbrauch

Listing 5.7: Abfrage der Aufnahme- und Entlassungsbewegung aus NBEW mit Hilfe der Fallnummer.

```
1  *Deklarationsteil
2  DATA:
3  t_admbew LIKE nbew,
4  t_disbew TYPE nbew.
5  *...
6  LOOP AT gt_micro INTO gs_micro.
7  SELECT * FROM nbew INTO t_admbew
8  WHERE falnr = gs_micro-falnr
9  AND storn = space
10 AND bewty = '1' "Aufnahme
11 AND storn = space
12 AND plane = space.
13 ENDSELECT.
14 IF sy-subrc = 0.
15   MOVE t_admbew-lfdnr TO gs_micro-admlfdnr.
16   MOVE t_admbew-bwidt TO gs_micro-admbwidt.
17   MOVE t_admbew-bwizt TO gs_micro-admbwizt.
18   MOVE t_admbew-bwgr1 TO gs_micro-admbwgr1.
19   MODIFY gt_micro FROM gs_micro.
20 ENDIF.
21 SELECT * FROM nbew INTO t_disbew
22 WHERE falnr = gs_micro-falnr
23 AND storn = space
24 AND bewty = '2' "Entlassung
25 AND plane = space.
26 ENDSELECT.
27 IF sy-subrc = 0.
28   MOVE t_disbew-lfdnr TO gs_micro-dislfdnr.
29   MOVE t_disbew-bwidt TO gs_micro-disbwidt.
30   MOVE t_disbew-bwizt TO gs_micro-disbwizt.
31   MODIFY gt_micro FROM gs_micro.
32 ENDIF.
33 ENDLOOP.
```

Listing 5.8: Generierung einer neuen Tabelle mit Zeileneinträgen für den jeweiligen Materialverbrauch.

```
1  *Deklarationsteil
2  DATA:
3  gt_final TYPE TABLE OF gty_micro, "Neue interne Tabelle
4  gs_nmatv TYPE nmatv, "Struktur der Tabelle NMATV
5  gs_rncoms TYPE rnmcons, "Instanz der Struktur
6  gs_ish_mm_consumpt_txt TYPE ish_mm_consump_txt, "Instanz
   der Struktur
7  gt_nmatv LIKE STANDARD TABLE OF gs_nmatv. "Tabelle aus
   gs_nmatv
8  *...
9  LOOP AT gt_micro INTO gs_micro. "Zu jeder Zeile aus
   gt_micro
10 CLEAR gt_nmatv. "Zurücksetzen der temporären Tabelle
11 *Erfassung des Materialverbrauchs zu dieser Zeile in
   gt_nmatv
12 SELECT * FROM nmatv INTO CORRESPONDING FIELDS OF TABLE
   gt_nmatv
13 WHERE einri = so_einri
14 AND falnr = gs_micro-falnr "Fallnummer
15 AND menge NE space
16 AND storn = space.
17 IF gt_nmatv IS INITIAL. "Kein Materialverbrauch
18 APPEND gs_micro TO gt_final. "Falls keine Medikation
19 ENDIF.
20 IF sy-subrc = 0. "SELECT Anweisung erfolgreich
21 *Durchgehen aller Materialverbrauchseinträge zu diesem
   Fall
22 LOOP AT gt_nmatv INTO gs_nmatv.
23 *Text zum Medikament (Produkt)
24 MOVE-CORRESPONDING gs_nmatv TO gs_rncoms. "Befüllung der
   Abfragestruktur
```

```
25 CALL FUNCTION 'ISH_MAT_CONSUMP_INFO' "Funktionsaufruf"
26 EXPORTING
27   ss_rnmcons = gs_rncoms "Parameter zum Funktionsaufruf"
28   ss_einri   = so_einri
29 IMPORTING
30   ss_cons_txt = gs_ish_mm_consumpt_txt. "Rückmeldung"
31 *Eintrag in die aktuelle Zeile aus gt_micro
32   gs_micro-maktx = gs_ish_mm_consumpt_txt-maktx.
33 MOVE gs_nmatv-matnr TO gs_micro-matnr.
34 MOVE gs_nmatv-menge TO gs_micro-menge.
35 MOVE gs_nmatv-meins TO gs_micro-meins.
36 MOVE gs_nmatv-consdt TO gs_micro-consdt.
37 MOVE gs_nmatv-constm TO gs_micro-constm.
38 MOVE gs_nmatv-lnrlm TO gs_micro-lnrlm.
39 *Aktuelle Zeile aus gt_micro inkl. Informationen zum
   Materialverbrauch wird in die neue interne Tabelle
   eingetragen.
40 APPEND gs_micro TO gt_final.
41 ENDLOOP.
42 ENDIF.
43 ENDLOOP.
```

Die interessante Tabelle ist ab diesem Zeitpunkt die `gt_final`, da darin die Informationen zur Medikamentenvergabe enthalten sind. Es fehlen bei der Medikation allerdings noch die Angaben zur Substanzgruppe und den Wirkstoffen bzw. deren Dosierung und die Art der Verabreichung (e.g., intravenös [70]). Die aus der Tabelle `NMATV` gewonnene Materialnummer verwende ich, um am Ende dieses Abschnitts die gesuchten Informationen aus den allgemeinen Materialdaten zu erhalten.

Es werden am UKS in einer eigenen Tabelle Texte für die Substanzgruppe abgelegt. Diese Texte enthalten ein freistehendes "iv", falls das Medikament intravenös verabreicht werden soll. Im anderen Falle erfolgt die Verabreichung per oral ("po"). Diese Abkürzungen sollen für die jeweiligen Medikamente in die Tabelle `gt_final`, daher ist bei der Definition des Zeilentyps in Listing 5.1 auf Seite 56 bereits das Textfeld *verabrt* mit zweistelliger Länge definiert. Zur Gewinnung der Verabreichungsart wird mit Hilfe der aus

der Tabelle MARA erhaltenen Warengruppe (*MATKL*) auf die Tabelle T023T zugegriffen. In der Bezeichnung für die Warengruppe (*WGBEZ*) wird mit dem Befehl *Contains String* [48] nach dem erwähnten freistehenden “iv“ gesucht.

Der Programmcode für das oben Beschriebene ist in Listing 5.9 auf der nächsten Seite zu finden. Hier verwende ich zum ersten Mal den Befehl *move-corresponding*, wodurch gleichnamige Komponenten zwischen Strukturen zugewiesen werden können.

5.5. Andere Daten bzw. Informationen

Zur Einrichtung selbst fehlen noch Informationen (e.g., Adresse, Institutskennzeichen). Mit den in diesem Kapitel benannten Methoden ist dies schnell zu ergänzen, da man die entsprechenden Informationen mit dem Kürzel der Einrichtung abfragen kann, siehe dazu Listing 5.10 auf Seite 72. Die Befüllung des Feldes für das Pseudonym wird in Kapitel 6 auf Seite 87 erklärt.

5.6. Ergebnisausgabe

Bisher habe ich in diesem Kapitel u.a. die Befüllung einer internen Tabelle beschrieben, welche die in Abschnitt 4.1 auf Seite 39 geforderten Werte enthält. In diesem Abschnitt erläutere ich drei Möglichkeiten zur Ergebnisausgabe und vergleiche anschließend deren Vorteile in Unterabschnitt 5.6.6 auf Seite 79. Die Vorbereitung der internen Tabelle zur Ergebnisausgabe wird in Unterabschnitt 5.6.1 erklärt.

5.6.1. Verarbeitung der internen Tabelle

Die vorliegende interne Tabelle im ABAP-Speicher kann vor der Ergebnisausgabe inhaltlich verarbeitet werden. Die gezeigte Darstellung des Ergebnisses der Abfrage aus Unterabschnitt 4.2.2 auf Seite 47 in Abbildung 4.6a auf Seite 52 kann man anpassen, indem man die Tabelle nach Zimmer und Bett sortiert. Dazu muss die interne Tabelle zuerst sortiert werden (siehe Listing 5.11 auf Seite 72) und kann daraufhin entsprechend abgespeichert oder im ALV gezeigt werden, wie in Abbildung 4.6b auf Seite 52 dargestellt. In Listing 4.2 auf Seite 51 erfolgt eine Sortierung der Tabelle, um anschließend mit *DELETE ADJACENT DUPLICATES* mehrfache Einträge zur gleichen Fallnummer zu löschen.

Listing 5.9: Ergänzung der Daten zur Medikation.

```
1  *Deklarationsteil
2  DATA:
3  t_mara LIKE mara,
4  l_wgbez TYPE wgbez.
5  *...
6  LOOP AT gt_final INTO gs_micro.
7  *Zugriff auf MARA (Allgemeine Materialdaten)
8  SELECT SINGLE * FROM mara INTO t_mara
9  WHERE matnr = gs_micro-matnr.
10 IF sy-subrc = 0. "SELECT auf MARA erfolgreich
11   MOVE-CORRESPONDING t_mara TO gs_micro. "inkl. matkl
12 *Zugriff auf T023T (Bezeichnungen zu Warengruppen)
13 *mit der aus mara gewonnen matkl (Warengruppe)
14   SELECT SINGLE wgbez INTO l_wgbez FROM t023t
15   WHERE matkl = t_mara-matkl.
16   IF sy-subrc = 0. "SELECT auf T023T erfolgreich
17     gs_micro-wgbez = l_wgbez. "Substanzgruppe
18     IF l_wgbez CS 'iv'. "Compare String
19       gs_micro-verabrart = 'iv'. "Gefunden
20     ELSE.
21       gs_micro-verabrart = 'po'. "Nicht gefunden
22     ENDIF.
23   ENDIF.
24   MODIFY gt_final FROM gs_micro.
25 ENDIF.
26 ENDLOOP.
```

Listing 5.10: Abruf von Information zur Einrichtung.

```
1  *Deklarationsteil
2  DATA: t_tn01 LIKE tn01.
3  *...
4  LOOP AT gt_final INTO gs_micro.
5  SELECT SINGLE * FROM tn01 INTO t_tn01
6  WHERE einri = gs_micro-einri.
7  IF sy-subrc = 0.
8    MOVE-CORRESPONDING t_tn01 TO gs_micro.
9    MODIFY gt_final FROM gs_micro.
10 ENDIF.
11 ENDLOOP.
```

Listing 5.11: Sortierung der in Listing 4.2 auf Seite 51 befüllten Tabelle.

```
1  *Nach der Befüllung der Tabelle gt_micro
2  SORT gt_micro by zimmr bett ASCENDING.
3  *Vor der Ausgabe
```


Listing 5.12: Übertragung der internen Tabelle in einen Text mit einem Separator für die Werte.

```
1  *Deklarationsteil
2  DATA: it_csvdata TYPE truxs_t_text_data.
3  *...
4  CALL FUNCTION 'SAP_CONVERT_TO_TEX_FORMAT'
5  EXPORTING
6    i_field_seperator      = '|'
7  TABLES
8    i_tab_sap_data         = gt_final
9  CHANGING
10   i_tab_converted_data   = it_csvdata
```

Einige Werte (e.g., Warengruppe) werden zur Erfassung anderer Werte benötigt, aber sind für die externe Weiterverarbeitung uninteressant. Eine Möglichkeit zur Teilausgabe von Werten ist die Definition und Deklaration einer weiteren internen Tabelle, welche lediglich die auszugebenden Werte enthält. In dem Fall benutzt man nach Beendigung der Datenbankabfragen den Befehl *move-corresponding*, um über die Zeilenstruktur die Werte von der bisherigen internen Tabelle in die neue verkleinerte Tabelle zu übertragen.

Bisher liegt eine interne Tabelle im ABAP-Speicher vor. Bevor diese als CSV-Datei abgelegt werden kann, müssen deren Zeileninformationen noch als Textausgabe angepasst werden. In ABAP verwende ich dazu die Funktion *SAP CONVERT TO TEX FORMAT*. Unter Angabe eines Separators (i.e., Trennzeichen) für die einzelnen Werte kann die interne Tabelle `gt_final` entsprechend angepasst bzw. in den mehrzeiligen Text `it_csvdata` übertragen werden. Als Separator wähle ich hier das Zeichen "|", siehe dazu Listing 5.12.

5.6.2. Darstellung im SAP List Viewer

Die Priorität für das Mikrobiologieszenario besteht zwar im Export einer CSV-Datei, aber bereits zu Kontrollzwecken ist die Möglichkeit einer visuellen Darstellung des Ergebnisses nützlich. Eine schnelle Ausgabe kann hier analog zu dem in Unterabschnitt 4.2.2 auf Seite 47 erklärten Listing 4.3 auf Seite 53 mit dem ALV erfolgen. Da hier allerdings

72 Spalten dargestellt werden müssen, ist eine Analyse damit erschwert. Entsprechende Anpassungsmöglichkeiten erläutere ich in Unterabschnitt 8.1.1 auf Seite 105. Die Abbildungen 5.1 auf der nächsten Seite und 5.2 auf Seite 76 zeigen eine komplette Ausgabe für die bisherigen Abfragen des Kapitels im ALV. Die verwendete interne Tabelle ist bereits sortiert, wie in der Zeile 356 des Listings A.1 auf Seite 149 im Anhang A zu sehen ist. Die Zeitangabe '00:00:00' erfolgt bei nicht befüllten Werten mit Zeitangaben, weil die Einträge Typ-formatiert sind. Die erste Spalte ist in der gezeigten Ausgabe leer, weil noch kein Pseudonym geliefert wurde.

5.6.3. Ausgabe als Datei auf dem Präsentationsserver

Die eigentliche CSV-Datei aus der internen Tabelle `gt_final` kann auf unterschiedliche Weise abgelegt werden. Hierbei ist relevant, ob die Erzeugung in der laufenden SAP-Schnittstelle (e.g., PC) auf dem Präsentationsserver [26] oder auf dem Applikationsserver (siehe Unterabschnitt 5.6.4) erfolgt. Für den Präsentationsserver wird an dieser Stelle die ABAP-Funktion `GUI_DOWNLOAD` verwendet, welche als Eingabe den erzeugten Text aus Unterabschnitt 5.6.1 auf Seite 70 und einen Dateinamen benötigt. Für den Dateinamen wird hier mit dem Befehl `CONCATENATE` das Präfix "microSAP_" mit anschließendem Datum und Uhrzeit aus der Systemangabe zusammengesetzt, versehen mit der Endung ".csv". Beides wird in Listing 5.13 auf Seite 77 realisiert.

Sollte die interne Tabelle im ABAP-List-Viewer dargestellt werden und dessen Ausgabe entsprechend (e.g., Optionale Speicherung als MIME HTML) eingerichtet sein, so ist für den Endnutzer innerhalb des ALV ein manuelles Speichern auf dem Präsentationsserver mit dem eingerichteten Format möglich. Darauf gehe ich in Kapitel 8 auf Seite 105 weiter ein. In diesem Fall ist der Inhalt und die Formatierung der Datei von der Einrichtung des Systems, der Anpassung des ALV im Programm und der individuellen Anpassung des Endnutzers abhängig.

5.6.4. Ausgabe als Datei auf dem Applikationsserver

Bei der Ausgabe auf dem Applikationsserver kann man die bereits in Unterabschnitt 5.6.1 auf Seite 70 vorgenommene Umwandlung mit dem Separator verwenden. Da man hier die Funktion `GUI_DOWNLOAD` nicht verwenden kann, wird der Text zeilenweise mit `OPEN DATASET` abgelegt. Es existieren in diesem Unternehmensinformationssystem von SAP mehrere Systeme (i.e., T21, C21), siehe auch Abschnitt 3.5 auf Seite 31. Daher treffe

Listing 5.13: Speicherung als CSV-Datei auf dem Präsentationsserver.

```
1  *Deklarationsteil
2  DATA: filename TYPE string.
3  *...
4  *Erzeugung des Dateinamens
5  CONCATENATE 'microSAP_' sy-datum sy-zeit '.csv' INTO
      filename.
6  *Ausgabe auf dem Präsentationsserver
7  CALL FUNCTION 'GUI_DOWNLOAD'
8  EXPORTING
9   filename      = filename
10 TABLES
11  data_tab      = it_csvdata.
```

ich Vorkehrungen, damit der Programmcode nach dem Transport aus dem Testsystem in das Produktivsystem nicht angepasst werden muss und die Dateien aus unterschiedlichen Systemen an die jeweils passende Stelle übertragen werden. Dazu wird der Name des SAP-Systems mit *sy-sysid* abgefragt. Alles zusammen ist in Listing 5.14 auf der nächsten Seite zu finden.

5.6.5. Beschreibung der Ausgabe

Die im ALV angezeigte interne Tabelle ist durch die Typdefinition der Zeile in Listing 5.1 auf Seite 56 deklariert. In der CSV-Datei sind diese Werte ebenso zu finden, separiert mit dem Zeichen "|". Die Zahlen sind allerdings nicht wie bei der Ausgabe mit ALV zu Anzeigezwecken beschönigend formatiert. Statt der Patientenummer "10001402" im ALV, wird man in der CSV-Datei den tatsächlichen Wert "0010001402" ohne Entfernung der führenden Nullen sehen. Gleiches gilt für die Fallnummer und die Nummern der Bewegungen. In Abbildung 5.3 auf Seite 79 ist ein Ausschnitt aus einer zur obigen ALV-Ausgabe passenden CSV-Datei zu finden. Sollte ein Wert nicht vorhanden sein, steht zwischen den Separatoren kein Wert bzw. lediglich die Standardformatierung (e.g., 00.00.0000 bei einem Datum). Um diese Datei wie in Kapitel 7 auf Seite 95 zur Weiterverarbeitung benutzen zu können, ist eine Erklärung zu den Stellen in den Spalten hilfreich. In der

Listing 5.14: Speicherung als CSV-Datei auf dem Applikationsserver.

```
1  *Deklarationsteil
2  DATA: systemname TYPE C LENGTH 3.
3  *...
4  *Ermittlung der System-ID
5  IF sy-sysid = 'C21'.
6     systemname = 'c21'. "Formatierung für den Speicherpfad
7  ELSEIF sy-sysid = 'T21'.
8     systemname = 't21'.
9  ENDIF.
10 *Erzeugung des Dateinamens
11 CONCATENATE '/sapuser/' systemname 'GEHEIM/microSAP_'
12             sy-datum sy-zeit '.csv' INTO filename.
13 *Ausgabe auf dem Applikationsserver
14 OPEN DATASET filename
15 IN LEGACY TEXT MODE FOR OUTPUT. "Speicherort öffnen
16 *Zeilenweise den Text in Zieldatei ablegen
17 LOOP AT it_csvdata into line_csvdata.
18     TRANSFER line_csvdata TO filename.
19 ENDLOOP.
20 CLOSE DATASET filename. "Speicherort schließen
```

```
|0010001402|2|01.01.1999|UKSH|UKS Homburg|...
|0010001402|2|01.01.1999|UKSH|UKS Homburg|...
|0010001402|2|01.01.1999|UKSH|UKS Homburg|...
|0010001402|2|01.01.1999|UKSH|UKS Homburg|...
|0010007692|1|14.12.2005|UKSH|UKS Homburg|...
|0010008733|1|01.01.1962|UKSH|UKS Homburg|...
|0010013213|1|14.03.1979|UKSH|UKS Homburg|...
```

...

Abbildung 5.3.: Ausschnitt mit fiktiven Patienten aus der CSV-Datei zur ALV-Ausgabe in Abbildung 5.1.

Tabelle 5.2 auf der nächsten Seite und der Tabelle 5.3 auf Seite 81 sind die Werte und die zugehörigen Stellen in der CSV-Datei aufgelistet. Den von Mehrzeiligkeit betroffenen Inhalt (i.e., Medikation) habe ich entsprechend markiert.

5.6.6. Vorteile der jeweiligen Ausgabe

Durch die ALV-Anzeige besteht der Vorteil des direkten Einblicks in die Werte, auch wenn es sich in diesem Fall um 72 Spalten handelt. Eine weitere Möglichkeit der Darstellung innerhalb von SAP wird in Kapitel 8 auf Seite 105 beschrieben. Der Export als Datei hat hingegen den Vorteil, dass man diese Datei wie in Kapitel 7 auf Seite 95 extern weiterverarbeiten oder wie in Kapitel 8 auf Seite 105 auf dem lokalen Rechner analysieren kann. Die Ausgabe auf dem Präsentationsserver ist besonders dann interessant, falls man dort entsprechende Analysetools zur Verfügung hat und direkt mit der Datei arbeiten möchte. Für eine Automatisierung der Weiterverarbeitung ist die Speicherung auf dem Applikationsserver vorzuziehen, da dieser in einem Krankenhaus - im Gegensatz zum Präsentationsserver - eine hohe Verfügbarkeit hat.

5.7. Externe Datensätze

Neben den in SAP generierten CSV-Dateien sind für dieses Szenario auch externe Daten von Interesse. In diesem Falle handelt es sich um Daten aus der Mikrobiologie, genauer gesagt um ein Zwischenformat in Form einer Datei (c.q., .lab-Datei). Ich erläutere in diesem Abschnitt eine Möglichkeit der Beiführung dieser Datensätze.

Die Mikrobiologie selbst ist ein Subsystem am UKS und stellt ein weiteres monolithisches System dar. Im normalen Krankenhausbetrieb werden angeforderte Werte bzw.

5. Erweiterte Datengewinnung für das Mikrobiologieszenario

<i>Stelle</i>	<i>Mehrzeilig</i>	<i>Bedeutung</i>
00		Pseudonym
01		Patientennummer
02		Geschlecht
03		Geburtsdatum
04		Einrichtung (Kürzel)
05		[Einrichtung] Kurzname der Einrichtung
06		[Einrichtung] Name der Einrichtung
07		[Einrichtung] Länderschlüssel
08		[Einrichtung] Postleitzahl
09		[Einrichtung] Ort
10		[Einrichtung] Straße und Hausnummer
11		[Einrichtung] Institutskennzeichen
12		[Einrichtung] Telefonnummer
13		Fallnummer
14		[Hauptdiagnose] Laufende Nummer Diagnose
15		[Hauptdiagnose] Schlüsselung einer Diagnose
16		[Hauptdiagnose] Identifikationskürzel für Diagnosekatalog
17		[Hauptdiagnose] Freitext einer Diagnose
18		[Hauptdiagnose] Datum, an dem der Satz hinzugefügt wurde
19		[Hauptdiagnose] Uhrzeit, zu der der Satz hinzugefügt wurde
20		[Hauptdiagnose] Text der Diagnose
21		[Aufnahmediagnose] Laufende Nummer Diagnose
22		[Aufnahmediagnose] Schlüsselung einer Diagnose
23		[Aufnahmediagnose] Identifikationskürzel für Diagnosekatalog
24		[Aufnahmediagnose] Freitext einer Diagnose
25		[Aufnahmediagnose] Datum, an dem der Satz hinzugefügt wurde
26		[Aufnahmediagnose] Uhrzeit, zu der der Satz hinzugefügt wurde
27		[Aufnahmediagnose] Text der Diagnose
28		[Entlassungsdiagnose] Laufende Nummer Diagnose
29		[Entlassungsdiagnose] Schlüsselung einer Diagnose
30		[Entlassungsdiagnose] Identifikationskürzel für Diagnosekatalog
31		[Entlassungsdiagnose] Freitext einer Diagnose
32		[Entlassungsdiagnose] Datum, an dem der Satz hinzugefügt wurde
33		[Entlassungsdiagnose] Uhrzeit, zu der der Satz hinzugefügt wurde
34		[Entlassungsdiagnose] Text der Diagnose
35		Laufende Nummer einer Bewegung

Tabelle 5.2.: Bedeutung der Einträge in der CSV-Datei, Teil 1. Der zweite Teil befindet sich in Tabelle 5.3 auf der nächsten Seite.

<i>Stelle</i>	<i>Mehrzeilig</i>	<i>Bedeutung</i>
36		Datum der Bewegung
37		Uhrzeit der Bewegung
38		Bewegungsendedatum
39		Bewegungsendezeit
40		OrgEinheit, die einem Fall zugewiesen wird
41		BauId eines Zimmers
42		BauId eines Bettenstellplatzes
43		[Aufnahmebewegung] Laufende Nummer einer Bewegung
44		[Aufnahmebewegung] Datum der Bewegung
45		[Aufnahmebewegung] Uhrzeit der Bewegung
46		[Aufnahmebewegung] Bewegungsgrund - 1. und 2. Stelle
47		[Entlassungsbewegung] Laufende Nummer einer Bewegung
48		[Entlassungsbewegung] Datum der Bewegung
49		[Entlassungsbewegung] Uhrzeit der Bewegung
50	X	Materialnummer in KH für Materialverbrauch
51	X	Materialkurztext
52	X	(angeforderte) Menge des Materials
53	X	Mengeneinheit
54	X	Datum der Verabreichung eines Materials
55	X	Uhrzeit der Verabreichung eines Materials
56	X	Substanzgruppe (Text)
57	X	Verabreichungsart (po. bzw. iv.)
58	X	Warengruppe
59	X	ATC-Code
60	X	Wirkstoff 1 (Hauptwirkstoff)
61	X	[Wirkstoff 1] Menge eines Wirkstoffs in einem Material
62	X	[Wirkstoff 1] Mengeneinheit eines Wirkstoffs in einem Material
63	X	Wirkstoff 2
64	X	[Wirkstoff 2] Menge eines Wirkstoffs in einem Material
65	X	[Wirkstoff 2] Mengeneinheit eines Wirkstoffs in einem Material
66	X	Wirkstoff 3
67	X	[Wirkstoff 3] Menge eines Wirkstoffs in einem Material
68	X	[Wirkstoff 3] Mengeneinheit eines Wirkstoffs in einem Material
69		Name
70		Vorname
71	X	Laufende Nummer der Materialanforderung

Tabelle 5.3.: Bedeutung der Einträge in der CSV-Datei, Teil 2. Der erste Teil befindet sich in Tabelle 5.2 auf der vorherigen Seite.

```
[Header]
Source=MLAB2
ImportDate=01.13.2023 14.00
...
[Data]
1|HOM|Universitätsklinikum des Saarlandes|20070851|Völlner,
Rudi|01.08.1949|M|...
2|
1|HOM|Universitätsklinikum des Saarlandes|20699280|Müller,
Max|06.11.1948|M|...
2|kwm|Kein Nachweis von MRSA (kulturell)||0||01.01.2023|
```

Abbildung 5.4.: Ausschnitt aus einer .lab-Datei mit fiktiven Testdaten.

Analysen aus der Mikrobiologie nach Anforderung übertragen, allerdings werden neben den Auftragsdaten (e.g., Patientenummer, Name) die eigentlichen Werte in vorbereiteter Form bereitgestellt, beispielsweise zur Generierung eines PDFs. Die Werte werden nicht strukturiert in das SAP-System übertragen. Daher sollen nach einem Entwurf des EU-Projektes täglich generierte Dateien innerhalb der Mikrobiologie zugänglich gemacht werden. Diese auf einem Server der Mikrobiologie befindlichen Dateien werden im folgenden kurz beschrieben. Der Server benutzt das Betriebssystem Windows.

Zieldateien

Die .lab-Dateien werden aus den entsprechenden aktuellen Analysen in der Mikrobiologie generiert und in einem Arbeitsverzeichnis auf dem dortigen Hybaseserver abgespeichert. In Kapitel 6 auf Seite 87 wird auf die weitere Verarbeitung dieser Dateien hinsichtlich des Datenschutzes eingegangen. Der Dateiname (e.g., “MLHybaseData 30-01-2015 145006.lab”) enthält das aktuelle Datum und die Uhrzeit. Die Leerzeichen werden in Abschnitt 5.8 auf der nächsten Seite nochmal relevant. Eine solche Textdatei wie in Abbildung 5.4 enthält die aktuellen Analysen zu einem oder mehreren Patienten. Sie besteht aus einem Kopf- und einem Datenteil. Die Werte im Datenteil sind mit einem Separator (c.q., “|”) getrennt. Die Einträge sind mehrzeilig und werden ab erster Stelle jeweils mit “1” beginnend nummeriert. Die erste Zeile beginnt mit Informationen zum Institut, gefolgt von Patientenstammdaten. Eine Datei kann mehrere Einträge zu mehreren Untersuchungsergebnissen zu einem Patienten enthalten. Es besteht die Möglichkeit diese Daten zu einem späteren Zeitpunkt mit anderen Daten (e.g., aus SAP) zusammenzubrin-

Stelle	Bedeutung bei Zeilennummer = 1
0	Zeilennummer
1	Abkürzung des Instituts der Quelle
2	Name des Instituts der Quelle
3	Patientennummer
4	Name, Vorname
5	Geburtsdatum
6	Geschlecht
...	...

Tabelle 5.4.: Bekannte personenbezogene Einträge in den .lab-Dateien.

gen, weil auch hier die Patientennummer vorhanden ist. Diese Werte müssen allerdings zuerst angepasst werden, weil im Gegensatz zu den Daten aus dem SAP die führenden Nullstellen fehlen. Es hat sich bei den anderen Werten in den .lab-Dateien allerdings nicht immer eine zwingend eindeutige Zuordnung der Werte in der vorliegenden Struktur ergeben. Die Tabelle 5.4 zeigt die hier verwendeten Werte.

5.8. Datentransport

Die CSV-Dateien aus Unterabschnitt 5.6.4 auf Seite 74 liegen auf dem SAP Applikationsserver und die Daten aus der Mikrobiologie auf dem sogenannten Hybaseserver. Um diese Daten wie in Kapitel 6 auf Seite 87 und Kapitel 7 auf Seite 95 unabhängig von ihren Ursprungssystemen weiterverarbeiten zu können, müssen sie zuerst an ein entsprechendes Ziel gelangen. Dieses Ziel kann ein zusätzlicher Server (c.q., Studienserver) darstellen. Es muss allerdings nicht zwingend eine neue Infrastruktur genutzt werden, um an die Daten aus den Quellsystemen zu gelangen. Stattdessen wurde schon für Tests in Zusammenarbeit mit dem ZIK des UKS ein Teil des dortigen Kommunikationsservers genutzt, siehe dazu auch Teil III auf Seite 115.

