

Universität Konstanz  
FB Informatik und Informationswissenschaft  
Bachelor-Studiengang Information Engineering

## **Bachelorarbeit**

zur Erlangung des akademischen Grades eines  
Bachelor of Science (B.Sc.)

### **Entwicklung eines Modells für die Repräsentation tabellenförmiger Daten aus Dokumenten.**

von

**Fabian Zintgraf**

(Matr.-Nr. 01 / 622978)

Erstgutachter: Prof. Dr. Marc Scholl

Zweitgutachter: Prof. Dr. Daniel A. Keim

Konstanz, den 17. Mai 2010

## **Abstract**

Tables can be found in almost every medium. In newspapers, books, invoices or electronic media. They provide the reader with structured data, which would be by far less obvious from text only. Usually humans do not have great difficulties with recognizing structured data in table forms. Automatic computer processes have to collect these information arduously from various sources to understand the data structure. Both the amount of different document types and the different types of table layouts complicate the process of automatic table extraction. An exchange of tabular structures between the most common document types is not supported. In particular, the transfer of pixel-based tables within documents is missing. The challenge consists of storing collected table content from various types of documents uniformly. This thesis describes the development of such a model that represents tabular data across different document types.

## **Kurzfassung**

Tabellen finden sich in fast jedem Medium wieder. In Zeitungen, Büchern, Rechnungen oder in elektronischen Medien vermitteln sie dem Leser strukturierte Daten, die aus bloßem Text viel weniger ersichtlich wären. Wie die Tabellen präsentiert werden, ist für den Menschen irrelevant. Nahezu ohne Schwierigkeiten ist er in der Lage, die Strukturen zu erkennen und aufzunehmen. Automatische Computer-Prozesse müssen sich hier entgegen mühsam an verschiedenen Informationsquellen bedienen, um die Datenstruktur verstehen zu können. Die Menge an verschiedenen Dokumententypen und die verschiedenen Arten von Tabellenlayouts erschweren den Prozess der automatischen Tabellenerkennung zusätzlich. Ein formatübergreifender Austausch von tabellarischen Strukturen, selbst bei den gängigsten Dokumentarten, wird nicht unterstützt. Insbesondere der Transfer von Tabellen innerhalb pixelbasierter Dokumente fehlt. Eine besondere Herausforderung besteht also darin, Tabellen aus verschiedenen Dokumentarten einheitlich abzulegen. Diese Bachelorarbeit beschreibt die Entwicklung eines solchen Modells, in dem tabellenförmige Daten dokumentübergreifend repräsentiert werden können.

# Inhaltsverzeichnis

<b>Abstract / Kurzfassung</b>	<b>1</b>
<b>1 Einleitung</b>	<b>5</b>
1.1 Einleitung . . . . .	5
1.2 Erkenntnisse aus bisherigen Arbeiten . . . . .	7
1.3 Strukturierung der Arbeit . . . . .	10
<b>2 Feststellungen</b>	<b>11</b>
2.1 Tabellen . . . . .	11
2.1.1 Definition . . . . .	11
2.1.2 Terminologie . . . . .	12
2.1.3 Tabellenarten . . . . .	14
2.1.4 Tabellenformate . . . . .	20
<b>3 Problemstellung</b>	<b>25</b>
3.1 Anforderungen . . . . .	26
3.1.1 Tabellenstruktur . . . . .	26
3.1.2 Tabellenlayout . . . . .	26
3.1.3 Generik . . . . .	27
3.1.4 Schema . . . . .	27
3.2 Folgerungen . . . . .	27
<b>4 Entwicklung</b>	<b>30</b>

<i>INHALTSVERZEICHNIS</i>	3
4.1 Tabellenmodell . . . . .	30
4.1.1 Überblick . . . . .	30
4.1.2 Programmaufbau . . . . .	32
4.2 Gesamtprozess . . . . .	36
4.2.1 Erkennung . . . . .	37
4.2.2 Extraktion . . . . .	37
4.2.3 Weiterverarbeitung . . . . .	38
4.2.4 Callback . . . . .	38
4.3 Simulationsoberfläche . . . . .	38
4.3.1 Überblick . . . . .	39
<b>5 Erkenntnisse</b>	<b>41</b>
5.1 Ergebnisse . . . . .	41
5.2 Ausblick . . . . .	43
<b>6 Zusammenfassung</b>	<b>46</b>
<b>7 Anhang</b>	<b>47</b>
7.1 Entwicklung . . . . .	47
<b>Literaturverzeichnis</b>	<b>48</b>

# Abbildungsverzeichnis

2.1	Unterschiede zwischen Tabellen und Formularen . . . . .	12
2.2	Die Terminologie einer Tabelle nach Wang . . . . .	13
2.3	Beispiel einer einfachen Tabelle . . . . .	15
2.4	Beispiel einer geschachtelten Tabelle . . . . .	16
2.5	Beispiel einer mehrseitigen Tabelle . . . . .	17
2.6	Beispiel einer Pivot-Tabelle . . . . .	17
2.7	Beispiel einer rekursiven Tabelle . . . . .	18
2.8	Beispiel einer komplexen Tabelle . . . . .	19
2.9	Das Periodensystem der Elemente . . . . .	19
4.1	Übersicht über das Tabellenmodell . . . . .	31
4.2	Eine mehrseitige Tabelle im Tabellenmodell . . . . .	32
4.3	Klassendiagramm der Trennerkonfiguration . . . . .	34
4.4	Klassendiagramm der Tabellenstruktur . . . . .	35
4.5	Übersicht über den Gesamtprozess . . . . .	36
4.6	Übersicht über die Simulationsoberfläche . . . . .	39
5.1	Beispiel für das zukünftige Tabellenmodell . . . . .	45

# Kapitel 1

## Einleitung

### 1.1 Einleitung

Tabellen begegnen uns jeden Tag. In Zeitungen, Büchern, Rechnungen oder in elektronischen Medien finden sie Verwendung, um dem Leser strukturierte Daten zu vermitteln, die aus der bloßen Textform viel weniger ersichtlich wären. Hierbei spielt es keine Rolle in welchem tabellarischen Layout die Daten repräsentiert werden - der Mensch ist ohne Schwierigkeiten in der Lage, selbst komplexeste Strukturen zu erkennen und aufzunehmen. Wohingegen sich automatische Prozesse mühsam an verschiedenen Informationsquellen bedienen müssen, um Tabellenstrukturen als solche überhaupt verstehen zu können. Die Vielzahl an unterschiedlichen Tabellenlayouts erschweren hierbei zusätzlich den Prozess der vollautomatischen Tabellenerkennung. Eine weitere Barriere besteht durch die nahezu unbegrenzte Menge an vorhandenen Dokumententypen, in denen auf verschiedenste Weisen tabellarische Strukturen ausgedrückt und gespeichert werden. Selbst die gängigsten Formate, wie Microsoft Office Word, HTML<sup>1</sup>, Adobe Acrobat PDF oder TeX, unterscheiden sich merklich in ihrer Repräsentationsweise und beschränken sich meist nur auf die interne Verarbeitung der Tabellendaten, wodurch ein formatübergreifender Austausch erschwert

---

<sup>1</sup>HTML: Hypertext Markup Language

oder sogar unmöglich gemacht wird. Insbesondere die Gewinnung von strukturierten Informationen aus pixelbasierten Dokumenten wird nicht allumfassend unterstützt. Hier kommt der Sektor der OCR<sup>2</sup> zum Tragen. Mittels spezieller Algorithmen ist es möglich, aus vereinzelt Pixeln Buchstaben zu erkennen, um somit auf Wörter schließen zu können. Zusätzlich müssen hier Hinweise, die zur Tabellengewinnung beitragen sollen, auf unterschiedlichste Weise gesammelt und interpretiert werden, da diese im Gegensatz zu anderen Formaten nicht direkt aus dem Kontext des Dokumentenformates heraus ablesbar sind.

Es besteht also eine besondere Herausforderung darin, in den verschiedensten Dokumententypen Tabellen zu identifizieren, zu extrahieren und deren Informationen zu erkennen, um sie dann in einem einheitlichen Tabellenmodell ablegen zu können. Hieraus entsteht zum einen die Schwierigkeit, die vielen verschiedenen Charakteristiken unterschiedlicher Tabellenformen auf ein Modell zu abstrahieren und zum anderen, schon existierende Modelle aus den verschiedensten Anwendungen mit zu unterstützen und sogar zu erweitern.

In dieser Arbeit wird nun die Entwicklung eines Modells aufgezeigt, welches tabellenförmige Daten aus verschiedenen Dokumenten repräsentiert, unter anderem auch mit besonderem Augenmerk auf die Problemstellung für pixelbasierte Dokumente. Die gewonnenen Ergebnisse sollen Anwendungsentwickler im Gebiet der Tabellenanalyse dabei unterstützen, sich mehr auf die Bedeutung globaler Lösungen zu konzentrieren, als individuelle Mittel und Wege für Teilprobleme zu generieren.

---

<sup>2</sup>OCR: Optical Character Recognition

## 1.2 Erkenntnisse aus bisherigen Arbeiten

Die Bedeutsamkeit der Erfassung und Repräsentation von tabellarischen Strukturen ist in den letzten Jahren immens angestiegen und bestätigt somit, wie wichtig Tabellen für die unterschiedlichsten Gebiete geworden sind. Daher ist es wenig verwunderlich, dass es sehr viele wissenschaftliche Publikationen gibt, die sich mit der Verarbeitung von tabellarischen Daten auseinandersetzen. Zum einen liegt das an der Menge an verschiedenen Dokumententypen, worauf die Arbeiten beruhen, und zum anderen an den unterschiedlichen Formen und Funktionen, die eine Tabelle annehmen kann.

Eine der wohl umfangreichsten Arbeiten in diesem Bereich beschreibt Wang in ihrer Dissertation [1]. Sie entwickelte ein Tabellenmodell, das die verschiedenen Stationen der Tabellenkomposition unterstützt, also die Beschreibung und Manipulation der logischen Struktur, die Spezifikation der Topologie und des Stils, sowie das Formatieren von konkreten Tabellen, jeweils wohl definiert durch mathematische Regeln. Auf diesem Modell basierend wurde eine Tabellen-gestaltungsanwendung implementiert, die dem Benutzer erlaubt, Tabellen mit hohem Qualitätsanspruch zu designen. Eine ähnliche Abstraktion der Tabelle in verschiedene Bereiche verwendet die Anwendung TARTAR [2]. Das zugrundeliegende Tabellenmodell besteht hier aus einer physikalischen, einer strukturellen, einer funktionalen und einer semantischen Komponente und basiert auf einer objektorientierten Sprache (F-Logik). Hierbei wurden HTML-kodierte Tabellen in das Tabellenmodellformat transformiert, um Ontologien aus Benutzerabfragen generieren zu können. Andere tabellenbezogene Anwendungen verwenden eigens entwickelte Modelle [3] [4] [5], oder setzen auf den ISO-Standard SGML (vgl. ISO 8879, 1986) zur Beschreibung logischer Strukturen von Dokumenten, und somit auch Tabellen [6].

Die Arbeit von Ramel et al. [4] beschäftigt sich mit einem flexiblen Repräsentationsschema für Tabellen und der Erkennung ebensolcher innerhalb verschiedener Inputdokumente. Der verwendete Korpus besteht aus Dokumenten, die für die

interne Kommunikation und für den Informationsaustausch zwischen Unternehmen bedacht waren, jeweils mit unterschiedlichsten Aufbauten in verschiedenen Dateiformaten. Die Dokumente werden zunächst in ein Austauschformat aus Druckanweisungen, wie Text, Linien, etc. umgewandelt, worauf das Tabellenmodell aufsetzt. Die Autoren beschreiben Tabellen durch ihr physikalisches Layout und ihre logische Struktur. Die physische Repräsentation wird durch eine reguläre Matrix aus virtuellen Zellen beschrieben, um mögliche Irregularitäten aus einer komplexeren Tabellenstruktur zu eliminieren. Virtuelle Zellen sind die kleinsten Rechtecke, die durch Verlängern aller Rahmengrenzen der Tabellenzellen bis hin zum Tabellenrand entstehen. Die logische Struktur beschreiben die realen Zellen, die demnach als Zusammenfassung von mindestens einer virtuellen Zelle gelten. Repräsentiert werden beide Strukturen hierbei durch ein bereits im Vorfeld [7] entwickelten DTDs<sup>3</sup> im XML-Format. In ihrer Arbeit beschreiben Ramel et al. zwei verschiedene Verfahren, um aus den Druckanweisungen Tabellen erkennen und aufbauen zu können. Die Strukturinformationen der Tabelle werden zum einen aus den horizontalen und vertikalen Linieninformationen innerhalb des Dokumentes abgeleitet oder zum anderen über die Analyse der Textelemente, indem im Dokument nach layoutspezifischen Regelmäßigkeiten gesucht wird. Beide Vorgehensweisen sind voneinander unabhängig und können zur Verfeinerung der Ergebnisse verknüpft werden. Der gezeigte Ansatz, verschiedene Dokumentarten in Druckanweisungen zu übersetzen und diese somit als Ursprungsinformationen zu verwenden, verringert zwar den Aufwand für die Anwendung, kann aber zu Informationsverlust führen. Denn Textelemente oder layoutspezifische Hinweise könnten ohne Umwege direkt aus dem Dokument extrahiert werden, zum Beispiel wenn es sich um eine HTML-Seite handelt. Die Verknüpfung von Wohlbergs Tabellenmodell [6] mit der Extrahierung aus HTML-Dokumenten zeigt die Arbeit von Gatterbauer et al. [8], wobei sie dieses als Zwischenschritt verwenden, um die Baumstruktur in ein zweidimensionales, optisches Boxen-Modell zu transformieren. Die Anwendung TabulaMagica [9]

---

<sup>3</sup>DTD: Document Type Definition

setzt hier entgegen auf das Wangsche Modell und bietet somit eine ähnliche Funktionalität. Interessant für die aktuelle Arbeit sind die Anmerkungen, wie Tabellen aus dem HTML-, RTF- und  $\text{\LaTeX}$ -Format importiert und die Informationen in DOM-Bäume, worauf die Anwendung aufbaut, konvertiert werden. Textbasierte Tabellen haben, bedingt durch ihr langes Bestehen, einen hohen Forschungsgrad erreicht. Beispielhaft sind hier die Arbeiten [10] [11], welche schon eine hohe Erkennungsrate liefern. Für PDF-Dokumente finden sich wenige wissenschaftliche Auseinandersetzungen [12] [3], während Arbeiten, wie zum Beispiel [13] [14] [15] [16], die auf pixelbasierten Formaten aufbauen, eher publiziert werden. Um sich in der Menge der vielen Publikationen zurecht zu finden, helfen hier die Ausarbeitungen von Lopresti und Nagy von 2000 [17], Zanibbi et al. von 2003 [18], und Embley, Hurst, Lopresti und Nagy aus dem Jahr 2005 [19]. Sie beschreiben jeweils eine Bestandsaufnahme der verschiedensten Studien und geben somit einen guten Überblick über aktuelle Forschungsprojekte.