Mit Hilfe eines Network-File-System kann vom Zielsystem auf Verzeichnisse zugegriffen werden, welche mit dem SAP-Applikationsserver und dem Hybaseserver geteilt werden [64]. Dazu wurde für erste Tests ein fester Zeitpunkt festgelegt, wann und auf welche Dateien vom Zielsystem aus durch das NFS zugegriffen werden kann. Zu diesem Zeitpunkt werden Dateien mit definiertem Namen (e.g., *Micro*.csv*) von den jeweiligen Quellverzeichnissen (e.g., Applikationsserver SAP) in ein mit dem Zielsystem geteiltes Verzeichnis

verschoben. Die Einrichtung des NFS unter Nutzung des Kommunikationsservers erfolgte durch Mitarbeiter des ZIK nach Absprache mit mir. Im folgenden erläutere ich die Bereitstellung und den Abruf der Dateien auf den Quell- und Zielservers, damit das NFS genutzt werden kann.

5.8.1. Bereitstellung auf dem Applikationsserver

In Listing 5.14 auf Seite 78 aus Unterabschnitt 5.6.4 auf Seite 74 werden die CSV-Dateien jeweils auf dem eingebundenen Zielverzeichnis des Applikationsservers gespeichert. Der Dateiname wurde zur Freigabe aus dem NFS entsprechend festgelegt, wobei beim Datumsformat im Dateinamen nicht die im deutschsprachigen Raum übliche Formatierung (i.e., Tag, Monat, Jahr) verwendet wird. Die Reihenfolge Jahr, Monat, Tag ermöglicht eine Sortierung der Dateien in Reihenfolge ihrer ursprünglichen Erstellung.

5.8.2. Bereitstellung auf dem Hybaseserver

Auf dem Hybaseserver ist seitens des ZIK ein Exportverzeichnis eingerichtet, welches wie das Verzeichnis auf dem Applikationsserver über das NFS mit dem Zielserver geteilt wird. Neben dem Kopieren der relevanten Dateien mit einem Skript aus dem Arbeitsverzeichnis der Mikrobiologie in dieses Exportverzeichnis ergibt sich allerdings eine kleine Schwierigkeit, nämlich die des Dateinamens. Unter Windows ist es möglich, Leerzeichen im Dateinamen zu verwenden, was zu Schwierigkeiten in der Verarbeitung der Dateien mit anderen Betriebssystemen (e.g., Linux) führen kann, aber auch unter Windows selbst [64]. Die Leerzeichen werden in diesem Falle durch Underscore (i.e., “_”) ersetzt, was einer auch in der Literatur genannten Vorgehensweise entspricht [7]. Zum Kopieren verwende ich das in Windowssystemen vorhandene *xcopy* [41]. Dabei werden ausschließlich Dateien aktuellen bzw. zukünftigen Datums erfasst. Eine tägliche Ausführung meines Testskripts wurde unter Verwendung der Systemverwaltung von Windows zusammen mit dem IT-Beauftragten der Mikrobiologie eingerichtet.

5.8.3. Abruf der Daten

Der Zielserver ist in diesem Falle ein Server mit dem Betriebssystem OPENSUSE. Der oben genannte Zugriff im NFS wird etabliert, indem die entsprechend bereitgestellten Verzeichnisse in die Verzeichnispfade des Zielservers eingebunden werden. Dazu werden

sie seitens des Betriebssystems mit dem Befehl *mount* eingehängt [98]. Zur Übertragung aus diesem NFS-Verzeichnis wird ein BASH-Skript [82] verwendet, wie in Listing 5.15 zu sehen ist.

Listing 5.15: BASH-Skript *transferscript01* zum Kopieren der Daten aus den Verzeichnissen des Network-File-Systems.

```
1 #!/bin/bash
2 ./home/eureca/.profile
3 mv /home/eureca/mountpoints/mountsap/sap-tsa/micro/*.csv
4   /home/eureca/store/
5 mv /home/eureca/mountpoints/mountsap/hybase/*.lab
6   /home/eureca/store/
```

So werden die ursprünglich aus dem SAP-Applikationsserver und Hybaseserver stammenden Dateien in ein eigenes Verzeichnis auf dem Zielsystem kopiert. Optional kann in diesem Skript auch ein Programm zur Weiterverarbeitung auf dem Zielsystem gestartet werden. Damit dieses Skript täglich ausgeführt werden kann, bietet sich unter LINUX ein Eintrag als *cron job* an [82], welcher hier wie folgt lautet:

```
1 30 3 * * * /home/eureca/transferscript01.sh
```

Dadurch wird das Listing 5.15 täglich um 3:30Uhr ausgeführt.

5.9. Manuelle Abfrage in SAP

Die obigen Programme mit den zugehörigen Datenabfragen (e.g., Listing 4.2 auf Seite 51) kann man auch manuell von einem Endnutzer des KIS starten lassen. In diesem Fall bietet sich für die Parametereingabe eine feinere Auswahl (e.g., Station) in einem Selektionsbild an, anstatt die Änderungen im Programmcode vornehmen zu müssen. Neben der Auswahl genau einer Station ist eine Option für mehrere Stationen ebenso interessant, besonders wenn man diese von einem Endnutzer vorgeben lassen kann. Abhängig von der Situation (e.g., Arbeit am Präsentationsserver) sind auch weitere Parameter für die Pseudonymisierung und die Speicherung relevant, siehe das Selektionsbild in Abbildung 5.5 auf der nächsten Seite. Die entsprechenden Änderungen im Deklarationsteil vor der ersten Abfrage aus der Tabelle NBEW sind in Anhang A auf Seite 149 zu finden.

5. Erweiterte Datengewinnung für das Mikrobiologieszenario

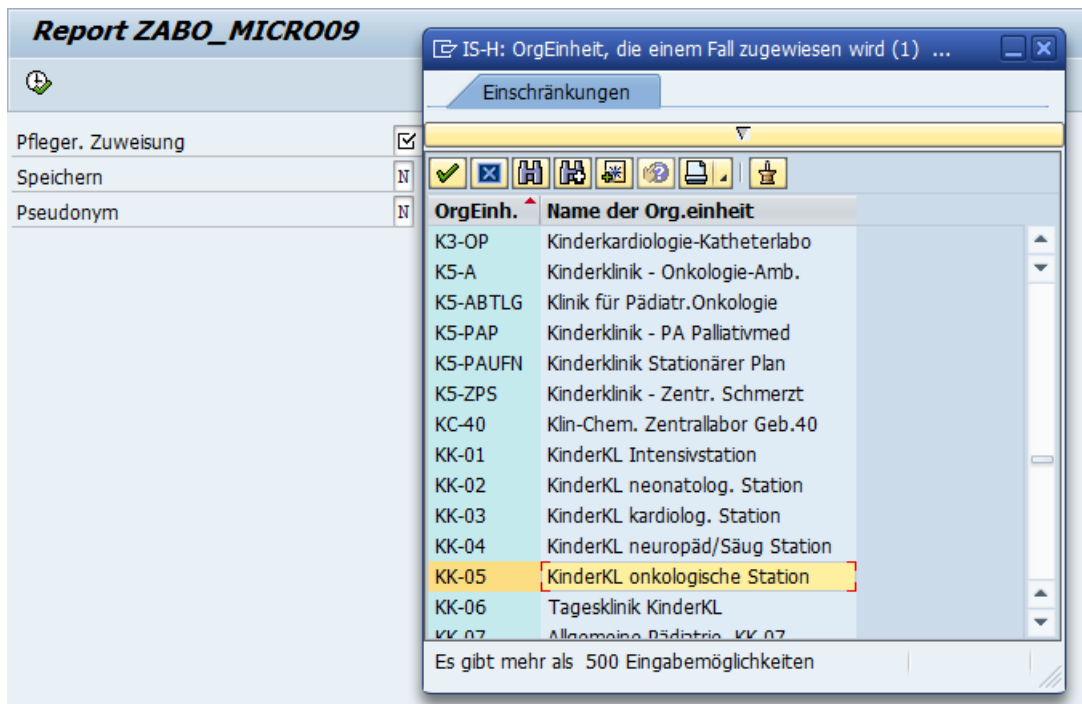


Abbildung 5.5.: Selektionsbild für das ABAP-Programm.

6. Datenschutz

In [99] wird der Begriff *Datenschutz* unter Verweis auf das Bundesdatenschutzgesetz als “Schutz des Einzelnen vor Beeinträchtigung seines Persönlichkeitsrechts beim Umgang mit seinen personenbezogenen Daten” definiert. Unter personenbezogenen Daten sind in den folgenden Abschnitten Daten (e.g., Name) zu verstehen, mit deren Hilfe eine Person bestimmt werden kann bzw. mit Zusatzinformationen bestimmbar werden könnte [88].

Beim Datenschutz ist in der Medizin zwischen den Situationen *Behandlung* und *Forschung* zu unterscheiden. Nach [67] greift bei der Behandlung eines Patienten für seine Gesundheitsdaten eine gesetzliche Ausnahmeregelung, welche sich aber nicht zwingend auf alle Bereiche der medizinischen Forschung übertragen lässt. In einer klinischen Studie ist eine einfache Erkennung der Identität eines Patienten aber möglicherweise gar nicht erwünscht. Nach [78] erfolgt nach der Patientenregistrierung in Studien mit Randomisation eine Pseudonymisierung, um den Selektionsbias zu vermeiden.

Das Thema Datenschutz ist in dieser Arbeit bei den Demonstrationsbeispielen zunächst vernachlässigbar, weil diese Beispiele fiktive Patientendaten beinhalten. Da ein Produktiveinsatz dieser oder ähnlicher Software angestrebt werden soll, befasse ich mich in diesem Kapitel mit einer Möglichkeit der informationstechnischen Aufbereitung der Daten zur späteren Pseudonymisierung. Im Abschnitt 6.1 werden die in dieser Hinsicht relevanten Werte aus Kapitel 4 auf Seite 39 und Kapitel 5 auf Seite 55 betrachtet. Ich nenne im Abschnitt 6.2 auf der nächsten Seite eine theoretische Möglichkeit der Umsetzung, während ich in Abschnitt 6.3 auf Seite 89 jeweils eine praktische Umsetzung für die Daten aus SAP und der Mikrobiologie erlautere.

6.1. Personenbezogene Daten im Mikrobiologieszenario

Bei Betrachtung der Tabelle 5.2 auf Seite 80 und Tabelle 5.3 auf Seite 81 fällt auf, dass tatsächlich ein guter Teil (e.g., Schlüsselung der Diagnose, Verabreichungsart) der erhobenen Werte für den Datenschutz gar nicht relevant ist. Name, Vorname und Geburtsdatum

sind dagegen eindeutig personenbezogene Daten. Mit Hilfe der Patientenummer ist ein Patient eindeutig bestimmbar, falls beim entsprechenden Krankenhaus ein Zugang zum KIS besteht und mit der Patientenummer gesucht werden kann. Mit der Fallnummer verhält es sich ähnlich. Die Notwendigkeit von Maßnahmen in der Verarbeitung sind von den konkreten Umständen abhängig, welche beispielsweise in Studien (siehe Kapitel 2 auf Seite 23) geregelt sind. Daher stellt die Auswahl der Werte in dieser Arbeit keine überall passende Referenz dar. Wichtige Ansprechpersonen sind in der Regel die zuständigen Personen für den Datenschutz vor Ort.

6.2. Maßnahmen während der Erhebung und Verarbeitung

Bei der Datenabfrage müssen einige Werte (e.g., Patientenummer, Fallnummer) erhoben werden, um damit durch weitere Abfragen Informationen aus anderen Tabellen zu erhalten. Dies ist unabhängig davon, ob die Werte später angezeigt oder exportiert werden sollen. Für weitere Abfragen nicht benötigte Werte (e.g., Patientename) sollte man innerhalb des Programms nur dann erheben, falls diese tatsächlich angezeigt werden sollen. Es ist hierbei zu beachten, dass die interne Tabelle aus Kapitel 5 auf Seite 55 in SAP angezeigt werden kann (e.g., ABAP Debugger [79]), bevor das Programm komplett durchgelaufen ist und einzelne Werte ausblenden kann.

Es kann in der Softwareentwicklung zu Kontrollzwecken hilfreich sein, je nach Problemstellung, zusätzliche Daten anzuzeigen. Dies sollte allerdings kein Dauerzustand werden, zumal die Chance für die Bestimmbarkeit einer Person mit jedem Datum ansteigen kann bzw. ein solcher Versuch vereinfacht wird. Manche Werte kann man so verarbeiten, dass der eigentliche Nutzen, je nach Vorhaben, vorhanden bleibt, aber die Bestimmbarkeit einer Person verringert bzw. erschwert wird. Als Beispiel ist hier die Verwendung des Geburtsjahres oder Alters anstatt des Geburtsdatums zu nennen.

Es kann der Bedarf bestehen, erhobene Daten weiter zu organisieren (e.g., Nutzung einer Datenbank) oder mit anderen Datenquellen zusammenführen. In obigem Fall kann dies mit der Patientenummer geschehen, welche sowohl in SAP als auch in der Mikrobiologie Verwendung findet. Ein Schritt in Richtung Anonymisierung [88] in Form einer ersatzlosen Löschung der Patientenummer ist daher keine wirkliche Option. Als Alternative kann man die Patientenummer mit einem gleichwertigen Ersatz (i.e., Pseudonym) austauschen, welcher die Bestimmung eines Patienten erschwert.

In dieser Arbeit verwende ich Hashwerte als Pseudonyme. Ein Hashwert bzw. eine

Hashadresse [37] wird mit Hilfe einer Hashfunktion [18] aus einem Objekt erzeugt, in diesem Falle die Patientenummer. Die Hashfunktion soll für unterschiedliche Eingabeobjekte möglichst keine identischen Hashwerte innerhalb einer Hashtabelle erzeugen, ansonsten spricht man von einer Kollision. Es existieren allerdings Hashfunktionen bzw. Verfahren, auf welche ein Entwickler an unterschiedlichen Stellen (c.q., SAP und JAVA) zurückgreifen kann. In Abschnitt 6.3 verwende ich daher das Message-Digest Verfahren (MD5) [18] von Ronald Rivest. Mit einem Hashwert kann das lokale Personal (e.g., Verwaltung, Pflege) keine zeitnahe Suche mit erfolgreicher Rückmeldung im KIS starten, womit auch der Einsatzzweck erfüllt ist.

Das Verfahren ist aus informationstechnischer Sicht nicht absolut sicher, zumal die Patientennummern als Eingangswerte kurz sind und lediglich aus Zahlen bestehen [100]. Es kann daher nicht völlig ausgeschlossen werden, dass ein Angreifer aus den Hashwerten wieder die Patientenummer bekommt, insbesondere bei vorhandenem Insiderwissen. Eine deutliche Verlängerung des Eingangswertes durch weitere Daten oder Wiederholung ist daher sinnvoll, um einen solchen Versuch weiter zu erschweren.

6.3. Praktische Umsetzung

Im folgenden beschreibe ich eine praktische Umsetzung der obigen Maßnahmen für die Daten aus SAP und der Mikrobiologie. Während dies in SAP schon vor Speicherung der CSV-Datei möglich ist, muss ich im Falle der Mikrobiologie mit den bereits erzeugten Dateien arbeiten. Es wird für identische Patienten (i.e., Patientennummern) das gleiche Pseudonym erzeugt, damit die Patientendaten zu einem späteren Zeitpunkt verbunden werden können.

6.3.1. Patientendaten aus SAP

In das ABAP-Programm aus Kapitel 4 auf Seite 39 wird als Auswahloption für eine Pseudonymisierung ein entsprechender Parameter hinzugefügt. Eine Überprüfung der positiven Auswahl kann daher jeweils in den relevanten Programmteilen erfolgen.

Der Name eines Patienten muss gar nicht erst erhoben werden, weil der Hashwert aus der Patientenummer gewonnen wird. Während in Listing 5.6 auf Seite 65 bei den personenbezogenen Daten immer die Namen abgefragt und gespeichert werden, enthält das Listing 6.1 auf der nächsten Seite eine entsprechende Anpassung. Sollte eine Pseudonymi-

Listing 6.1: Gewinnung von personenbezogenen Daten unter Vernachlässigung des Namens.

```
1  *Deklarationsteil
2  PARAMETERS: pa_pseu DEFAULT 'N'.
3  *...
4  LOOP AT gt_micro INTO gs_micro.
5    SELECT SINGLE * FROM npat INTO t_rednpat WHERE patnr =
        gs_micro-patnr.
6    IF sy-subrc = 0.
7      MOVE t_rednpat-gschl TO gs_micro-gschl.
8      MOVE t_rednpat-gbdat TO gs_micro-gbdat.
9    *Wurde eine Pseudonymisierung gewünscht?
10     IF NOT ( pa_pseu = 'J' OR pa_pseu = 'Y' ).
11       MOVE t_rednpat-nname TO gs_micro-nname.
12       MOVE t_rednpat-vname TO gs_micro-vname.
13     ENDIF.
14     MODIFY gt_micro FROM gs_micro.
15   ENDIF.
16 ENDLOOP.
```

sierung gewünscht sein, wird der Name eines Patienten gar nicht erst in die Zeilenstruktur eingetragen und wird somit auch nicht an die zu befüllende interne Tabelle weiter gegeben. Die entsprechenden Stellen der Tabelle und der CSV-Datei bleiben in dem Falle leer.

Zur Nutzung des Geburtsjahres anstelle des Geburtsdatums muss lediglich die entsprechende Stelle im Datum abgefragt und damit das Geburtsdatum überschrieben werden.

In ABAP ist bereits eine Funktion zur Generierung von Hashwerten enthalten, ebenso wie das Message-Digest Verfahren. In Listing 6.2 auf der nächsten Seite wird nach der Befüllung der Tabelle jede einzelne Zeile durchgegangen und aus der Patientenummer der Hashwert generiert. Anschließend wird die Patientenummer in der aktuellen Zeile gelöscht und das Geburtsdatum zum Geburtsjahr verändert. In Abbildung 6.1 auf Seite 93 ist ein Ergebnis dieses Vorgehens in der Ausgabe mit dem ALV zu sehen.

6.3.2. Patientendaten aus der Mikrobiologie

Bei den Dateien aus der Mikrobiologie ist eine Vermeidung der Erhebung von einzelnen Werten nicht möglich, da auf die Erstellung dieser Dateien von meiner Seite kein Einfluss besteht. Das Gleiche gilt für die die Generierung eines Hashwerts als Pseudonym. Die Dateien müssen daher nachträglich eingelesen und verarbeitet werden.

Ich habe eine Software (i.e., LABANOM) zur Pseudonymisierung der .lab-Dateien in JAVA entwickelt und objektorientiert programmiert. Die Auswahl von JAVA als plattformunabhängige Programmiersprache ist der Tatsache geschuldet, dass zum Zeitpunkt der Implementierung nicht festgestanden hat, ob das Programm zu einem späteren Zeitpunkt auf dem Hybaseserver mit Windows oder dem Zielservers mit Linux laufen soll [1]. Die Vorgehensweise von LABANOM ist wie folgt:

- Es wird nach .lab-Dateien im aktuellen Verzeichnis gesucht.
- Jede Datei wird nacheinander verarbeitet.
- Für alle Dateien:
 - Die aktuelle Datei wird zeilenweise eingelesen.
 - Für alle Zeilen:
 - * Jede Zeile wird nach dem Separator aufgetrennt und die einzelnen Werte werden in einem Array [21] in einer Instanz der Klasse *Zeile* gespeichert.

Listing 6.2: Generierung eines Hashwertes zur Ersetzung der Patientennummer. Zudem wird das Geburtsdatum zum Geburtsjahr geändert.

```
1  *Deklarationsteil
2  DATA: l_sig TYPE string,
3         l_hash TYPE hash160. "Hashwert
4  *...
5  IF pa_pseu = 'J' OR pa_pseu = 'Y'.
6    LOOP AT gt_final INTO gs_micro.
7      MOVE-CORRESPONDING gs_micro TO gs_pseud.
8      l_sig = gs_pseud-patnr.
9      CALL FUNCTION 'CALCULATE_HASH_FOR_CHAR'
10     EXPORTING
11       alg      = 'MD5'
12       data     = l_sig
13       length   = 0
14     IMPORTING
15       hash     = l_hash.
16     gs_pseud-pseud = l_hash.
17     gs_pseud-patnr = space.
18     gs_pseud-gbdat = gs_pseud-gbdat(4).
19     IF sy-subrc = 0.
20       MODIFY gt_final FROM gs_pseud.
21     ENDIF.
22   ENDLOOP.
23 ENDIF.
```

Patient	Geschl	Geburtsdatum	Einr.
10001402	2	01.01.1999	UKSH
10001402	2	01.01.1999	UKSH
10001402	2	01.01.1999	UKSH
10001402	2	01.01.1999	UKSH
10007692	1	14.12.2005	UKSH
10008733	1	01.01.1962	UKSH
10013213	1	14.03.1979	UKSH
10013417	2	07.07.2007	UKSH
10013417	2	07.07.2007	UKSH
10013417	2	07.07.2007	UKSH
10013417	2	07.07.2007	UKSH
10013417	2	07.07.2007	UKSH
10013417	2	07.07.2007	UKSH
10013612	1	04.02.1994	UKSH
10014114	2	07.08.2002	UKSH
10015012	1	11.11.1955	UKSH
10015012	1	11.11.1955	UKSH
10015012	1	11.11.1955	UKSH
10015012	1	11.11.1955	UKSH
10015012	1	11.11.1955	UKSH
10015012	1	11.11.1955	UKSH
10015012	1	11.11.1955	UKSH
10015312	3	11.11.2011	UKSH
10016418	2	08.08.2008	UKSH

(a) Reguläre fiktive Testdaten.

	Patient	Geschl	GebDatum	Einr.
CBDC77138266924B3FAE9981884807E5		2	. .1999	UKSH
CBDC77138266924B3FAE9981884807E5		2	. .1999	UKSH
CBDC77138266924B3FAE9981884807E5		2	. .1999	UKSH
CBDC77138266924B3FAE9981884807E5		2	. .1999	UKSH
EF86E17AD0CFD33BC97F8A7A40B5DF89		1	. .2005	UKSH
2CC59EB27B2EFB30AA0D6DF8FB7D3B21		1	. .1962	UKSH
0A0049E1C3C36163598C6314E22FC542		1	. .1979	UKSH
C79331AA1720AAE8F075D10F8C7843B7		2	. .2007	UKSH
C79331AA1720AAE8F075D10F8C7843B7		2	. .2007	UKSH
C79331AA1720AAE8F075D10F8C7843B7		2	. .2007	UKSH
C79331AA1720AAE8F075D10F8C7843B7		2	. .2007	UKSH
C79331AA1720AAE8F075D10F8C7843B7		2	. .2007	UKSH
ABB69DD072B34AD91D9B316FD4C94A0F		1	. .1994	UKSH
EE31794D9466E5DB9E59CCBB16E02BDD		2	. .2002	UKSH
FCE2F3D91377664C28C09E048377C5E6		1	. .1955	UKSH
FCE2F3D91377664C28C09E048377C5E6		1	. .1955	UKSH
FCE2F3D91377664C28C09E048377C5E6		1	. .1955	UKSH
FCE2F3D91377664C28C09E048377C5E6		1	. .1955	UKSH
FCE2F3D91377664C28C09E048377C5E6		1	. .1955	UKSH
FCE2F3D91377664C28C09E048377C5E6		1	. .1955	UKSH
FCE2F3D91377664C28C09E048377C5E6		1	. .1955	UKSH
3553B81A86A96C0DC70C778796FB8CD7		3	. .2011	UKSH
5763F15B644B5D59641E8F49052CF87B		2	. .2008	UKSH

(b) Pseudonymisierte fiktive Testdaten.

Abbildung 6.1.: Ausschnitt des Ergebnisses der Generierung eines Hashwertes für die Patientenummer und der Konvertierung des Geburtsdatums im ABAP-List-Viewer.

Zeile mit personenbezogenen Daten:

```
1|HOM|Universitätsklinikum des Saarlandes|10123499|Meier, Christine|11.11.1975|W|...
```

Obige Zeile nach Anwendung von Labanom

```
1|HOM|Universitätsklinikum des Saarlandes|46bac12a84d77faf31b777de0fbd2a0c|cccccccccccccc|1975|W|...
```

Abbildung 6.2.: Pseudonymisierung der entsprechenden fiktiven Testwerte einer .lab Datei mit LABANOM.

- * Der Patientename in mit 1| beginnenden Zeilen wird mit der gleichen Anzahl an Zeichen ersetzt.
 - * Ein MD5 Hashwert wird aus der eingelesenen Patientenummer (plus hinzugefügter führender Nullen) generiert und an gleicher Stelle eingesetzt.
 - * Das Geburtsdatum wird zum Geburtsjahr umgerechnet.
 - * Aus den Zeilenklassen wird wieder ein Text erzeugt.
- Der Text wird als Datei mit einem entsprechenden Präfix (c.q., “pseud_”) gespeichert.

Der relevante Programmcode ist in Anhang C auf Seite 183 zu finden. Die Notwendigkeit der vorherigen Anpassung (c.q., zusätzliche führende Nullen) ergibt sich aus der Tatsache, dass in SAP mit den eigentlichen Werten aus der Datenbank gerechnet wird und nicht mit optisch verschönerten Formatierungen für die Bildschirmausgabe. Bei der Patientenummer aus der .lab-Datei fehlen diese führenden Nullen. Daher müssen diese Werte angepasst werden, um bei der Erzeugung eines Hashwertes das gleiche Ergebnis wie in SAP zu erhalten. Die Abbildung 6.2 zeigt das Resultat der Anwendung des oben beschriebenen Programms.

7. Weiterverarbeitung der gewonnenen Daten

In Kapitel 5 auf Seite 55 wurden CSV-Dateien mit dem SAP-System generiert. Dieses Kapitel zeigt eine Möglichkeit zur Weiterverarbeitung dieser Dateien. In diesem Zusammenhang erläutere ich, wie die Mehrzeiligkeit beim Materialverbrauch (c.q., Medikamentengabe) in der CSV-Datei verwendbar ist.

Im Rahmen des EURECA-Projektes werden als Eingabe für ein Data Warehouse (DWH) CDA-Dokumente [12] gefordert. Ein Data Warehouse erwirbt regelmäßig Daten aus verschiedenen Quellen und führt diese zusammen [68], um sie für spätere Analysen zur Verfügung zu stellen. Meist wird ein ETL-System (Extract, Transform, Load) verwendet, um die Daten aus verschiedenen Systemen in das Speichersystem des DWH zu bringen. Dabei wird das DWH nicht zwingend zeitgleich zum Quellsystem aktualisiert.

Ein CDA-Dokument basiert auf dem XML¹-Format [19] und wird für klinische Informationen eingesetzt [12]. Damit möchte man dem HL7-Standard entsprechen, welcher einen internationaler Kommunikationsstandard für den elektronischen Austausch medizinischer Daten darstellt [69].

7.1. Inhaltliche Vorarbeit

Für die Eingabe in das DWH wurden innerhalb von EURECA mehrere CDA-Dokumente vorgegeben, deren Werte zu befüllen sind. Aus der CSV-Datei wird daher für jeden einzelnen Patienten wie in Abbildung 7.1 auf Seite 97 ein CDA (i.e., Antibiotika-CDA) mit der Medikation (e.g., Zeitpunkt der Medikamentengabe) und ein CDA (i.e., KIS-CDA) mit den anderen Daten (e.g., belegtes Zimmer, Diagnose) erstellt. Diese Beispiel-CDAs kamen allerdings ohne genauere Beschreibung der enthaltenen Werte. Daher musste von

¹Extensible Markup Language

mir unter Betrachtung der vorhandenen Werte im SAP zuerst eine Zuordnung aufgestellt werden, welche dann unter Mitarbeit von Projektteilnehmern in EURECA überprüft wurde. Die Zuordnungsbeschreibung ist im Anhang E auf Seite 207 zu finden.

Die Strukturierung der CDA-Dokumente stimmt nicht vollkommen mit der Strukturierung im Krankenhaus überein. Da einige Werte aus den CDA-Dokumenten in dieser Form nicht direkt im SAP vorhanden sind, müssen sie unter Verwendung der vorhandenen Werte im SAP erstellt werden. Dies erfolgt durch Umrechnung oder Kombination der vorhandenen Werte, siehe die folgenden Beispiele:

Umwandlung von Werten Während das männliche Geschlecht in SAP mit der Zahl "1" gekennzeichnet ist, wird für die CDA-Dokumente die Umwandlung in das Zeichen "M" notwendig. Der aktuelle ICD-Katalog ist in SAP mit einer zweistelligen Abkürzung gegeben, die im CDA natürlich kein Gegenstück hat. So muss beispielsweise für "IE" noch eine Zuordnung (c.q., "ICD10-2014 ICD-10-GM, Version 2014") erfolgen.

"ID OF THE ADMISSION ACT" Dieser geforderte Wert erscheint zunächst schlüssig. Es wird nach dem Identitätscode (e.g., Nummer) des Aufnahmeakts für einen Patienten gefragt. Im Falle des UKS existiert ein solcher Identitätscode in der gewünschten Form allerdings nicht. Es ist möglich, einen beliebigen anderen Identifikator [10] zu generieren. Dessen Eindeutigkeit ist ohne weitere Maßnahmen aber nicht garantiert, und es gibt im Krankenhaus bei existentiellen Rückfragen dazu kein Gegenstück. Die Bewegungsnummer zu einer Aufnahmebewegung ist im SAP auch nicht eindeutig, weil die Bewegung zu einem Fall gehört. Im CDA-Dokument ist die Fallnummer aber nicht vorhanden. Für diesen Wert habe ich daher folgende Lösung gewählt: Verwendung der Fallnummer plus die laufende Nummer der als Aufnahme gekennzeichneten Bewegung.