### 1.3 Strukturierung der Arbeit

In Kapitel 2 wird zunächst erläutert, was eine Tabelle definiert und welche terminologischen Begriffe hierbei wichtig sind. Anhand von Beispielen wird verdeutlicht, in welchen verschiedenen Facetten tabellenförmige Daten repräsentiert werden können, die eine automatische Extrahierung erschweren. Zuletzt werden verschiedene Dokumentenformate angesprochen und Gemeinsamkeiten und Unterschiede erläutert, um die Problematik für die Tabellenextraktion aus diversen Dokumentarten zu verdeutlichen. In Kapitel 3 werden die Problemstellung und die Anforderungen für die Entwicklung des Tabellenmodells vorgestellt. Kritische Punkte werden im Vorfeld analysiert und Lösungsvorschläge aufgezeigt. Daraufhin wird in Kapitel 4 im Einzelnen auf die Entwicklung des Tabellenmodells eingegangen. Nach der detaillierten Beschreibung des Modells und dessen Funktionsweise, wird es im globalen Kontext stehend beschrieben und die Komponenten, die mit dem Modell kommunizieren, werden vorgestellt. Danach folgt die Vorstellung der eigens entwickelten Simulationsoberfläche. In Kapitel 5 werden die Ergebnisse dieser Arbeit kritisch betrachtet und es wird ausblickend gezeigt, wie man noch bestehende Probleme beheben kann. Abschließend werden die Hauptaspekte dieser Arbeit nochmals zusammengefasst.

# Kapitel 2

## Feststellungen

Um verstehen zu können, was ein Tabellenmodell leisten und bezwecken soll, muss zunächst auf das eigentliche ursprüngliche Objekt, die Tabelle, eingegangen werden.

### 2.1 Tabellen

#### 2.1.1 Definition

Tabellen können über ihren Hauptaspekt, strukturierte Daten zu repräsentieren, definiert werden. Es gibt jedoch keine Beschreibung, die alle verschiedenen Tabellenarten und deren Aussehen zusammenfasst, denn die Erscheinungsformen können sehr verschieden ausfallen, je nachdem welche Daten dargestellt werden sollen und in welchem Layout sich die Tabelle befindet.

Für ein genaueres Bild, sollte man zunächst Tabellen von Formularen abgrenzen, da sie sich grundsätzlich unterscheiden [17]. Abbildung 2.1 gibt die aufgezeigten Unterschiede wieder. Sie unterscheiden sich zum einen in ihrem Erscheinungsbild, da Formulare keinen strukturierten Gesetzmäßigkeiten unterliegen und unter Umständen auch von Hand ausgefüllt werden können. Zum anderen unterscheiden sie sich durch die Art der Verwendung. Tabellen repräsentieren Daten, wohingegen Formulare dem Zweck der Datengewinnung dienen.

Tables	Forms
For output	For input
Frame and content created simultaneously	Frame created before content
Tabular structure	Rectilinear structure
Machine-printed	Machine- or hand-printed
Sometimes unique	Frame rarely unique, content often unique

Abbildung 2.1: Der Unterschied zwischen Tabellen und Formularen besteht zum einen in ihrem Erscheinungsbild und zum anderen in der Art ihrer Verwendung [17]

Listen unterscheiden sich hingegen deutlich weniger von Tabellen. Sie entsprechen vielmehr einer ebenfalls regelmäßigen Struktur, jedoch mit wenigen Spalten. Deutliche Merkmale von Listen sind Aufzählungssymbole oder Nummerierungszeichen, welche sich an jeder neuen Position befinden, sowie die flache Struktur. In [19] wird der Bezug aufgestellt, dass, wenn Listen Vektoren entsprächen, man Tabellen sinngemäß als Matrizen sehen müsste.

Fasst man die Abgrenzungen gegenüber Formularen und Listen zusammen, ergibt sich die Definition einer Tabelle zum einen über die regelmäßige Struktur und zum anderen über die ausschließliche Repräsentation von Daten. Die Terminologie, also aus welchen Objekten eine Tabelle bestehen kann, soll nun im nächsten Abschnitt beschrieben werden.

### 2.1.2 Terminologie

Tabellen liegen in einer bestimmten Form vor und werden in einem Medium repräsentiert. Hauptsächlich werden sie als Zeilen- und Spaltenstruktur dargestellt und befinden sich in einem ebenen Medium, wie Papier oder elektronisch am Bildschirm. Die hier verwendeten terminologischen Begriffe beziehen sich auf die Arbeit von Wang [1], mit der Ausnahme, dass die Beschreibung um den Begriff 'footer' erweitert wurde.

Grundsätzlich ist eine Tabelle in fünf Hauptregionen unterteilt. Der Tabellenstumpf (stub) befindet sich auf der linken Seite und enthält die Zeilenüberschri-

ten (row headings), der Tabellenkopf (boxhead) befindet sich rechts oben und enthält die Spaltenüberschriften, die Region links oben in der Ecke entspricht der Tabellenbeschreibung (stub head), der Tabellenfuß (footer) befindet sich unter der Tabelle und der Tabellenrumpf (body) wird von Tabellenstumpf, Tabellenkopf und Tabellenfuß eingeschlossen und enthält die eigentlichen Einträge. Der Schnitt einer Spalte mit einer Zeile entspricht einer Zelle und eine rechteckige Vereinigungsmenge von Zellen werden Block genannt. Einer schemenhaften Darstellung hierfür entspricht die Abbildung 2.2.

Term	Assignments			Examinations		Final Grade
	Ass1	Ass2	Ass3	Midt.	Final	
1991						
Winter	85	80	75	60	75	75
Spring	80	65	75	60	70	70
Fall	80	85	75	55	80	75
1992						
Winter	85	80	70	70	75	75
Spring	80	80	70	70	75	75
Fall	75	70	65	60	80	70
mean	81	77	72	63	76	73
<i>student's grades Version 2.0</i>						

Abbildung 2.2: Terminologische Begriffe für eine Tabelle nach Wang [1]

Aus dieser Darstellung wird ersichtlich, wie die einzelnen Tabelleneinträge einer regelmäßigen Struktur unterliegen. Die Zellen lassen sich in horizontaler Richtung zu Zeilen und in vertikaler Richtung zu Spalten zusammenfassen. Die Tabelleninformationen befinden sich hierbei im Tabellenrumpf, während die Zusammenhänge über die Zeilenköpfe, beziehungsweise Spaltenköpfe definiert werden. Die regelmäßigen Strukturen können mitunter in den Überschriftenblöcken gebrochen werden, bedingt durch Verschachtelung der Kategorien und Untergruppierungen einzelner Überschriften, wie zum Beispiel die Einteilung

für 'Assignments' in 'Ass1', 'Ass2' und 'Ass3'. Der Tabellenfuß enthält meist Meta-Informationen zu den Daten der Tabelle und kann ebenfalls Brüche in der regelmäßigen Struktur aufweisen.

Je nachdem wie strukturierte Daten repräsentiert werden, variieren die einzelnen Gruppen einer Tabelle in ihrem Erscheinungsbild. Je mehr Varianten auftreten, desto komplexer und undurchsichtiger wird die Struktur für automatische Erkennungsprozesse. Auf welche unterschiedlichen Weisen diese auftreten können und wie im einzelnen diese Darstellungen aussehen, wird nun im folgenden aufgezeigt.

### 2.1.3 Tabellenarten

Grundsätzlich ist festzuhalten, dass tabellenförmige Strukturen auf nahezu unbegrenzte Weise in ihrer Form, ihrem Layout und ihrer Formatierung veränderbar sind. Klammert man formatspezifische Faktoren aus, dann resultieren die Unterschiede zum einen aus der Dimensionalität der Daten und zum anderen aus der Anordnung dieser innerhalb der Tabelle. Des Weiteren kann auch der eigentliche Zelleninhalt über die Form der Tabelle bestimmen. Denn dieser kann sowohl Text, als auch Bilder, Diagramme, Graphen und andere Tabellen beschreiben. Während der Mensch sehr gut in der Lage ist, Informationen aus komplexen Konstrukten mittels seiner kognitiven Fähigkeiten extrahieren zu können, bedarf es bei automatischen Prozessen einer aufwendigen Interpretation des Kontextes und der Gegebenheiten innerhalb der tabellarischen Struktur. Gestützt auf die Aussagen von Lopresti und Nagy [17], sowie Embley et al. [19], werden nun die in tabellarischen Strukturen vorkommenden Komplexitätsstufen aufgezeigt und deren Merkmale verdeutlicht.

Einfache Tabellen, wie in Abbildung 2.3, besitzen keine Unordnung in ihrer Struktur, was bedeutet, dass einzelne Tabellenelemente, wie Spalten, Zeilen und Zellen klar voneinander getrennt sind. Dies kann sowohl durch Linien, durch Trennungszeichen oder allein durch einen größeren Abstand zu Nachbarzellen gegeben sein. Ebenso setzt es voraus, dass keine Zellen miteinander verschmolzen

sind. Einzelne Spalten zeichnen sich durch ihre gleiche Ausrichtung, Schriftgröße, Schriftart, und ähnliches aus. Es existiert folglich kein formatspezifischer Bruch. Der Tabellenrumpf beschreibt zusammen mit Tabellenkopf und Tabellenfuß eine regelmäßige Matrix, worin einzelne Spalten durch eine Überschrift referenziert werden können. Tabellenkopf und Tabellenfuß enthalten als zusätzliche Bedingung maximal eine Zeile, ebenfalls die regelmäßige Spaltenstruktur aufgreifend.

Platz	Mannschaft	G	U	V	Tore	Diff.	Punkte
1	Bayern München	18	10	4	66:29	+37	64
2	FC Schalke 04	19	7	6	53:29	+24	64
3	Werder Bremen	16	9	7	68:39	+29	57
4	Bayer Leverkusen	15	12	5	63:36	+27	57
5	Borussia Dortmund	16	8	8	52:38	+14	56

Abbildung 2.3: Eine einfache Tabelle, wie hier die Top-5 Vereine aus der Ersten Bundesliga, beschreibt eine sehr regelmäßige Struktur ohne formatspezifischen Brüche

Nach [4] folgen aber nur wenige Tabellen diesem einfachen Schema. Des öfteren finden sich geschachtelte Überschriften, wie in Abbildung 2.4 zu sehen, welche die Regelmäßigkeit bei Tabellen verletzen. Grund hierfür ist die Zusammenfassung mehrerer Spalten in eine Überkategorie. Um diese bilden zu können, werden Zellen verschmolzen. Das Phänomen besteht aber ausschließlich im Tabellenkopf oder in den Zeilenüberschriften, da lediglich Spalten oder Zeilen logisch gruppiert werden müssen, was aber keinen Einfluss auf die Darstellung der Daten im Tabellenrumpf hat. Diese befinden sich nach wie vor in einer regelmäßigen und strukturierten Form. Betrachtet man den Tabellenfuß, so beinhaltet er meist nur Meta-Informationen über die Tabelle und muss die eigentliche Struktur nicht zwangsläufig übernehmen. Zanibbi konstatiert, dass weitere Informationen im Tabellentitel, in der Tabellenbeschreibung, in Fußnoten oder im Text, der auf die Tabelle verweist, zu finden sein können [18].

Über- schreitung in km/h	Regelsatz bei Begehung				Fahrverbot bei Begehung	
	innerhalb geschl. Ortschaften		außerhalb geschl. Ortschaften		innerhalb	außerhalb
	Euro	Punkte	Euro	Punkte	Monate	Monate
bis 10	15		10			
11-15	25		20			
16-20	35		30			
21-25	80	1	70	1		
26-30	100	3	80	3		
31-40	160	3	120	3	1	
41-50	200	4	160	3	1	1
51-60	280	4	240	4	2	1
61-70	480	4	440	4	3	2
über 70	680	4	600	4	3	3

Abbildung 2.4: Der aktuelle Bußgeldkatalog des Bundesverkehrsministeriums für Verkehr, Bau und Stadtentwicklung (<http://www.bmvbs.de/>) beschreibt, welche Tatbestände bei zu hoher Geschwindigkeit entstehen. Eine Schachtelung tritt in der Spalte 'Regelsatz bei Begehung' und 'Fahrverbot bei Begehung' auf, indem nach 'innerorts' und 'außerorts' separiert wird

In bestimmten Fällen kommt es vor, dass der erforderliche Platz für die Darstellung der Tabelle auf nur einer Seite nicht ausreicht und sie auf den nächsten Seiten fortgeführt werden muss. Dieses Phänomen tritt häufig in seitenorientierten Dokumentarten, wie Microsoft Office Word, Adobe Acrobat PDF oder in Papierform auf. Zum Beispiel die Wetterdaten in Abbildung 2.5. Hierfür beschreibt die Tabelle auf jeder fortgeführten Seite eine einheitliche und gleich bleibende Struktur, wobei meist der Tabellenkopf zur Benutzerorientierung wieder aufgegriffen wird. Der Tabellenfuß kann Anmerkungen zur Fortführung enthalten, oder eine Zwischensumme, welche hier lediglich als Meta-Informationen dient. Die im Tabellenrumpf befindlichen Daten aller Seiten bilden eine globale, fiktive Tabelle, die nicht unterbrochen ist.