"DATE OF ADMISSION (Datum & Uhrzeit)" Da es keinen *Admission Act* (siehe oben) gibt, existiert folglich auch kein Datum dazu. Es existiert ein Datum ohne Zeitangabe, wann ein Fall im SAP angelegt wurde. Dies hat mit der aktuellen Aufnahme oder Verlegung Tage später wenig zu tun. Daher bietet sich analog zu oben die folgende Lösung an: Zu einer als Aufnahmebewegung gekennzeichneten Bewegung im jeweiligen Fall, wird das Datum der Bewegung plus die Uhrzeit der Bewegung verwendet.

Es wird an diesen Beispielen bereits deutlich, dass man die Werte nicht alle eins zu eins übertragen kann und zumindest einiges an Recherche notwendig ist.

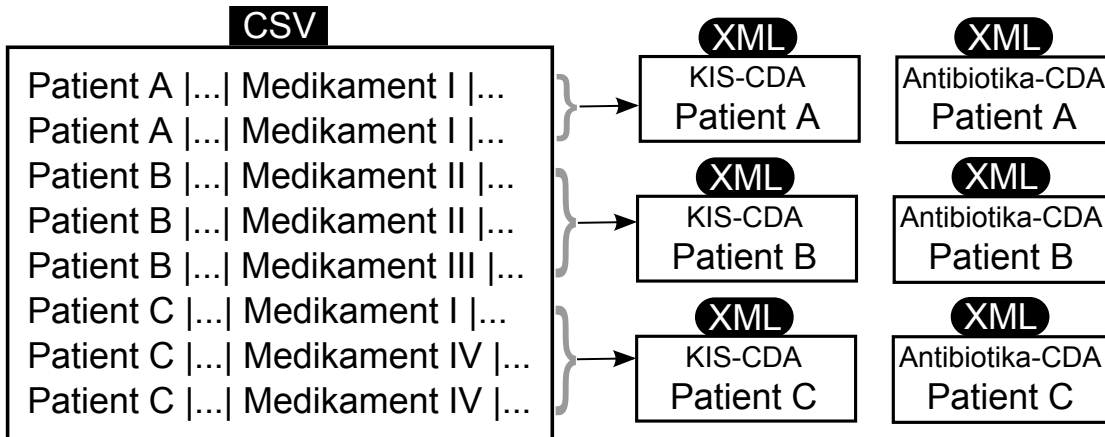


Abbildung 7.1.: Erstellung von CDA-Dokumenten für einzelne Patienten aus der CSV-Datei des SAP.

7.2. Praktische Umsetzung

Die Übertragung der Werte aus den CSV-Dateien in die CDA-Dokumente erfolgt hier in JAVA. In JAVA sind bereits einige Möglichkeiten (e.g., *jdom2*, *dom4j*) zu Generierung von XML gegeben [62].

Zum Einlesen der Werte verwende ich das Vorgehen aus Unterabschnitt 6.3.2 auf Seite 91. Ebenso werden die Werte in einer Instanz einer Klasse (i.e., *SapZeile*) als Array gespeichert. Die CSV-Datei enthält alle Werte aus den gewählten Stationen in seinen Zeilen, und die Liste aus *SapZeilen* ist dazu das inhaltliche Äquivalent. Dagegen sollen die XML-Dateien aber zu den einzelnen Patienten erzeugt werden. Daher werden aus der kompletten Liste mit *SapZeilen* für jeden einzelnen Patienten eine Liste mit den zugehörigen *SapZeilen* erzeugt. Der Aufwand ist überschaubar, weil die CSV-Datei schon nach der Patientennummer sortiert ist, siehe Listing 7.1 auf der nächsten Seite. Im folgenden geht es nun um die Erfassung der benötigten Werte.

7.2.1. Zugriff auf einzelne Werte

Die Bedeutung der Einträge in der CSV-Datei aus dem SAP ist bekannt, wie in der Tabelle 5.2 auf Seite 80 und der Tabelle 5.3 auf Seite 81 zu sehen ist.

Ein erstes Beispiel für die Wertezuordnung ist in Listing 7.2 auf Seite 99 zu sehen, bei dem das Zimmer eines Patienten ermittelt wird. Einem gut zu erkennenden Bezeichner

Listing 7.1: Erzeugung von Listen mit *SapZeilen* zu einzelnen Patienten.

```
1 public class Saptransfer {
2   //...
3   String idvar = null; //Aktuelle Patientennummer
4   String idprev = ""; //Vorherige Patientennummer
5   int listsize = zeilen.size(); //Größe der Liste aller
   SapZeilen
6   int counter = 1; //Zähler für die Listenstelle
7   List<SapZeile> patientzeilen = null; //Leere Liste mit
   SapZeilen
8   for (SapZeile n:zeilen) { //Alle Zeilen durchgehen
9     idvar = n.getID(); //Patientennummer oder Hashwert der
   Zeile
10    if (idvar.equals(idprev)) { //Patientennummer der
   vorherigen Zeile identisch?
11      patientzeilen.add(n); //Die aktuelle Zeile gehört zum
   gleichen Patienten
12    }
13    else { //Keine neuen Zeilen mehr zu diesem Patienten
14      if (patientzeilen != null ) { //z.B. Start
15        //Weiterverarbeitung der vorherigen Liste...
16      }
17      //Eine neue Liste wird angelegt
18      //Die aktuelle Zeile ist der erste Eintrag
19      patientzeilen = new ArrayList<SapZeile>();
20      patientzeilen.add(n);
21    }
22    //Am Ende der Liste
23    if (counter == listsize) { //Terminiert.
24      if (patientzeilen != null ) {
25        //Weiterverarbeitung der letzten Liste...
26      }
27    } idprev = idvar; counter++;
28  }
```

Listing 7.2: Beispielfunktion für die Zuordnung der Werte aus der CSV-Datei zu einem Bezeichner. Der eigentliche Wert kann mit der Funktion *getcsvstring* abgerufen werden, hier am Beispiel der Zimmernummer.

```
1 public class SapZeile
2 {
3     private int length;
4     public String[] valuessep; //Werte aus der CSV-Datei
5     private int POSITION_PATNR = 1;
6     private int POSITION_PSEUD = 0;
7     private int POSITION_GSCHL = 2;
8     private int POSITION_GBDAT = 3;
9     //...
10    private int POSITION_ZIMMR = 41;
11    private int POSITION_BETT = 42;
12    //...
13    private int POSITION_VERABRART = 57;
14    //...
15    public String getcsvstring(int n)
16    {
17        String direktstring = "";
18        if (length > n) //Ende des Arrays nicht erreicht
19        {
20            if (valuessep[n] != null) {
21                return valuessep[n];
22            }
23        }
24        return direktstring;
25    }
26    //...
27    public String getzimmer()
28    {
29        return (""+getcsvstring(POSITION_ZIMMR));
30    }
31 }
```

wird der entsprechende Platz des Wertes aus der CSV-Datei zugeordnet. Der entsprechende Bezeichner kann in die Funktion *getcsvstring* eingesetzt werden, um den gesuchten Wert zu erhalten. Dies geschieht über Variablen, damit bei einer Änderung der Erstellung der CSV-Dateien lediglich die Zuordnung beim Bezeichner geändert werden muss. Die Übertragung in einen Wert vor dem Einsatz in das XML Konstrukt erfolgt wie in Listing 7.3.

Listing 7.3: Zuordnung eines Wertes vor der XML Generierung. Dazu wird die Funktion aus Listing 7.2 auf der vorherigen Seite verwendet.

```
1 public class cdacreator
2 {
3     private static Element hiscreateClinicalDocument(SapZeile
         zeile){
4     String UNIQUE_IDENTIFIER_OF_THE_ROOM = zeile.getzimmer();
5     //Andere Werteübertragungen.
6     //XML Generierung ...
7     }
8 }
```

Es wird aufwändiger, falls der Wert nicht eins zu eins übertragen werden kann. Ein Beispiel für die Verarbeitung eines Wertes aus der CSV-Datei ist in Listing 7.4 bei der Verabreichungsart zu sehen. Hier werden verschiedene Abkürzungen aus dem SAP abgefragt und entsprechend "oral" oder "intravenous" zurückgegeben. Die vollständige Klasse *SapZeile* ist in Listing D.1 auf Seite 191 im Anhang zu finden.

Listing 7.4: Abfrage der Verabreichungsart aus einer Zeile der CSV-Datei.

```
1 public class SapZeile
2 {
3     public String getconslit()
4     {
5     String LITERAL_OF_THE_USED_TERM = "";
6     String einnahmeart = (""+getcsvstring(POSITION_VERABRART
         ));
7     if (einnahmeart.equals("iv")||einnahmeart.equals("iv.")
         ||
```

```
8     einnahmeart.equals("IV") || einnahmeart.equals("IV."))
9     {LITERAL_OF_THE_USED_TERM = "INTRAVENOUS";}
10    else
11    {
12        if (einnahmeart.equals("po") || einnahmeart.equals("po.")
13            ||
14            einnahmeart.equals("PO") || einnahmeart.equals("PO."))
15            {LITERAL_OF_THE_USED_TERM = "ORAL";}
16    }
17    return LITERAL_OF_THE_USED_TERM;
18 }
```

7.2.2. Zugriff auf die Medikation

Während für die gesuchten Werte aus dem KIS-CDA eine *SapZeile* zum Patienten ausreichend ist, werden für eine Darstellung der Medikation in der Antibiotika-CDA alle aus der CSV-Datei gewonnenen *SapZeilen* zum jeweiligen Patienten benötigt. Da ein Patient parallel mit mehreren Medikamenten behandelt werden kann, lohnt es sich, je nach Verarbeitungsziel, die Liste mit *SapZeilen* zu einem Patienten in weitere Listen zu den jeweiligen Medikamenten aufzuteilen. Die Erstellung dieser Medikamentenliste pro Patient ist, dank der mitgelieferten Materialnummer, wie in Listing 7.5 auf der nächsten Seite direkt realisierbar. Diese Liste kann wie in Listing 7.6 auf Seite 103 iterativ für andere Zwecke (e.g., XML Generierung) abgearbeitet werden.

Listing 7.5: Generierung einer Medikamentenliste für einen Patienten.

```
1 public class cdacreator {
2 //Die Eingabeliste besteht aus Zeilen zu einem Patienten.
3 public static List<List<SapZeile>> getmediclist(List<
4     SapZeile> zeilen){
5     List<List<SapZeile>> mediclist = new ArrayList<List<
6         SapZeile>>();
7     //Liste aus Materialnummern
8     List<String> matnrs = new ArrayList<String>();
9     //Durchlaufen aller SapZeilen des Patienten.
10    for (SapZeile line:zeilen){
11        if ((matnrs != null) && matnrs.contains(line.getmatnr()
12            )) { //Wurde diese Materialnummer schon verwendet?
13            for (List<SapZeile> donemedic:mediclist){
14                //Suche nach der schon vorhandenen Liste zum
15                Medikament
16                if (donemedic.get(0).getmatnr().equals(line.
17                    getmatnr())){
18                    //Zeile wird zur passenden Liste hinzugefügt
19                    donemedic.add(line);
20                }
21            };
22        }
23        else { //Da diese Materialnummer noch nicht verwendet
24            wurde, wird eine neue Liste angelegt.
25            List<SapZeile> medikament = new ArrayList<SapZeile>();
26            //Zeile wird der Liste zu diesem Medikament
27            hinzugefügt
28            medikament.add(line);
29            //Die Liste wird der Medikamentenliste zugefügt
30            mediclist.add(medikament);
31            matnrs.add(line.getmatnr()); //Materialnummer
32        }} return mediclist;
33    }}
```

Listing 7.6: Abarbeitung einer Medikamentenliste aus Listing 7.5.

```
1 public class cdacreator {
2   public static Element entrysubstanceAdministration(List<
      SapZeile> substzeilen){
3     List<List<SapZeile>> mediclist = getmediclist(
      substzeilen);
4     //Neuer XML-Teil für die Medikation
5     Element section = new Element("section");
6     //Abarbeiten der Medikamentenliste
7     for (List<SapZeile> medikament:mediclist){
8       String firsttake = medikament.get(0).getconsm();
9       for (SapZeile line:medikament){
10        //Verarbeitung der Verabreichungszeiten eines
      Medikaments
11      }
12    }
13    //...
14    return section;
15  }
16 }
```

7. Weiterverarbeitung der gewonnenen Daten

8. Analyse der gewonnenen Daten

Ähnlich wie bei den unterschiedlichen Ergebnisausgaben in Abschnitt 5.6 auf Seite 70, gibt es auch entsprechende Möglichkeiten zur Analyse.

8.1. Innerhalb von SAP

8.1.1. Tabellenausgabe

Die Darstellung im ALV aus den Abbildungen 5.1 auf Seite 75 und 5.2 auf Seite 76 gibt bereits eine erste Möglichkeit zur Analyse, da hier Patientendaten auf dem Bildschirm so zusammengebracht werden, wie es sonst im System nicht geschieht. Dies kann dadurch weiter optimiert werden, indem redundante Informationen ausgeblendet werden bzw. zu vernachlässigende Spalten (e.g., Adresse des Instituts) nicht dargestellt werden.

In Unterabschnitt 5.6.1 auf Seite 70 habe ich erläutert, dass man die interne Tabelle innerhalb des Programms vor jeglicher Ausgabe automatisch vorbereiten (e.g., sortieren) kann. Dies wirkt sich allerdings nicht nur auf die ALV-Ausgabe aus, sondern auch auf den Export als CSV-Datei. Falls dies nicht gewünscht ist, kann man beispielsweise eine weitere interne Tabelle innerhalb des Programm deklarieren und im Programmcode für die Tabellenausgabe in SAP verarbeiten. Im folgenden gehe ich noch auf eine weitere Möglichkeit ein, nämlich individuelle Anpassungen des Endbenutzers in der ALV-Ausgabe. Diese Anpassungen sind unabhängig von der internen Tabelle und dem Export als CSV-Datei.

Zur besseren Einstiegsübersicht des Endnutzers passe ich die ALV-Ausgabe in Anhang A in Listing A.1 auf Seite 149 ab Zeile 440 weiter an. Dort setze ich den Patientennamen weiter nach vorne und gebe den unterschiedlichen (e.g., Aufnahme, Entlassung) Diagnosen und Bewegungen ein entsprechendes Präfix. Zudem ermögliche ich dem Endnutzer eine Anpassung seines individuellen Layouts, so dass die Reihenfolge, Sortierung und Ausblendung der Spalten im ALV angepasst werden kann. Diese Art der Ausgabe

8. Analyse der gewonnenen Daten

Report ZABO_MICRO11_write

SAP-Systemfunktionsleiste

Nachname	Vorname	Geschl	Geburtsdatum	Fall	Hd.Diag...	Pr. OE	Zimmer	Bett	Materialkurztext	Menge	ME	Verabreichung	Zeit	Verabr.	ATC-Code	Wirkstoff 1	Menge	Einheit		
April	Scherz	1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Ambisome 50 mg Tris	10 Dfl	1,000	ST	19.06.2013	13:06:51	po	AJ02AA01	Amphotericin B	50,000	MG	
		1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Ambisome 50 mg Tris	10 Dfl	1,000	ST	19.06.2013	13:25:57	po	AJ02AA01	Amphotericin B	50,000	MG	
		1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Ambisome 50 mg Tris	10 Dfl	1,000	ST	20.06.2013	10:08:14	po	AJ02AA01	Amphotericin B	50,000	MG	
		1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Ambisome 50 mg Tris	10 Dfl	1,000	ST	21.06.2013	10:08:43	po	AJ02AA01	Amphotericin B	50,000	MG	
		1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Candidas 50 mg Tris	1 Dfl	1,000	ST	19.06.2013	13:22:19	iv	AJ02AX04	Caspofungin	50,000	MG	
		1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Candidas 50 mg Tris	1 Dfl	1,000	ST	21.06.2013	10:09:01	iv	AJ02AX04	Caspofungin	50,000	MG	
Kanns	Achim	1	04.02.1994	28760		KK-05	9-0354	9-0354B		0,000			00:00:00				0,000			
Mai	Anfang	3	11.11.2011	30647		KK-05	9-0356	9-0356A	Thymoglobulin 25 mg Tris	5 ml	1 Dfl	1,000	ST	20.02.2015	14:43:18	po	AL04AA04	Antithymozytäres Imm	25,000	MG
Maria	Kopf	2	07.07.2007	28562		KK-05	9-0355	9-0355B	Mucosolvan Tropfen	1,000	ML	03.04.2014	14:22:24	po	AR05CB06	Ambroxol	0,000			
		2	07.07.2007	28562		KK-05	9-0355	9-0355B	Antmykoticum zur Pilzbehandlung allgem.	1,000	ST	19.06.2013	16:00:38	po			0,000			
		2	07.07.2007	28562		KK-05	9-0355	9-0355B	Antmykoticum zur Pilzbehandlung allgem.	1,000	ST	19.06.2013	16:00:52	po			0,000			
		2	07.07.2007	28562		KK-05	9-0355	9-0355B	Antmykoticum zur Pilzbehandlung allgem.	1,000	ST	19.06.2013	16:01:41	po			0,000			
		2	07.07.2007	28562		KK-05	9-0355	9-0355B	Antmykoticum zur Pilzbehandlung allgem.	1,000	ST	19.06.2013	16:02:38	po			0,000			
		2	07.07.2007	28562		KK-05	9-0355	9-0355B	Antmykoticum zur Pilzbehandlung allgem.	1,000	ST	20.06.2013	08:36:06	po			0,000			
		2	07.07.2007	28562		KK-05	9-0355	9-0355B	Antmykoticum zur Pilzbehandlung allgem.	25,000	ST	20.06.2013	08:37:12	po			0,000			
Müller	Kerstin	2	01.01.1999	2206	Q23.0	KK-05	9-0358	9-0358B	Lindenblüten	1,000	ST	19.11.2013	08:30:00	po	AA01AA05	Blutplasma	1,000	G		
		2	01.01.1999	2206	Q23.0	KK-05	9-0358	9-0358B	Lindenblüten	1,000	ST	19.11.2013	09:06:08	po	AA01AA05	Blutplasma	1,000	G		
		2	01.01.1999	2206	Q23.0	KK-05	9-0358	9-0358B	Candidas 50 mg Tris	1 Dfl	1,000	ST	19.11.2013	08:45:00	iv	AJ02AX04	Caspofungin	50,000	MG	
		2	01.01.1999	2206	Q23.0	KK-05	9-0358	9-0358B	Candidas 50 mg Tris	1 Dfl	1,000	ST	19.11.2013	09:08:38	iv	AJ02AX04	Caspofungin	50,000	MG	
Netzer	Günther	1	14.12.2005	17779		KK-05	9-0358	9-0358A	Lavendelöl	1,000	ST	20.02.2015	13:04:47	iv	AA02AD04	Amiodaron	0,000			
Thomas	Testmann	1	02.01.2014	40145	H83.3	KK-05	9-0351	9-0351A	Codein Filmpillen 400 mg	2,000	ST	04.04.2014	14:00:21	po	AR05DA04	Codein	0,000			
Utöpie	Max	1	01.01.1962	21959	H90.7	KK-05	9-0351	9-0351B		0,000			00:00:00				0,000			

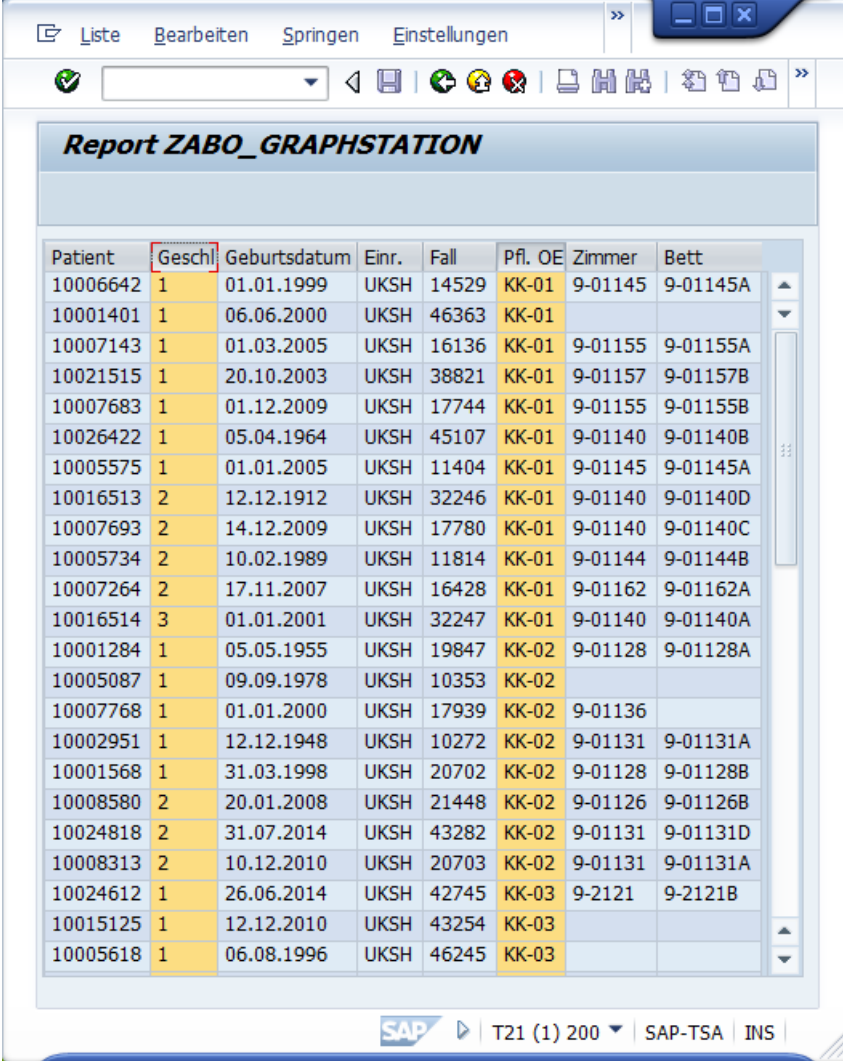
1 Details für ausgewählte Zeile **7** Druckvorschau
2, 3 Sortieren aufsteigend, absteigend **8, 9, 10** Layout ändern, auswählen, sichern
5, 6 Filter setzen, löschen

Abbildung 8.1.: Verbesserte Ausgabe mit dem ABAP-List-Viewer. Es handelt sich um fiktive Daten. Mehrere Spalten sind ausgeblendet und die Tabelle ist nach Name und Vorname sortiert. Nach dem Auslösen der *Druckvorschau* in der *Systemfunktionsleiste* kann diese Tabelle im Folgebildschirm als Tabellenkalkulation exportiert werden.

mit einem bereits angepassten individuellen Layout ist in Abbildung 8.1 zu sehen. Hinzu kommt die Möglichkeit einer Speicherung dieses Layouts. Neben einer Druckausgabe habe ich in Listing A.1 auf Seite 149 zusätzlich den Export als Tabellenkalkulation (c.q., MIME HTML) aktiviert, welche dann auf dem Präsentationsserver gespeichert werden kann.

8.1.2. Präsentationsgrafik

In SAP ist es möglich - aus aufbereiteten Daten - zusätzliche Grafiken bzw. Diagramme zu erzeugen. Zur Demonstration habe ich daher das Programm GRAPHSTATION in ABAP entwickelt. Mit diesem Programm kann sich ein Endnutzer für mehrere von ihm ausgewählte Stationen die Anzahl der Patienten nach Geschlecht anzeigen lassen. Dazu wende ich die Abfragen aus Kapitel 5 auf Seite 55 bis einschließlich Abschnitt 5.2 auf Seite 63 auf mehrere Stationen (e.g., KK-01 bis KK-05) an. Die zu befüllende interne Tabelle ist kleiner als in Kapitel 5 auf Seite 55, weil weniger Werte abgefragt werden. Ein Ergebnis in der ALV-Ausgabe im T21 ist in Abbildung 8.2 auf der nächsten Seite zu sehen.



The screenshot displays the SAP ALV interface for the report 'Report ZABO_GRAPHSTATION'. The table contains 20 rows of patient data, sorted by station and gender. The columns are: Patient, Geschl., Geburtsdatum, Einr., Fall, Pfl. OE, Zimmer, and Bett. The 'Geschl.' column is highlighted in yellow, and the 'Pfl. OE' column is highlighted in orange. The table is displayed in a window with a menu bar (Liste, Bearbeiten, Springen, Einstellungen) and a toolbar with various icons. The SAP logo and system information (T21 (1) 200, SAP-TSA, INS) are visible at the bottom.

Patient	Geschl.	Geburtsdatum	Einr.	Fall	Pfl. OE	Zimmer	Bett
10006642	1	01.01.1999	UKSH	14529	KK-01	9-01145	9-01145A
10001401	1	06.06.2000	UKSH	46363	KK-01		
10007143	1	01.03.2005	UKSH	16136	KK-01	9-01155	9-01155A
10021515	1	20.10.2003	UKSH	38821	KK-01	9-01157	9-01157B
10007683	1	01.12.2009	UKSH	17744	KK-01	9-01155	9-01155B
10026422	1	05.04.1964	UKSH	45107	KK-01	9-01140	9-01140B
10005575	1	01.01.2005	UKSH	11404	KK-01	9-01145	9-01145A
10016513	2	12.12.1912	UKSH	32246	KK-01	9-01140	9-01140D
10007693	2	14.12.2009	UKSH	17780	KK-01	9-01140	9-01140C
10005734	2	10.02.1989	UKSH	11814	KK-01	9-01144	9-01144B
10007264	2	17.11.2007	UKSH	16428	KK-01	9-01162	9-01162A
10016514	3	01.01.2001	UKSH	32247	KK-01	9-01140	9-01140A
10001284	1	05.05.1955	UKSH	19847	KK-02	9-01128	9-01128A
10005087	1	09.09.1978	UKSH	10353	KK-02		
10007768	1	01.01.2000	UKSH	17939	KK-02	9-01136	
10002951	1	12.12.1948	UKSH	10272	KK-02	9-01131	9-01131A
10001568	1	31.03.1998	UKSH	20702	KK-02	9-01128	9-01128B
10008580	2	20.01.2008	UKSH	21448	KK-02	9-01126	9-01126B
10024818	2	31.07.2014	UKSH	43282	KK-02	9-01131	9-01131D
10008313	2	10.12.2010	UKSH	20703	KK-02	9-01131	9-01131A
10024612	1	26.06.2014	UKSH	42745	KK-03	9-2121	9-2121B
10015125	1	12.12.2010	UKSH	43254	KK-03		
10005618	1	06.08.1996	UKSH	46245	KK-03		

Abbildung 8.2.: Screenshot eines Abfrageresultats von GRAPHSTATION im ALV, sortiert nach Station und Geschlecht. Es handelt sich hierbei um fiktive Patienten und Stationen.

Aus der gewonnenen Tabelle gewinne ich die Gesamtzahl an Patienten jeden Geschlechts (i.e., (1) männlich, (2) weiblich, (3) unbekannt) für die jeweilige Station. Diese Informationen nutze ich in GRAPHSTATION, um in SAP mit Hilfe der Funktion *Graph_Matrix_3D* [72] Diagramme zu erzeugen. Einige Beispiele für diese Ausgabe aus dem T21 sind in Abbildung 8.3 auf der nächsten Seite und Abbildung 8.4 auf Seite 110 zu sehen. Der Programmcode zu GRAPHSTATION ist in Anhang B auf Seite 177 zu finden.

8.2. Daten auf dem Präsentationsserver

Sollte man die Daten auf dem Präsentationsserver (e.g., PC) als CSV-Datei ablegen, besteht die Möglichkeit für einen direkten Informationsgewinn mit lokalen Mitteln. Dies ist beispielsweise mit der frei erhältlichen OfficeSuite LIBREOFFICE¹ [86] möglich. Beim Öffnen mit LIBREOFFICE CALC muss bei der ersten Nutzung bei den Trennoptionen “Getrennt” und beim Trennzeichen ausschließlich “Andere” unter Eintrag des Separators (i.e., “[”]) gewählt werden. In MICROSOFT EXCEL² funktioniert dies ähnlich aus dem laufenden Programm mit “Daten - Aus Text”. Die Erstellung einer Pivot-Tabelle in LIBREOFFICE CALC liefert nach Vorbereitung (siehe unten) eine Ergebnisdarstellung zu den Daten aus der CSV-Datei. Die Abbildung 8.5 auf Seite 111 zeigt das Datum und die Menge bzw. Mengeneinheit jeder Vergabe von Medikamenten für einen ausgewählten Patienten innerhalb der CSV-Datei. Man vergleiche dazu die Darstellung in Abbildung 5.1 auf Seite 75 und in Abbildung 5.2 auf Seite 76.

Die CSV-Datei ist auf die Weiterverarbeitung in Kapitel 7 auf Seite 95 ausgelegt, und diese Daten sollen dann kontinuierlich gesammelt werden. Daher wird bislang keine Kopfzeile mitgeliefert, was den Umgang für den Endnutzer in einer Tabellenkalkulation aber erschwert. Bei LIBREOFFICE kann man zur Vorbereitung die Kopfzeile dadurch ersetzen, dass man manuell die erste Zeile kopiert. Es gibt dazu für die Betrachtung der aktuellen Einzeldatei eine gute Alternative, nämlich den in Unterabschnitt 8.1.1 auf Seite 105 geschilderten Export als Tabellenkalkulation. Die Abbildung 8.6 auf Seite 111 zeigt in EXCEL die aus dem ALV im MIME HTML Format exportierte Tabelle zur Abbildung 8.1 auf Seite 106.