Tag	Datum	Durch-	Temp.	Temp.	Niederschl.	Luftdruck
		Schnitt	Max	Min	in mm	in hPa
Do	01.04.10	4.1	5.9	2.1	0.4	1011.9
Fr	02.04.10	4.0	11.1	-2.3	0.0	1016.7
Sa	03.04.10	8.4	15.1	1.7	1.2	1012.2
So	04.04.10	8.5	11.6	6.3	0.0	1012.4
Mo	05.04.10	7.1	11.1	4.5	0.0	1024.3
Di	06.04.10	7.4	14.6	-0.3	0.0	1024.9
Mi	07.04.10	10.3	18.6	3.7	0.0	1018.9

Fortsetzung auf Seite 2

Tag	Datum	Durch-	Temp.	Temp.	Niederschl.	Luftdruck
		Schnitt	Max	Min	in mm	in hPa
Do	08.04.10	11.1	19.6	2.5	0.0	1020.5
Fr	09.04.10	10.1	14.2	6.5	0.0	1026.4
Sa	10.04.10	7.2	12.7	2.2	0.0	1025.6
So	11.04.10	4.1	9.2	0.2	0.4	1019.7
Mo	12.04.10	5.0	9.1	1.7	0.0	1015.5
Di	13.04.10	6.5	13.6	1.5	1.0	1013.0

Abbildung 2.5: Das Layout von mehrseitigen Tabellen verhält sich auf jeder Seite gleich. Der Tabellenkopf wird wiederholt und im Tabellenfuß können sich Meta-Informationen oder Anmerkungen zur Fortführung der Tabelle befinden

Komplexere Tabellen, wie Pivot-Tabellen oder Kreuztabellen in Abbildung 2.6, besitzen Strukturschwankungen auch im Tabellenrumpf, bedingt durch die hohe Anzahl an Dimensionen, die dargestellt werden. Die Schwankungen resultieren aus der Gruppierung der Daten, wodurch übergeordnete Zeilen entstehen. Diese enthalten Informationen, die in untergeordneten Zeilen erweitert oder verfeinert werden. Die jeweiligen Gruppen beschreiben untereinander eine geordnete Struktur. Bei dieser Tabellenform kann es zu Unterbrechungen durch Wiederholen des Tabellenkopfes oder Darstellungen von Zwischensummen kommen.

Summe von Betrag		
Kategorie	Unterkategorie	Summe
<b>Reise</b>	Bahn, Bus	28,00
	Flug	1088,90
	Taxi	12,00
		<b>1128,90</b>
<b>Unterkunft</b>	Hotel	822,50
		<b>822,50</b>
<b>Essen</b>	Restaurant	152,60
		<b>152,60</b>
<b>Gesamtergebnis</b>		<b>2104,00</b>

Abbildung 2.6: Eine komplexere Tabelle beschreibt eine fiktive Reisekostenabrechnung. Hierbei tauchen Kategorien und Unterkategorien auf, die nach ausgegebenen Betrag aufgelistet sind. In jeder Hauptkategorie werden die Zwischensummen dargestellt und die Gesamtsumme befindet sich im Tabellenfuß

Eine Erweiterung der Pivot-Tabellen führt zur Definition von rekursiven Tabellen (Vergleiche Abbildung 2.7). Hier können Zellen wiederum Tabellen enthalten,

welche die Informationen aus der übergeordneten Tabelle erweitern. Diese Form findet meist bei mehrdimensionalen Daten Verwendung, die in sich selbst eine hohe Komplexität aufweisen. Tabellen der gleichen Rekursionsstufe teilen sich häufig ihr Layout, um sie auch von menschlicher Seite aus deuten zu können.

Tabellen	können		
rekursiv sein	Tabellen	können	
	rekursiv sein	Tabellen	können
		rekursiv sein	Tabellen
		rekursiv sein	

Abbildung 2.7: Ein fiktives Beispiel einer rekursiven Tabelle

In seltenen Fällen beschreiben Tabellen eine vollkommen unregelmäßige Struktur [19], zum Beispiel wenn sie in Diagrammen eingesetzt werden oder nur für Anordnungszwecken Verwendung finden. In Abbildung 2.8 ist beispielsweise nicht mehr gegeben, dass Zeilen waagrecht und Spalten senkrecht verlaufen und auch nicht, dass benachbarte Zellen eine Beziehung untereinander haben. Eine geordnete Struktur kann zwar zu erkennen sein, bedarf aber eines großen Interpretationsaufwandes, sogar auf menschlicher Seite.

Eine der wohl komplexesten Tabellen beschreibt das Periodensystem der Elemente in Figur 2.9. Zum einen bestehen die Spalten nicht aus einer gleichen Anzahl von Zellen und zum anderen befinden sich in zwei Zellen (Lanthanoide, beziehungsweise Actinoide) Verweise auf eine andere Tabelle. Um diese interpretieren zu können, bedarf es eines umfangreichen gebietsspezifischen Vorwissens [19] und gilt somit als höchste Herausforderung für zukünftige Tabellenanwendungen.

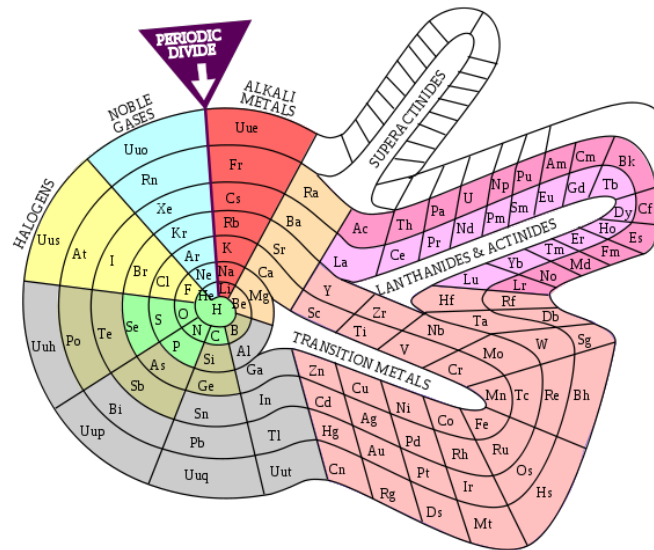


Abbildung 2.8: Die Periodentabelle wird hier abstrahiert auf eine Spiralforn. Somit entsteht eher ein Diagramm als eine Tabelle. Die zugrundeliegenden Daten sind aber strukturiert (aus 'The periodic spiral of Professor Thedor Benfey' in [20]).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18																																																																																																																																																																																																																																																																																																																																																																													
IA	IIA	IIIB	IVB	VB	VIB	VIB	VIB	VIB	IB	IB		IIIA	IVA	VA	VIA	VIA	VIA																																																																																																																																																																																																																																																																																																																																																																													
1 H Hydrogen 1.00794	2 He Helium 4.002602											3 Li Lithium 6.941	4 Be Beryllium 9.012182	5 B Bor 10.811	6 C Kohlenstoff 12.011	7 N Stickstoff 14.00643	8 O Sauerstoff 15.999	9 F Fluor 18.9984032	10 Ne Neon 20.1797																																																																																																																																																																																																																																																																																																																																																																											
3 Na Natrium 22.98976928	4 Mg Magnesium 24.304											11 Al Aluminium 26.9815386	12 Si Silicium 28.0855	13 P Phosphor 30.973762	14 S Schwefel 32.06	15 Cl Chlor 35.453	16 Ar Argon 39.948	17 K Kalium 39.0983	18 Ca Calcium 40.078																																																																																																																																																																																																																																																																																																																																																																											
19 K Kalium 39.0983	20 Ca Calcium 40.078	21 Sc Scandium 44.955912	22 Ti Titan 47.867	23 V Vanadium 50.9415	24 Cr Chrom 51.9961	25 Mn Mangan 54.938044	26 Fe Eisen 55.845	27 Co Kobalt 58.933195	28 Ni Nickel 58.6934	29 Cu Kupfer 63.546	30 Zn Zink 65.39	31 Ga Gallium 69.723	32 Ge Germanium 72.64	33 As Arsen 74.9216	34 Se Selen 78.96	35 Br Brom 79.904	36 Kr Krypton 83.799	37 Rb Rubidium 85.4678	38 Sr Strontium 87.62																																																																																																																																																																																																																																																																																																																																																																											
39 Y Yttrium 88.90584	40 Zr Zirkon 91.224	41 Nb Niob 92.90638	42 Mo Molybdän 95.94	43 Tc Technetium 98	44 Ru Ruthenium 101.07	45 Rh Rhenium 101.07	46 Pd Palladium 106.36	47 Ag Silber 107.8682	48 Cd Cadmium 112.411	49 In Indium 114.818	50 Sn Zinn 118.710	51 Sb Antimon 121.757	52 Te Tellur 127.6	53 I Jod 126.90547	54 Xe Xenon 131.29	55 Ba Baryum 137.327	56 La Lanthan 138.90547	57 Ce Cerium 140.12	58 Pr Praseodym 140.90766	59 Nd Neodym 144.24	60 Pm Promethium 144.9126	61 Sm Samarium 150.36	62 Eu Europium 151.964	63 Gd Gadolinium 157.25	64 Tb Terbium 158.92532	65 Dy Dysprosium 162.5003	66 Ho Holmium 164.93032	67 Er Erbium 167.259	68 Tm Thulium 168.9304	69 Yb Ytterbium 173.04	70 Lu Lutetium 174.967	71 Uuo Ununoktium 289																																																																																																																																																																																																																																																																																																																																																														
72 Hf Hafnium 178.49	73 Ta Tantal 180.9479	74 W Wolfram 183.84	75 Re Rhenium 186.207	76 Os Osmium 190.23	77 Ir Iridium 192.222	78 Pt Platin 195.078	79 Au Gold 196.966569	80 Hg Quecksilber 200.59	81 Tl Thallium 204.3833	82 Pb Blei 207.2	83 Bi Bismut 208.9804	84 Po Polonium 209	85 At Astat 210	86 Rn Radon 222	87 Fr Francium 223	88 Ra Radium 226	89 to 103 Lanthanoids	104 Rf Rutherfordium 261	105 Db Dubnium 262	106 Sg Seaborgium 263	107 Bh Bohrium 264	108 Hs Hassium 265	109 Mt Meitnerium 266	110 Ds Darmstadtium 271	111 Rg Roentgenium 272	112 Uub Ununbium 277	113 Uut Ununtrium 284	114 Uuq Ununquadium 285	115 Uup Ununpentium 286	116 Uuq Ununhexium 287	117 Uus Ununseptium 288	118 Uuo Ununoctium 289	119 to 110 Superactinoids	120 Uue Ununennium 289																																																																																																																																																																																																																																																																																																																																																												
89 Ac Actinium 227	90 Th Thorium 232.0377	91 Pa Protactinium 231.03688	92 U Uran 238.02891	93 Np Neptunium 237	94 Pu Plutonium 244	95 Am Americium 243	96 Cm Curium 247	97 Bk Berkelium 247	98 Cf Californium 251	99 Es Einsteinium 252	100 Fm Fermium 257	101 Md Mendelevium 258	102 No Nobelium 259	103 Lr Lawrencium 260	104 to 118 Actinoids	119 Uuh Ununheptium 285	120 Uuo Ununoctium 286	121 Uuq Ununquadium 287	122 Uup Ununpentium 288	123 Uuq Ununhexium 289	124 Uup Ununseptium 290	125 Uuq Ununoktium 291	126 Uup Ununennium 292	127 Uuq Ununheptium 293	128 Uuq Ununoktium 294	129 Uup Ununseptium 295	130 Uuq Ununoktium 296	131 Uup Ununseptium 297	132 Uuq Ununoktium 298	133 Uup Ununseptium 299	134 Uuq Ununoktium 300	135 Uup Ununseptium 301	136 Uuq Ununoktium 302	137 Uup Ununseptium 303	138 Uuq Ununoktium 304	139 Uup Ununseptium 305	140 Uuq Ununoktium 306	141 Uup Ununseptium 307	142 Uuq Ununoktium 308	143 Uup Ununseptium 309	144 Uuq Ununoktium 310	145 Uup Ununseptium 311	146 Uuq Ununoktium 312	147 Uup Ununseptium 313	148 Uuq Ununoktium 314	149 Uup Ununseptium 315	150 Uuq Ununoktium 316	151 Uup Ununseptium 317	152 Uuq Ununoktium 318	153 Uup Ununseptium 319	154 Uuq Ununoktium 320	155 Uup Ununseptium 321	156 Uuq Ununoktium 322	157 Uup Ununseptium 323	158 Uuq Ununoktium 324	159 Uup Ununseptium 325	160 Uuq Ununoktium 326	161 Uup Ununseptium 327	162 Uuq Ununoktium 328	163 Uup Ununseptium 329	164 Uuq Ununoktium 330	165 Uup Ununseptium 331	166 Uuq Ununoktium 332	167 Uup Ununseptium 333	168 Uuq Ununoktium 334	169 Uup Ununseptium 335	170 Uuq Ununoktium 336	171 Uup Ununseptium 337	172 Uuq Ununoktium 338	173 Uup Ununseptium 339	174 Uuq Ununoktium 340	175 Uup Ununseptium 341	176 Uuq Ununoktium 342	177 Uup Ununseptium 343	178 Uuq Ununoktium 344	179 Uup Ununseptium 345	180 Uuq Ununoktium 346	181 Uup Ununseptium 347	182 Uuq Ununoktium 348	183 Uup Ununseptium 349	184 Uuq Ununoktium 350	185 Uup Ununseptium 351	186 Uuq Ununoktium 352	187 Uup Ununseptium 353	188 Uuq Ununoktium 354	189 Uup Ununseptium 355	190 Uuq Ununoktium 356	191 Uup Ununseptium 357	192 Uuq Ununoktium 358	193 Uup Ununseptium 359	194 Uuq Ununoktium 360	195 Uup Ununseptium 361	196 Uuq Ununoktium 362	197 Uup Ununseptium 363	198 Uuq Ununoktium 364	199 Uup Ununseptium 365	200 Uuq Ununoktium 366	201 Uup Ununseptium 367	202 Uuq Ununoktium 368	203 Uup Ununseptium 369	204 Uuq Ununoktium 370	205 Uup Ununseptium 371	206 Uuq Ununoktium 372	207 Uup Ununseptium 373	208 Uuq Ununoktium 374	209 Uup Ununseptium 375	210 Uuq Ununoktium 376	211 Uup Ununseptium 377	212 Uuq Ununoktium 378	213 Uup Ununseptium 379	214 Uuq Ununoktium 380	215 Uup Ununseptium 381	216 Uuq Ununoktium 382	217 Uup Ununseptium 383	218 Uuq Ununoktium 384	219 Uup Ununseptium 385	220 Uuq Ununoktium 386	221 Uup Ununseptium 387	222 Uuq Ununoktium 388	223 Uup Ununseptium 389	224 Uuq Ununoktium 390	225 Uup Ununseptium 391	226 Uuq Ununoktium 392	227 Uup Ununseptium 393	228 Uuq Ununoktium 394	229 Uup Ununseptium 395	230 Uuq Ununoktium 396	231 Uup Ununseptium 397	232 Uuq Ununoktium 398	233 Uup Ununseptium 399	234 Uuq Ununoktium 400	235 Uup Ununseptium 401	236 Uuq Ununoktium 402	237 Uup Ununseptium 403	238 Uuq Ununoktium 404	239 Uup Ununseptium 405	240 Uuq Ununoktium 406	241 Uup Ununseptium 407	242 Uuq Ununoktium 408	243 Uup Ununseptium 409	244 Uuq Ununoktium 410	245 Uup Ununseptium 411	246 Uuq Ununoktium 412	247 Uup Ununseptium 413	248 Uuq Ununoktium 414	249 Uup Ununseptium 415	250 Uuq Ununoktium 416	251 Uup Ununseptium 417	252 Uuq Ununoktium 418	253 Uup Ununseptium 419	254 Uuq Ununoktium 420	255 Uup Ununseptium 421	256 Uuq Ununoktium 422	257 Uup Ununseptium 423	258 Uuq Ununoktium 424	259 Uup Ununseptium 425	260 Uuq Ununoktium 426	261 Uup Ununseptium 427	262 Uuq Ununoktium 428	263 Uup Ununseptium 429	264 Uuq Ununoktium 430	265 Uup Ununseptium 431	266 Uuq Ununoktium 432	267 Uup Ununseptium 433	268 Uuq Ununoktium 434	269 Uup Ununseptium 435	270 Uuq Ununoktium 436	271 Uup Ununseptium 437	272 Uuq Ununoktium 438	273 Uup Ununseptium 439	274 Uuq Ununoktium 440	275 Uup Ununseptium 441	276 Uuq Ununoktium 442	277 Uup Ununseptium 443	278 Uuq Ununoktium 444	279 Uup Ununseptium 445	280 Uuq Ununoktium 446	281 Uup Ununseptium 447	282 Uuq Ununoktium 448	283 Uup Ununseptium 449	284 Uuq Ununoktium 450	285 Uup Ununseptium 451	286 Uuq Ununoktium 452	287 Uup Ununseptium 453	288 Uuq Ununoktium 454	289 Uup Ununseptium 455	290 Uuq Ununoktium 456	291 Uup Ununseptium 457	292 Uuq Ununoktium 458	293 Uup Ununseptium 459	294 Uuq Ununoktium 460	295 Uup Ununseptium 461	296 Uuq Ununoktium 462	297 Uup Ununseptium 463	298 Uuq Ununoktium 464	299 Uup Ununseptium 465	300 Uuq Ununoktium 466	301 Uup Ununseptium 467	302 Uuq Ununoktium 468	303 Uup Ununseptium 469	304 Uuq Ununoktium 470	305 Uup Ununseptium 471	306 Uuq Ununoktium 472	307 Uup Ununseptium 473	308 Uuq Ununoktium 474	309 Uup Ununseptium 475	310 Uuq Ununoktium 476	311 Uup Ununseptium 477	312 Uuq Ununoktium 478	313 Uup Ununseptium 479	314 Uuq Ununoktium 480	315 Uup Ununseptium 481	316 Uuq Ununoktium 482	317 Uup Ununseptium 483	318 Uuq Ununoktium 484	319 Uup Ununseptium 485	320 Uuq Ununoktium 486	321 Uup Ununseptium 487	322 Uuq Ununoktium 488	323 Uup Ununseptium 489	324 Uuq Ununoktium 490	325 Uup Ununseptium 491	326 Uuq Ununoktium 492	327 Uup Ununseptium 493	328 Uuq Ununoktium 494	329 Uup Ununseptium 495	330 Uuq Ununoktium 496	331 Uup Ununseptium 497	332 Uuq Ununoktium 498	333 Uup Ununseptium 499	334 Uuq Ununoktium 500	335 Uup Ununseptium 501	336 Uuq Ununoktium 502	337 Uup Ununseptium 503	338 Uuq Ununoktium 504	339 Uup Ununseptium 505	340 Uuq Ununoktium 506	341 Uup Ununseptium 507	342 Uuq Ununoktium 508	343 Uup Ununseptium 509	344 Uuq Ununoktium 510	345 Uup Ununseptium 511	346 Uuq Ununoktium 512	347 Uup Ununseptium 513	348 Uuq Ununoktium 514	349 Uup Ununseptium 515	350 Uuq Ununoktium 516	351 Uup Ununseptium 517	352 Uuq Ununoktium 518	353 Uup Ununseptium 519	354 Uuq Ununoktium 520	355 Uup Ununseptium 521	356 Uuq Ununoktium 522	357 Uup Ununseptium 523	358 Uuq Ununoktium 524	359 Uup Ununseptium 525	360 Uuq Ununoktium 526	361 Uup Ununseptium 527	362 Uuq Ununoktium 528	363 Uup Ununseptium 529	364 Uuq Ununoktium 530	365 Uup Ununseptium 531	366 Uuq Ununoktium 532	367 Uup Ununseptium 533	368 Uuq Ununoktium 534	369 Uup Ununseptium 535	370 Uuq Ununoktium 536	371 Uup Ununseptium 537	372 Uuq Ununoktium 538	373 Uup Ununseptium 539	374 Uuq Ununoktium 540	375 Uup Ununseptium 541	376 Uuq Ununoktium 542	377 Uup Ununseptium 543	378 Uuq Ununoktium 544	379 Uup Ununseptium 545	380 Uuq Ununoktium 546	381 Uup Ununseptium 547	382 Uuq Ununoktium 548	383 Uup Ununseptium 549	384 Uuq Ununoktium 550	385 Uup Ununseptium 551	386 Uuq Ununoktium 552	387 Uup Ununseptium 553	388 Uuq Ununoktium 554	389 Uup Ununseptium 555	390 Uuq Ununoktium 556	391 Uup Ununseptium 557	392 Uuq Ununoktium 558	393 Uup Ununseptium 559	394 Uuq Ununoktium 560	395 Uup Ununseptium 561	396 Uuq Ununoktium 562	397 Uup Ununseptium 563	398 Uuq Ununoktium 564	399 Uup Ununseptium 565	400 Uuq Ununoktium 566	401 Uup Ununseptium 567	402 Uuq Ununoktium 568	403 Uup Ununseptium 569	404 Uuq Ununoktium 570	405 Uup Ununseptium 571	406 Uuq Ununoktium 572	407 Uup Ununseptium 573	408 Uuq Ununoktium 574	409 Uup Ununseptium 575	410 Uuq Ununoktium 576	411 Uup Ununseptium 577	412 Uuq Ununoktium 578	413 Uup Ununseptium 579	414 Uuq Ununoktium 580	415 Uup Ununseptium 581	416 Uuq Ununoktium 582	417 Uup Ununseptium 583	418 Uuq Ununoktium 584	419 Uup Ununseptium 585	420 Uuq Ununoktium 586	421 Uup Ununseptium 587	422 Uuq Ununoktium 588	423 Uup Ununseptium 589	424 Uuq Ununoktium 590	425 Uup Ununseptium 591	426 Uuq Ununoktium 592	427 Uup Ununseptium 593	428 Uuq Ununoktium 594	429 Uup Ununseptium 595	430 Uuq Ununoktium 596	431 Uup Ununseptium 597	432 Uuq Ununoktium 598	433 Uup Ununseptium 599	434 Uuq Ununoktium 600	435 Uup Ununseptium 601	436 Uuq Ununoktium 602	437 Uup Ununseptium 603	438 Uuq Ununoktium 604	439 Uup Ununseptium 605	440 Uuq Ununoktium 606	441 Uup Ununseptium 607	442 Uuq Ununoktium 608	443 Uup Ununseptium 609	444 Uuq Ununoktium 610	445 Uup Ununseptium 611	446 Uuq Ununoktium 612	447 Uup Ununseptium 613	448 Uuq Ununoktium 614	449 Uup Ununseptium 615	450 Uuq Ununoktium 616	451 Uup Ununseptium 617	452 Uuq Ununoktium 618	453 Uup Ununseptium 619	454 Uuq Ununoktium 620	455 Uup Ununseptium 621	456 Uuq Ununoktium 622	457 Uup Ununseptium 623	458 Uuq Ununoktium 624	459 Uup Ununseptium 625	460 Uuq Ununoktium 626	461 Uup Ununseptium 627	462 Uuq Ununoktium 628	463 Uup Ununseptium 629	464 Uuq Ununoktium 630	465 Uup Ununseptium 631	466 Uuq Ununoktium 632	467 Uup Ununseptium 633	468 Uuq Ununoktium 634	469 Uup Ununseptium 635	470 Uuq Ununoktium 636	471 Uup Ununseptium 637	472 Uuq Ununoktium 638	473 Uup Ununseptium 639	474 Uuq Ununoktium 640	475 Uup Ununseptium 641	476 Uuq Ununoktium 642	477 Uup Ununseptium 643	478 Uuq Ununoktium 644	479 Uup Ununseptium 645	480 Uuq Ununoktium 646	481 Uup Ununseptium 647	482 Uuq Ununoktium 648	483 Uup Ununseptium 649	484 Uuq Ununoktium 650	485 Uup

### 2.1.4 Tabellenformate

Unter den vielen verschiedenen Dateiformaten befindet sich ein großer Anteil an Dokumententypen, in denen Tabellenstrukturen repräsentiert werden können. In der Literatur sind hierbei mehrere Einteilungen in Dokumentengruppen zu finden, die jeweils unterschiedliche Gesichtspunkte in den Vordergrund stellen. Embley et al. legen zugrunde, in welchem Medium Tabellen präsentiert werden [19]. Sie unterteilen hierfür in elektronische Medien und Medien in Papierform, wobei die erstgenannte Gruppe zusätzlich nach Kodierungs-Schemata getrennt wird. Es resultieren folglich drei Gruppen:

- ASCII kodierte Dateien enthalten nur einen reinen, linguistischen Inhalt. Abstände müssen hier auf Zeichenebene modelliert werden.
- Seiten beschreibende Dateien (Page-descriptor-Dateien) enthalten ebenfalls einen linguistischen Inhalt, besitzen aber auch Formatierungsoptionen und eigene Kodierungsformen.
- Bitmapdateien enthalten Bilder von Tabellen.

Im Unterschied hierzu steht die Interpretation von Lopresti und Nagy aus [17], welche Dokumente nach ihrem Repräsentationslevel unterteilt. Das unterste Level beinhaltet Dokumentarten, in denen Tabellen nur dargestellt werden können. Im mittleren Level werden alle editierbaren Dokumente zusammengefasst und das oberste Level sammelt Dokumente, die abgefragt werden können. Je nach Level unterscheiden die Autoren nach einer morphologischen, syntaktischen oder semantischen Verwendung der Dokumentarten.

Fasst man beide Aussagen zusammen, so entstehen vier verschiedene Dokumentengruppen, jeweils unterscheidbar durch deren Darstellungs- und Veränderungsgrad. Im weiteren wird nun aufgezeigt, welche Dateiformate repräsentativ für die jeweilige Gruppe stehen und wie diese im einzelnen Tabellen kodieren.

### **Textformate**

Tabellen in Textdateien oder E-Mails bestehen aus einzelnen Zeichen, werden also auch durch diese strukturiert. Das Zeichen 'CR' (carriage return) ist meist Indiz für eine neue Zeile und durch Einrücken mit Tabulator-Zeichen oder Auffüllen mit Leerzeichen wird die Spaltenstruktur modelliert. In bestimmten Fällen, zum Beispiel in CSV-Dateien<sup>4</sup>, werden Spaltentrenner ohne Einrückung mit nur einem Zeichen ausgedrückt. Ebenso kann es vorkommen, dass die Rahmen von Tabelle, Zeile, Spalte oder Zelle als Schriftzeichen dargestellt werden. Da textbasierte Formate eine der am längsten existierenden Formate sind, finden sich viele wissenschaftliche Auseinandersetzungen in Anwendungen und Publikationen [10] [11], die schon merklich gute Ergebnisse in Hinsicht auf die Extrahierung der Tabelleninhalte liefern.

### **Page-descriptor-Formate**

Zu den Page-descriptor-Formaten gehören unter anderem Mark-up-Sprachen, wie HTML, SGML, XML oder  $\LaTeX$ , Formate aus den Office-Paketen, wie Word, Excel oder PowerPoint und andere eigenständige Formate, wie RTF oder PDF.

Mark-up-Sprachen besitzen spezielle Konventionen wie Tabellen definiert werden sollen. Dies wird durch eine spezifische Grammatik gewährleistet. Die Tabellenobjekte werden durch verschiedene 'Tags' beschrieben, die dann bei HTML zum Beispiel vom Browser interpretiert und dargestellt werden. Nach [19] kann man nicht völlig sicher sein, dass diese 'Tags' nicht missbraucht werden, da sie wie in [21] beschrieben, auch als Layout in Webseiten und nicht als Datenrepräsentation verwendet werden können. Nach [4] verfügt XHTML [22] über die besten Möglichkeiten, auch komplexe Tabellen repräsentieren zu können. Hier existieren Kennzeichnungen für den Tabellenkopf, den Tabellenrumpf und den Tabellenfuß. Es können Zellen über bestimmte Bereiche miteinander verschmolzen werden und der Strukturbaum kann leicht erzeugt werden. Deswegen wird

---

<sup>4</sup>CSV: Comma Separated Values

dieses Format auch häufig in Anwendungen, wie zum Beispiel [9], als Eingabe-, sowie als Ausgabeformat verwendet. Zusätzlich können die einzelnen Objekte der Tabelle mit Formatierungseigenschaften versehen werden.