¹Version: 4.2.7.2

²Version: 14.0.7143.5000

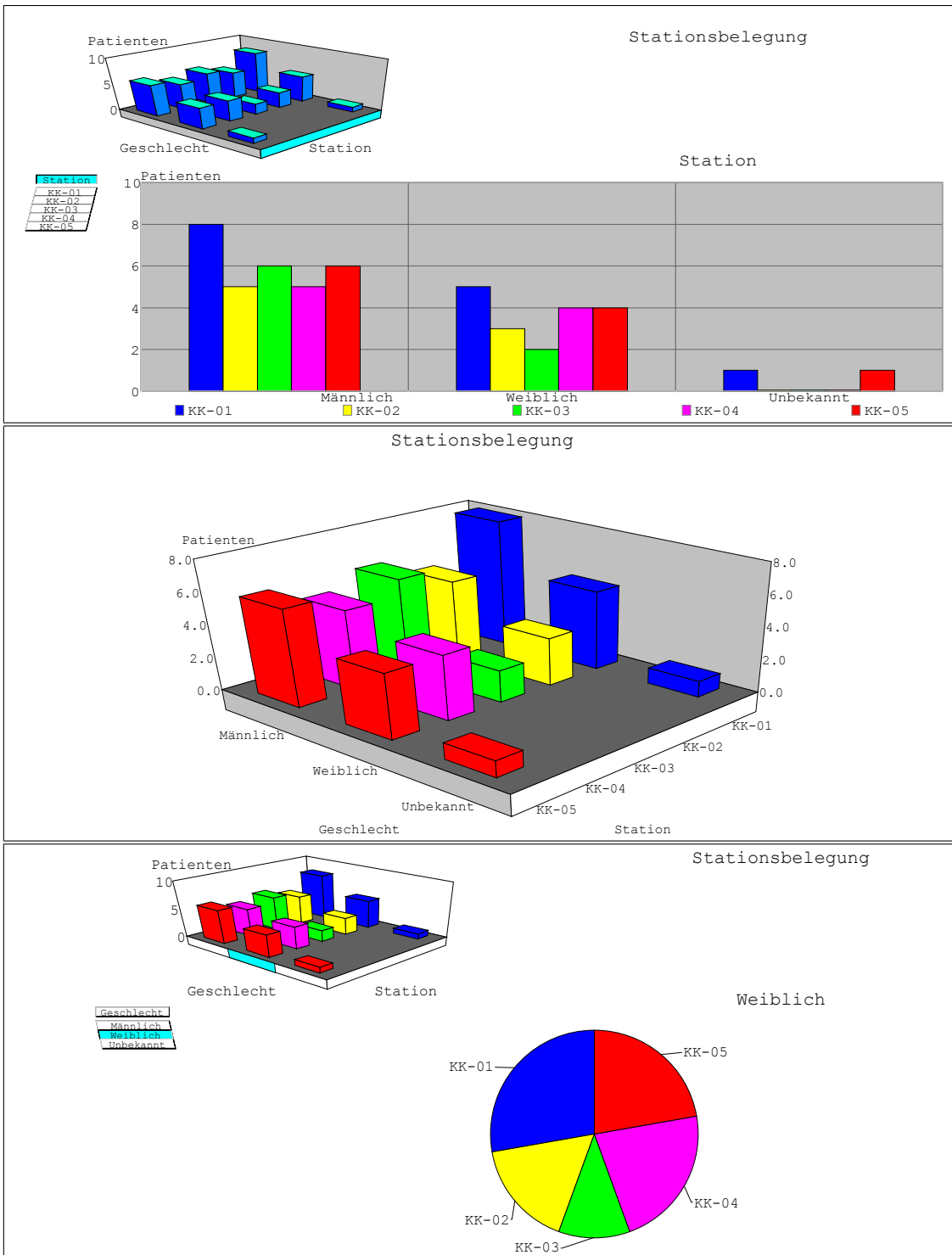


Abbildung 8.3.: Visuelle Darstellungsbeispiele zur Analyse in SAP mit GRAPHSTATION. Für diese Abbildung wurden 5 fiktive Stationen ausgewählt.

8. Analyse der gewonnenen Daten

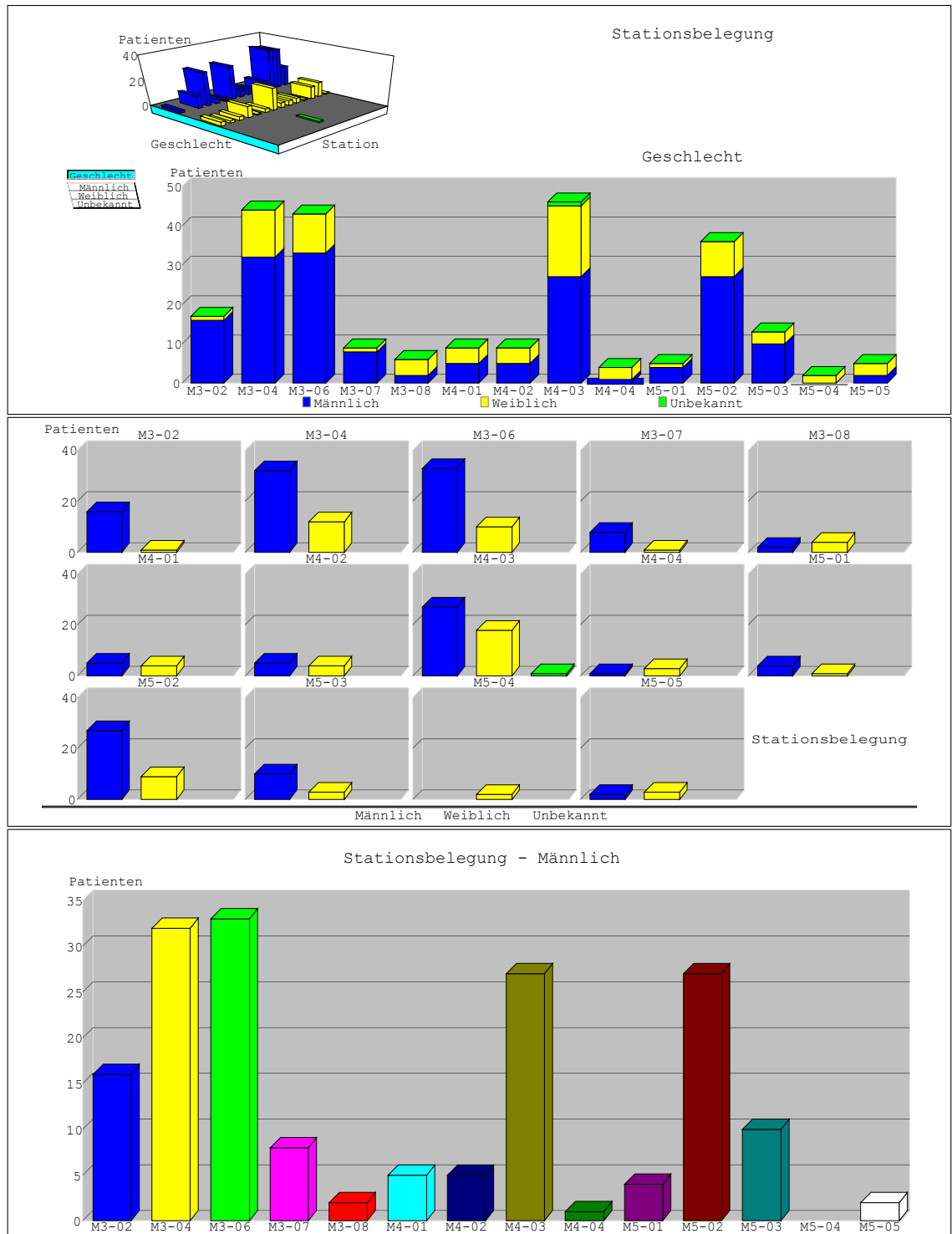


Abbildung 8.4.: Visuelle Darstellungsbeispiele zur Analyse in SAP mit GRAPHSTATION.
Für diese Abbildung wurden 14 fiktive Stationen ausgewählt.

	Maria				
	Kopf				
	03.04.2014	19.06.2013	20.06.2013	28.02.2014	Gesamt Ergebnis
	ML	ST	ST	ST	
Antimykoticum zur Pilzbehandlung allgem.		4	26	3	33
Mucosolvan Tropfen	1				1

Abbildung 8.5.: Darstellung eines Teils der CSV-Datei als Pivot-Tabelle mithilfe von LIBREOFFICE CALC. Die Patientendaten sind fiktiv.

	A	B	C	D	E	F	G	H	I	J	K	L	M	
	Nachname	Vorname	Gesch	Geburtsdatum	Fall	Hauptdiag	Pfleger	Zimmer	Bett	Materialkurztext	Materialmenge	Mengeneinheit	Verabreichungsdatum	
2	April	Scherz	1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Ambisome 50 mg TrS	10 Dfl	1,000	ST	19.06.2013
3	April	Scherz	1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Ambisome 50 mg TrS	10 Dfl	1,000	ST	19.06.2013
4	April	Scherz	1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Ambisome 50 mg TrS	10 Dfl	1,000	ST	20.06.2013
5	April	Scherz	1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Ambisome 50 mg TrS	10 Dfl	1,000	ST	21.06.2013
6	April	Scherz	1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Cancidas 50 mg TrS	1 Dfl	1,000	ST	19.06.2013
7	April	Scherz	1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Cancidas 50 mg TrS	1 Dfl	1,000	ST	20.06.2013
8	April	Scherz	1	11.11.1955	30245	C92.00	KK-05	9-0351	9-0351C	Cancidas 50 mg TrS	1 Dfl	1,000	ST	21.06.2013
9	Kanns	Achim	1	04.02.1994	28760		KK-05	9-0354	9-0354B		0,000			
10	Mai	Anfang	3	11.11.2011	30647		KK-05	9-0356	9-0356A	Thymoglobulin 25 mg TrS	5 ml	1,000	ST	20.02.2015
11	Maria	Kopf	2	07.07.2007	28562		KK-05	9-0355	9-0355B	Mucosolvan Tropfen		1,000	ML	03.04.2014

Abbildung 8.6.: Ausschnitt eines Screenshots aus MICROSOFT EXCEL mit fiktiven Daten. Dies ist eine Teildarstellung der aus dem ABAP-List-Viewer exportierten Tabelle aus Abbildung 8.1 auf Seite 106 als Tabellenkalkulation.

8.3. Daten auf dem Applikationsserver

Das Kapitel 7 auf Seite 95 behandelt die Weiterverarbeitung der CSV-Dateien mit dem Ziel, um beispielsweise CDA-Dokumente im XML-Format erstellen zu können. Die dazu verwendeten CSV-Dateien entsprechen in diesem Falle den wie in Abschnitt 5.8 auf Seite 83 transportierten Daten auf dem Applikationsserver.

In EURECA wird von einem anderen Projektpartner ein DWH entwickelt, welches als Quelldaten u.a. die obigen CDA-Dokumente verarbeiten soll. Durch die tägliche Generierung des aktuellen Standes wie in Kapitel 4 auf Seite 39 und Kapitel 5 auf Seite 55, sammelt sich nach der Weiterverarbeitung und der Einspielung in das DWH das Potential eines informationellen Mehrwerts.

Auf ein DWH werden in der Regel Analysetools (e.g., SAS³) angewendet, um die große Datenmenge zum Informationsgewinn erfassen zu können [51]. Dadurch können Daten als Resultat zusammengefasst werden oder bestimmte Teilaspekte analysiert werden. Im Gegensatz zum KIS liegt ein Vorteil in der Nutzung eines externen DWH auch darin, dass Zeit und Belastung keine übergeordnete Rolle spielen.

³<http://www.sas.com>

9. Übertragbarkeit

Der hier gewählte Ansatz mit generierten CSV-Dateien aus dem SAP System ist anpassbar und für andere Aufgabenstellungen nutzbar. Besonders CSV-Dateien lassen sich in einer Mehrzahl der KIS erstellen [60]. Daher habe ich ebenso eine CSV-Datei als Schnittstelle verwendet [81]. Eine Beschreibung der CSV-Datei ist in Unterabschnitt 5.6.5 auf Seite 77 zu finden. Einige Möglichkeiten der Weiterverwendung sind in Kapitel 7 auf Seite 95 und Kapitel 8 auf Seite 105 veranschaulicht.

Auch die Vorgehensweise innerhalb des SAP Systems wird für ein solches Szenario in vielen Fällen übertragbar sein. Gerade in anderen Krankenhäusern, welche ebenfalls ein KIS betreiben, welches auf SAP oder SAP in Kombination mit einem anderen Anbieter (i.e., SIEMENS) basiert. Im Segment der großen Krankenhäuser ab 800 Betten wird nach [57] zum damaligen Stand der Erhebung die Verwendung von SAP bei ca. 26% liegen. Die entsprechenden Standardtabellen sind also auch andernorts zu finden. Bei obigem System müsste man in einem Krankenhaus mit der Kombination SAP HEALTHCARE + I.S.H. MED lediglich geringfügige Erweiterungen einrichten, um eine entsprechende Ausgangsbasis zu haben. Für andere Systeme wird durch die Definition des Zeilentyps aus den passenden Tabellen in Kapitel 4 auf Seite 39 und Kapitel 5 auf Seite 55 deutlich, welche Werte gesucht werden.

Der Ansatz zur Datengewinnung wird analog zu dieser Arbeit im Allgemeinen damit verbunden sein, dass man eine Menge von relevanten Patienten definiert. Der Zeitrahmen der Erhebung (e.g., täglich, monatlich) und der Zeitraum (e.g., rückwirkend) der gesuchten Daten spielen dabei eine wesentliche Rolle. Ebenso benötigt man eine Liste der gesuchten Werte. Sollte das KIS eine entsprechende Exportfunktion nicht bereitstellen, wird man es mit *Software Engineering* [66] entsprechend erweitern. Für Datenbankzugriffe werden Abfragen in SQL oder einer anderen DML zum Einsatz kommen.

Ein etwas anderer Fall ist die vollständige Pflege einer Studie innerhalb eines KIS. Sollte dies nicht bereits vorgesehen sein, müssen dementsprechende Erweiterungen erst konzipiert werden. Es muss eine Aufstellung einer Studie innerhalb des System durch

Softwaretechnik möglich gemacht werden. Dies wird mit zahlreichen Schreibzugriffen auf die Datenbank und entsprechender Rechtevergabe innerhalb des KIS verbunden sein.

Die Datenextrahierung bei schon vorhandener oder erweiterter informationstechnischer Infrastruktur setzt natürlich voraus, dass diese auch genutzt wird. Sollten einzelne Werte oder Beobachtungen nicht zu den Pflichtfeldern eines KIS gehören, können diese Patientendaten für Studienzwecke nur dann verwendet werden, wenn man von der Möglichkeit der Dateneingabe Gebrauch macht.

Teil III.
Ergebnisse

Inzwischen werden in nahezu allen Kliniken in Deutschland Krankenhausinformationssysteme verwendet. Dabei fallen neben rein organisatorischen Belangen (e.g., Krankenkassenzugehörigkeit zur Abrechnung) auch medizinische Informationen an. Während diese Patientendaten in der Vergangenheit lediglich in einer schriftlichen Krankenakte erfasst wurden, sind sie nun vielerorts elektronisch hinterlegt. Patientendaten haben für sich genommen einen Wert für die Behandlung des Patienten selbst, aber eine Kombination mit anderen Patientendaten kann zu einem weiteren Mehrwert führen. Dies ist sowohl bei medizinischen Studien der Fall, als auch bei praktischen Erwägungen (e.g., Behandlung) innerhalb eines Krankenhauses.

Innerhalb eines Szenarios für ein EU-Projekt sollen auf täglicher Basis bestimmte Patientendaten (siehe Tabelle 4.1 auf Seite 42 und Tabelle 4.2 auf Seite 43) einer Station aus dem SAP System des Krankenhausinformationssystems in Homburg gewonnen werden. Anschließend werden sie mit den Daten aus anderen monolithischen Systemen (i.e., Mikrobiologie) zusammengeführt und weiterverarbeitet, um später in ein extern entwickeltes Data Warehouse transferiert zu werden. Verbunden mit entsprechenden Programmen werden dann Analysen seitens des Projektes erfolgen.

Das Auswahlkriterium für die obigen Daten sind alle Patienten, die zum aktuellen Stichtag stationär in der Klinik für Pädiatrische Onkologie und Hämatologie behandelt werden. Zur Gewinnung dieser Daten musste ich zuerst analysieren, an welcher Stelle diese Daten im SAP System vorhanden sind und wie man sie abfragen kann. Der Schlüssel dazu ist in diesem Fall die Suche nach allen aktuellen gültigen (e.g., nicht stornierten) stationären Patientenbewegungen auf dieser Station. Auf diese Weise kann mit der jeweils ersten Abfrage die Menge an Behandlungsfällen auf die relevanten reduziert werden, bevor damit weitere Daten abgefragt werden. Mit Hilfe dieser Daten wird eine mit den gesuchten Datentypen deklarierte interne Tabelle in SAP befüllt und Schritt für Schritt wie in Abbildung 9.1 auf der nächsten Seite ergänzt. Zum Zusammenstellen der Daten innerhalb des SAP-Systems habe ich ein Programm mit der Programmiersprache ABAP geschrieben. Anschließend erfolgt in diesem Programm der Export als CSV-Datei, weil dieser Ansatz auf viele andere Einrichtungen übertragbar ist [60]. Die CSV-Datei dient dabei als Schnittstelle für externe Systeme [81].

Es musste eine Kommunikationsstruktur geschaffen werden, um die Dateien auf einem Zielsever für Studienzwecke zusammenzubringen. Als Verbindung wurde der Kommunikationsserver des Krankenhausinformationssystems unter dankenswerter Unterstützung

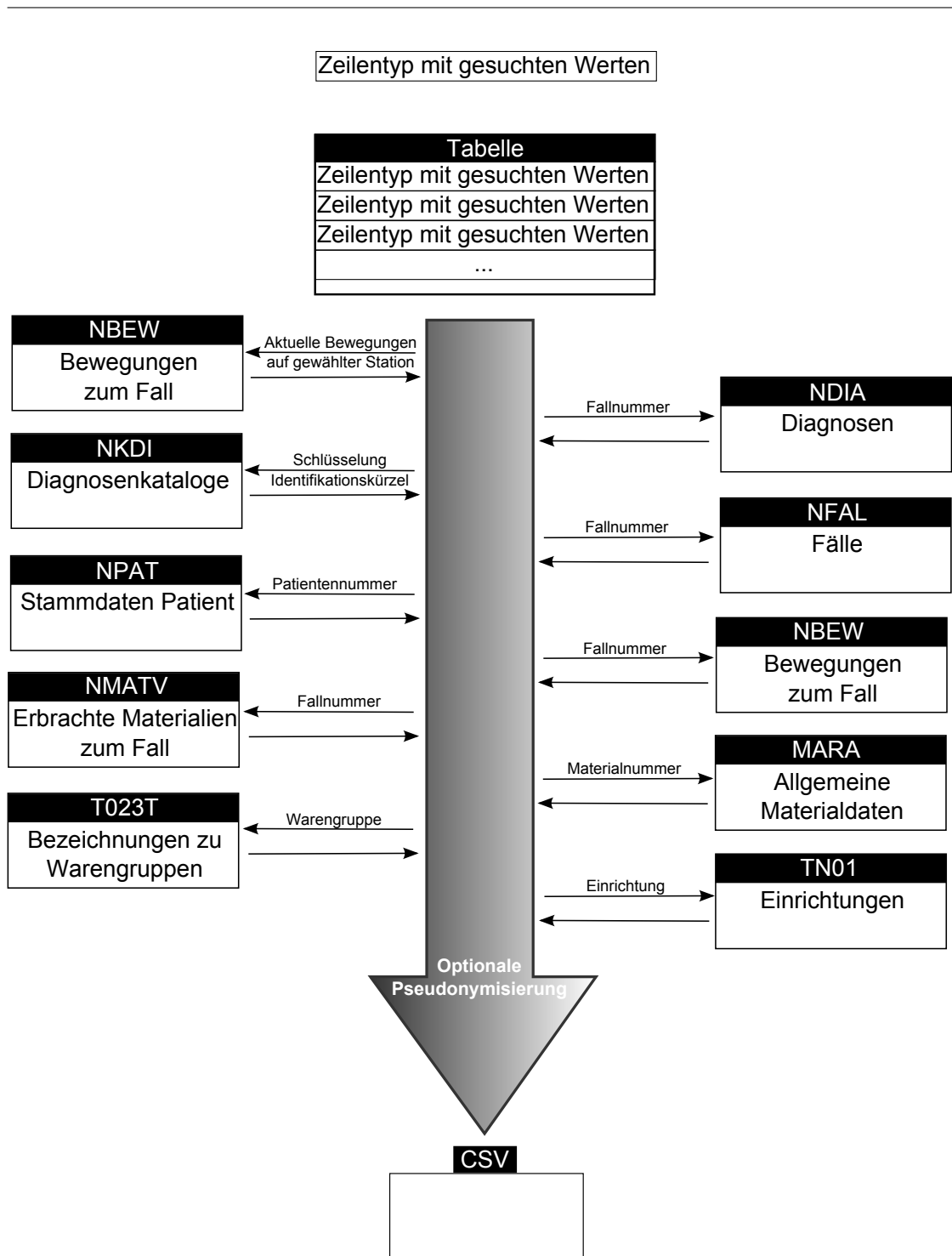
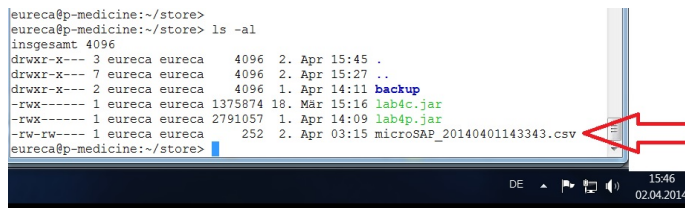


Abbildung 9.1.: Vereinfachte Darstellung der Befüllung einer internen Tabelle in SAP mit den gesuchten Daten. Bei den abgefragten Tabellen wird jeweils der wichtigste Schlüssel genannt.



(a) Manuelle Erstellung einer CSV-Datei im SAP-Testsystem am 1. April 2014.



(b) Betrachtung des Zielverzeichnis auf dem Zielsystem am 2. April 2014.

Abbildung 9.2.: Test der Kommunikationsstruktur aus dem SAP-System zum Zielsystem für Studien. Die enthaltenen Patientendaten sind fiktiv.

der Mitarbeiter aus dem Zentrum für Informations- und Kommunikationstechnik genutzt. Die geteilten Verzeichnisse wurden ab dem Zeitpunkt jede Nacht automatisch aktualisiert. Ergänzt wurde diese Struktur dann meinerseits durch die Bereitstellung der Exportdateien auf den jeweiligen Ausgangsservern und die Datenverschiebung auf dem Zielsystem. Ein erster Test zur Nutzung dieser Struktur verlief mit einer manuell auf dem SAP-Testsystem erstellten Datei erfolgreich, siehe Abbildung 9.2. Ein analoger Test fand später mit Erfolg in der Mikrobiologie mit einer Testdatei statt. Ab der zweiten Jahreshälfte 2014 wurde für einige Monate ein Vorversion des obigen ABAP-Programms täglich automatisch im SAP-Testsystem ausgeführt und die damit erzeugten CSV-Dateien dementsprechend täglich automatisch auf den Zielsystem für Studien transferiert. Dabei wurden Patientendaten ohne Bezug zu real existierenden Personen genutzt.

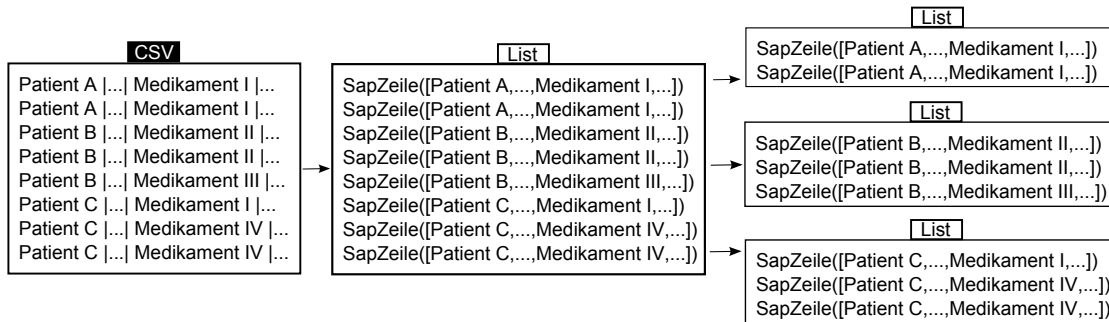
Bevor die Daten aus dem SAP von dem Zielsystem in das Data Warehouse gepusht werden können, sollen aus der exportierten CSV-Datei für jeden einzelnen Patienten zwei *Clinical Document Architecture* (CDA)-Dokumente in XML Form generiert werden. Dabei beinhaltet eine Datei die Medikation, während die andere Datei die restlichen Daten

(e.g., Station, Diagnose) enthält. In diesem Zusammenhang habe ich verdeutlicht, wie mit der CSV-Datei aus dem SAP gearbeitet werden kann, siehe dazu auch Abbildung 9.3 auf der nächsten Seite. Als Programmiersprache habe ich JAVA verwendet.

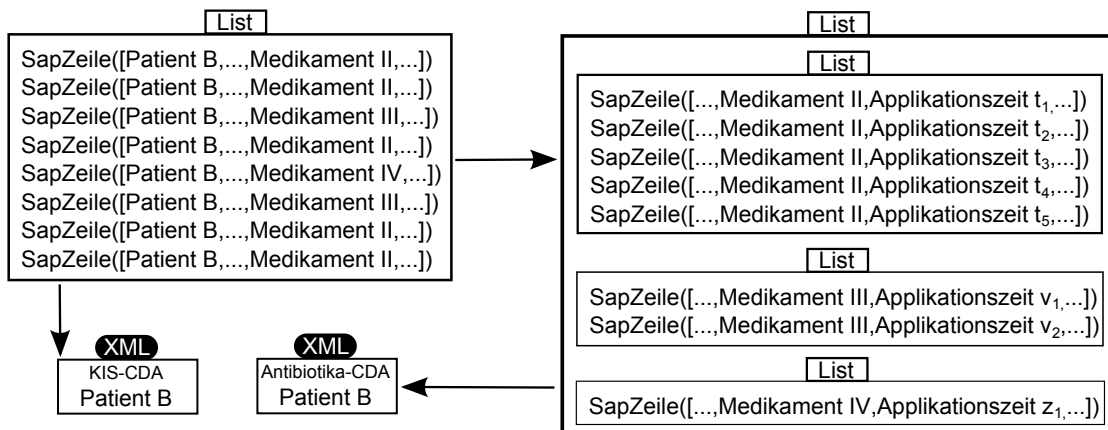
Neben dem erwähnten analytischen Potential eines Data Warehouse, bestehen auch andere Möglichkeiten zur Untersuchung der gewonnenen Patientendaten. Ich demonstrierte in dieser Arbeit sowohl eine Möglichkeit der Weiterverarbeitung der CSV-Datei mit einer Tabellenkalkulation, als auch SAP-interne Möglichkeiten (e.g., SAP List Viewer, Diagramme).

Maßnahmen hinsichtlich des Datenschutzes habe ich in dieser Arbeit besprochen und an der lokalen Situation demonstriert, inklusive einer Möglichkeit der Pseudonymisierung. Dabei wird ein Teil der Patientenstammdaten (e.g., Name) entfernt oder verarbeitet (e.g., Geburtsjahr statt Geburtsdatum) und die Patientenummer durch einen Hashwert ersetzt. Für den letzten Punkt wird eine Message-Digest Hashfunktion verwendet, was sowohl in ABAP als auch JAVA funktionierte. Entsprechend habe ich ein Programm zur Pseudonymisierung der Textdateien aus der Mikrobiologie in JAVA erstellt und das ABAP-Programm im SAP-System erweitert.

Mit der obigen Vorgehensweise wurde in dieser Arbeit die Gewinnung von Patientendaten aus einem vorhanden Krankenhausinformationssystem erklärt. Diese Vorgehensweise wäre auch an anderen Kliniken möglich, was natürlich eine entsprechende Nutzung (e.g., Eingabe von Daten) der Krankenhausinformationssysteme voraussetzt.



(a) Inhaltlicher Transfer der Patientenzeilen aus der CSV-Datei zu einzelnen Patienten.



(b) Extraktion der Informationen zur Medikamentenapplikation zu einem Patienten mit anschließender CDA-Generierung.

Abbildung 9.3.: Inhaltlicher Ablauf der Weiterverarbeitung einer CSV-Datei.

Teil IV.
Diskussion

In dieser Arbeit war es u.a. das Ziel, ein schon vorhandenes Krankenhausinformationssystem in Homburg in Form von SAP HEALTHCARE & I.S.H.MED so zu erweitern, dass es gesuchte Patientendaten für ein bestimmtes Szenario extrahieren kann. Dies geschah hier in Teilen am Beispiel eines EU-Projektes.

Beim Verschaffen eines ersten Überblicks zum Krankenhausinformationssystem wurde mir bewusst, dass es sich hierbei um ein sehr komplexes System handelt. Es stellte sich heraus, dass neben dem Hauptsystem in Form von SAP noch weitere monolithische Systeme existieren, welche bestenfalls lose gekoppelt sind. Daher musste, neben gar nicht erst elektronisch erfassten Daten, in Kauf genommen werden, dass andere Daten nicht ohne weiteres in strukturierter Form erreichbar sind. Dies ist der Tatsache geschuldet, dass diese Systeme für organisatorische oder wirtschaftliche Belange entwickelt worden sind, bzw. auf den aktuellen Tagesbetrieb ausgelegt wurden und weniger auf die externe informationstechnische Weiterverwendung der medizinischen Daten.