Anwendungen aus der Office Familie, wie Word oder Excel, verwenden ein internes Tabellenmodell auf Objektbasis, worin zahlreiche Formatierungsoptionen möglich sind. Innerhalb der Pakete sind verschiedene Konvertierungsroutinen für Tabellen implementiert, welche in der Arbeit von Embley et al. [23] näher beschrieben werden. Es können mit wenig ersichtlichem Verlust an Informationen zum Beispiel HTML-Tabellen in Word eingefügt werden, Word-Tabellen in das Excel-Format konvertiert werden oder Excel-Tabellen in ein Datenbankformat geschrieben werden. Das objektbasierte System ist mittels der COM-Interop-Schnittstelle [24] ansprechbar. Diese bietet einen kompletten Zugriff auf die im Dokument befindlichen Inhalte und deren Manipulation, wodurch eine Tabelle schnell erzeugt oder extrahiert werden kann. Das neue Office-Format<sup>5</sup> beschreibt ein XML basiertes Mark-up-Format und dient der Interoperabilität und einem verbesserten Dateiaustausch. Das interne Tabellenmodell unterstützt, ähnlich wie bei XHTML, ein Verschmelzen von Zellen und die Kennzeichnung von Zeilen als Überschriften. Die genauere Analyse des Tabellenobjekts und dessen Verhalten gibt Aufschluss über die interne Modellierung der Objekte. Die Tabelle wird als einheitliche Matrix aufgefasst, wobei diese Struktur verloren geht, sobald Zellen miteinander verschmolzen werden. Geht zum Beispiel eine Zelle über mehrere Spalten, besteht nicht mehr die Möglichkeit, durch die Zellen einer Spalte zu iterieren. Es lässt sich also vermuten, dass hier eine Zeile als Array von Zellen und Spalten als Array von Zellen gespeichert werden. Um die Konsistenz zu bewahren, werden im Open XML Format so genannte virtuelle Zellen eingefügt, die eine regelmäßige Struktur wieder gewährleisten können. Des Weiteren werden in den Applikationen Word und Excel die Zeilen- beziehungsweise Spaltenanzahlen technisch bedingt limitiert<sup>6</sup>.

PDF-Dokumente ermöglichen es, elektronische Dokumente leicht und zuverlässig

<sup>5</sup>ECMA TC45 - Office Open XML Formats

<sup>6</sup>Word: 64 Spalten, 32.768 Zeilen; Excel: 16.384 Spalten, 1.048.576 Zeilen

sig plattformunabhängig zu betrachten und auszutauschen [25]. Dokumente werden durch low-level-Objekte, wie Textzeichen, Pfade oder Bilder ausgedrückt, die dann von einem Anzeigegerät zur Darstellung gerendert werden. Über Bibliotheken, wie zum Beispiel xpdf [26], können die Instruktionen aus den Dokumenten extrahiert werden. Aufbauend auf diesen Informationen muss die Tabellenstruktur somit interpretiert werden. Beispielfhaft sind hierfür die Anwendungen in [27], [3] und [28].

### **Datenbankformate**

Relationale Datenbanken beschreiben Tabellen über Relationen, welche durch mathematische Definitionen beschrieben werden. Deren Struktur ist wohldefiniert und obliegt somit einer regelmäßigen Natur. Komplexe Strukturen, wie geschachtelte Überschriften, das Verschmelzen von Zellen oder Vergeben von Formatierungen sind nicht erlaubt. Die Tabellenstruktur ist schon beim Erzeugen der Tabelle bekannt und kann auf einfache Weise abgefragt werden. Die Anwendung phpMyAdmin<sup>7</sup> zum Beispiel dient zur Administration von MySQL-Datenbanken und übersetzt die Benutzer-Anfragen in eine HTML-Tabelle, welche dann im Browser dargestellt wird.

### **Bildformate**

Bilder besitzen lediglich eine darstellende Funktion, je nach Format können sie entweder durch Bildpunkte (BMP, JPEG, GIF) oder durch Vektoren (SVG, EPS) realisiert werden, wobei Vektorbilder im gerasterten Zustand wieder Pixelbildern entsprechen. Sie bestehen demnach aus Pixeln an bestimmten Positionen, versehen mit Farbinformationen oder Grauwerten. Diese entstehen entweder durch Computer generierte Prozesse oder einscannen von Papierdokumenten, wobei letztere mit Verdrehungen, Verzerrungen und Rauschen behaftet sein kann. Um eine Tabelle hieraus erkennen zu können, bedarf es einer Interpretation der Pixeldaten. Mittels verschiedener OCR-Verfahren können somit

---

<sup>7</sup><http://www.phpmyadmin.net/>

Linien, Weißräume, Zeichen oder Blöcke extrahiert werden, worauf die Tabellenstrukturerkennung aufsetzen kann.

Nachdem gezeigt wurde, was eine Tabelle definiert, welche grundlegenden Begriffe für eine Tabelle gelten, wie Tabellen unterschiedlich formuliert werden können und wie strukturierte Daten in verschiedenen Dokumentarten repräsentiert werden, kann nun im weiteren die Themenstellung dieser Ausarbeitung erörtert werden.

## Kapitel 3

# Problemstellung

Die Themenstellung dieser Arbeit entstand innerhalb der Firma Open Text Document Technologies GmbH aus dem Wunsch heraus, ein Tabellenmodell zu entwickeln. Sie beschäftigt sich mit der intelligenten Texterkennung, also Inhalte aus eingescannten Formularen oder Rechnungen zu erkennen und in Text umzuwandeln. Mittels der Dokumentenanalyse ist es möglich, die Daten zu interpretieren, um zum Beispiel den Datensatz aus einer Rechnung zu extrahieren und ihn mittels eines Workflowsystems automatisch in einen Geschäftsprozess einzubetten. Die Fragestellung ist für sie besonders relevant, da ein großer Teil ihrer verwendeten Dokumente Rechnungen sind, die nahezu immer Tabellen enthalten. Algorithmen, wie Linienerkennungen, Blockbildungen oder Interpretationen der Tabellenpositionen finden schon Verwendung in ihren kommerziellen Systemen. Es fehlt lediglich ein Tabellenmodell, welches darin eingebettet werden kann.

Im folgenden Kapitel wird nun erläutert, welche Anforderungen an das Tabellenmodell gestellt werden und welche Folgerungen man hierzu aus dem vorgestellten Vorwissen aus Kapitel 1.2 zur Bearbeitung der Themenstellung ziehen kann.

## 3.1 Anforderungen

Die Anforderungen an das Tabellenmodell wurden in mehreren Diskussionsrunden mit ausgewählten Mitarbeitern der Firma Open Text Document Technologies GmbH aufgestellt. Unterteilt sind sie hierbei in Anforderungen an die Tabellenstruktur, an das Tabellenlayout und generische, sowie schemabezogene Anforderungen.

### 3.1.1 Tabellenstruktur

Die Tabellenstruktur soll durch Objekte wie Spalten, Zeilen und Zellen modelliert werden, da diese die gängigste Repräsentationsform darstellt. Hierbei werden Zellen als Träger der Tabellendaten beschrieben, welche aber auch im Modell über die Spalten und Zeilen erreichbar sein sollen. Zelleninhalte beschränken sich lediglich auf Textdaten. Zusätzlich sollen Beziehungen zwischen benachbarten Tabellenobjekten erhalten bleiben, um zum Beispiel durch die Zellen innerhalb der Zeile iterieren zu können. Der logische Bezug der Tabellen, also, in welche Kategorien die Daten eingeteilt sind, ist hier unwichtig. Vielmehr soll die Tabelle als sichtbares Objekt modelliert werden.

### 3.1.2 Tabellenlayout

Das Layout der Tabelle, also wie die Daten präsentiert werden, ist ebenso wichtig für die Tabellenrepräsentation, wie deren Struktur. Denn dort sind Zusatzinformationen enthalten, die in einem späteren Interpretationsschritt hinzugezogen werden können. Hierzu gehören die Unterteilung in Tabellenkopf, Tabellenrumpf und Tabellenfuß und das Einbeziehen von verschiedenen Formatierungselementen, wie Hintergrundfarbe, Ausrichtung, Schriftart oder Rahmeneigenschaften der Tabellenobjekte.

### 3.1.3 Generik

Betrachtet man pixelbasierte Eingabedokumente, bestehen zu Anfang nur Informationen über einzelne Pixel. Es liegt kein Anwendungswissen vor, ob eine Tabelle im Dokument existiert und wie diese aufgebaut ist. Es bedarf somit einer Interpretation der Pixelinformationen, um diese beschreiben zu können. Da diese Entscheidungen häufig nicht mit völliger Sicherheit getroffen werden können, sollen deren Vertrauenswerte mit in das Modell einfließen, denn spätere Interpretationen über die Beschaffenheit der Tabelle können die aktuell modellierte Struktur verändern. Es können somit falsche oder unsichere Entscheidungen erkannt und zu einem späteren Zeitpunkt, wenn mehr Informationen verfügbar sind, sogar behoben werden. Das Tabellenmodell soll also flexibel genug sein, um die Tabellenstruktur schrittweise generisch verändern zu können.

### 3.1.4 Schema

Das Schema des Tabellenmodells soll in Design und Benutzbarkeit einfach gehalten werden. Das bedeutet, dass bewusst auf die Modellierung komplexer Tabellen verzichtet werden kann, um das Modell weitestgehend mit simplen Methoden bedienen zu können. Im Vordergrund steht also die Modellierung der Tabelle als sichtbares Objekt. Es soll die Struktur und das Layout der Tabelle und nicht die Logik der Tabellendaten repräsentiert werden. Das Befüllen des Modells und die Interpretation der Inhalte finden außerhalb des eigentlichen Modells statt und sollen, wenn möglich, die schon verwendeten Methoden innerhalb der Firma unterstützen.

## 3.2 Folgerungen

Die Anforderungen aus Kapitel 3.1 sollen nun in den thematischen Kontext dieser Arbeit eingebettet werden. Hierfür steht zunächst die Tabelle im Vordergrund. Es bedarf einer Erörterung, welche Komplexitätsstufen im Modell realisiert werden sollen. Entscheidungsträger sind hier zum einen die Forderung

nach einem einfachen Modell und zum anderen das Vorkommen der verschiedenen Tabellenformen innerhalb der Dokumente. Nach Aussagen aus [1] beschreiben die häufigsten Tabellen eine regelmäßige und einfach gehaltene Struktur. Kreuztabellen oder hierarchisch organisierte Tabellen werden deshalb im Tabellenmodell nicht unterstützt. Zur Vollständigkeit wird die Problematik mit komplexen Tabellen in Kapitel 5.2 aufgegriffen und Lösungsvorschläge hierfür aufgezeigt. Zusätzlich werden mehrseitige Tabellen modellierbar sein.

Um Tabellen aus verschiedenen Dokumententypen repräsentieren zu können, bedarf es zuvor einer Analyse der Informationen, die in den unterschiedlichen Formaten zur Modellierung der Tabelle beitragen. Kapitel 2.1.4 gab hierzu schon einen umfangreichen Einblick, der nun mit den Anforderungen verknüpft wird. Es gilt also zu klären, auf welchen Informationen das Tabellenmodell aufbaut:

- Textformate enthalten keine Formatierungsoptionen und die Tabellenstrukturierung wird oftmals durch Blockbildung, wie in [11], erreicht.
- Aus Page-descriptor-Formaten kann die Tabellenstruktur häufig durch eigens entworfene Extraktionsroutinen entnommen werden. Innerhalb von PDF-Dokumenten setzen viele Arbeiten auf die Blockbildung oder interpretieren Linieninformationen.
- Der Bezug der Tabellenstrukturinformationen aus Datenbankformaten ist trivial, da die Tabellen flach strukturiert sind und deren Schema direkt entnommen werden kann.
- Um Tabellen aus Bildern extrahieren zu können, müssen verschiedene Informationen gesammelt und interpretiert werden. Mögliche Ansätze verwenden Linieninformationen oder bilden Blöcke, indem sie Abstände einzelner Objekte analysieren.

Betrachtet man die Blockbildung als Verfahren, das unsichtbare Linien als Informationsträger nutzt, so kann man dieses auch als Linienverfahren interpretieren. Eine Linie, ob sichtbar oder unsichtbar, entspricht somit einer physikali-

schen Trennung zwischen zwei Objekten. In einer Tabelle können diese Trennungen zwischen Spalten, Zeilen, und somit auch Zellen, auftreten. Ist bereits die Struktur einer Tabelle bekannt, kann diese mittels sichtbaren oder unsichtbaren Trennern modelliert werden. Unter dieser Voraussetzung ist es möglich, einfach strukturierte Tabellen aus den verschiedensten Dokumentarten, nur anhand ihrer Linieninformationen zu repräsentieren.

Wie das Tabellenmodell nun genau aufgebaut ist, welche Objekte definiert werden müssen und wie diese im einzelnen zusammenspielen, zeigt das folgende Kapitel. Hier werden die Aspekte zur Entwicklung des Tabellenmodells beschrieben und es wird zusätzlich erörtert, welche Komponenten es geben muss, um das Modell in einer Anwendung integrieren zu können.

# Kapitel 4

## Entwicklung

In diesem Kapitel wird nun die Entwicklung des Tabellenmodells vorgestellt. Zunächst wird das Modell alleinstehend betrachtet und einzelne Komponenten werden vorgestellt. Danach folgt die Einordnung in einen Tabellenverarbeitungsprozess in welchem das Modell eingebettet werden kann und zuletzt wird die Simulationsoberfläche beschrieben.

### 4.1 Tabellenmodell

Die Entwicklung des Tabellenmodells stellt das zentrale Kerngebiet dieser Arbeit dar und das Modell wurde selbstständig vom Autor konzipiert und implementiert. Das Modell baut auf den in Kapitel 3.1 erwähnten Anforderungen auf und beschreibt mit welchen Funktionen eine generische Modellierung von Tabellen ohne Anwendungswissen möglich ist. Hierbei wird besonders der Aspekt aus 2.1.1, dass Tabellen Träger von gleichartigen Informationen sind, hervorgehoben.

#### 4.1.1 Überblick

Die Position und Lage der Tabelle innerhalb des Dokuments wird durch ein Parallelogramm beschrieben. Es besteht aus einem Ursprungspunkt und zwei

Richtungsvektoren, wodurch es möglich ist, auch gedrehte Tabellen modellieren zu können. Dies ist wichtig, da eingescannte Dokumente verdreht sein können, diese Feststellung aber erst zu einem späteren Zeitpunkt getroffen wird. Oder, dass verdrehte Dokumente mittels digitalen Signaturen zur Bewahrung ihrer Echtheit geschützt werden, folglich also nicht mehr verändert werden dürfen. Als Maßeinheit wird der zehnte Teil eines Millimeters verwendet. Die Struktur der Tabelle wird durch vertikale Spaltentrenner und horizontale Zeilentrenner definiert. Diese durchlaufen die gesamte Fläche des Parallelogramms, jeweils parallel zu den aufspannenden Richtungsvektoren. Diese Trenner werden nur durch ihren Abstand vom Ursprung bestimmt. Abbildung 4.1 veranschaulicht diesen Sachverhalt.

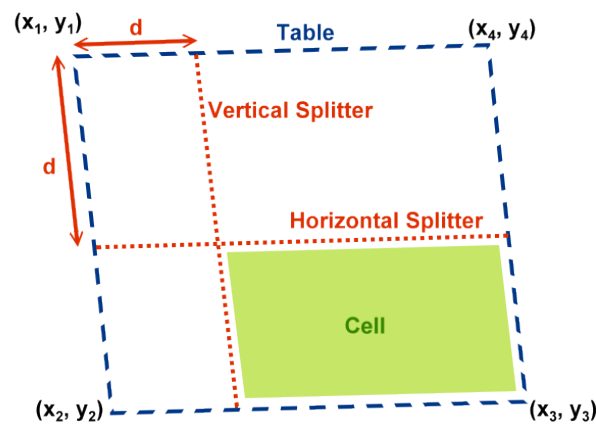


Abbildung 4.1: Übersicht über das Tabellenmodell

Aus der Konfiguration aller Trenner im Modell resultiert die Zeilen-, Spalten- und Zellenstruktur der Tabelle. Sie wird hierdurch im Modell simuliert und kann jederzeit abgerufen werden. Sie ist statisch und auf dem Modell aufgesetzt, wird also bei jeder Veränderung am Modell neu berechnet. Um die Flexibilität des Tabellenmodells zu gewährleisten, besitzt jedes Trennerobjekt eine Vertrauenseigenschaft im Wertebereich zwischen 0 und 100. Über einen minimalen Vertrauenswert der Tabelle kann bestimmt werden, welche Trenner die Struktur simulieren sollen. Liegt der Vertrauenswert unter dem minimalen Vertrauenswert der Tabelle, wird der Trenner deaktiviert und für den Aufbau der Tabellenstruk-

tur ignoriert, wodurch keine Objekte aus dem Modell gelöscht werden müssen. Zusätzlich kann den Trennerobjekten eine Linienart, sichtbar oder unsichtbar, zugeordnet werden. Mehrseitige Tabellen können zu einem Multi-Page-Objekt zusammengefasst werden. Dieses beinhaltet die vertikalen Spaltentrenner, da verwandte Tabellen die gleiche Spaltenstruktur besitzen. Die horizontalen Zeilentrenner werden in den seitenspezifischen Tabellen modelliert, können also unterschiedlich ausfallen. Die Dimensionen des Parallelogramms müssen hierzu aufgeteilt werden. Das Multi-Page-Objekt bestimmt die Breite und die seitenspezifische Tabelle bestimmt jeweils ihre Höhe. Abbildung 4.2 beschreibt diesen Prozess anhand eines Beispiels.

Wir bedanken uns für Ihren Auftrag und stellen Ihnen folgende Positionen in Rechnung:

Pos.	Menge	Artikelbezeichnung	Bestellnummer	Einheit	Einzelpreis	Gesamtpreis
10	2	Actuation	4500017278	Stk.	10.00 €	20.00 €
15	3	Slug for Shaft	4500017278	Stk.	47.50 €	142.50 €
20	2	Ball Bearing	4500017278	Stk.	25.00 €	50.00 €
25	3	Electronic	4500017278	Stk.	4.00 €	12.00 €
30	1	Bearing Case	4500017278	Stk.	20.00 €	20.00 €
35	2	Actuation	4500017234	Stk.	10.00 €	20.00 €
40	3	Slug for Shaft	4500017234	Stk.	47.50 €	142.50 €
45	2	Ball Bearing	4500017234	Stk.	25.00 €	50.00 €
50	3	Electronic	4500017234	Stk.	4.00 €	12.00 €
55	1	Bearing Case	4500017234	Stk.	20.00 €	20.00 €

**Seite 1:**

Pos.	Menge	Artikelbezeichnung	Bestellnummer	Einheit	Einzelpreis	Gesamtpreis
60	2	Actuation	4500017223	Stk.	10.00 €	20.00 €
65	3	Slug for Shaft	4500017223	Stk.	47.50 €	142.50 €
70	2	Ball Bearing	4500017223	Stk.	25.00 €	50.00 €
75	3	Electronic	4500017223	Stk.	4.00 €	12.00 €
80	1	Bearing Case	4500017223	Stk.	20.00 €	20.00 €

**Seite 2:**

Abbildung 4.2: Das Beispiel zeigt eine mehrseitige Tabelle aus einer realen Rechnung. Die gleiche Struktur, sowie die gleiche Breite sind auf beiden Seiten klar ersichtlich

#### 4.1.2 Programmaufbau

Das Tabellenmodell befindet sich in einer Microsoft .NET 3.5 Umgebung und besteht aus eigens entwickelten C#-Klassen, welche die vorgestellten Funktionen in vollem Umfang unterstützen. Um das Modell und deren Daten konsistent ablegen zu können, ist es möglich, dieses in eine XML-Struktur zu übersetzen. Hierfür wird die im .NET Framework enthaltene Objekt-Serialisierungs-

Klasse [29] verwendet. In diesem Kapitel sollen nun einzelne Komponenten des Objektmodells vorgestellt und analysiert werden. Dieses kann in zwei Bereiche eingeteilt werden. Der erste Teil beleuchtet die Strukturgenerierung über die Zeilen- und Spaltentrenner, während der zweite Teil die simulierte Tabellenstruktur beschreibt.

Zentrales Objekt im Modell (vergleiche Abbildung 4.3) ist die *MultiPageTable*, welche für jede Seite ein *Table*-Objekt enthält. Über die *INotifyPropertyChanged*-Schnittstelle können Anwendungen, die das Tabellenmodell verwenden, auf Änderungen im Modell aufmerksam gemacht werden. Die *MultiPageTable* besitzt eine *SplitterCollection*, welche die vertikalen Trenner modelliert, während *Table* über eine *SplitterCollection* die horizontalen Trenner definiert. Die *ISplitterOperations*-Schnittstelle unterstützt alle gängigen Methoden auf die *SplitterCollection*, wie zum Beispiel das Hinzufügen neuer Trenner, das Deaktivieren oder Aktivieren bestimmter Trenner, sowie das Entfernen von Trennern. Demnach besitzt das einzelne *Splitter*-Objekt Eigenschaften zu Status (*State*) und Trennerart (*SplitterType*), sowie einen Distanz- (*Distance*) und Vertrauenswert (*Confidence*). Die *SplitterCollection*-Klasse beschreibt eine sortierte Liste, wobei das Sortierkriterium durch die Abstände der Trenner gegeben ist. Die in der *Table*-Klasse befindliche *Parallelogram*-Eigenschaft beschreibt die Geometrie der Tabelle, welche im Tabellenmodell zusätzlich mit der Splitterkonfiguration verwurzelt ist. Um sicher zu gehen, dass die im Modell definierten Trenner zu jeder Zeit innerhalb der Tabellengeometrie liegen, müssen die Veränderungen überwacht werden, um gegebenenfalls die Eigenschaften der Trenner oder der Geometrie verändern zu können. Werden die Vektoren des Parallelogramms vergrößert, müssen die Trenner an ihrer Position stehen bleiben. Da diese durch den Abstand des Ursprungspunktes gegeben ist, muss dieser demnach angepasst werden, sobald sich der Ursprungspunkt beim Vergrößern verschiebt. Dies geschieht ebenfalls beim Verkleinern der Vektoren. Hier kann es aber zusätzlich vorkommen, dass die Parallelogrammgrenze über Trenner hinweg verläuft, wo-

mit diese dann außerhalb der Geometrie liegen würden. Ist dies der Fall, wird die Parallelogrammgrenze mit dem ersten passierten Splitter getauscht. Dieser Tausch findet ebenfalls statt, wenn die Distanz eines Trenners negativ, oder größer als der jeweilige Vektorbetrag wird.

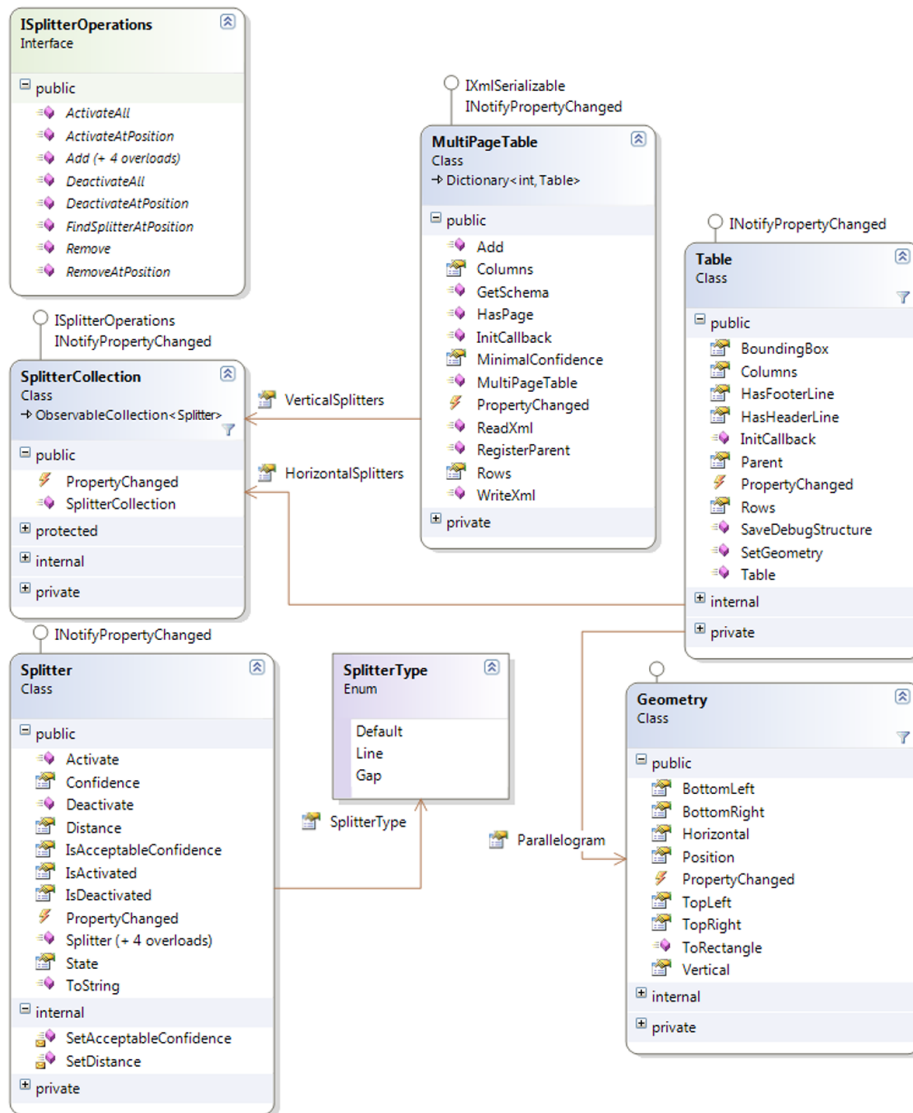


Abbildung 4.3: Klassendiagramm der Trennerkonfiguration

Die simulierte Tabellenstruktur, wie sie in Abbildung 4.4 abgebildet ist, wird im Modell über die Klassen *Row*, *Column* und *Cell* modelliert. Alle Objekte sind von außen nicht veränderbar, die Struktur kann also nur abgefragt werden. Er-

gibt sich eine Veränderung der Trenner-Konfiguration, wird die Struktur intern neu berechnet.

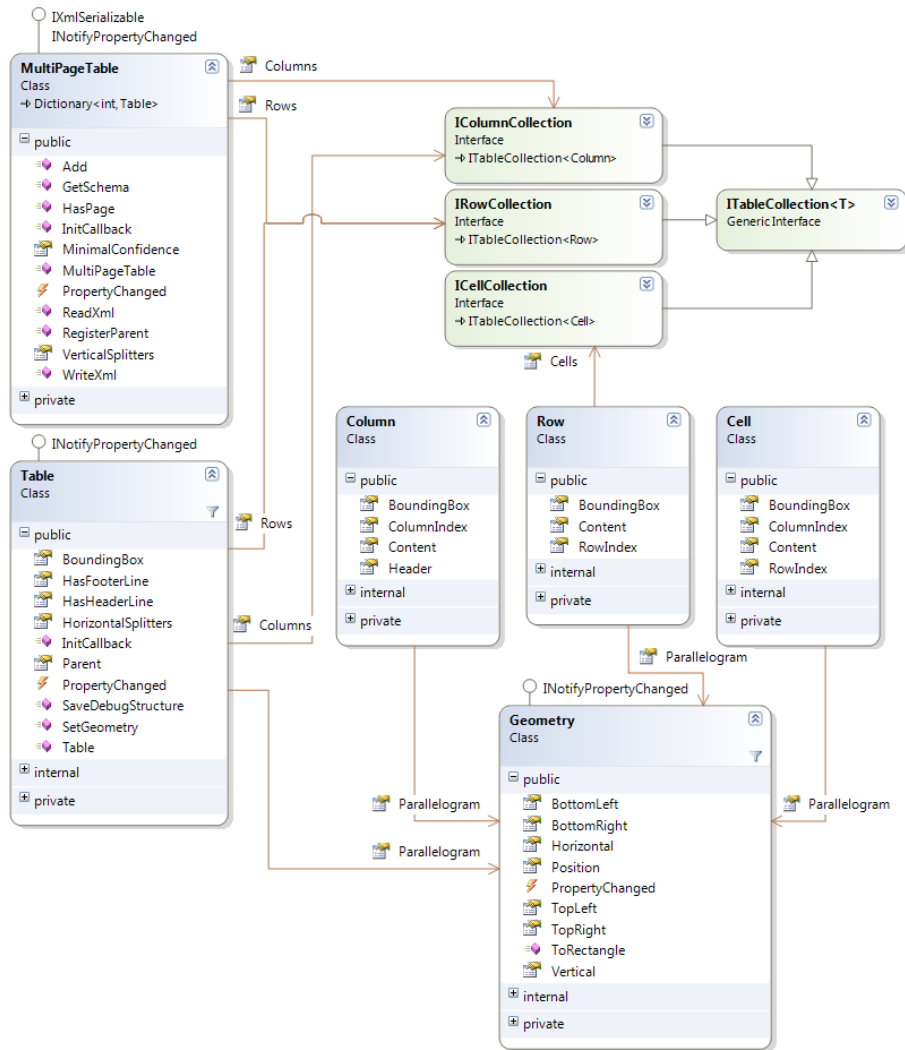


Abbildung 4.4: Klassendiagramm der Tabellenstruktur

Die *MultiPageTable*-Klasse und die *Table*-Klasse enthalten hierfür eine Liste von Spalten (*IColumnCollection*) und Zeilen (*IRowCollection*), und die *Row*-Klasse enthält eine Liste von Zellen (*ICellCollection*). Eine Spalte definiert sich durch Eigenschaften, wie einen Spaltenindex (*ColumnIndex*), eine Spaltenüberschrift (*Header*), einen Text-Inhalt (*Content*) und eine Geometrie (*Parallelogram*), während sich Zeilen über Zeilenindex (*RowIndex*), Text-Inhalt und Geo-

metrie beschreiben lassen. Zellen besitzen sowohl einen Zeilenindex, als auch einen Spaltenindex, sowie einen Text-Inhalt und eine Geometrie. Anhand der Anzahl der Trenner und deren Abstände kann somit die Struktur der Tabelle leicht aufgebaut werden. Der Algorithmus im Anhang unter 7.1.1 gibt diesen Vorgang wieder.