Die Situationsanalyse und der Gewinnungsprozess innerhalb des SAP Systems war davon gezeichnet, dass sich die Anforderungen an die Daten seitens Teilnehmern des EU-Projektes zum Teil noch änderten. Was an passenden Daten im SAP vorhanden war, wurde von mir gefunden und konnte auch exportiert werden. Dabei spielte das Selektionskriterium in Form der aktuell behandelten Patienten auf einer Station eine wesentliche Rolle.

Die Werte sind, jeweils einzeln betrachtet, möglicherweise noch nicht aussagekräftig, so dass ein Abarbeiten einer Liste gesuchter Werte nicht immer zum Ziel führt. Es werden in dieser Arbeit beispielsweise die ICD-Codes zu Diagnosen abgefragt, aber diese Kodierung umfasst nicht alle Gegebenheiten zu einem einzelnen Patienten. Wie in [2, 47] beschrieben, lässt sich damit alleine recht wenig über den individuellen Patienten und die Ausprägung einer Erkrankung sagen. Mit Hilfe der in meinem Programm zusätzlich abgefragten Informationen (e.g., Alter) kann man dem gerecht werden, doch letztlich muss diese Kombination von Informationen immer noch von einem Arzt bewertet werden. Die Abfrage der ICD-Codes habe ich um den jeweils verwendeten Katalog erweitert. Diese Notwendigkeit entsteht aus der Existenz verschiedener Versionen und länderspezifischer ICD-Erweiterungen.

Ein nicht zu unterschätzender Faktor ist dabei die Veränderbarkeit von Werten. Beispielsweise können Diagnosen seitens des Klinikpersonals noch angepasst bzw. ausformuliert werden, bevor der Patient abgerechnet wird. Eine erste Notiz im System ist daher

eventuell noch unpräzise. Eine zeitverzögerte Lösung, nämlich auf eine “finale” Eingabemodifikation zu warten, wäre nur eine Annäherung. Eine nächtliche Abfrage reduziert zumindest die Wahrscheinlichkeit auf Daten eines laufenden Eingabeprozesses (e.g., neuer Patient) zuzugreifen. Die kontinuierliche (e.g., tägliche) Abfrage aktueller Falldaten wird die gesammelten Informationen letztlich auf den aktuellsten Stand bringen. Eine finale Anpassung einer Kodierung innerhalb des Krankenhausinformationssystems in Zusammenhang mit Abrechnungszwecken hat aber nicht das gleiche Ziel, wie beispielsweise die spätere Abfrage von Diagnosen zu medizinischen Studienzwecken.

Andere Systeme

In [8] wird das VPH¹-Share Projekt innerhalb der VPH Initiative der Europäischen Kommission vorgestellt. Dieses soll in Zukunft ein Framework zum Verständnis der physiologischen Prozesse innerhalb des menschlichen Körpers anbieten. Dazu konzentrieren sich die Autoren auf einen Patienten-Avatar, welcher das digitale Gegenstück zum entsprechenden Menschen darstellen soll. Es wird in [8] auch von einer Reihe von Herausforderungen berichtet, beispielsweise die Erstellung passender Modelle für die Daten. Daher sieht man es dort auch als “ultimatives Ziel” an, eine Standardisierung von entsprechenden Vorgehensweisen zu erreichen. Das Projekt wird zwar selbst vieles über *Cloud Computing* [83] anbieten, doch die Daten werden auch bei den Krankenhäusern verbleiben. Bei der Dateninfrastruktur greift man dort unter anderem auf CSV-Dateien zurück.

In meiner Arbeit wird, im Gegensatz zu oben, die Gewinnung von bestimmten akut vorhandenen Daten zu einer bestimmten Gruppe (c.q., derzeit stationär behandelt) von Patienten konkret angegangen. Beides kann man zusammenbringen, entweder durch Annäherung an die praktische Situation von Kliniken, oder durch weitreichende Lösungsansätze (e.g., Pflichtstandards für Kliniken).

Das in Abschnitt 3.6 auf Seite 33 beschriebene System in der Neurologie Püttlingen geht einen anderen Weg. Das Ziel ist dort eine einfachere und genauere Generierung von Arztbriefen, unter Nutzung eines dazu entwickelten Krankenhausinformationssystems (i.e., HoWoS). Während der in dieser Arbeit beschriebene Ansatz auf eine vorhandene Struktur zugreift, ist der Ansatz in der Neurologie Püttlingen durchweg nach vorne gerichtet bzw. alte Strukturen wurden schlicht ersetzt. Während das System in meiner

¹Virtual Physiological Human

Arbeit eher ein sanft eingeführte Ergänzung darstellt, musste in der Neurologie Püttlingen seitens des Personals mit einem komplett neuen Krankenhausinformationssystem gearbeitet werden. Es wird dort mit Hilfe komplexer Auswahlstrukturen innerhalb von dynamisch begleitenden Textbausteinen gearbeitet, um die vorherigen Informationsdefizite bei weichen Daten zu vermindern. Auf diese Weise werden die Eingaben standardisiert, zudem gibt es ein Kontrollsystem zur tatsächlichen Eingabe dieser Daten. Ein Arztbrief kann zeitnah aus dem laufenden System generiert werden und die Diagnosen sind weniger von der individuellen Ausbildung bzw. Erfahrung des Arztes abhängig. Die Idee der Freitextgenerierung aus kodierten Informationen innerhalb eines Krankenhausinformationssystems ist in [65] weiter beschrieben.

Letztlich werden in der Neurologie Püttlingen einige zu Beginn dieses Kapitels geäußerte Herausforderungen durch eine Art Eingabezwang, innerhalb eines vorgegebenen Rahmens, gegenüber dem behandelnden Arzt gelöst. Im Artikel [85] wird dagegen darüber berichtet, dass viele Kliniker die papierbasierte Patientendokumentation “gar nicht schlecht” finden, da die elektronische Erfassung und Verarbeitung nicht immer einen Zugewinn bringt. Man muss sich also um die weitere Akzeptanz der Arbeitsweise mit einem Krankenhausinformationssystem bemühen.

Ein Vorteil in der Vorgehensweise dieser Arbeit ist allerdings die Nutzung eines bereits vorhandenen Systems. Beim Start einer neuen klinischen Studie oder anderen Projekten wird kaum die Option bestehen, bei teilnehmenden Kliniken die Entwicklung, Anschaffung und Einführung eines dazu passenden Krankenhausinformationssystems abzuwarten.

Weiterverarbeitung

Zur später erfolgenden Weitergabe der Daten in das Data Warehouse des EU-Projektes, werden aus den CSV-Dateien noch *Clinical Document Architecture* (CDA)-Dokumente im XML-Format generiert. Dies ist allerdings nur zum Teil eine direkte Übertragung von Werten. Die in Kapitel 7 auf Seite 95 angesprochenen Herausforderungen (e.g., Strukturunterschiede) ergaben sich an mehreren Stellen. Daher sollte der Ersteller eines solchen vorgegebenen CDA-Dokuments zunächst definieren, welche Werte benötigt werden und wie diese berechnet werden sollen. Dabei wird es schwierig sein, alle möglichen Gegebenheiten vor Ort vorausszusehen. Die Struktur der gegebenen CDA-Dokumente entspricht unter Umständen nicht der Struktur des Krankenhausinformationssystems im Kranken-

haus. Je größer die Zahl der teilnehmenden Organisationen innerhalb eines Projektes wird, desto schwieriger wird es in den meisten Fällen. Es wird in vielen Krankenhäusern zumindest leichte Unterschiede geben. Neben technischen Aspekten wäre hier auch die Eingabe von Werten zu nennen (e.g., zeitnah oder verspätet) oder das Vernachlässigen vorhandener Optionen des Systems (e.g., keine Zeit zur Eingabe von Detailinformationen).

Auf der Seite des Krankenhauses wird sich die Frage stellen, wie gewissenhaft man die Interpretation eines vorgegebenen Zwischenformats gestalten kann. Dabei können Rückfragen entstehen, die ohne Zugänglichkeit aller Beteiligten zu Verzögerungen führen können. Die zeitintensive Recherche zur Erkundung der gesuchten Werte, wie sie dann in Homburg stattgefunden hat, kann nicht überall geleistet werden. Das kann im Einzelfall zu einer unbeabsichtigten Fehlinterpretation eines Zwischenformats führen.

Bei einer geringen Zahl an teilnehmenden Kliniken, ist eine Ausrichtung der Zwischenformate an den Teilnehmern sinnvoll. Bei einer größeren Teilnehmerzahl muss die Leistungsfähigkeit des Data Warehouse vergrößert werden, um möglichst viele CDA-Dokumente unterschiedlichen Inhalts und weitere Formate verarbeiten zu können. Die Kliniken können in dem Falle das CDA-Dokument entsprechend ihrer Struktur anpassen und nutzen. Bei der Anwendung von lokalen Analysetools ist die Weiterverarbeitung der CSV-Datei oder die Füllung eines Data Warehouse ohnehin vermeidbar.

Literaturverzeichnis

- [1] ABTS, D. : *Grundkurs JAVA: Von den Grundlagen bis zu Datenbank- und Netzanwendungen*. Springer, 2013
- [2] BAIERL, M. : *Herausforderung Alltag: Praxishandbuch für die pädagogische Arbeit mit psychisch gestörten Jugendlichen*. Vandenhoeck & Ruprecht, 2014
- [3] BAIRU, M. ; CHIN, R. : *Global Clinical Trials Playbook: Capacity and Capability Building*. Academic Press, 2012
- [4] BARNES, D. J. ; KÖLLING, M. : *Java lernen mit BlueJ: Eine Einführung in die objektorientierte Programmierung*. Pearson Deutschland, 2009
- [5] BAROLIN, G. S.: *Integrierte Psychotherapie: Anwendungen in Der Gesamtmedizin und Benachbarten Sozialberufen*. Springer-Verlag, 2006
- [6] BAUER, A. ; GRATZL, G. : *mySAP SCM Materialwirtschaft*. Pearson Deutschland GmbH, 2004
- [7] BEHRENS, G. ; KUZ, V. ; BEHRENS, R. : *Softwareentwicklung Von Telematikdiensten: Konzepte, Entwicklung und Zukünftige Trends*. Springer, 2011
- [8] BENKNER, S. ; BISBAL, J. ; ENGELBRECHT, G. ; HOSE, R. D. ; KANIOVSKYI, Y. ; KOEHLER, M. ; PEDRINACI4, C. ; WOOD, S. : Towards Collaborative Data Management in the VPH-Share Project. In: *Euro-Par 2011 Workshops, Part I, LNCS 7155* (2012), S. 54–63
- [9] BENSON, T. : *Principles of Health Interoperability HL7 and SNOMED, 2nd edition*. Springer London, 2012
- [10] BÖHME, S. ; NOHR, R. F. ; WIEMER, S. : *Sortieren, Sammeln, Suchen, Spielen: Die Datenbank als mediale Praxis*. LIT Verlag Münster, 2012

- [11] BIETHAHN, J. ; MUCKSCH, H. ; RUF, W. : *Ganzheitliches Informationsmanagement Band 1: Grundlagen (6. Auflage)*. Oldenbourg, 2004
- [12] BOONE, K. W.: *The CDA TM book*. Springer Science & Business Media, 2011
- [13] BROCHHAUSEN, M. ; WEILER, G. ; SCHERA, F. ; RAUCH, J. ; GRAF, N. ; KIEFER, S. : *Ontology-based Trial Management System (ObTiMA)*. Available from Nature Precedings. <http://dx.doi.org/10.1038/npre.2009.3753.1>. Version: Sep 2009
- [14] CODD, E. F.: *The Relational Model for Database Management Version 2*. Addison-Wesley, 1990
- [15] COHRS, M. : *Ein Architekturmodell für SAP-Anwendungen: Leicht wartbare, erweiterbare und teamorientierte SAP-Eigenentwicklungen mit ABAP Objects*. Springer-Verlag, 2011
- [16] DER, G. ; EVERITT, B. S.: *Statistical Analysis of Medical Data Using SAS*. CRC Press, 2005
- [17] EARL-SLATER, A. : *The Handbook of Clinical Trials and Other Research*. Radcliffe Medical Press, 2002
- [18] ECKERT, C. : *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. Oldenbourg, 2011
- [19] ECKSTEIN, R. ; CASABIANCA, M. : *XML: kurz & gut, 2. Auflage*. O'Reilly, 2003
- [20] FENSTERMACHER, D. ; STREET, C. ; MCSHERRY, T. ; NAYAK, V. ; OVERBY, C. ; FELDMAN, M. : The Cancer Biomedical Informatics Grid (caBIG). In: *Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference Shanghai, China, September 1-4, 2005*, S. 743–746
- [21] FISCHER, P. ; HOFER, P. : *Lexikon Der Informatik*. Springer, 2011
- [22] GELL, G. ; GITTER, T. : Hospital Information System/ Electronic Health Record (HIS/HER) and clinical research. In: *Digital Excellence: University Meets Economy (2008)*, S. 137–146
- [23] GOEPFERT, A. ; CONRAD, C. : *Unternehmen Krankenhaus*. Georg Thieme Verlag KG, 2013

- [24] GRÄBER, S. ; RICHTER, S. ; FOLZ, J. ; PHAM, T. ; JACOB, P. ; SCHILLING, M. K.: Clinical Pathways in General Surgery - Development, Implementation, and Evaluation. In: *Methods Inf Med, Sch* 46 (2007), S. 574–579
- [25] GRÄBER, S. : *Rahmenkonzept für das Klinikinformationssystem der Universitätskliniken des Saarlandes (1. Fortschreibung)*. pdf auf Webseite des ZIK, abgerufen am 24.10.2014, 15:00Uhr. http://www.uniklinikum-saarland.de/fileadmin/UKS/Einrichtungen/Zentrale_Dienste/ZIK/Projekte/Rahmenkonzept/rahmenkonzept2000.pdf. Version: 5 2000
- [26] GRONAU, N. : *Enterprise resource planning und Supply-chain-Management: Architektur und Funktionen*. Oldenbourg Verlag, 2004
- [27] GUBBELS, H. : *SAP ERP - Praxishandbuch Projektmanagement*. Springer, 2013
- [28] GULOVA, V. : *Pflegeinformatik: Die Rolle der Informations- und Kommunikationstechnologien (Ikt) in der Gesundheits- und Krankenpflege*. disserta, 2012
- [29] HAAS, P. : *Medizinische Informationssysteme und Elektronische Krankenakten*. Springer-Verlag Berlin, 2005
- [30] HARING, R. : *Der überforderte Patient: Gesund bleiben im Zeitalter der Hightech-Medizin*. C.H.Beck, 2014
- [31] HEISS, F. J. ; GRATZL, G. ; WEIRICH, E. : *SAP NetWeaver Web Application Server*. Pearson Deutschland GmbH, 2005
- [32] HELP.SAP.COM: *i.s.h.med Klinisches System - Datenschnittstelle*. Webseite abgerufen am 11.11.2014, 13:35Uhr. http://help.sap.com/saphelp_afs64/helpdata/de/9c/2adebb2fce11d39b0d00805a163521/frameset.htm
- [33] HERBIG, B. : *Informations- und Kommunikationstechnologien im Krankenhaus: Grundlagen, Umsetzung, Chancen und Risiken*. Schattauer Verlag, 2006
- [34] HEUER, A. J. (Hrsg.) ; SCANLAN, C. L. (Hrsg.): *Wilkins' Clinical Assessment in Respiratory Care, 7th Edition*. Elsevier Health Sciences, 2013
- [35] HÖPKEN, A. ; NEUMANN, H. : *Datenschutz in der Arztpraxis: ein Leitfaden für den Umgang mit Patientendaten*. C. F. Müller, 2008

- [36] HÄRDER, T. ; RAHM, E. : *Datenbanksysteme - Konzepte und Techniken der Implementierung*. Springer, 1999
- [37] HUBWIESER, P. ; AIGLSTORFER, G. : *Fundamente der Informatik: Ablaufmodellierung, Algorithmen und Datenstrukturen*. Oldenbourg, 2004
- [38] HÄUSSLER, B. : *Arzneimittel-Atlas 2013*. Spinger, 2013
- [39] INTRUP, D. : *Umsetzung Der Datenarchivierung Im Sap-Erp-System*. Diplomica Verlag, 2009
- [40] JÄGER, H. : *Persönliches Interview, geführt vom Autor dieser Arbeit. Püttlingen, 18.9.2014*.
- [41] JOOS, T. : *Windows 7 - Platin Edition*. Pearson Deutschland, 2010
- [42] KELLER, H. : *ABAP-Referenz*. SAP PRESS, 2010
- [43] KELLER, H. ; KRÜGER, S. : *ABAP Objects: ABAP-Programmierung mit SAP NetWeaver*. SAP PRESS, 2006
- [44] KEMPER, A. ; EICKLER, A. : *Datenbanksysteme: Eine Einführung*. Oldenbourg, 2006
- [45] KH-CIRS-NETZ: *Ärztliches Zentrum für Qualität in der Medizin. Fall des Monats 'Mai 2013': Patientenidentifikation in großen Krankenhaus-Datenbanken*. Abgerufen am 22.10.2014, 16:00Uhr. <http://www.kh-cirs.de/faelle/mai13.html>. Version: 5 2013
- [46] KÜHNHAUSER, K.-H. ; FRANZ, T. : *Discover ABAP: Der praktische Einstieg, 3. Auflage*. SAP PRESS, 2011
- [47] KOCH, O. : *Kontextorientierte Informationsversorgung in Medizinischen Behandlungsprozessen: Informationslogistische Konzeption Eines Lösungsansatzes Für Ärzte*. Springer, 2010
- [48] KOAGENT: *SAP ABAP - Questions and Answers*. Jones & Bartlett Learning, 2009
- [49] KOKOL, P. ; ZUPAN, B. ; STARE, J. : *Medical Informatics Europe '99*. IOS Press, 1999

-
- [50] KOPP, T. ; SCHÖCHLIN, J. : *Die Arztpraxis der Zukunft: Ein ganzheitliches IT-Konzept zur Unterstützung der ambulanten Gesundheitsversorgung*. EUL Verlag, 2012
- [51] KRISHNAN, K. : *Data Warehousing in the Age of Big Data*. Newnes, 2013
- [52] KRYKWINSKI, C. ; SCHEPPER, K. D. ; VANSUYT, C. ; ALONSO, R. ; REY, D. P. ; FÓRGÓ, N. ; GORALCZYK, M. ; LODZIG, B. ; RUEPING, S. ; TRABOLD, D. ; ROHM, K. ; KIEFER, S. ; KONDYLAKIS, H. ; KOUMAKIS, L. ; HOLLINK, F. B. L. ; HUANG, Z. ; TEIJE, A. ten ; LEEUWEN, J. V.: *EURECA - Enabling information re-Use by linking clinical Research and CAre - Deliverable: D1.2 Definition of relevant user scenarios based on input from users*. Abgerufen am 04.03.2015. http://share.ecancer.org/EURECA/Deliverables/D1%20_V1%200%20Definition%20of%20relevant%20user%20scenarios%20based%20on%20input%20from%20users.pdf. Version: 2012
- [53] KUJATH, B. : *Workflow-Management im Krankenhaus*, Fachhochschule für Technik und Wirtschaft Berlin, Diplomarbeit, 2003
- [54] LANGE, L. ; JAEGER, H. ; SEIFERT, W. : *Good Clinical Practice I: Grundlagen und Strategie*. Springer, 2013
- [55] LANGENBERG, S. ; UERLICH, M. ; NEUGEBAUER, M. ; SCHNEIDER, C. : EAI im Krankenhaus - Ein Erfahrungsbericht zur Koppelung von SAP IS-H mit dem Klinischen Arbeitsplatzsystem ORBIS. In: *Enterprise application integration: Tagungsband des 2. GI-/GMDS-Workshops zum Thema Enterprise Application Integration, Philipps-Universität Marburg, 30.6. - 1.7. (2005)*, S. 28–35
- [56] LÖBER, N. : *Fehler und Fehlerkultur im Krankenhaus*. Springer-Verlag, 2012
- [57] LEIMEISTER, J. M. ; KLAPDOR, S. ; HÖRMANN, C. ; KRUMHOLTZ, H. : *IT-Management in deutschen Krankenhäusern: Eine empirische Untersuchung unter IT-Entscheidungssträgern*. Books on Demand, 2011
- [58] LEITNER, F. ; GAUS, W. ; HAUS, R. ; KNAUP-GREGORI, P. ; PFEIFFER, K.-P. ; WAGNER, J. : *Medizinische Dokumentation: Grundlagen einer qualitätsgesicherten integrierten Krankenversorgung, 6. Aufl.* Schattauer Verlag, 2012

- [59] MAIKISCH, H. : *Einführung von SAP R/3 im Krankenhaus*, Donau-Universität Krems, Diplomarbeit, 2001
- [60] MATE, S. : *Ein Semantic-Web-Ansatz zum Mapping klinischer Metadaten am Beispiel eines Bioproben-/Projektvermittlungs-Portals für das DPKK auf der Basis von i2b2*, Friedrich-Alexander-Universität Erlangen-Nürnberg Lehrstuhl für Medizinische Informatik, Diplomarbeit, 2011
- [61] MATZKE, B. : *ABAP: die Programmiersprache des SAP-Systems R/3*. Pearson Deutschland, 2002
- [62] McLAUGHLIN, B. ; EDELSON, J. : *Java and XML*. O'Reilly, 2006
- [63] MERZ, T. ; DRÜMMER, O. : *Die PostScript- & PDF-Bibel, 2. Auflage*. PDFlib GmbH, 2002
- [64] NEMETH, E. ; SNYDER, G. ; HEIN, T. : *Handbuch zur Linux-Systemverwaltung*. Pearson Education Deutschland GmbH, 2004
- [65] PANZARASA, S. ; QUAGLINI, S. ; PESSINA, M. ; CAVALLINI, A. ; MICIELI, G. : GIFT: a tool for generating free text reports from encoded data. In: *11th Mediterranean Conference on Medical and Biomedical Engineering and Computing IFMBE Proceedings* Bd. 16, Springer, June 2007, S. 152–156
- [66] PRESSMAN, R. S.: *Software Engineering: A Practitioner's Approach*. Mcgraw-Hill, 2009
- [67] PÖTTGEN, N. : *Schriften zum deutschen und europäischen öffentlichen Recht*. Bd. 20: *Medizinische Forschung und Datenschutz*. Lang, 2009
- [68] RAINARDI, V. : *Building a Data Warehouse: With Examples in SQL Server*. Apress, 2007
- [69] RAMPETSREITER, C. : *E-Health in der biomedizinischen Analytik: Ein Grobkonzept für die elektronische Übertragung von Labordaten*. disserta, 2012
- [70] REICHE, D. ; BINDING, M. ; BOSS, N. ; WANGERIN, G. : *Roche Lexikon Medizin*. Elsevier Health Sciences, 2003

- [71] RÖHRIG, B. ; PREL, J.-B. D.: Erkenntnisgewinn in der Medizin durch Studien. In: *Krankenhaushygiene up2date* 5 (2010), S. 133–147
- [72] RIEKERT, R. : *ABAP-Programmierung - Fortgeschrittene Programmieretechniken für ABAP*. Addison-Wesley, 2001
- [73] ROBBINS, A. ; BEEBE, N. H. F.: *Klassische Shell-Programmierung*. O'Reilly Germany, 2006
- [74] SÖBBING, T. : *IT/IP-Rechte im Unternehmenskauf: Leitfaden für Information Technology & Software*. Diplomica Verlag, 2010
- [75] SCHLEGEL, H. : *Steuerung der IT im Klinikmanagement: Methoden und Verfahren*. Vieweg Teubner, 2010
- [76] SCHÜLLER, A. M. ; DUMONT, M. : *Die erfolgreiche Arztpraxis*. Spin, 2013
- [77] SCHLOSSER, J. : *Wissenschaftliche Arbeiten schreiben mit LaTeX: Leitfaden für Einsteiger*. mitp Professional, 2013
- [78] SCHUMACHER, M. ; SCHULGEN, G. : *Methodik klinischer Studien: Methodische Grundlagen der Planung, Durchführung und Auswertung, 3. Auflage*. Springer, 2008
- [79] SCHWAIGER, R. : *Schrödinger programmiert ABAP*. Galileo Press, SAP Press, 2013
- [80] SILVERSTON, L. : *The Data Model Resource Book, Vol. 1: A Library of Universal Data Models for All Enterprises*. Wiley, 2001
- [81] SKULSCHUS, M. ; WIEDERSTEIN, M. : *SQL Server 2012: XML-Integration mit T-SQL*. Comelio Medien, 2012
- [82] SMITH, L. : *Bash Shell: Essential Programs for Your Survival at Work: For Computer Programmers and Script-Writers*. Larry L. Smith, 2006
- [83] SRINIVASAN, S. : *Cloud Computing Basics*. Springer, 2014
- [84] STAUSBERG, J. ; KOCH, D. ; INGENERF, J. ; BETZLER, M. : Comparing Paper-based with Electronic Patient Records: Lessons Learned during a Study on Diagnosis and

- Procedure Codes. In: *J Am Med Inform Assoc.* 10 (2003), Sep-Oct, Nr. 5, S. 470–477
- [85] TANGE, H. : The paper-based patient record: Is it really so bad? In: *Computer Methods and Programs in Biomedicine* 48 (1995), S. 127–131. – ISSN 0169–2607. – {MIE} 94 Selected Papers
- [86] TEAM, L. D.: *LibreOffice 4.0 Getting Started Guide.* Lulu.com, 2013
- [87] THEOBALD, P. : *Profikurs ABAP, 2.Auflage.* Vieweg, 2007
- [88] TINNEFELD, M.-T. ; EHMANN, E. ; GERLING, R. W.: *Einführung in das Datenschutzrecht: Datenschutz und Informationsfreiheit in europäischer Sicht.* Old, 2005
- [89] TÖNNIES, K. : *Grundlagen der Bildverarbeitung.* Addison-Wesley, 2005
- [90] ULLRICH, M. : *SAP R 3 - der schnelle Einstieg.* Pearson Deutschland, 2000
- [91] VENOT, A. ; BURGUN, A. ; QUANTIN, C. : *Medical Informatics, e-Health: Fundamentals and Applications.* Springer Science & Business Media, 2013
- [92] VIETEN, M. : *Laborwerte verstehen leicht gemacht: Alle wichtigen Werte von A-Z.* Georg Thieme Verlag, 2009
- [93] WAGNER, M. : *Pharmazeutische Informationssysteme: Modellierung, Informationsstrukturen und Kommunikation interdisziplinär ausgerichteter Datenbanksysteme.* Vieweg, 2000
- [94] WENZEL, P. : *Betriebswirtschaftliche Anwendungen mit SAP R/3.* Spring, 2001
- [95] WENZEL, P. : *Rechnungswesen mit SAP R/3.* Vieweg, 2001
- [96] WHO: *International Statistical Classification of Diseases and Related Health Problems - 10th Revision - Volume 3 Alphabetical index - Second Edition.* World Health Organization Geneva, 2004
- [97] WIEKEN, J.-H. : *SQL: Einstieg für Anspruchsvolle.* Pearson Deutschland GmbH, 2009
- [98] WILLEMER, A. : *Linux-Server für Einsteiger: Mit Debian GNU/Linux und Ubuntu Server.* John Wiley & Sons, 2014

- [99] WITT, B. C.: *Datenschutz kompakt und verständlich: Eine praxisorientierte Einführung*. Springer, 2010
- [100] YUEN, P. ; LAU, V. : *Practical Web Technologies*. Addison Wesley, 2003
- [101] ZIK: *Klinikinformationssystem Überblick*. Webseite abgerufen am 23.10.2014, 15:45Uhr. http://www.uniklinikum-saarland.de/de/einrichtungen/zentrale_einrichtungen/zik/klinikinformationssystem/

Abbildungsverzeichnis

1.1.	Ausschnitt eines Screenshots mit fiktiven Patientendaten aus dem Testsystem von Homburg in der Belegungsübersicht einer fiktiven Station. Das Dreieck mit Ausrufezeichen weist auf besondere Ansteckungsgefahr hin, während das darunterliegende blaue Symbol auf andere Risikofaktoren hinweist. Das Klicken auf dieses blaue Symbol führt zur Abbildung 1.2 auf Seite 16.	16
1.2.	Ausschnitt eines Screenshots aus dem Testsystem von Homburg. Dies ist eine detaillierte Ansicht für die Risikofaktoren eines fiktiven Patienten aus Abbildung 1.1 auf Seite 16.	16
3.1.	Darstellung eines Datenbanksystems.	29
3.2.	Ausschnitt eines Screenshots aus dem Testsystem T21 des UKS Homburg zur Produktversion.	31
4.1.	Datenerhebung und Transport im Mikrobiologieszenario bis zum Studienserver.	41
4.2.	Screenshot eines Teils der Tabelle NPAT aus dem SAP Diktionär des T21.	45
4.3.	Screenshot eines Teils der Tabelle NFAL aus dem SAP Diktionär des T21 mit Schlüsselfeld <i>FALNR</i> (Fallnummer). Die Tabelle enthält mit <i>PATNR</i> auch die dem Fall zugehörige Patientenummer.	46
4.4.	Screenshot eines Teils der Tabelle NBEW aus dem SAP Diktionär des T21. Markiert sind die Felder <i>EINRI</i> , <i>FALNR</i> und <i>LFDNR</i> aus Tabelle 4.3 auf Seite 47.	48
4.5.	Screenshot eines Teils der Tabelle NBEW aus dem SAP Diktionär des T21. Markiert sind die Felder <i>ORGPF</i> , <i>ZIMMR</i> und <i>BETT</i> aus Tabelle 4.3 auf Seite 47.	49