Da das Tabellenmodell, alleinstehend betrachtet, nur die Repräsentation tabellenförmiger Daten übernimmt, soll es nun in einen Tabellenverarbeitungsprozess eingebettet werden. Dieser stellt aus den Dokumentarten die benötigten Informationen zur Strukturierung der Tabelle zur Verfügung und verwendet das Modell, um Tabelleninhalte zu interpretieren oder zu exportieren. Diese werden nun im weiteren vorgestellt.

## 4.2 Gesamtprozess

Der Gesamtprozess in Abbildung 4.5 bettet das Tabellenmodell in einen Tabellenverarbeitungsprozess ein. Er beschreibt also, mit welchen Komponenten dieses befüllt, geändert oder exportiert werden kann. Er verwendet 4 Komponenten, welche in den folgenden Unterkapiteln genauer beschrieben werden. Drei davon finden bereits in den Produkten der Firma Open Text Document Technologies GmbH Verwendung.

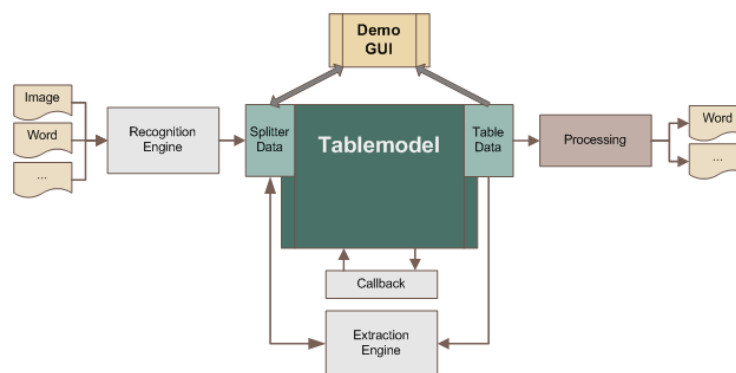


Abbildung 4.5: Der Gesamtprozess zeigt, wie das Tabellenmodell in eine Tabellenverarbeitungsumgebung eingebettet werden kann und welche Komponenten hierfür wichtig sind.

Ausgangspunkt in einem Tabellenverarbeitungsprozess ist ein beliebiges Ursprungsdokument. Hier bedarf es zu allererst einer Identifizierung der Tabellen. Die Strukturinformationen werden dann in einem Erkennungsschritt in das Tabellenmodell übertragen. Die Roh-tabelle kann nun schrittweise verfeinert werden, indem deren Struktur und deren Inhalt interpretiert werden. Es entsteht somit ein immer genaueres, exakteres Bild. Geht man von einer Tabelle aus, die nicht in einem pixelbasierten Format vorliegt, kann unter Umständen der Verfeinerungsschritt entfallen. Es gilt lediglich aufzuzeigen, dass das Tabellenmodell unabhängig von der Art der Eingabedokumente ist. Zuletzt besteht in einem Tabellenverarbeitungsprozess die Möglichkeit, die verfeinerte Tabelle zur Weiterverarbeitung zu extrahieren.

#### 4.2.1 Erkennung

Die firmeneigene Erkennung erfasst die Tabelleninformationen aus beliebigen Eingabedokumenten. Sie ist hier als Blackbox zu sehen, welche Tabellen an beliebigen Positionen lokalisieren kann, um somit Informationen zu Lage und Größe des Objekts zu liefern. Zusätzlich besitzt sie die Fähigkeit, Linien innerhalb der Tabelle zu erkennen. Hierbei können Linien sowohl sichtbare, wie auch nicht sichtbare Objekte sein. Wobei letztere als Weißflächen erkennbar sind. Die Lage, die Größe und die erkannten Linien werden dann dem Tabellenmodell mitgeteilt. Zur Zeit werden noch keine Formatierungen, wie Liniendicke oder äußere Erscheinungen unterstützt. Eine Erweiterung der Erkennung könnte dann auch die Linien-Formatierungen liefern.

#### 4.2.2 Extraktion

Die firmeneigene Extraktion interpretiert die schon vorhandenen Tabellenstrukturinformationen und reichert sie mit logischen Informationen an. Sie ist für die schrittweise Verfeinerung des Tabellenmodells zuständig und beinhaltet zahlreiche Algorithmen, die das Layout und den Inhalt der Tabelle analysieren. Hierfür werden die Eigenschaften der Trenner, wie der Vertrauenswert oder deren Status

verändert, um die Struktur der Tabelle zu verbessern. Zum Beispiel kann die Analyse der Überschriften Hinweise liefern, ob eine Spalte doch lieber getrennt werden soll.

### 4.2.3 Weiterverarbeitung

Die Weiterverarbeitung schreibt die Tabellen in verschiedene Ausgabedokumente. Je nach Ausgabeformat werden bestimmte Eigenschaften der Tabelle ignoriert, da diese dort nicht unterstützt werden. Zum Beispiel wird der Export in eine CSV-Datei<sup>8</sup> keine Informationen über die Linienart verwenden können, während ein Export in das Microsoft Office Word-Format [30] alle Tabelleninformationen verarbeiten kann. Über verschiedene Schnittstellen kann diese Funktionalität zur Verfügung gestellt werden.

### 4.2.4 Callback

Der firmeneigene Callback liefert zu einer bestimmten Zone den Text aus einem Bild. Hierfür wurde der gesamte Text im Vorfeld bereits erkannt und die Erkennungsergebnisse in eine XML-Datei gespeichert. Eine Schnittstelle übersetzt die Anfragen des Tabellenmodells innerhalb einer bestimmten Zone und schaut die Ergebnisse in der Ergebnisdatei nach. Der Callback wird benötigt, um Inhaltsinformationen innerhalb der Tabelle aus Bildern zu gewinnen. Für andere Dokumentenformate müssen andere Schnittstellen bereitgestellt werden. Um Formatierungen, wie Ausrichtung oder Schriftart im Tabellenmodell einbinden zu können, kann hierzu der Callback verwendet werden. Dieser müsste dann lediglich formatspezifische Anfragen unterstützen.

## 4.3 Simulationsoberfläche

Da die bisherigen Methoden der Erkennung und der Extraktion noch nicht mit dem entwickelten Tabellenmodell harmonieren und das Umschreiben dieser

<sup>8</sup>RFC 4180: <http://tools.ietf.org/html/rfc4180>



seitige Dokumente können durchgeblättert werden. Wurde eine Tabelle angelegt, kann deren Eigenschaften im Fenster rechts eingesehen und auch verändert werden. Es stellt die Eigenschaften der Objekte innerhalb des Tabellenmodells dar. Hierdurch lässt sich das Modell gut simulieren und ist leicht durch den Benutzer veränderbar. Die Menüleiste beinhaltet Funktionen zur Dateiverwaltung des Tabellenprojektes und zur Auswahl des Dokuments. Die Tabellen können später in eine Worddatei exportiert werden, wo eine Weiterverarbeitung stattfinden kann.

Die Entwicklung des Tabellenmodells soll nun im folgenden, abschließenden Kapitel analysiert und erörtert werden.

# Kapitel 5

## Erkenntnisse

Im weiteren werden nun die Ergebnisse der Arbeit aufgezeigt und das Tabellenmodell hinsichtlich der Themenstellung evaluiert. Die in 3.1 aufgestellten Anforderungen und die Ergebnisse aus anderen wissenschaftlichen Publikationen dienen hierbei als Grundlage. Zusätzlich wird betrachtet, inwieweit sich das Tabellenmodell in realen Anwendungen einsetzen lässt und welche Tabellenarten modellierbar sind. Da bereits im Vorhinein auf komplexe Tabellen verzichtet wurde, wird anschließend erörtert, welche Veränderungen im Modell realisiert werden müssen, um auch diese Strukturen repräsentieren zu können.

### 5.1 Ergebnisse

Zunächst einmal bedarf es einer alleinstehenden Betrachtung des Modells. Hierbei steht die Modellierung der Tabellenstruktur nur durch Linieninformationen im Vordergrund. Nach Lopresti und Nagy findet sich für pixelbasierte Dokumente in der Literatur entweder die Analyse der Geometrieeigenschaften oder der Zelleninhalte [17]. Abstrahiert man die Blockbildung als Problemlösung, in der versucht wird, einzelne Blöcke zu identifizieren, die durch unsichtbare Linienobjekte getrennt werden, so kann man dieses Verfahren ebenso als Linienverfahren bezeichnen. Unter diesen Umständen ist es möglich, Tabellen nur durch diese

Informationen zu betrachten.

Hervorzuheben ist die flexible Trennung zwischen den Strukturinformationen, also den Trennern, und der eigentlichen daraus resultierenden Tabellenstruktur. Diese beschreibt lediglich eine Momentaufnahme und nicht zwingend das endgültige Ergebnis. Auch durch die Vergabe von Vertrauenswerten entsteht eine höhere Flexibilität innerhalb des Modells. Durch die Anbindung verschiedener Interpretationsalgorithmen ist es möglich, schrittweise Veränderungen vornehmen zu lassen, um die Tabellenstruktur zu perfektionieren. Diese Gegebenheit ist bisher zwar nur über die Simulationsoberfläche realisiert, macht aber deutlich, welche Vorteile ein flexibles Tabellenmodell hat und welche Probleme hierdurch behoben werden können.

Formateigenschaften, wie Ausrichtung, Farbe, Schriftart oder Schriftgröße sind nicht direkt im Modell integriert, sondern werden über externe Schnittstellen bereitgestellt. Dies liegt zum einen an der gewählten Definition der Tabellenstruktur und zum anderen an der Art, wie die Erkennung arbeitet. Antworten über formatspezifische Eigenschaften in Zeilen, Spalten oder Zellen können erst nach der Strukturerkennung gegeben werden, da die Objekte davor noch nicht existieren. Einzig die Trenner beinhalten Informationen über das Format der Linien, die sie repräsentieren. Für pixelbasierte Dokumente macht dies durchaus Sinn, betrachtet man aber zum Beispiel eine Wordtabelle, so könnten die Informationen direkt entnommen werden. Stattdessen muss ein Umweg über spezifische Schnittstellen gemacht werden, der den Overhead des Tabellenmodells womöglich vergrößert. Jedoch erhält man nur dann formatspezifische Informationen, wenn man danach verlangt.

Der Aspekt der Arbeit, tabellenförmige Daten aus verschiedenen Dokumententypen zu repräsentieren, bezieht sich auf die Verknüpfung von pixelbasierten und elektronischen Dokumenten. Hierbei wurde in Kapitel 2.1.4 bereits erläutert, wie sie im einzelnen Tabellen modellieren und mit welchen Verfahren deren Layoutinformationen extrahiert werden können. Natürlich entstehen durch den hier gewählten Ansatz Nachteile. Es werden zum Beispiel Informationen, die in

anderen Formaten vorhanden sein können, im Tabellenmodell nicht unterstützt und fallen somit weg. Auf der anderen Seite wird aber das Modell hierdurch möglichst einfach und benutzbar gehalten. Die Modellierung von mehrseitigen Tabellen erweist sich als sehr hilfreich, da ein Großteil der in der Firma vorkommenden Dokumente ebensolche beinhalten.

Man kann nicht mit völliger Sicherheit behaupten, dass ein Tabellenmodell existiert, womit jedes erdenkliche Tabellenkonstrukt erkannt und repräsentiert werden kann. Die Forschung ist hier in den letzten Jahren zwar deutlich vorangekommen, es warten aber immer noch vorhandene Probleme auf eine Lösung. Jedoch zeigt sich, dass durchaus ein Hauptteil der verschiedenen Tabellenarten einheitlich repräsentiert werden kann. Da das bisherige Modell nur für einfache Tabellen gilt, soll nun im Folgenden analysiert werden, welche Veränderungen vollzogen werden müssen, um hier auch komplexere Strukturen zu unterstützen.

## 5.2 Ausblick

Um im Modell mehrzeilige Kopf- beziehungsweise Fußzeilen ausdrücken zu können, werden die horizontalen Trenner mit einer zusätzlichen Eigenschaft versehen. Diese gibt an, in welchem Tabellenbereich sich das Trennerobjekt befindet oder welchen Tabellenbereich es trennt. Gültige Eigenschaftswerte sind dann 'Tabellenkopf', 'Tabellenrumpf' oder 'Tabellenfuß'. Die Tabelleneigenschaft *Header* muss hierbei bestehen bleiben, um die erste Zeile zu beschreiben. Ein zusätzlicher Vorteil entsteht dadurch, dass auch Zwischenüberschriften, welche keine Daten beschreiben, modelliert werden können. Somit können diese Zeilen aus der Datenansicht eliminiert werden und die Tabellenstruktur beinhaltet nur noch die reinen Daten. Die Komplexität des Modells erhöht sich nur wenig, wodurch die Benutzbarkeit erhalten bleibt.

Für die Modellierung von geschachtelten Tabellen muss ein höherer Aufwand betrieben werden. Da die einzigen Informationen zur Strukturierung der Daten auf den Trennungen zwischen Zeilen und Spalten beruhen, können auch nur die

Trennerobjekte hierfür herangezogen werden. Ramel et al. vermeiden dieses Problem, indem sie die Tabelle 'virtualisieren' und somit Irregularitäten innerhalb der Tabelle beseitigen [4]. In ihrem Ansatz speichern sie aber die Verknüpfung von virtuellen und realen Zellen direkt in der Tabellenstruktur und nicht in den Trennerobjekten. Der nun im weiteren vorgestellte Ansatz unterteilt die Tabelle in verschiedene Gitterstrukturen, abhängig von den dargestellten Hierarchien. Existiert eine Schachtelung in einer Tabelle, so werden die im Tabellenrumpf befindlichen Daten auf einer logischen Ebene abstrahiert. Es entsteht somit eine Hierarchisierung der Daten. Eine Tabelle kann also folglich durch verschiedene Gitterstrukturen ausgedrückt werden, die je nach Hierarchiestufe gröber oder feiner unterteilt ist. Im Tabellenmodell werden deshalb die Trenner durch eine zusätzliche Eigenschaft erweitert. Sie gibt an, welche Hierarchiestufe der Trenner vertritt. Um auch Informationen über die erste Zeile der Tabelle zu erhalten, muss diese eine Anfangsstufe besitzen, da nicht immer gegeben ist, dass die Tabelle mit der ersten Stufe beginnt. Die Struktur in Stufe  $x$  wird durch die Trenner der Stufen  $1 \dots x$  bestimmt und ein horizontaler Trenner der Stufe  $x$  gibt an, in welche Struktur die darunterliegende Zeile geteilt wird. Das folgende Beispiel in Abbildung 5.1 veranschaulicht den Prozess für geschachtelte Tabellen.

Betrachtet man die Tabelle nur in der ersten Hierarchiestufe, so ergeben sich drei Spalten und eine Zeile. In der zweiten Hierarchiestufe entstehen 5 Spalten und 8 Zeilen. Im Tabellenkopf entsteht bei 'Erststimmen' und 'Zweitstimmen' ein Bruch der Zelle. Hier müsste also ein Verschmelzen der Zellen möglich gemacht werden. Geht man von der Tatsache aus, dass die Tabelle in Stufe eins beginnt, der erste horizontale Trenner zur zweiten Hierarchiestufe gehört und zwischen den roten vertikalen Trennern ein zusätzlicher Trenner höherer Stufe verläuft, so lässt sich daraus schließen, dass die Zellen 'Erststimmen' und 'Zweitstimmen' verschmolzen sein müssen. Für die Spalte 'Liste' kann man nur vermuten, dass durch die nicht vorhandenen vertikalen Trennern sie eine horizontal verschmolzene Zelle beschreibt. Es besteht also die Möglichkeit, über die Trennerkonfiguration bestimmen zu können, wo Schachtelungen innerhalb der Tabelle auftreten.

a)

Liste	Erststimmen		Zweitstimmen	
	2009	2005	2009	2005
CDU	43.1 %	43.9 %	31.8 %	37.3 %
SPD	21.6 %	31.7 %	18.9 %	29.8 %
BÜNDNIS 90/DIE GRÜNEN	12.7 %	8.9 %	14.9 %	12.1 %
FDP	14.8 %	10.5 %	21.3 %	13.7 %
Die Linke.	6.5 %	3.7 %	6.8 %	3.8 %
Sonstige	1.3 %	1.3 %	6.3 %	3.3 %

b)

Liste	Erststimmen		Zweitstimmen	
	2009	2005	2009	2005
CDU	43.1 %	43.9 %	31.8 %	37.3 %
SPD	21.6 %	31.7 %	18.9 %	29.8 %
BÜNDNIS 90/DIE GRÜNEN	12.7 %	8.9 %	14.9 %	12.1 %
FDP	14.8 %	10.5 %	21.3 %	13.7 %
Die Linke.	6.5 %	3.7 %	6.8 %	3.8 %
Sonstige	1.3 %	1.3 %	6.3 %	3.3 %

Abbildung 5.1: Abgebildet sind hier die Wahlergebnisse der Wahlkreises 287 Konstanz der Bundestagswahl 2005 und 2009. Abbildung a) beschreibt die tatsächliche Tabelle, Abbildung b) veranschaulicht die Trennerkonfiguration der geschachtelten Tabelle. Rote Trenner definieren die erste Hierarchiestufe, grüne Trenner definieren die zweite Hierarchiestufe. Die Tabelle beginnt in Hierarchiestufe eins.

Diese Informationen können dann in der simulierten Tabellenstruktur aufgenommen werden, indem ähnlich wie in [4] virtuelle Zellen verwendet werden, welche die regelmäßige Struktur nicht zerstören. Für Spaltenüberschriften folgt, dass diese ebenfalls geschachtelte Tabellen unterstützen müssen. Die Spaltenüberschriften der Tabelle beschreiben die Logik, welche die Daten ausdrücken sollen. Es würde sich der Wangsche Ansatz [1] zur Modellierung der Tabellenlogik eignen. Über die Hierarchisierung und die Trennerkonfiguration können also die einzelnen Gruppen identifiziert werden. Die Spaltenüberschrift enthält somit die Inhalte der Zellen der einzelnen Gruppe im Tabellenkopf. Nochmal zurückgreifend auf das Beispiel in Abbildung 5.1 folgt dann in Spalte zwei die Überschrift 'Erststimmen.2009'. Mit Hilfe des gezeigten Ansatzes können nun auch komplexere Tabellen modelliert werden, besonders für Tabellen, die verschiedene Gitterstrukturen aufweisen. Wird das Tabellenmodell hierdurch erweitert, entsteht folglich ein größerer Abdeckungsrahmen für verschiedene Tabellenarten, die Benutzbarkeit des Modells wird aber herabgesetzt.

## Kapitel 6

# Zusammenfassung

Diese Arbeit beschäftigte sich mit der Entwicklung eines Modells, das tabellenförmige Daten aus Dokumenten repräsentiert. Anfangs wurde die Tabelle beleuchtet. Durch die Abgrenzung von Formularen und Listen entwickelte sich die Definition, was eine tabellarische Struktur auszeichnet. Es wurde aufgezeigt, welche terminologischen Begriffe hierfür wichtig sind und auf welche unterschiedlichen Weisen strukturierte Daten ausgedrückt werden können. Anhand mehrerer Beispiele wurde beschrieben, welche Gegebenheiten eine geordnete Struktur für Mensch und Maschine komplex gestalten. Unterschiedliche Dokumentarten wurden in Dokumentengruppen eingeteilt und beleuchtet, wie diese Tabellen modellieren und durch automatische Prozesse herausgefiltert werden können. Danach wurde die Themenstellung aufgegriffen und die Anforderungen zur Entwicklung des Tabellenmodells aufgestellt und Folgerungen hiervon abgeleitet. Im Detail wurden einzelne Komponenten des Tabellenmodells vorgestellt und der Gesamtprozess zur Verarbeitung von tabellenförmiger Daten analysiert. Anhand einer Simulationsoberfläche konnte gezeigt werden, wie das Tabellenmodell mit den simulierten Entscheidungen aus Erkennung und Extraktion zurecht kommt. Schließlich wurden die Ergebnisse dieser Arbeit zusammengetragen und ausblickend gezeigt, wie noch bestehende Probleme des Tabellenmodells behoben werden können.

# Kapitel 7

## Anhang

### 7.1 Entwicklung

---

**Algorithm 7.1.1** Aufbau der simulierten Tabellenstruktur im Modell

---

**Require:** Minimal Confidence:  $0 \leq \text{confidence} \leq 100$

```
for  $i = 0$  to VerticalSplitters.Count +1 do
  splitter  $\leftarrow$  VerticalSplitters[i]
  if splitter is active  $\wedge$  splitter.Confidence  $\geq$  confidence
   $\vee$  all VerticalSplitters passed then
    generate new column
    add column to table
  end if
end for
for  $i = 0$  to HorizontalSplitters.Count +1 do
  splitter  $\leftarrow$  HorizontalSplitters[i]
  if splitter is active  $\wedge$  splitter.Confidence  $\geq$  confidence
   $\vee$  all splitters passed then
    generate new row
    for  $i = 0$  to Columns do
      if table has header  $\wedge i == 0$  then
        set current column's header
      else
        generate new cell
        add cell to current row
      end if
    end for
    add row to table
  end if
end for
```

---

# Literaturverzeichnis

- [1] Wang, X.: Tabular Abstraction, Editing, and Formatting. PhD thesis, University of Waterloo (1996)
- [2] Pivk, A., Cimiano, P., Sure, Y., Gams, M., Rajkovič, V., Studer, R.: Transforming arbitrary tables into logical form with tartar. *Data Knowl. Eng.* **60** (2007) 567–595
- [3] Oro, E., Ruffolo, M.: Pdf-trex: An approach for recognizing and extracting tables from pdf documents. *International Conference on Document Analysis and Recognition* **0** (2009) 906–910
- [4] Ramel, J.Y., Crucianu, M., Vincent, N., Faure, C.: Detection, extraction and representation of tables. In: *ICDAR '03: Proceedings of the 7th International Conference on Document Analysis and Recognition, Washington, DC, USA, IEEE Computer Society* (2003) 374
- [5] Watanabe, T., Luo, Q., Sugie, N.: Layout recognition of multi-kinds of table-form documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17** (1995) 432–445
- [6] Wohlberg, T.: Hypertables: Entwicklung einer strukturbeschreibungssprache für tabellen in xml. Master's thesis, Fachbereich Informatik, Universität Hamburg (1999)
- [7] Crucianu, M., El Ayadi, R., Vincent, N.: On the representation of tables in xml. Internal Report 244, Laboratoire d'Informatique, University of Tours (2001)
- [8] Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., Pollak, B.: Towards domain-independent information extraction from web tables. In: *WWW '07: Pro-*

- ceedings of the 16th international conference on World Wide Web, New York, NY, USA, ACM (2007) 71–80
- [9] Silberhorn, H.: Tabulamagica: an integrated approach to manage complex tables. In: DocEng '01: Proceedings of the 2001 ACM Symposium on Document engineering, New York, NY, USA, ACM (2001) 68–75
- [10] Khamsi, M.A., Misane, D.: Using natural language processing for identifying and interpreting tables in plain text. In: In 4th Annual Symposium on Document Analysis and Information Retrieval. (1995) 535–545
- [11] Hu, J., Kashi, R., Lopresti, D., Wilfong, G.: A system for understanding and reformulating tables. In: 4th IAPR International Workshop on Document Analysis Systems, Rio de Janeiro, Brazil (2000) 361–372
- [12] Liu, Y., Bai, K., Mitra, P., Giles, C.L.: Tableseer: automatic table metadata extraction and searching in digital libraries. In: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries. JCDL '07, New York, NY, USA, ACM (2007) 91–100
- [13] Kieninger, T., Dengel, A.: The t-recs table recognition and analysis system. In: Selected Papers from the Third IAPR Workshop on Document Analysis Systems: Theory and Practice. DAS '98, London, UK, Springer-Verlag (1999) 255–269
- [14] Ishitani, Y., Fume, K., Sumita, K.: Table structure analysis based on cell classification and cell modification for xml document transformation. International Conference on Document Analysis and Recognition **0** (2005) 1247–1252
- [15] Wasserman, H.C., Yukawa, K., Sy, B.K., Kwok, K.L., Phillips, I.T.: A theoretical foundation and a method for document table structure extraction and decomposition. In: Proceedings of the 5th International Workshop on Document Analysis Systems. Volume 5 of DAS '02., London, UK, Springer-Verlag (2002) 291–294
- [16] Laurentini, A., Viada, P.: Identifying and understanding tabular material in compound documents. In: 11th International Conference Pattern Recognition, The Hague (1992) 405–409
- [17] Lopresti, D.P., Nagy, G.: A tabular survey of automated table processing. In: GREC '99: Selected Papers from the 3rd International Workshop on Graphics Recognition, Recent Advances, London, UK, Springer-Verlag (2000) 93–120

- [18] Zanibbi, R., Blostein, D., Cordy, J.: A survey of table recognition: Models, observations, transformations, and inferences. *International Journal of Document Analysis and Recognition* **7** (2003) 1–16
- [19] Embley, D., Hurst, M., Lopresti, D., Nagy, G.: Table-processing paradigms: a research survey. *International Journal on Document Analysis and Recognition* **8** (2006) 66–86
- [20] Marchese, F.T.: The chemical table: An open dialog between visualization and design. *Information Visualisation, International Conference on* **0** (2008) 75–81
- [21] Mao, A.Y., Cordy, J.R., Dean, T.R.: Automated conversion of table-based websites to structured stylesheets using table recognition and clone detection. In: *CASCON '07: Proceedings of the 2007 conference of the center for advanced studies on Collaborative research*, New York, NY, USA, ACM (2007) 12–26
- [22] W3C: XHTML 1.1 - Module-based XHTML. W3C Recommendation, <http://www.w3.org/TR/xhtml1/> (2001)
- [23] Embley, D.W., Lopresti, D., Nagy, G.: Notes on contemporary table recognition. (2006)
- [24] Microsoft: Primäre Interop-Assemblys in Office. Microsoft, [http://msdn.microsoft.com/de-de/library/15s06t57\(VS.80\).aspx](http://msdn.microsoft.com/de-de/library/15s06t57(VS.80).aspx). (2010)
- [25] Adobe: PDF Reference version 1.7. 6 edn. Adobe Systems Incorporated (2006)
- [26] Noonburg, D.: xpdf: A C++ library for accessing PDF, [www.foolabs.com/xpdf](http://www.foolabs.com/xpdf). (2009)
- [27] Anjewierden, A.: Aidas: Incremental logical structure discovery in pdf documents. In: *6th International Conference on Document Analysis and Recognition (ICDAR)*. (2001) 374–378
- [28] Chao, H., Fan, J.: Layout and content extraction for pdf documents. In: *Document Analysis Systems VI*. (2004) 213–224
- [29] Microsoft: .NET Framework Developer's Guide - Serializing Objects. Microsoft, [http://msdn.microsoft.com/en-us/library/7ay27kt9\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/7ay27kt9(VS.71).aspx). (2010)
- [30] Microsoft: Microsoft Office Word Format Specification. Microsoft, <http://download.microsoft.com/>. (2007)