4.6.	Darstellung der Abfrage aus Listing 4.2 auf Seite 51 mit Hilfe des ABAP-List-Viewers in Listing 4.3 auf Seite 53.	52
5.1.	Erster Teil der Ergebnisausgabe im ABAP-List-Viewer des T21. Die markierten Spalten sind der jeweilige Anschluss zur Fortsetzung der Tabelle. Hierbei ist zu beachten, dass es sich um fiktive Daten aus einem Testsystem handelt. Daher sind manche Einträge (e.g., Medikamente) nicht vollständig vorhanden. Die Fortsetzung befindet sich in Abbildung 5.2 auf Seite 76.	75
5.2.	Zweiter Teil der Ergebnisausgabe im ABAP-List-Viewer des T21 mit fiktiven Patienten. Die markierten Spalten sind der jeweilige Anschluss zur Fortsetzung der Tabelle. Der vorherige Teil befindet sich in Abbildung 5.1 auf Seite 75.	76
5.3.	Ausschnitt mit fiktiven Patienten aus der CSV-Datei zur ALV-Ausgabe in Abbildung 5.1.	79
5.4.	Ausschnitt aus einer .lab-Datei mit fiktiven Testdaten.	82
5.5.	Selektionsbild für das ABAP-Programm.	86
6.1.	Ausschnitt des Ergebnisses der Generierung eines Hashwertes für die Patientennummer und der Konvertierung des Geburtsdatums im ABAP-List-Viewer.	93
6.2.	Pseudonymisierung der entsprechenden fiktiven Testwerte einer .lab Datei mit LABANOM.	94
7.1.	Erstellung von CDA-Dokumenten für einzelne Patienten aus der CSV-Datei des SAP.	97
8.1.	Verbesserte Ausgabe mit dem ABAP-List-Viewer. Es handelt sich um fiktive Daten. Mehrere Spalten sind ausgeblendet und die Tabelle ist nach Name und Vorname sortiert. Nach dem Auslösen der <i>Druckvorschau</i> in der <i>Systemfunktionsleiste</i> kann diese Tabelle im Folgebildschirm als Tabellenkalkulation exportiert werden.	106
8.2.	Screenshot eines Abfrageresultats von GRAPHSTATION im ALV, sortiert nach Station und Geschlecht. Es handelt sich hierbei um fiktive Patienten und Stationen.	107

8.3.	Visuelle Darstellungsbeispiele zur Analyse in SAP mit GRAPHSTATION. Für diese Abbildung wurden 5 fiktive Stationen ausgewählt.	109
8.4.	Visuelle Darstellungsbeispiele zur Analyse in SAP mit GRAPHSTATION. Für diese Abbildung wurden 14 fiktive Stationen ausgewählt.	110
8.5.	Darstellung eines Teils der CSV-Datei als Pivot-Tabelle mithilfe von LIBREOFFICE CALC. Die Patientendaten sind fiktiv.	111
8.6.	Ausschnitt eines Screenshots aus MICROSOFT EXCEL mit fiktiven Daten. Dies ist eine Teildarstellung der aus dem ABAP-List-Viewer exportierten Tabelle aus Abbildung 8.1 auf Seite 106 als Tabellenkalkulation.	111
9.1.	Vereinfachte Darstellung der Befüllung einer internen Tabelle in SAP mit den gesuchten Daten. Bei den abgefragten Tabellen wird jeweils der wich- tigste Schlüssel genannt.	118
9.2.	Test der Kommunikationsstruktur aus dem SAP-System zum Zielsystem für Studien. Die enthaltenen Patientendaten sind fiktiv.	119
9.3.	Inhaltlicher Ablauf der Weiterverarbeitung einer CSV-Datei.	121

Tabellenverzeichnis

1.1. Relationen als flache Tabellen, mit unterstrichenen Schlüsselattributen. Darin enthalten ist unter anderem folgende Information: Die Patientin Schmidt (<i>PATNR 1341</i>) hat (<i>PATNR 1341 / TERMNR 2306</i>) einen Termin (<i>TERMNR 2306</i>) zum Zeitpunkt 10:30Uhr bei Arzt Schneider in Zimmer 4.02.	20
3.1. Beispiele für SQL-Fragen zu den Relationen aus Tabelle 1.1 auf Seite 20. (a) Abfrage der Namen aller Patienten. (b) Es werden Name, Geschlecht und Geburtsdatum des Patienten abgefragt, welcher um 10:30Uhr einen Termin hat.	30
4.1. Gesuchte Werte für das Mikrobiologieszenario (Stand 10.11.2014) Teil 1. Der zweite Teil ist in Tabelle 4.2 auf Seite 43 zu finden.	42
4.2. Gesuchte Werte für das Mikrobiologieszenario (Stand 10.11.2014) Teil 2. Der zweite Teil ist in Tabelle 4.1 auf Seite 42 zu finden. Der <i>ATC-Code</i> (Anatomisch-Therapeutisch-Chemisches Klassifikationssystem) [38] ist eine Klassifikation für Arzneistoffe.	43
4.3. Werte für die Typzeile der zu befüllenden Tabelle für das Programm <code>selectoptions_demo1</code> . Die entsprechenden Werte in der Diktionär-Tabelle sind in Abbildung 4.4 auf Seite 48 und Abbildung 4.5 auf Seite 49 zu sehen.	47
5.1. Bedeutung der Tabellen aus Listing 5.1 auf Seite 56.	59
5.2. Bedeutung der Einträge in der CSV-Datei, Teil 1. Der zweite Teil befindet sich in Tabelle 5.3 auf Seite 81.	80
5.3. Bedeutung der Einträge in der CSV-Datei, Teil 2. Der erste Teil befindet sich in Tabelle 5.2 auf Seite 80.	81
5.4. Bekannte personenbezogene Einträge in den <code>.lab</code> -Dateien.	83

Programm-Listings

4.1. Definition und Deklaration einer Struktur und einer internen Tabelle. . . .	50
4.2. Datenabfrage aus NBEW zur Befüllung der internen Tabelle aus Listing 4.1 auf Seite 50.	51
4.3. ABAP-List-Viewer zur Anzeige einer internen Tabelle innerhalb von SAP. Ein Beispielresultat ist in Abbildung 4.6a auf Seite 52 zu sehen.	53
5.1. Definition des Strukturtyps zur späteren Deklaration einer internen Tabelle in SAP für das Mikrobiologieszenario. Die ungekürzte Definition ist in Anhang A auf Seite 149 zu finden.	56
5.2. Abfrage zur Hauptdiagnose. Es wird für jeden Eintrag der zu befüllenden internen Tabelle die darin enthaltene Fallnummer verwendet, um die Information zur Diagnose aus NDIA zu gewinnen. Zuerst wird nach einer Krankenhaushauptdiagnose gesucht und beim Fehlen derselben die Fachabteilungshauptdiagnose. Danach werden die entsprechenden Werte auf die interne Tabelle übertragen.	60
5.3. Abfrage zur Aufnahmediagnose.	62
5.4. Suche nach den Bezeichnungen zu ICD-Codes, welche in den Abfragen in Listing 5.2 auf Seite 60 und 5.3 auf Seite 62 gewonnen wurden.	64
5.5. Gewinn der Patientenummer zum Fall über die Tabelle NFAL.	65
5.6. Gewinnung von personenbezogenen Daten.	65
5.8. Generierung einer neuen Tabelle mit Zeileneinträgen für den jeweiligen Materialverbrauch.	66
5.7. Abfrage der Aufnahme- und Entlassungsbewegung aus NBEW mit Hilfe der Fallnummer.	67
5.9. Ergänzung der Daten zur Medikation.	71
5.10. Abruf von Information zur Einrichtung.	72
5.11. Sortierung der in Listing 4.2 auf Seite 51 befüllten Tabelle.	72

5.12. Übertragung der internen Tabelle in einen Text mit einem Separator für die Werte.	73
5.13. Speicherung als CSV-Datei auf dem Präsentationsserver.	77
5.14. Speicherung als CSV-Datei auf dem Applikationsserver.	78
5.15. BASH-Skript <i>transferscript01</i> zum Kopieren der Daten aus den Verzeichnissen des Network-File-Systems.	85
6.1. Gewinnung von personenbezogenen Daten unter Vernachlässigung des Namens.	90
6.2. Generierung eines Hashwertes zur Ersetzung der Patientenummer. Zudem wird das Geburtsdatum zum Geburtsjahr geändert.	92
7.1. Erzeugung von Listen mit <i>SapZeilen</i> zu einzelnen Patienten.	98
7.2. Beispielfunktion für die Zuordnung der Werte aus der CSV-Datei zu einem Bezeichner. Der eigentliche Wert kann mit der Funktion <i>getcsvstring</i> abgerufen werden, hier am Beispiel der Zimmernummer.	99
7.3. Zuordnung eines Wertes vor der XML Generierung. Dazu wird die Funktion aus Listing 7.2 auf Seite 99 verwendet.	100
7.4. Abfrage der Verabreichungsart aus einer Zeile der CSV-Datei.	100
7.5. Generierung einer Medikamentenliste für einen Patienten.	102
7.6. Abarbeitung einer Medikamentenliste aus Listing 7.5.	103
A.1. ABAP-Programm <i>zabo_micro11</i>	149
A.2. Funktionsbaustein <i>z_abo_alvcolumnprefixadd</i>	174
B.1. ABAP-Programm <i>zabo_graphstation</i>	177
C.1. Hauptprogramm <i>labanom4.java</i>	183
C.2. <i>Zeile.java</i> zum Erstellen einer Instanz für jede Zeile.	186
D.1. <i>SapZeile.java</i>	191

Teil V.
Anhang

A. Programmcode aus Micro11

Das Programm in Listing A.1 fragt in SAP die Patientendaten aus den Datenbanken des Krankenhausinformationssystems ab und exportiert diese wahlweise als CSV-Datei. Die Selektionstexte für den Eingabebildschirm sind nicht enthalten. Zur besseren Übersicht in der ALV-Ausgabe habe ich zur Unterscheidung der unterschiedlichen Diagnosen und Bewegungen den Funktionsbaustein *z_abo_alvcolumnprefixadd* in Listing A.2 auf Seite 174 programmiert.

Listing A.1: ABAP-Programm *zabo_micro11*.

```
1 REPORT  zabo_micro11_write.
2
3
4 TYPES: BEGIN OF gty_micro ,
5     pseud TYPE c LENGTH 40, "Platz für ein Pseudonym"
6     patnr TYPE npat-patnr, "Patientennummer"
7     gschl TYPE npat-gschl, "Geschlecht"
8     gbdat TYPE npat-gbdat, "Geburtsdatum"
9     einri TYPE nfal-einri, "Einrichtung (Kürzel)"
10    *Institut, Adresse, Telefon
11     einkb TYPE tn01-einkb, "[Einrichtung] Kurzname der  

12         Einrichtung"
13     einbz TYPE tn01-einbz, "[Einrichtung] Name der  

14         Einrichtung"
15     land TYPE tn01-land, "[Einrichtung] Länderschlüssel"
16     pstlz TYPE tn01-pstlz, "[Einrichtung] Postleitzahl"
17     ort TYPE tn01-ort, "[Einrichtung] Ort"
18    *ort2 TYPE tn01-ort2,
19     stras TYPE tn01-stras, "[Einrichtung] Straße und Hausnr"
```

```
18   instnr TYPE tn01-instnr, "[Einrichtung]
      Institutskennzeichen
19   telf1 TYPE tn01-telf1, "[Einrichtung] Telefonnummer
20   falnr TYPE nfal-falnr, "Fallnummer
21 *KH Hauptdiagnose oder Fachabteilungshauptdiagnose
22   hfdialfdnr TYPE ndia-ldnr, "[Hauptdiagnose] Laufende
      Nummer Diagnose
23   hfdkat1 TYPE ndia-dkat1, "[Hauptdiagnose] Schlüsselung
      einer Diagnose
24   hfdkey1 TYPE ndia-dkey1, "[Hauptdiagnose]
      Identifikationskürzel für Diagnosekatalog
25   hfditxt TYPE ndia-ditxt, "[Hauptdiagnose] Freitext einer
      Diagnose
26   hferdat TYPE ndia-erdat, "[Hauptdiagnose] Datum, an dem
      der Satz hinzugefügt wurde
27   hfertim TYPE ndia-ertim, "[Hauptdiagnose] Uhrzeit, zu
      der der Satz hinzugefügt wurde
28   hfdtext1 TYPE nkdi-dtext1, "[Hauptdiagnose] Text der
      Diagnose
29 *Aufnahmediagnose
30   adm_dialfdnr TYPE ndia-ldnr, "[Aufnahmediagnose]
      Laufende Nummer Diagnose
31   adm_dkat1 TYPE ndia-dkat1, "[Aufnahmediagnose]
      Schlüsselung einer Diagnose
32   adm_dkey1 TYPE ndia-dkey1, "[Aufnahmediagnose]
      Identifikationskürzel für Diagnosekatalog
33   adm_ditxt TYPE ndia-ditxt, "[Aufnahmediagnose] Freitext
      einer Diagnose
34   adm_erdat TYPE ndia-erdat, "[Aufnahmediagnose] Datum, an
      dem der Satz hinzugefügt wurde
35   adm_ertim TYPE ndia-ertim, "[Aufnahmediagnose] Uhrzeit,
      zu der der Satz hinzugefügt wurde
```

36 adm_dtext1 TYPE nkdi-dtext1, "[Aufnahmediagnose] Text
der Diagnose

37 *Entlassungsdiagnose

38 dis_dialfdnr TYPE ndia-ldfnr, "[Entlassungsdiagnose]
Laufende Nummer Diagnose

39 dis_dkat1 TYPE ndia-dkat1, "[Entlassungsdiagnose]
Schlüsselung einer Diagnose

40 dis_dkey1 TYPE ndia-dkey1, "[Entlassungsdiagnose]
Identifikationskürzel für Diagnosekatalog

41 dis_ditxt TYPE ndia-ditxt, "[Entlassungsdiagnose]
Freitext einer Diagnose

42 dis_erdat TYPE ndia-erdat, "[Entlassungsdiagnose] Datum,
an dem der Satz hinzugefügt wurde

43 dis_ertim TYPE ndia-ertim, "[Entlassungsdiagnose]
Uhrzeit, zu der der Satz hinzugefügt wurde

44 dis_dtext1 TYPE nkdi-dtext1, "[Entlassungsdiagnose] Text
der Diagnose

45 *Aktuelle Bewegung

46 lfdnr TYPE nbew-lfdnr, "Laufende Nummer einer Bewegung

47 bwidt TYPE nbew-bwidt, "Datum der Bewegung

48 bwizt TYPE nbew-bwizt, "Uhrzeit der Bewegung

49 bwedt TYPE nbew-bwedt, "Bewegungsendedatum

50 bwezt TYPE nbew-bwezt, "Bewegungsendezeit

51 orgpf TYPE nbew-orgpf, "Stationäre OrgEinheit, die einem
Fall zugewiesen wird

52 zimmr TYPE nbew-zimmr, "BauId eines Zimmers

53 bett TYPE nbew-bett, "BauId eines Bettenstellplatzes

54 *Admission Bewegung

55 admlfdnr TYPE nbew-lfdnr, "[Aufnahmebewegung] Laufende
Nummer einer Bewegung

56 admbwidt TYPE nbew-bwidt, "[Aufnahmebewegung] Datum der
Bewegung

```
57   admbwizt TYPE nbew-bwizt, "[Aufnahmebewegung] Uhrzeit
      der Bewegung
58   admbwgr1 TYPE nbew-bwgr1, "[Aufnahmebewegung]
      Bewegungsgrund - 1. und 2. Stelle
59 *Entlassung
60   dislfdnr TYPE nbew-lfdnr, "[Entlassungsbewegung]
      Laufende Nummer einer Bewegung
61   disbwidt TYPE nbew-bwidt, "[Entlassungsbewegung] Datum
      der Bewegung
62   disbwizt TYPE nbew-bwizt, "[Entlassungsbewegung] Uhrzeit
      der Bewegung
63 *Materialverbrauch
64   matnr TYPE nmatv-matnr, "Materialnummer in KH für
      Materialverbrauch
65   maktx TYPE ish_mm_consump_txt-maktx, "Materialkurztext
66   menge TYPE nmatv-menge, "(Angeforderte) Menge des
      Materials
67   meus TYPE nmatv-meus, "Mengeneinheit
68   consdt TYPE nmatv-consdt, "Datum der Verabreichung eines
      Materials
69   constm TYPE nmatv-constm, "Uhrzeit der Verabreichung
      eines Materials
70   wgbez TYPE t023t-wgbez, "Warengruppenbezeichnung
71   verabrart TYPE c LENGTH 2, "Platz für die
      Verabreichungsart, po oder iv.
72   matkl TYPE mara-matkl, "Warengruppe
73   ish_atc TYPE mara-ish_atc, "ATC-Code
74   ish_agent1 TYPE mara-ish_agent1, "Wirkstoff 1 (
      Hauptwirkstoff)
75   ish_ag1quant TYPE mara-ish_ag1quant, "[Wirkstoff 1]
      Menge eines Wirkstoffs in einem Material
76   ish_ag1unit TYPE mara-ish_ag1unit, "[Wirkstoff 1]
      Mengeneinheit eines Wirkstoffs in einem Material
```

```

77 ish_agent2 TYPE mara-ish_agent2, "Wirkstoff 2"
78 ish_ag2quant TYPE mara-ish_ag2quant, "[Wirkstoff 2]"
      Menge eines Wirkstoffs in einem Material
79 ish_ag2unit TYPE mara-ish_ag2unit, "[Wirkstoff 2]"
      Mengeneinheit eines Wirkstoffs in einem Material
80 ish_agent3 TYPE mara-ish_agent3, "Wirkstoff 3"
81 ish_ag3quant TYPE mara-ish_ag3quant, "[Wirkstoff 3]"
      Menge eines Wirkstoffs in einem Material
82 ish_ag3unit TYPE mara-ish_ag3unit, "[Wirkstoff 3]"
      Mengeneinheit eines Wirkstoffs in einem Material
83 nname TYPE npat-nname, "Name der Patienten"
84 vname TYPE npat-vname, "Vorname der Patienten"
85 lnrlm TYPE nmatv-lnrlm, "Laufende Nummer der"
      Materialanforderung
86 END OF gty_micro.
87
88 DATA: gt_micro TYPE TABLE OF gty_micro,
89       gt_final TYPE TABLE OF gty_micro,
90       gs_micro TYPE gty_micro,
91       gs_pseud TYPE gty_micro.
92 DATA gs_nmatv TYPE nmatv.
93 DATA gs_rncoms TYPE rnmcons.
94 DATA gs_ish_mm_consumpt_txt TYPE ish_mm_consump_txt.
95 DATA flag TYPE c VALUE space.
96 DATA: go_alv_grid TYPE REF TO cl_salv_table,
97       columns TYPE REF TO cl_salv_columns_list,
98       column TYPE REF TO cl_salv_column_list,
99       lv_scrtext_s TYPE scrtext_s,
100      lv_scrtext_m TYPE scrtext_m,
101      lv_scrtext_l TYPE scrtext_l,
102      lv_lvc_tip TYPE lvc_tip.
103 DATA: t_ndia LIKE ndia.
104 DATA: t_rednpat LIKE npat.

```

A. Programmcode aus Micro11

```
105 DATA: t_fall LIKE nfal. "s.o.
106 DATA: gt_nmatv LIKE STANDARD TABLE OF gs_nmatv.
107 DATA: t_znmat_nmmc1 LIKE znmat_nmmc1.
108 DATA: t_mara LIKE mara,
109         t_tn01 LIKE tn01.
110 DATA: t_admbew LIKE nbew,
111         t_disbew TYPE nbew,
112         t_nkdi LIKE nkdi.
113 DATA: l_sig TYPE string,
114         l_hash TYPE hash160.
115 DATA: it_csvdata TYPE truxs_t_text_data,
116         line_csvdata LIKE LINE OF it_csvdata,
117         filename TYPE string,
118         systemname TYPE c LENGTH 3.
119 DATA: l_wgbez TYPE wgbez.
120 DATA: lr_functions TYPE REF TO cl_salv_functions_list.
121 DATA: go_layout TYPE REF TO cl_salv_layout,
122         gs_key TYPE salv_s_layout_key.
123
124 PARAMETERS so_einri LIKE nbew-einri DEFAULT 'UKSH' NO-
        DISPLAY.
125 SELECT-OPTIONS so_orgpf FOR gs_micro-orgpf OBLIGATORY
        MEMORY ID station.
126 PARAMETERS: pa_save. "DEFAULT 'N'.
127 PARAMETERS: pa_pseu. "DEFAULT 'N'.
128
129 START-OF-SELECTION.
130
131 SELECT * FROM nbew INTO CORRESPONDING FIELDS OF TABLE
        gt_micro
132 WHERE einri = so_einri
133 AND ( orgfa = 'K1-ABTLG' OR
134        orgfa = 'K2-ABTLG' OR
```

```

135         orgfa = 'K3-ABTLG' OR
136         orgfa = 'K5-ABTLG' )
137 AND orgpf IN so_orgpf
138 AND bwedt GE sy-datum
139 AND bwidt LE sy-datum
140 AND ( bewty = '1' OR bewty = '3' OR bewty = '7' )
141 AND storn = space
142 AND planb = space.
143 SORT gt_micro DESCENDING BY falnr bwedt bwezt.
144 DELETE ADJACENT DUPLICATES FROM gt_micro COMPARING falnr.
145
146 *Diagnosen
147 LOOP AT gt_micro INTO gs_micro.
148 *Krankenhaustauptdiagnose
149     SELECT SINGLE * FROM ndia INTO t_ndia
150     WHERE einri = so_einri
151     AND falnr = gs_micro-falnr
152     AND storn = space
153     AND khdia = 'X'.
154     IF sy-subrc = 0.
155         MOVE t_ndia-lfdnr TO gs_micro-hfdialfdnr.
156         MOVE t_ndia-dkat1 TO gs_micro-hfdkat1.
157         MOVE t_ndia-dkey1 TO gs_micro-hfdkey1.
158         MOVE t_ndia-ditxt TO gs_micro-hfditxt.
159         MOVE t_ndia-erdat TO gs_micro-hferdat.
160         MOVE t_ndia-ertim TO gs_micro-hfertim.
161         MODIFY gt_micro FROM gs_micro.
162     ELSE .
163 *Fachabteilungshauptdiagnose
164     SELECT SINGLE * FROM ndia INTO t_ndia
165     WHERE einri = so_einri
166     AND falnr = gs_micro-falnr
167     AND storn = space

```

```
168         AND fhdia = 'X'.
169         IF sy-subrc = 0.
170             MOVE t_ndia-lfdnr TO gs_micro-hfdialfdnr.
171             MOVE t_ndia-dkat1 TO gs_micro-hfdkat1.
172             MOVE t_ndia-dkey1 TO gs_micro-hfdkey1.
173             MOVE t_ndia-ditxt TO gs_micro-hfditxt.
174             MOVE t_ndia-erdat TO gs_micro-hferdat.
175             MOVE t_ndia-ertim TO gs_micro-hfertim.
176             MODIFY gt_micro FROM gs_micro.
177         ENDIF.
178     ENDIF.
179     *Aufnahmediagnose
180     SELECT SINGLE * FROM ndia INTO t_ndia
181     WHERE einri = so_einri
182     AND falnr = gs_micro-falnr
183     AND storn = space
184     AND afdia = 'X'.
185     IF sy-subrc = 0.
186         MOVE t_ndia-lfdnr TO gs_micro-adm_dialfdnr.
187         MOVE t_ndia-dkat1 TO gs_micro-adm_dkat1.
188         MOVE t_ndia-dkey1 TO gs_micro-adm_dkey1.
189         MOVE t_ndia-ditxt TO gs_micro-adm_ditxt.
190         MOVE t_ndia-erdat TO gs_micro-adm_erdat.
191         MOVE t_ndia-ertim TO gs_micro-adm_ertim.
192         MODIFY gt_micro FROM gs_micro.
193     ENDIF.
194     *Entlassungsdiagnose
195     SELECT SINGLE * FROM ndia INTO t_ndia
196     WHERE einri = so_einri
197     AND falnr = gs_micro-falnr
198     AND storn = space
199     AND endia = 'X'.
200     IF sy-subrc = 0.
```

```

201     MOVE t_ndia-lfdnr TO gs_micro-dis_dialfdnr.
202     MOVE t_ndia-dkat1 TO gs_micro-dis_dkat1.
203     MOVE t_ndia-dkey1 TO gs_micro-dis_dkey1.
204     MOVE t_ndia-ditxt TO gs_micro-dis_ditxt.
205     MOVE t_ndia-erdat TO gs_micro-dis_erdat.
206     MOVE t_ndia-ertim TO gs_micro-dis_ertim.
207     MODIFY gt_micro FROM gs_micro.
208     ENDIF.
209     *Stammdaten zum Patient
210     SELECT SINGLE * FROM nfal INTO t_fall
211     WHERE falnr = gs_micro-falnr.
212     IF sy-subrc = 0.
213         MOVE t_fall-patnr TO gs_micro-patnr.
214         MODIFY gt_micro FROM gs_micro.
215     ENDIF.
216     SELECT SINGLE * FROM npat INTO t_rednpat
217     WHERE patnr = gs_micro-patnr.
218     IF sy-subrc = 0.
219         MOVE t_rednpat-gschl TO gs_micro-gschl.
220         MOVE t_rednpat-gbdat TO gs_micro-gbdat.
221         IF NOT ( pa_pseu = 'J' OR pa_pseu = 'Y' ).
222             MOVE t_rednpat-nname TO gs_micro-nname.
223             MOVE t_rednpat-vname TO gs_micro-vname.
224         ENDIF.
225         MODIFY gt_micro FROM gs_micro.
226     ENDIF.
227 ENDLOOP.
228
229 CLEAR gs_micro.
230 *Admission Bewegung, Dischargebewegung
231 LOOP AT gt_micro INTO gs_micro.
232     SELECT * FROM nbew INTO t_admbew
233     WHERE falnr = gs_micro-falnr

```

A. Programmcode aus Micro11

```
234     AND storn = space
235     AND bewty = '1'
236     AND storn = space
237     AND plane = space.
238     ENDSELECT.
239 IF sy-subrc = 0.
240     MOVE t_admbew-lfdnr TO gs_micro-admlfdnr.
241     MOVE t_admbew-bwidt TO gs_micro-admbwidt.
242     MOVE t_admbew-bwizt TO gs_micro-admbwizt.
243     MOVE t_admbew-bwgr1 TO gs_micro-admbwgr1.
244     MODIFY gt_micro FROM gs_micro.
245 ENDIF.
246 SELECT * FROM nbew INTO t_disbew
247     WHERE falnr = gs_micro-falnr
248     AND storn = space
249     AND bewty = '2'
250     AND storn = space
251     AND plane = space.
252 ENDSELECT.
253 IF sy-subrc = 0.
254     MOVE t_disbew-lfdnr TO gs_micro-dislfdnr.
255     MOVE t_disbew-bwidt TO gs_micro-disbwidt.
256     MOVE t_disbew-bwizt TO gs_micro-disbwizt.
257     MODIFY gt_micro FROM gs_micro.
258 ENDIF.
259 ENDLOOP.
260
261 *Auslesen des Texts zum ICD-Code
262 LOOP AT gt_micro INTO gs_micro.
263     SELECT dtext1 FROM nkdi INTO gs_micro-hfdtext1
264     WHERE spras = 'DE'
265     AND dkat = gs_micro-hfdkat1
266     AND dkey = gs_micro-hfdkey1 .
```

```

267  ENDSELECT.
268  IF sy-subrc = 0.
269    MODIFY gt_micro FROM gs_micro.
270  ENDIF.
271  SELECT dtext1 FROM nkdi INTO gs_micro-adm_dtext1
272    WHERE spras = 'DE'
273    AND dkat = gs_micro-adm_dkat1
274    AND dkey = gs_micro-adm_dkey1 .
275  ENDSELECT.
276  IF sy-subrc = 0.
277    MODIFY gt_micro FROM gs_micro.
278  ENDIF.
279  SELECT dtext1 FROM nkdi INTO gs_micro-dis_dtext1
280    WHERE spras = 'DE'
281    AND dkat = gs_micro-dis_dkat1
282    AND dkey = gs_micro-dis_dkey1 .
283  ENDSELECT.
284  IF sy-subrc = 0.
285    MODIFY gt_micro FROM gs_micro.
286  ENDIF.
287  ENDLOOP.
288
289  *Materialverbrauch
290  LOOP AT gt_micro INTO gs_micro.
291    CLEAR gt_nmatv.
292    SELECT * FROM nmatv INTO CORRESPONDING FIELDS OF TABLE
293      gt_nmatv
294      WHERE einri = so_einri
295      AND falnr = gs_micro-falnr
296      AND menge NE space
297      AND storn = space.
298    IF gt_nmatv IS INITIAL.
299      APPEND gs_micro TO gt_final. "Kein Verbrauch

```

A. Programmcode aus Micro11

```
299     ENDIF .
300     IF sy-subrc = 0.
301         LOOP AT gt_nmatv INTO gs_nmatv.
302             MOVE-CORRESPONDING gs_nmatv TO gs_rncoms.
303             CALL FUNCTION 'ISH_MAT_CONSUMP_INFO'
304                 EXPORTING
305                     ss_rnmcons = gs_rncoms
306                     ss_einri   = so_einri
307             IMPORTING
308                 ss_cons_txt = gs_ish_mm_consumpt_txt.
309             gs_micro-maktx = gs_ish_mm_consumpt_txt-maktx.
310             MOVE gs_nmatv-matnr TO gs_micro-matnr.
311             MOVE gs_nmatv-menge TO gs_micro-menge.
312             MOVE gs_nmatv-meins TO gs_micro-meins.
313             MOVE gs_nmatv-consdt TO gs_micro-consdt.
314             MOVE gs_nmatv-constm TO gs_micro-constm.
315             MOVE gs_nmatv-lnrlm TO gs_micro-lnrlm.
316             APPEND gs_micro TO gt_final.
317         ENDLOOP.
318     ENDIF .
319 ENDLOOP.
320
321 CLEAR gs_micro.
322 *Substanzgruppe
323 *Verabreichungsart
324 LOOP AT gt_final INTO gs_micro.
325     SELECT SINGLE * FROM mara INTO t_mara
326         WHERE matnr = gs_micro-matnr.
327     IF sy-subrc = 0.
328         MOVE-CORRESPONDING t_mara TO gs_micro.
329         SELECT SINGLE wgbez INTO l_wgbez FROM t023t
330             WHERE matkl = t_mara-matkl.
331         IF sy-subrc = 0.
```

```

332     gs_micro-wgbez  = l_wgbez.
333     IF l_wgbez CS 'iv'.
334         gs_micro-verabrart = 'iv'.
335     ELSE.
336         gs_micro-verabrart = 'po'.
337     ENDIF.
338 ENDIF.
339     MODIFY gt_final FROM gs_micro.
340 ENDIF.
341 ENDLOOP.
342
343 CLEAR gs_micro.
344 *Einrichtung
345 LOOP AT gt_final INTO gs_micro.
346     SELECT SINGLE * FROM tn01 INTO t_tn01
347         WHERE einri = gs_micro-einri.
348     IF sy-subrc = 0.
349         MOVE-CORRESPONDING t_tn01 TO gs_micro.
350         MODIFY gt_final FROM gs_micro.
351     ENDIF.
352 ENDLOOP.
353
354 CLEAR gs_micro.
355 * Ausgabetabelle wird sortiert.
356 SORT gt_final. " by patnr ASCENDING as text.
357
358 * Pseudonymerstellung
359 * Für die PatNr wird ein Pseudonym erstellt.
360 IF pa_pseu = 'J' OR pa_pseu = 'Y'.
361     LOOP AT gt_final INTO gs_micro.
362         MOVE-CORRESPONDING gs_micro TO gs_pseud.
363         l_sig = gs_pseud-patnr.
364         CALL FUNCTION 'CALCULATE_HASH_FOR_CHAR'

```

```
365     EXPORTING
366         alg      = 'MD5'
367         data     = l_sig
368         length  = 0
369     IMPORTING
370         hash     = l_hash.
371     gs_pseud-pseud = l_hash.
372     gs_pseud-patnr = space.
373 *Geburtsdatum zu Geburtsjahr
374     gs_pseud-gbdat = gs_pseud-gbdat(4).
375     IF sy-subrc = 0.
376         MODIFY gt_final FROM gs_pseud.
377     ENDIF.
378 ENDLOOP.
379 ENDIF.
380 * Ende Pseudonym
381
382 * Speichern als CSV
383 IF pa_save = 'J' OR pa_save = 'Y'.
384 * Vorbereitung für CSV-Speicherung
385     CALL FUNCTION 'SAP_CONVERT_TO_TEX_FORMAT'
386     EXPORTING
387         i_field_seperator      = '|'
388     TABLES
389         i_tab_sap_data         = gt_final
390     CHANGING
391         i_tab_converted_data   = it_csvdata.
392     IF sy-subrc <> 0.
393 * MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
394 *         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
395     ENDIF.
396
397 * Name des aktuellen Systems
```

```

398 * Case sensitive!
399 IF sy-sysid = 'C21'.
400     systemname = 'c21'.
401 ELSEIF sy-sysid = 'T21'.
402     systemname = 't21'.
403 ENDIF.
404 * Ende Vorbereitung
405
406 **Speichern auf dem Applikationsserver
407 *   CONCATENATE '/sapuser/' systemname 'adm/SAP_COM/
      ishmed/eureca/micro/microSAP_' sy-datum sy-zeit '.csv'
      INTO filename.
408 *   OPEN DATASET filename IN LEGACY TEXT MODE FOR OUTPUT.
409 *   LOOP AT it_csvdata into line_csvdata.
410 *       TRANSFER line_csvdata TO filename.
411 *       "CLEAR it_csvdata.
412 *   ENDLOOP.
413 *   CLOSE DATASET filename.
414
415 **Speichern auf dem Präsentationsserver
416   CONCATENATE 'microSAP_' sy-datum sy-zeit '.csv' INTO
      filename.
417   CALL FUNCTION 'GUI_DOWNLOAD'
418     EXPORTING
419       filename      = filename
420     TABLES
421       data_tab      = it_csvdata.
422   IF sy-subrc <> 0.
423     MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
424       WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
425   ENDIF.
426 ENDIF.
427 * Ende Speichern

```

```
428
429 * Anzeige im ALV
430 cl_salv_table⇒factory(
431     IMPORTING
432         r_salv_table    = go_alv_grid
433     CHANGING
434         t_table        = gt_final
435     ).
436     columns = go_alv_grid→get_columns( ).
437
438 *Namen, Tooltips, Anzeige, Position
439 *Typen ohne Namensgebung
440 column ?= columns→get_column( columnname = 'PSEUD').
441 lv_scrtext_s = 'Pseud.'.
442 lv_scrtext_m = 'Pseudonym'.
443 lv_scrtext_l = 'Pseudonym'.
444 lv_lvc_tip = lv_scrtext_l.
445 column→set_short_text( value = lv_scrtext_s ).
446 column→set_medium_text( value = lv_scrtext_m ).
447 column→set_long_text( value = lv_scrtext_l ).
448 column→set_tooltip( value = lv_lvc_tip ).
449 IF pa_pseu NE 'J' AND pa_pseu NE 'Y'.
450     column→set_technical( ).
451 ENDIF.
452 IF pa_pseu = 'J' OR pa_pseu = 'Y'.
453     column ?= columns→get_column( columnname = 'PATNR').
454     column→set_technical( ).
455     column ?= columns→get_column( columnname = 'NNAME').
456     column→set_technical( ).
457     column ?= columns→get_column( columnname = 'VNAME').
458     column→set_technical( ).
459 ENDIF.
460 column ?= columns→get_column( columnname = 'VERABRART').
```

```

461 lv_scrtext_s = 'Verabr.'.
462 lv_scrtext_m = 'Verabr.art'.
463 lv_scrtext_l = 'Verabreichungsart'.
464 lv_lvc_tip = lv_scrtext_l.
465 column→set_short_text( value = lv_scrtext_s ).
466 column→set_medium_text( value = lv_scrtext_m ).
467 column→set_long_text( value = lv_scrtext_l ).
468 column→set_tooltip( value = lv_lvc_tip ).
469
470 *Position
471 columns→set_column_position(
472     columnname = 'NNAME'
473     position = 3
474 ).
475 columns→set_column_position(
476     columnname = 'VNAME'
477     position = 4
478 ).
479
480 *Für eine bessere Lesbarkeit im ALV:
481 *Prefix Hauptdiagnose
482 column ?= columns→get_column( columnname = 'HFDIALFDNR').
483 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
484     EXPORTING
485         shortname      = 'Hd.'
486         mediumname     = 'Hd.'
487         longname       = 'Hauptdiagnose_Kh./F.abt.'
488         column2change = column.
489 column ?= columns→get_column( columnname = 'HFDKAT1').
490 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
491     EXPORTING
492         shortname      = 'Hd.'
493         mediumname     = 'Hd.'

```

```
494     longname      = 'Hauptdiagnose_Kh./F.abt.'
```

```
495     column2change = column.
```

```
496 column ?= columns→get_column( columnname = 'HFDKEY1').
```

```
497 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
498     EXPORTING
```

```
499     shortname      = 'Hd.'
```

```
500     mediumname     = 'Hd.'
```

```
501     longname       = 'Hauptdiag.Kh./F.abt.'
```

```
502     column2change = column.
```

```
503 column ?= columns→get_column( columnname = 'HFDITXT').
```

```
504 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
505     EXPORTING
```

```
506     shortname      = 'Hd.'
```

```
507     mediumname     = 'Hd.'
```

```
508     longname       = 'Hauptdiag.Kh./F.abt.'
```

```
509     column2change = column.
```

```
510 column ?= columns→get_column( columnname = 'HFERDAT').
```

```
511 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
512     EXPORTING
```

```
513     shortname      = 'Hd.'
```

```
514     mediumname     = 'Hd.'
```

```
515     longname       = 'Hauptdiagnose_Kh./F.abt.'
```

```
516     column2change = column.
```

```
517 column ?= columns→get_column( columnname = 'HFERTIM').
```

```
518 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
519     EXPORTING
```

```
520     shortname      = 'Hd.'
```

```
521     mediumname     = 'Hd.'
```

```
522     longname       = 'Hauptdiagnose_Kh./F.abt.'
```

```
523     column2change = column.
```

```
524 column ?= columns→get_column( columnname = 'HFDTEXT1').
```

```
525 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
526     EXPORTING
```

```

527     shortname      = 'Hd.'
528     mediumname     = 'Hd.'
529     longname       = 'Hauptdiagnose_Kh./F.abt.'
530     column2change  = column.
531
532     *Prefix Aufnahmediagnose
533     column ?= columns→get_column( columnname = 'ADM_DIALFDNR')
534     .
534     CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
535     EXPORTING
536         shortname      = 'Aufn.'
537         mediumname     = 'Aufn.'
538         longname       = 'Aufnahmediagnose.'
539         column2change  = column.
540     column ?= columns→get_column( columnname = 'ADM_DKAT1').
541     CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
542     EXPORTING
543         shortname      = 'Aufn.'
544         mediumname     = 'Aufn.'
545         longname       = 'Aufnahmediagnose.'
546         column2change  = column.
547     column ?= columns→get_column( columnname = 'ADM_DKEY1').
548     CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
549     EXPORTING
550         shortname      = 'Aufn.'
551         mediumname     = 'Aufn.'
552         longname       = 'Aufnahmediagnose.'
553         column2change  = column.
554     column ?= columns→get_column( columnname = 'ADM_DITXT').
555     CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
556     EXPORTING
557         shortname      = 'Aufn.'
558         mediumname     = 'Aufn.'

```

```
559     longname      = 'Aufnahmediagnose.'
```

```
560     column2change = column.
```

```
561 column ?= columns→get_column( columnname = 'ADM_ERDAT').
```

```
562 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
563     EXPORTING
```

```
564     shortname      = 'Aufn.'
```

```
565     mediumname     = 'Aufn.'
```

```
566     longname       = 'Aufnahmediagnose.'
```

```
567     column2change = column.
```

```
568 column ?= columns→get_column( columnname = 'ADM_ERTIM').
```

```
569 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
570     EXPORTING
```

```
571     shortname      = 'Aufn.'
```

```
572     mediumname     = 'Aufn.'
```

```
573     longname       = 'Aufnahmediagnose.'
```

```
574     column2change = column.
```

```
575 column ?= columns→get_column( columnname = 'ADM_DTEXT1').
```

```
576 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
577     EXPORTING
```

```
578     shortname      = 'Aufn.'
```

```
579     mediumname     = 'Aufn.'
```

```
580     longname       = 'Aufnahmediagnose.'
```

```
581     column2change = column.
```

```
582
```

```
583 *Prefix Entlassungsdiagnose
```

```
584 column ?= columns→get_column( columnname = 'DIS_DIALFDNR')
```

```
585 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
586     EXPORTING
```

```
587     shortname      = 'Entl.'
```

```
588     mediumname     = 'Entl.'
```

```
589     longname       = 'Entlassungsdiagnose.'
```

```
590     column2change = column.
```

```
591 column ?= columns→get_column( columnname = 'DIS_DKAT1').
592 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
593     EXPORTING
594         shortname      = 'Entl.'
595         mediumname     = 'Entl.'
596         longname       = 'Entlassungsdiagnose.'
597         column2change  = column.
598 column ?= columns→get_column( columnname = 'DIS_DKEY1').
599 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
600     EXPORTING
601         shortname      = 'Entl.'
602         mediumname     = 'Entl.'
603         longname       = 'Entlassungsdiagnose.'
604         column2change  = column.
605 column ?= columns→get_column( columnname = 'DIS_DITXT').
606 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
607     EXPORTING
608         shortname      = 'Entl.'
609         mediumname     = 'Entl.'
610         longname       = 'Entlassungsdiagnose.'
611         column2change  = column.
612 column ?= columns→get_column( columnname = 'DIS_ERDAT').
613 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
614     EXPORTING
615         shortname      = 'Entl.'
616         mediumname     = 'Entl.'
617         longname       = 'Entlassungsdiagnose.'
618         column2change  = column.
619 column ?= columns→get_column( columnname = 'DIS_ERTIM').
620 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
621     EXPORTING
622         shortname      = 'Entl.'
623         mediumname     = 'Entl.'
```

```
624     longname      = 'Entlassungsdiagnose.'
```

```
625     column2change = column.
```

```
626 column ?= columns→get_column( columnname = 'DIS_DTEXT1').
```

```
627 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
628     EXPORTING
```

```
629     shortname      = 'Entl.'
```

```
630     mediumname     = 'Entl.'
```

```
631     longname       = 'Entlassungsdiagnose.'
```

```
632     column2change = column.
```

```
633
```

```
634 *Prefix Aufnahmebewegungen
```

```
635 column ?= columns→get_column( columnname = 'ADMLFDNR').
```

```
636 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
637     EXPORTING
```

```
638     shortname      = 'Aufn.'
```

```
639     mediumname     = 'Aufn.'
```

```
640     longname       = 'Aufnahmebewegung.'
```

```
641     column2change = column.
```

```
642 column ?= columns→get_column( columnname = 'ADMBWIDT').
```

```
643 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
644     EXPORTING
```

```
645     shortname      = 'Aufn.'
```

```
646     mediumname     = 'Aufn.'
```

```
647     longname       = 'Aufnahmebewegung.'
```

```
648     column2change = column.
```

```
649 column ?= columns→get_column( columnname = 'ADMBWIZT').
```

```
650 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
```

```
651     EXPORTING
```

```
652     shortname      = 'Aufn.'
```

```
653     mediumname     = 'Aufn.'
```

```
654     longname       = 'Aufnahmebewegung.'
```

```
655     column2change = column.
```

```
656 column ?= columns→get_column( columnname = 'ADMBWGR1').
```

```

657 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
658     EXPORTING
659         shortname      = 'Aufn.'
660         mediumname     = 'Aufn.'
661         longname       = 'Aufnahmebewegung.'
662         column2change  = column.
663
664 *Prefix Entlassungsbewegungen
665 column ?= columns→get_column( columnname = 'DISLFDNR').
666 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
667     EXPORTING
668         shortname      = 'Entl.'
669         mediumname     = 'Entl.'
670         longname       = 'Entlassungsbewegung.'
671         column2change  = column.
672 column ?= columns→get_column( columnname = 'DISBWIDT').
673 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
674     EXPORTING
675         shortname      = 'Entl.'
676         mediumname     = 'Entl.'
677         longname       = 'Entlassungsbewegung.'
678         column2change  = column.
679 column ?= columns→get_column( columnname = 'DISBWIZT').
680 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
681     EXPORTING
682         shortname      = 'Entl.'
683         mediumname     = 'Entl.'
684         longname       = 'Entlassungsbewegung.'
685         column2change  = column.
686
687 *Prefix für aktuelle Bewegungen
688 column ?= columns→get_column( columnname = 'LFDNR').
689 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'

```

```
690     EXPORTING
691         shortname      = 'Bew.'
692         mediumname     = 'Bew.'
693         longname       = 'Aktuelle_Bewegung.'
694         column2change = column.
695 column ?= columns→get_column( columnname = 'BWIDT').
696 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
697     EXPORTING
698         shortname      = 'Bew.'
699         mediumname     = 'Bew.'
700         longname       = 'Aktuelle_Bewegung.'
701         column2change = column.
702 column ?= columns→get_column( columnname = 'BWIZT').
703 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
704     EXPORTING
705         shortname      = 'Bew.'
706         mediumname     = 'Bew.'
707         longname       = 'Aktuelle_Bewegung.'
708         column2change = column.
709 column ?= columns→get_column( columnname = 'BWEDT').
710 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
711     EXPORTING
712         shortname      = 'Bew.'
713         mediumname     = 'Bew.'
714         longname       = 'Aktuelle_Bewegung.'
715         column2change = column.
716 column ?= columns→get_column( columnname = 'BWEZT').
717 CALL FUNCTION 'Z_ABO_ALVCOLUMNPREFIXADD'
718     EXPORTING
719         shortname      = 'Bew.'
720         mediumname     = 'Bew.'
721         longname       = 'Aktuelle_Bewegung.'
722         column2change = column.
```

```
723
724 *Layouts
725 go_layout = go_alv_grid→get_layout( ).
726 gs_key-report = sy-cprog.
727 go_layout→set_key( value = gs_key ).
728 go_layout→set_save_restriction(
729     value = if_salv_c_layout⇒restrict_user_dependant
730 ).
731 go_layout→set_default( value = 'X' ).
732
733 columns→set_optimize( abap_true ).
734
735 *Funktionen der ALV-Ausgabe
736 lr_functions = go_alv_grid→get_functions( ).
737 lr_functions→set_all( ).
738 lr_functions→set_group_aggregation( value =
739     if_salv_c_bool_sap⇒false ).
739 lr_functions→set_print( value = if_salv_c_bool_sap⇒true ).
740 lr_functions→set_print_preview( value = if_salv_c_bool_sap
741     ⇒true ).
741 lr_functions→set_group_export( value = if_salv_c_bool_sap⇒
742     true ).
742 lr_functions→set_export_localfile( value =
743     if_salv_c_bool_sap⇒false ).
743 lr_functions→set_export_mail( value = if_salv_c_bool_sap⇒
744     false ).
744 lr_functions→set_export_wordprocessor( value =
745     if_salv_c_bool_sap⇒false ).
745 lr_functions→set_view_excel( value = if_salv_c_bool_sap⇒
746     false ).
746 lr_functions→set_ABC_Analysis( value = if_salv_c_bool_sap⇒
747     false ).
```

```
747 lr_functions→set_export_send( value = if_salv_c_bool_sap⇒
      false ).
748 lr_functions→set_graphics( value = if_salv_c_bool_sap⇒
      false ).
749
750 DATA: lo_display TYPE REF TO cl_salv_display_settings.
751 lo_display = go_alv_grid→get_display_settings( ).
752 lo_display→set_stripped_pattern( abap_true ).
753 go_alv_grid→display( ).
```

Listing A.2: Funktionsbaustein *z_abo_alvcolumnprefixadd*.

```
1 FUNCTION z_abo_alvcolumnprefixadd.
2 *"-----
3 *"Lokale Schnittstelle:
4 *"  IMPORTING
5 *"      REFERENCE(SHORTNAME) TYPE  SCRTEXT_S OPTIONAL
6 *"      REFERENCE(MEDIUMNAME) TYPE SCRTEXT_M OPTIONAL
7 *"      REFERENCE(LONGNAME) TYPE  SCRTEXT_L OPTIONAL
8 *"      REFERENCE(COLUMN2CHANGE) TYPE REF TO
      CL_SALV_COLUMN_LIST
9 *"-----
10 DATA:
11 lv_scrtext_s TYPE scrtext_s,
12 lv_scrtext_m TYPE scrtext_m,
13 lv_scrtext_l TYPE scrtext_l,
14 lv_lvc_tip TYPE lvc_tip.
15
16 lv_scrtext_s = column2change→get_short_text( ).
17 lv_scrtext_m = column2change→get_medium_text( ).
18 lv_scrtext_l = column2change→get_long_text( ).
19
20 CONCATENATE shortname lv_scrtext_s INTO lv_scrtext_s.
21 CONCATENATE mediumname lv_scrtext_m INTO lv_scrtext_m.
22 CONCATENATE longname lv_scrtext_l INTO lv_scrtext_l.
```

```
23
24     column2change→set_short_text( value = lv_scrtext_s ).
25     column2change→set_medium_text( value = lv_scrtext_m ).
26     column2change→set_long_text( value = lv_scrtext_l ).
27     column2change→set_tooltip( value = lv_lvc_tip ).
28
29 ENDFUNCTION.
```


B. Programmcode aus Graphstation

Listing B.1: ABAP-Programm *zabo_graphstation*.

```
1 REPORT    zabo_graphstation3_write .
2
3
4 TYPES: BEGIN OF gty_stationswerte ,
5     patnr TYPE npat-patnr , "Patientennummer
6     gschl TYPE npat-gschl , "Geschlecht
7     gbdat TYPE npat-gbdat , "Geburtsdatum
8     einri TYPE nfal-einri , "Einrichtung (Kürzel)
9     falnr TYPE nfal-falnr , "Fallnummer
10    orgpf TYPE nbew-orgpf , "Stationäre OrgEinheit, die einem
        Fall zugewiesen wird
11    zimmr TYPE nbew-zimmr , "BauId eines Zimmers
12    bett TYPE nbew-bett , "BauId eines Bettenstellplatzes
13 END OF gty_stationswerte ,
14
15 BEGIN OF gty_einzelstationen ,
16     orgpf TYPE nbew-orgpf ,
17     geschlcount_1m TYPE i ,
18     geschlcount_2w TYPE i ,
19     geschlcount_3u TYPE i ,
20 END OF gty_einzelstationen ,
21
22 BEGIN OF gty_orgpftab ,
23     station TYPE nbew-orgpf ,
24 END OF gty_orgpftab .
```

B. Programmcode aus Graphstation

```
25
26 PARAMETERS so_einri LIKE nbew-einri DEFAULT 'UKSH' NO-
    DISPLAY.
27
28 DATA: gt_stationswerte TYPE TABLE OF gty_stationswerte,
29         gt_final TYPE TABLE OF gty_stationswerte,
30         gs_stationswerte TYPE gty_stationswerte,
31         gs_pseud TYPE gty_stationswerte,
32         gs_einzelstation TYPE gty_einzelstationen,
33         gt_einzelstationen TYPE TABLE OF gty_einzelstationen
    ,
34 * Zur Nutzung der Funktion "Graph.." muss zumindest eine
    leere Tabelle
35 * als Ausgabeoption übergeben werden. Die Ausgabe ist dann
    "default".
36     BEGIN OF gt_ausgabeoption OCCURS 0,
37         option(1),
38     END OF gt_ausgabeoption.
39 DATA: go_alv_grid TYPE REF TO cl_salv_table,
40         columns TYPE REF TO cl_salv_columns_list.
41 DATA: t_rednpat LIKE npat.
42 DATA: t_fall LIKE nfal. "
43 DATA: counter_1 TYPE i,
44         counter_2 TYPE i,
45         counter_3 TYPE i.
46 DATA station TYPE nbew-orgpf.
47 DATA: iv_station TYPE nbew-orgpf.
48 DATA: it_stationsliste TYPE TABLE OF gty_orgpftab.
49 DATA: gs_element TYPE gty_orgpftab.
50 DATA stationszahl TYPE i.
51 SELECT-OPTIONS so_orgpf FOR gs_stationswerte-orgpf
    OBLIGATORY MEMORY ID station.
52
```

```

53 START-OF-SELECTION .
54
55 SELECT * FROM nbew INTO CORRESPONDING FIELDS OF TABLE
      gt_stationswerte
56 WHERE einri = so_einri
57 AND orgpf IN so_orgpf
58 AND bwedt GE sy-datum
59 AND bwidt LE sy-datum
60 AND ( bewty = '1' OR
61 bewty = '3' )
62 AND storn = space
63 AND plane = space.
64
65 LOOP AT gt_stationswerte INTO gs_stationswerte.
66     SELECT SINGLE * FROM nfal INTO t_fall
67     WHERE falnr = gs_stationswerte-falnr.
68     IF sy-subrc = 0.
69         MOVE t_fall-patnr TO gs_stationswerte-patnr.
70         MODIFY gt_stationswerte FROM gs_stationswerte.
71     ENDIF.
72
73     SELECT SINGLE * FROM npat INTO t_rednpat
74     WHERE patnr = gs_stationswerte-patnr.
75     IF sy-subrc = 0.
76         MOVE t_rednpat-gschl TO gs_stationswerte-gschl.
77         MOVE t_rednpat-gbdat TO gs_stationswerte-gbdat.
78         MODIFY gt_stationswerte FROM gs_stationswerte.
79     ENDIF.
80 ENDLOOP.
81
82 SORT gt_stationswerte BY orgpf gschl.
83
84 *Erzeugen einer Liste der Stationen mit Inhalt

```

B. Programmcode aus Graphstation

```
85  CLEAR gs_stationswerte.
86  LOOP AT gt_stationswerte INTO gs_stationswerte.
87      IF gs_stationswerte-orgpf NE iv_station.
88          APPEND gs_stationswerte-orgpf TO it_stationsliste.
89          iv_station = gs_stationswerte-orgpf.
90          stationszahl = stationszahl + 1.
91      ENDIF.
92  ENDLOOP.
93
94  IF it_stationsliste IS INITIAL.
95      MESSAGE 'Die ausgewählten OE enthalten keine
96              entsprechenden Daten.' TYPE 'A'.
97  ENDIF.
98
99  IF stationszahl > 28.
100     MESSAGE 'Die Zahl der gewählten Stationen liegt
101             ausserhalb des getesteten Bereichs (28).' TYPE 'A'.
102  ENDIF.
103
104  LOOP AT it_stationsliste INTO gs_element.
105      CLEAR gs_stationswerte.
106      CLEAR: counter_1, counter_2, counter_3, station,
107             gs_einzelstation.
108      station = gs_element-station.
109      LOOP AT gt_stationswerte INTO gs_stationswerte
110          WHERE orgpf = station.
111          IF gs_stationswerte-gschl = '1'.
112              counter_1 = counter_1 + 1.
113          ELSEIF gs_stationswerte-gschl = '2'.
114              counter_2 = counter_2 + 1.
115          ELSEIF gs_stationswerte-gschl = '3'.
116              counter_3 = counter_3 + 1.
117          ENDIF.
```

```

115     ENDLLOOP .
116     gs_einzelstation-orgpf = station .
117     gs_einzelstation-geschlcount_1m = counter_1 .
118     gs_einzelstation-geschlcount_2w = counter_2 .
119     gs_einzelstation-geschlcount_3u = counter_3 .
120     APPEND gs_einzelstation TO gt_einzelstationen .
121 ENDLLOOP .
122
123 CALL FUNCTION 'GRAPH_MATRIX_3D'
124     EXPORTING
125         col1     = 'Männlich'
126         col2     = 'Weiblich'
127         col3     = 'Unbekannt'
128         titl     = 'Stationsbelegung'
129         dim1     = 'Geschlecht'
130         dim2     = 'Station'
131         valt     = 'Patienten'
132     TABLES
133         data     = gt_einzelstationen
134         opts     = gt_ausgabeoption
135     EXCEPTIONS
136         OTHERS  = 1 .
137
138 * Anzeige im ALV
139 * cl_salv_table=>factory(
140 *     IMPORTING
141 *         r_salv_table     =     go_alv_grid
142 *     CHANGING
143 *         t_table         = gt_einzelstationen
144 **      t_table         = gt_stationswerte
145 *     ).
146 * columns = go_alv_grid->get_columns( ).
147 * columns->set_optimize( abap_true ).

```

B. Programmcode aus Graphstation

```
148 * DATA: lo_display TYPE REF TO cl_salv_display_settings.  
149 * lo_display = go_alv_grid→get_display_settings( ).  
150 * lo_display→set_striped_pattern( abap_true ).  
151 * go_alv_grid→display( ).
```

C. Programmcode aus Labanom

Im folgenden sind einige Klassen zum in JAVA 7 geschriebenen Programm LABANOM 4 zu sehen. Die Klasse *MD5HashBackup* benutzt zur Erstellung der Hashkeys das Paket *java security MessageDigest*. Zum Einlesen und Speichern wird das Paket *java io BufferedReader* bzw. *java io BufferedWriter* verwendet.

Listing C.1: Hauptprogramm *labanom4.java*.

```
1 import java.io.*;
2 import java.net.URL;
3 import java.lang.Object;
4 import java.io.File.*;
5 import java.awt.Component.*;
6 import java.awt.Container;
7 import javax.swing.JComponent;
8 import javax.swing.JFileChooser;
9 import java.util.*;
10 import java.io.BufferedReader;
11 import java.io.BufferedWriter;
12 import java.io.IOException;
13 import java.nio.charset.Charset;
14 import java.nio.charset.StandardCharsets;
15 import java.nio.file.Files;
16 import java.nio.file.Path;
17 import java.nio.file.Paths;
18 import java.util.Arrays;
19 import java.util.List;
20 import java.util.Scanner;
21 import java.io.BufferedReader;
```

```
22 public class Labano4
23 {
24     private TextRW fileworker;
25     private String SEPARATOR = "\\|"; //Delimiter
26     private String JOINER = "|";
27     private String ANONYM = "anonym_";
28     private String PSEU = "pseud_";
29     private String TRIGGER = "1";
30     private String ENDUNG = ".lab";
31     private MD5HashBackup md5generator;
32     private int flag;
33     public static void main(String... aArgs) throws
        IOException{
34         Labano4 run = new Labano4();
35         run.scan();
36     }
37     public Labano4()
38     {
39         //flag = 0; //Anonymize
40         flag = 1; //Pseudonymize
41         fileworker = new TextRW();
42         md5generator = new MD5HashBackup();
43     }
44     public Labano4(int auftrag)
45     {
46         flag = auftrag;
47         fileworker = new TextRW();
48         md5generator = new MD5HashBackup();
49     }
50     public void scan() throws IOException
51     {
52         File directory = new File(System.getProperty("user.
            dir"));
```

```

53     File[] currentFiles = directory.listFiles();
54     scanDirectory(currentFiles);
55 }
56 private void scanDirectory(File[] currentFiles)
57     throws IOException
58 {
59     if (currentFiles==null || currentFiles.length==0) {
60         return;
61     }
62     for (File file : currentFiles) {
63         if (file.isDirectory()) {
64             continue;
65         }
66         // Aktuelle Datei
67         String filename = file.getName();
68         if (filename.endsWith(ENDUNG) && !filename.
69             startsWith(ANONYM)
70             && !filename.
71             startsWith(
72                 PSEU)) {
73             replace(file);
74             //file.delete();
75         }
76     }
77 }
78 public void replace(File inputFile) throws IOException
79 {
80     List<Zeile> zeilen = new ArrayList<Zeile>();
81     String filename = inputFile.getName();
82     String filepfad = inputFile.getAbsolutePath();
83     List<String> lines = fileworker.readLargerTextFile
84         ((inputFile));
85     //Zeileninstanz für jede Linie im Text

```

```
81     for (String line:lines) {
82         String[] values = line.split(SEPARATOR, -1);
83         zeilen.add(new Zeile(values));
84     }
85     List<String> outlines = new ArrayList<String>();
86     for (Zeile zeile:zeilen) { //Anonymisieren
87         if (flag == 0) {
88             zeile.anonymize(TRIGGER);
89         }
90         if (flag == 1) {
91             zeile.pseudonymize(TRIGGER, md5generator);
92         }
93         outlines.add(zeile.rebuildline(JOINER));
94     }
95     Path exportpath = Paths.get(filepfad);
96     String altername = exportpath.toString();
97     String prefix = "";
98     if (flag == 0) prefix = ANONYM;
99     else if (flag == 1) prefix = PSEU;
100    String neuename = altername.replace(filename,
101        prefix + filename);
102    System.out.println("File created "+neuename);
103    fileworker.writeLargerTextFile(neuename, outlines)
104        ;
105    }
106 }
```

Listing C.2: *Zeile.java* zum Erstellen einer Instanz für jede Zeile.

```
1 import java.util.*;
2 import java.util.Arrays;
3 public class Zeile
4 {
5     private int length;
6     public String[] valuessep;
```

```

7     public Zeile()
8     {
9         length = 0;
10        valuessep= null;
11    }
12        public Zeile(String[] values){
13            valuessep = values;
14            length = valuessep.length;
15        }
16    public String[] getvalues(){
17        return valuessep;
18    }
19    public void anonymize(String trigger){
20        if ((valuessep != null) && (valuessep.length>1)) {
21            //Zeile vorhanden und nicht leer
22            if (valuessep[0].equals(trigger) ) { //Check
23                interessante Zeile
24                for (int i=1; i<=7; i++){ //Trigger bleibt
25                    drin
26                    StringBuilder sB = new StringBuilder
27                        ();
28                    sB.setLength(valuessep[i].length());
29                    String s = sB.toString().replaceAll("
30                        \u0000", "a");
31                    valuessep[i] = s;
32                }
33            }
34        }
35    public void pseudonymize(String trigger,
36        MD5HashBackup md5gen){
37        if ((valuessep != null) && (valuessep.length>1)) { //
38            Zeile vorhanden und nicht leer

```

```
33         if (valuessep[0].equals(trigger) ) { //Check
           interessante Zeile
34           //ACHTUNG, in NPAT im SAP sind zwei
           führende Nullen.
35           String hashcreate = "00"+valuessep[3]; //
           da .lab
36           String md5key = md5gen.md5create(
           hashcreate);
37           String jahr = datumjahr(valuessep[5]);
38           for (int i=4; i<6; i++){
39               StringBuilder sB = new StringBuilder
               ();
40               sB.setLength(valuessep[i].length());
41               String s = sB.toString().replaceAll("
               \u0000", "c");
42               valuessep[i] = s;
43           }
44           valuessep[3] = md5key;
45           valuessep[5] = jahr;
46       }
47   }
48 }
49 public String datumjahr(String datum){
50     String jahr = "";
51     String[] strArray = datum.split("\\.");
52     if (strArray.length == 3) {
53         jahr = strArray[2];
54     }
55     return jahr;
56 }
57 public String rebuildline(String seperator){
58     String s = "";
```

```
59         if ((valuessep != null) && (this.length>0)) { //
           Zeile vorhanden und nicht leer
60             StringBuilder sb = new StringBuilder();
61             if (this.length > 0 ) { //Aenderung 1 zu 0,
               Aenderung ist Zusatzbedingung
62                 sb.append(valuessep[0]);
63                 for (int i=1; i<this.length; i++){
64                     sb.append(seperator);
65                     sb.append(valuessep[i]);
66                 }}
67             s = sb.toString();
68         }
69     return s;
70 }
71 }
```


D. Programmcode aus Saptransfer

Für jede Zeile in der CSV-Datei aus dem SAP wird eine Instanz der Klasse *SapZeile* erzeugt. Diese Klasse bietet Funktionen zum Erhalt der in Tabelle 5.2 auf Seite 80 und Tabelle 5.3 auf Seite 81 beschriebenen Werte. Die inhaltliche Verwendung der Informationen aus den *SapZeilen* ist in Anhang E auf Seite 207 am Beispiel der gesuchten Werte für die CDA-Dokumente beschrieben.

Listing D.1: *SapZeile.java*.

```
1 import java.util.*;
2 import java.util.Arrays;
3 public class SapZeile
4 {
5     private int length;
6     public String[] valuessep;
7     private int POSITION_PATNR = 1;
8     private int POSITION_PSEUD = 0;
9     private int POSITION_GSCHL = 2;
10    private int POSITION_GBDAT = 3;
11    private int POSITION_EINRI = 4;
12    private int POSITION_EINRIKNAME = 5;
13    private int POSITION_EINRINAME = 6;
14    private int POSITION_LANDKEY = 7;
15    private int POSITION_PLZ = 8;
16    private int POSITION_ORT = 9;
17    private int POSITION_STREET = 10;
18    private int POSITION_INST = 11;
19    private int POSITION_PHONE = 12;
20    private int POSITION_FALNR = 13;
```

D. Programmcode aus Saptransfer

```
21     private int POSITION_HLFDNR = 14; // bis 20
22     private int POSITION_HDKAT1 = 15;
23     private int POSITION_HDKEY1 = 16;
24     private int POSITION_HDITXT = 17;
25     private int POSITION_HERDAT = 18;
26     private int POSITION_HERTIM = 19;
27     private int POSITION_HDTEXT1 = 20;
28     private int POSITION_ALFDNR = 21; // bis 27
29     private int POSITION_ADKAT1 = 22;
30     private int POSITION_ADKEY1 = 23;
31     private int POSITION_ADITXT = 24;
32     private int POSITION_AERDAT = 25;
33     private int POSITION_AERTIM = 26;
34     private int POSITION_ADTEXT1 = 27;
35     private int POSITION_ELFDNR = 28; // bis 34
36     private int POSITION_EDKAT1 = 29;
37     private int POSITION_EDKEY1 = 30;
38     private int POSITION_EDITXT = 31;
39     private int POSITION_EERDAT = 32;
40     private int POSITION_EERTIM = 33;
41     private int POSITION_EDTEXT1 = 34;
42     private int POSITION_LFDNR = 35;
43     private int POSITION_BWIDT = 36;
44     private int POSITION_BWIZT = 37;
45     private int POSITION_BWEDT = 38;
46     private int POSITION_BWEZT = 39;
47     private int POSITION_ORGPF = 40; // bis 42
48     private int POSITION_ZIMMR = 41;
49     private int POSITION_BETT = 42;
50     private int POSITION_alfdnr = 43;
51     private int POSITION_abwidt = 44;
52     private int POSITION_abwizt = 45;
53     private int POSITION_abwgr1 = 46;
```

```
54     private int POSITION_elfdnr = 47;
55     private int POSITION_ebwidt = 48;
56     private int POSITION_ebwizt = 49;
57     private int POSITION_MATNR = 50; //bis 55 bzw. bis 58
58     private int POSITION_MAKTX = 51;
59     private int POSITION_MENGE = 52;
60     private int POSITION_MEINS = 53;
61     private int POSITION_CONSDT = 54;
62     private int POSITION_CONSTM = 55;
63     private int POSITION_SUBSTGRP = 56;
64     private int POSITION_VERABRART = 57;
65     private int POSITION_ATCGRUPPE = 58;
66     private int POSITION_ATCMAT = 59;
67     private int POSITION_AGENT1 = 60; //bis68
68     private int POSITION_QUANT1 = 61;
69     private int POSITION_UNIT1 = 62;
70     private int POSITION_AGENT2 = 63;
71     private int POSITION_QUANT2 = 64;
72     private int POSITION_UNIT2 = 65;
73     private int POSITION_AGENT3 = 66;
74     private int POSITION_QUANT3 = 67;
75     private int POSITION_UNIT3 = 68;
76     private int POSITION_NNAME = 69;
77     private int POSITION_VNAME = 70;
78     private int POSITION_LNRLM = 71;
79     public SapZeile()
80     {
81         length = 0;
82         valuessep= null;
83     }
84     public SapZeile(String[] values){
85         valuessep = values;
86         length = valuessep.length;
```

```
87     }
88     public String[] getvalues(){
89         return valuessep;
90     }
91     public String getcsvstring(int n){
92         String direktstring = "";
93         if (length > n) {
94             if (valuessep[n] != null) {
95                 return valuessep[n];
96             }
97         }
98         return direktstring;
99     }
100    public String getname(){
101        if (getcsvstring(POSITION_NNAME).equals(""))
102            return "";
103        return getcsvstring(POSITION_NNAME)+" ,□"+
104            getcsvstring(POSITION_VNAME);
105    }
106    public String admreason(){
107        return getcsvstring(POSITION_ADTEXT1);
108    }
109    public String admdate(){
110        return (""+datumsumwandlung(getcsvstring(
111            POSITION_abwidt))+uhrzeitumwandlung(
112            getcsvstring(POSITION_abwizt)));
113    }
114    public String admID(){
115        return (""+getcsvstring(POSITION_FALNR)+" "+
116            getcsvstring(POSITION_alfdnr));
117    }
118    public String substadm(){
```

```
114         return (""+getcsvstring(POSITION_FALNR)+" "+
115             getcsvstring(POSITION_LNRLM));
116     }
117     public String getstreet(){
118         return (""+getcsvstring(POSITION_STREET));
119     }
120     public String getLANDKEY(){
121         return (""+getcsvstring(POSITION_LANDKEY));
122     }
123     public String getORT(){
124         return (""+getcsvstring(POSITION_ORT));
125     }
126     public String getPHONE(){
127         return (""+getcsvstring(POSITION_PHONE));
128     }
129     public String getPLZ(){
130         return (""+getcsvstring(POSITION_PLZ));
131     }
132     public String getINST(){
133         return (""+getcsvstring(POSITION_INST));
134     }
135     public String getinstname(){
136         return (""+getcsvstring(POSITION_EINRINAME));
137     }
138     public String getbed(){
139         return (""+getcsvstring(POSITION_BETT));
140     }
141     public String getzimmer(){
142         return (""+getcsvstring(POSITION_ZIMMR));
143     }
144     public String getorgpf(){
145         return (""+getcsvstring(POSITION_ORGPF));
```

```
146     }
147     public String getdisletdes(){
148         return (""+getcsvstring(POSITION_EDTEXT1));
149     }
150     public String getkhletdes(){
151         return (""+getcsvstring(POSITION_HDTEXT1));
152     }
153     public String disdate(){
154         return (""+datumsumwandlung(getcsvstring(
155             POSITION_ebwidt))+uhrzeitumwandlung(
156             getcsvstring(POSITION_ebwizt)));
157     }
158     public String disID(){
159         return (""+getcsvstring(POSITION_FALNR)+" "+
160             getcsvstring(POSITION_elfdnr));
161     }
162     public String khdiagid(){
163         return (""+getcsvstring(POSITION_HLFDNR));
164     }
165     public String khdiagdate(){
166         return (""+datumsumwandlung(getcsvstring(
167             POSITION_HERDAT))+uhrzeitumwandlung(
168             getcsvstring(POSITION_HERTIM)));
169     }
170     public String disdiagid(){
171         return (""+getcsvstring(POSITION_ELFDR));
172     }
173     public String disdiagdate(){
174         return (""+datumsumwandlung(getcsvstring(
175             POSITION_EERDAT))+uhrzeitumwandlung(
176             getcsvstring(POSITION_EERTIM)));
177     }
178 }
```

```

171     public String getconsm(){ //Datum einer
        Medikamentengabe
172         return (""+datumsumwandlung(getcsvstring(
            POSITION_CONSDT))+uhrzeitumwandlung(
            getcsvstring(POSITION_CONSTM)));
173     }
174     public String HDKEY1(){
175         return (""+getcsvstring(POSITION_HDKEY1));
176     }
177     public String HDTEXT1(){
178         return (""+getcsvstring(POSITION_HDTEXT1));
179     }
180     public String hdvocicd(){
181         String hdkat = getcsvstring(POSITION_HDKAT1);
182         String umw = icdtrans(hdkat);
183         return umw;
184     }
185     public String EDKEY1(){
186         return (""+getcsvstring(POSITION_EDKEY1));
187     }
188     public String EDTEXT1(){
189         return (""+getcsvstring(POSITION_EDTEXT1));
190     }
191     public String edvocicd(){
192         String hdkat = getcsvstring(POSITION_EDKAT1);
193         String umw = icdtrans(hdkat);
194         return umw;
195     }
196     public String icdtrans(String icdsap){
197         String umw = "";
198         String dkat = icdsap;
199         switch(dkat)
200         {

```

```
201         case "I4":
202             umw = "ICD10 -2004□ICD -10 -GM ,□Version□2004";
203             break;
204         case "I5":
205             umw = "ICD10 -2005□ICD -10 -GM ,□Version□2005";
206             break;
207         case "I6":
208             umw = "ICD10 -2006□ICD -10 -GM ,□Version□2006";
209             break;
210         case "I7":
211             umw = "ICD10 -2007□ICD -10 -GM ,□Version□2007";
212             break;
213         case "I8":
214             umw = "ICD10 -2008□ICD -10 -GM ,□Version□2008";
215             break;
216         case "I9":
217             umw = "ICD10 -2009□ICD -10 -GM ,□Version□2009";
218             break;
219         case "IA":
220             umw = "ICD10 -2010□ICD -10 -GM ,□Version□2010";
221             break;
222         case "IB":
223             umw = "ICD10 -2011□ICD -10 -GM ,□Version□2011";
224             break;
225         case "IC":
226             umw = "ICD10 -2012□ICD -10 -GM ,□Version□2012";
227             break;
228         case "ID":
229             umw = "ICD10 -2013□ICD -10 -GM ,□Version□2013";
230             break;
231         case "IE":
232             umw = "ICD10 -2014□ICD -10 -GM ,□Version□2014";
233             break;
```

```

234         default:
235             umw = dkat;
236             break;
237     }
238     return umw;
239 }
240 public String getconslit(){
241     String LITERAL_OF_THE_USED_TERM = "";
242     String einnahmeart = (""+getcsvstring(
        POSITION_VERABRART));
243     if (einnahmeart.equals("iv") || einnahmeart.equals
        ("iv.") || einnahmeart.equals("IV") ||
        einnahmeart.equals("IV."))
244         {LITERAL_OF_THE_USED_TERM = "INTRAVENOUS"
            ;}
245     else {
246         if (einnahmeart.equals("po") ||
            einnahmeart.equals("po.") ||
            einnahmeart.equals("PO") || einnahmeart
            .equals("PO."))
247             {LITERAL_OF_THE_USED_TERM = "ORAL";}
248     }
249     return LITERAL_OF_THE_USED_TERM;
250 }
251 public String getconscodcode(){
252     String CONCEPT_CODE = "";
253     String einnahmeart = (""+getcsvstring(
        POSITION_VERABRART));
254     if (einnahmeart.equals("iv") || einnahmeart.equals
        ("iv.") || einnahmeart.equals("IV") ||
        einnahmeart.equals("IV."))
255         {CONCEPT_CODE = "C38276";}
256     else {

```

```
257         if (einnahmeart.equals("po") ||
                einnahmeart.equals("po.") ||
                einnahmeart.equals("PO") || einnahmeart
                .equals("PO."))
258             {CONCEPT_CODE = "C38288";}
259         }
260     return CONCEPT_CODE;
261 }
262 public String getmatnr(){
263     return (""+getcsvstring(POSITION_MATNR));
264 }
265 public String getmenge(){
266     return (""+getcsvstring(POSITION_MENGE));
267 }
268 public String getmeins(){
269     return (""+getcsvstring(POSITION_MEINS));
270 }
271 public String getatccode(){
272     return (""+getcsvstring(POSITION_ATCMAT));
273 }
274 public String getsbstgrp(){
275     return (""+getcsvstring(POSITION_SUBSTGRP));
276 }
277 public String datumjahr(String datum){
278     String jahr = "";
279     String[] strArray = datum.split("\\.");
280     if (strArray.length == 3) {
281         jahr = strArray[2];
282     }
283     return jahr;
284 }
285 public String datumsumwandlung(String datum){
286     String neudatum = "";
```

```

287         String[] strArray = datum.split("\\.");
288         if (strArray.length == 3) {
289             String jahr = strArray[2];
290             String monat = strArray[1];
291             String tag = strArray[0];
292             neudatum = ""+jahr+monat+tag;
293             if (strArray[1].equals("_")) {
294                 neudatum = ""+jahr;
295             }
296         }
297         return neudatum;
298     }
299     public String uhrzeitumwandlung(String zeit){
300         String uszeit = "";
301         String[] strArray = zeit.split("\\:");
302         if (strArray.length == 3) {
303             String second = strArray[2];
304             String minute = strArray[1];
305             String hour = strArray[0];
306             uszeit = ""+hour+minute+second;
307         }
308         return uszeit;
309     }
310     public String getID(){
311         int y = POSITION_PSEUD;
312         int x = POSITION_PATNR;
313         String id = "";
314         if (length > x) {
315             if (valuessep[x] != null && !(
316                 valuessep[x].equals("")) {
317                 id = valuessep[x];
318             }
319             else {

```

```
319         if (length > y) {
320             if (valuessep[y] != null) {
321                 id = valuessep[y];
322             }
323         }
324     }
325 }
326 return id;
327 }
328 public String getpseud(){
329     int x = POSITION_PSEUD;
330     int y = POSITION_PATNR;
331     String pseud = "";
332     if (length > x) {
333         if (valuessep[x] != null && !(
334             valuessep[x].equals("")) {
335             pseud = valuessep[x];
336         }
337         else {
338             if (length > y) {
339                 if (valuessep[y] != null) {
340                     pseud = valuessep[y];
341                 }
342             }
343         }
344     }
345     return pseud;
346 }
347 public String getgschl(){
348     /*
349     * 1 männlich
350     * 2 weiblich
```

```
351         * 3 unbekannt
352         */
353         int x = POSITION_GSCHL;
354         String geschl = "";
355         if (length > x) {
356             if (valuessep[x] != null) {
357                 if (valuessep[x].equals("1")) geschl = "M";
358                 if (valuessep[x].equals("2")) geschl = "F";
359                 if (valuessep[x].equals("3")) geschl = "UN";
360             }
361         }
362         return geschl;
363     }
364
365     public String getgbdat(){
366         int x = POSITION_GBDAT;
367         if (length > x) {
368             if (valuessep[x] != null) {
369                 return datumsumwandlung(valuessep[x]);
370             }
371             else return "";
372         }
373         else return "";
374     }
375     public String geteinri(){
376         int x = POSITION_EINRI;
377         if (length > x) {
378             if (valuessep[x] != null) {
379                 return valuessep[x];
380             }
381             else return "";
382         }
383         else return "";
```

```
384     }
385     public String getfalnr(){
386         int x = POSITION_FALNR;
387         if (length > x) {
388             if (valuessep[x] != null) {
389                 return valuessep[x];
390             }
391             else return "";
392         }
393         else return "";
394     }
395     //Hauptdiagnose, Schlüssel
396     public String[] gethdiagnose(){
397         int z = POSITION_HDKAT1; //Start
398         int x = POSITION_HDKEY1; //Ende
399         int length1 = x - z + 1;
400         if (length > x) {
401             if (valuessep[x] != null) {
402                 String[] diagnose = new String[
403                     length1];
404                 for (int i=0; i<length1; i++){
405                     diagnose[i] = valuessep[z+i];
406                 }
407                 return diagnose;
408             }
409             else return null;
410         }
411     }
412     //Bewegung
413     public String[] getbewegung(){
414         int z = POSITION_LFDNR; //Start
415         int x = POSITION_BETT; //Ende
```

```

416         int length1 = x - z + 1;
417         if (length > x) {
418             if (valuessep[x] != null) {
419                 String[] bewegung = new String[
420                     length1];
421                 for (int i=0; i<length1; i++){
422                     bewegung[i] = valuessep[z+i];
423                 }
424                 return bewegung;
425             }
426         }
427         else return null;
428     }
429     public String[] getsubstance(){
430         int z = POSITION_MATNR; //Start
431         int x = POSITION_UNIT3; //Ende
432         int length1 = x - z + 1;
433         if (length > x) {
434             if (valuessep[x] != null) {
435                 String[] substance = new String[
436                     length1];
437                 for (int i=0; i<length1; i++){
438                     substance[i] = valuessep[z+i];
439                 }
440                 return substance;
441             }
442         }
443         else return null;
444     }
445 }

```


E. Wertezuordnung zur Clinical Document Architecture

Wie in Kapitel 7 auf Seite 95 erklärt, werden aus der im SAP System erzeugten CSV-Datei für jeden Patienten ein KIS-CDA und ein Antibiotika-CDA generiert. Im folgenden werden die Wertzuordnung bzw. Generierung zum Stand des 7.11.2014 beschrieben. Die Überschriften stehen jeweils für die gesuchten Werte im CDA-Dokument. Darunter folgt der zugeordnete Wert aus der in SAP erzeugten CSV-Datei.

E.1. Krankenhausinformationssystem

MESSAGE_ID

Institutskennzeichen + Fallnummer + Aktuelles Datum.

TITLE_FOR_THIS_DOCUMENT

Patientennummer + „_HIS_“ + Aktuelles Datum.

DATE_OF_THE_CREATION_OF_THIS_DOCUMENT (YYYYMMDDHHMMSS)

Aktuelles Datum.

PATIENT_ID

Patientennummer.

PATIENT_NAME

Nachname + „,“ + Vorname.

M(Male), F(Female), UN (Undifferentiated)

“1” -> “M”.

“2” -> “F”.

“3” -> “UN”.

BIRTHDATE(YYYYMMDDHHMMSS)

Geburtsdatum in Formatierung “TT.MM.JJJJ” wird umgewandelt in obiges Format unter Wegfall der Uhrzeit.

OPTIONAL:BRIEF_LITERAL_DESCRIPTION_OF_THE_REASON_OF_THE_ADMISSION_INTO_THE_HOSPITAL

Falls eine als Aufnahmediagnose gekennzeichnete Diagnose vorhanden ist, wird der im SAP zum ICD Code entsprechende Text verwendet.

DATE_OF_ADMISSION(YYYYMMDDHHMMSS)

Zu einer als Aufnahmebewegung gekennzeichneten Bewegung in diesem Fall: Datum der Bewegung + Uhrzeit der Bewegung.

ID_OF_THE_ADMISSION_ACT

Fallnummer + Laufende Nummer der als Aufnahme gekennzeichneten Bewegung.

DATE_OF_ADMISSION(YYYYMMDDHHMMSS)

Zu einer als Aufnahmebewegung gekennzeichneten Bewegung in diesem Fall: Datum der Bewegung + Uhrzeit der Bewegung.

STREET

Straße und Hausnummer des Instituts.

CITY

Ort.

STATE

Wird nicht befüllt.

POSTALCODE

Postleitzahl.

COUNTRY

Länderschlüssel (z.B. „DE“).

TELEPHONE

Telefonnummer.

UNIQUE_IDENTIFIER_OF_THE_CLINIC/HOSPITAL

Institutskennzeichen.

NAME_OF_THE_CLINIC/HOSPITAL

Name der Einrichtung.

UNIQUE_IDENTIFIER_OF_THE_BED

BauId eines Bettenstellplatzes.

UNIQUE_IDENTIFIER_OF_THE_ROOM

BauId eines Zimmers.

UNIQUE_IDENTIFIER_OF_THE_STATION

OrgEinheit, die einem Fall zugewiesen wird.

OPTIONAL:BRIEF_LITERAL_DESCRIPTION_OF_THE_DIAGNOSIS

Falls eine als Entlassungsdiagnose gekennzeichnete Diagnose vorhanden ist, wird der im SAP zum ICD Code entsprechende Text verwendet.

DATE_OF_DISCHARGE(YYYYMMDDHHMMSS)

Zu einer als Entlassungsbewegung gekennzeichnete Bewegung in diesem Fall: Datum der Bewegung + Uhrzeit der Bewegung.

ID_OF_THE_DISCHARGE_ACT

Fallnummer + Laufende Nummer der als Entlassung gekennzeichneten Bewegung.

DATE_OF_DISCHARGE(YYYYMMDDHHMMSS)

Zu einer als Entlassungsbewegung gekennzeichneten Bewegung in diesem Fall: Datum der Bewegung + Uhrzeit der Bewegung.

ID_OF_THE_OBSERVATION

Laufende Nummer Diagnose der als Entlassungsdiagnose gekennzeichneten Diagnose für diesen Fall. Sollte diese nicht vorhanden sein, wird hier und unten die Krankenhaus-hauptdiagnose (bzw. Fachabteilungsdiagnose) verwendet.

DATE_OF_DIAGNOSIS(YYYYMMDDHHMMSS)

Zu einer als Entlassungsdiagnose gekennzeichneten Diagnose in diesem Fall: Datum, an dem der Satz hinzugefügt wurde + Uhrzeit, zu der der Satz hinzugefügt wurde

DIAGNOSIS_COREDATASET_CONCEPT_CODE

Identifikationskürzel für den Diagnosekatalog bei einer als Entlassungsdiagnose gekennzeichneten Diagnose.

HL7_CODE_FOR_THE_VOCABULARY

„2.16.840.1.113883.6.3“, falls ICD10.

NAME_OF_THE_VOCABULARY_USED

Schlüsselung der Diagnose wird umgerechnet:

"I4": "ICD10-2004 ICD-10-GM, Version 2004".

"I5": "ICD10-2005 ICD-10-GM, Version 2005".

"I6": "ICD10-2006 ICD-10-GM, Version 2006".

"I7": "ICD10-2007 ICD-10-GM, Version 2007".

"I8": "ICD10-2008 ICD-10-GM, Version 2008".

"I9": "ICD10-2009 ICD-10-GM, Version 2009".

"IA": "ICD10-2010 ICD-10-GM, Version 2010".

"IB": "ICD10-2011 ICD-10-GM, Version 2011".

"IC": "ICD10-2012 ICD-10-GM, Version 2012".

"ID": "ICD10-2013 ICD-10-GM, Version 2013".

"IE": "ICD10-2014 ICD-10-GM, Version 2014".

LITERAL_OF_THE_USED_TERM

Zur vorhandenen Entlassungsdiagnose wird der im SAP zum ICD Code hinterlegte Text verwendet.

E.2. Antibiotika

TITLE_FOR_THIS_DOCUMENT

Patientennummer + „_ANTIBIOTIC_“ + Aktuelles Datum.

DATE_OF_THE_CREATION_OF_THIS_DOCUMENT (YYYYMMDDHHMMSS)

Aktuelles Datum.

PATIENT_ID

Patientennummer.

PATIENT_NAME

Nachname + „,“ + Vorname.

M(Male), F(Female), UN (Undifferentiated)

„1“ -> „M“.

„2“ -> „F“.

„3“ -> „UN“.

BIRTHDATE(YYYYMMDDHHMMSS)

Geburtsdatum in Formatierung „TT.MM.JJJJ“ wird umgewandelt in obiges Format unter Wegfall der Uhrzeit.

Entry

Jede neue Gabe eines Medikaments hat einen neuen Entry zur Folge, begleitet mit den anderen Informationen zur Medikamentenvergabe in diesem Dokument.

ID_OF_THE_SUBSTANCE_ADMINISTRATION

Fallnummer + Laufende Nummer der Materialanforderung.

LITERAL_DESCRIPTION_OF_THE_SUBSTANCEADMINISTRATION

Wird nicht befüllt.

STATUS_OF_THE_SUBSTANCEADMINISTRATION

Wird nicht befüllt.

DATE_OF_THE_START_OF_THE_TREATMENT

Für den ersten Eintrag dieses Materials beim Patienten in diesem Fall: Datum der Verabreichung eines Materials + Uhrzeit der Verabreichung eines Materials.

DATE_OF_THE_END_OF_THE_TREATMENT

Wird nicht befüllt.

value="%%TIME_NUMBER%%" unit="%%TIME_UNIT%%"

Stattdessen wird das Schema für die aktuelle einzelne Vergabe verwendet und dafür entsprechend eingetragen: Datum der Verabreichung eines Materials + Uhrzeit der Verabreichung eines Materials.

Beispiel: "<effectiveTime xsi:type="TS" value="20131119083000" />".

ROUTE_COREDATASET_CONCEPT_CODE

„C38276“ bei iv, „C38288“ bei po.

HL7_CODE_FOR_THE_VOCABULARY

„2.16.840.1.113883.3.26.1.1“.

NAME_OF_THE_VOCABULARY_USED

„NCI Thesaurus“.

LITERAL_OF_THE_USED_TERM

„INTRAVENOUS“ bei iv, „ORAL“ bei po.

DOSE; UNITS_OF_DOSE

(Angeforderte) Menge des Materials; Mengeneinheit.

RATE; UNITS_PER_TIME

Wird nicht befüllt.

PRODUCT_COREDATASET_CONCEPT_CODE

Wird nicht befüllt.

HL7_CODE_FOR_THE_VOCABULARY

Wird nicht befüllt.

NAME_OF_THE_VOCABULARY_USED

Wird nicht befüllt.

LITERAL_OF_THE_USED_TERM

Wird nicht befüllt.

GENERIC_COREDATASET_CONCEPT_CODE

ATC Code für das Medikament

HL7_CODE_FOR_THE_VOCABULARY

„1.2.276.0.76.5.319“.

NAME_OF_THE_VOCABULARY_USED

„ATC“.

LITERAL_OF_THE_USED_TERM

Substanzgruppe.