Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 97

Plausibilitätsprüfung implizit gekoppelter Spreadsheet-Daten

Wolfgang Kraus

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. rer. nat. Stefan Wagner
Betreuer/in:	Daniel Kulesz, M.Sc
Beginn am:	2013-07-11
Beendet am:	2014-05-09
CR-Nummer:	D.2.5, E.m, H.4.1

Kurzfassung

Spreadsheets („Excel Programme“) sind tabellenförmige Dateien, die nach diversen Studien weit verbreitet sind und eine hohe Fehlerquote aufweisen. An der Universität Stuttgart wurde zur Unterstützung der Prüfung vom Spreadsheets das Spreadsheet Inspection Framework (SIF) entwickelt und in diversen Arbeiten erweitert. Für eine einfachere Benutzung wurde zudem eine Integration (das SIFEI) für Microsoft Excel entwickelt, das eine Benutzerschnittstelle für das SIF bereitstellt.

Im Rahmen dieser Bachelorarbeit soll das SIF um die Möglichkeit einer Plausibilitätsprüfung von implizit gekoppelten Spreadsheet-Daten erweitert werden, um die Erhöhung der Datenqualität zu unterstützen. Für die Benutzung soll zudem das SIFEI erweitert werden, um diese Prüfung aus Excel heraus ausführen zu können.

Abschließend wurde das umgesetzte Konzept evaluiert. Das Konzept ist für die Prüfung gut geeignet, jedoch benötigt der Benutzer anfangs Unterstützung und die Verfügbarkeit aller Daten stellt ein Problem dar.

Abstract

Spreadsheets (“Excel programs“) are files with a tabular presentation. According to several studies they are both widely used and error-prone. To support the testing of those spreadsheet, the Spreadsheet Inspection Framework (SIF) was developed and expanded throughout severeal thesis projects. Furthermore an integration for Microsoft Excel was developed as a user interface for the SIF, the SIF Excel Integration (SIFEI).

Goal of the thesis is to expand the SIF to support a plausibility check of implicitly coupled spreadsheet data to improve the data quality in spreadsheets. Also the SIFEI is to be expanded to support defining and running those tests through Excel.

Concluding, the concept and the implementation were evaluated. The concept showed to be applicable but the user needs help when using the program for the first time and the availability for the needed data might pose a problem.

Inhaltsverzeichnis

1. Einleitung

Spreadsheets („Excel-Programme“) werden oft für wichtige Aufgaben eingesetzt [Cro07], enthalten jedoch häufig Fehler. Diese Fehler können zum Teil gravierende Folgen haben, so gab es beispielsweise bei der Analyse von 25 Spreadsheets, die in der Finanzindustrie benutzt wurden, einen maximalen prozentualen Fehler von 416,5%, die höchste absolute Abweichung betrug $110.543.305 [PBL09]. Panko fasste einige Artikel zu gefundenen Fehlern zusammen und gab als Durchschnitt für Spreadsheets im „Field Audit“ ab 1997 eine Fehlerquote von 91% an. Selbst wenn diese unter Laborbedingungen erstellt werden, enthielten 51% der Spreadsheet Fehler [Pan08].

Im Hintergrund der Aufgabenstellung dieser Bachelorarbeit stehen, konträr zur Formel-Fixierung der Forschung in der Finanzindustrie, auch schlechte Datenqualität.

1.1. Motivation

Eine Fehlerminimierung wird somit auch hier, ähnlich wie bei der professionellen Softwareentwicklung, benötigt. Um diese zu vereinfachen gibt es mehrere kommerziell verfügbare Programme, die jedoch auf Finanzanwendungen zugeschnitten sind und in der Regel wenig Einstellungsmöglichkeiten mit sich bringen [Zit12].

Als Alternative wurde an der Universität Stuttgart das Spreadsheet Inspection Framework (SIF) entwickelt, um statische [Zit12] und dynamische [Lem13] Prüfungen durchzuführen. Die statischen Prüfungen ermöglichen es, die verwendeten Formeln ähnlich der statischen Codeanalyse zu prüfen, beispielsweise, dass die Operatoren eine bestimmte Verschachtelungstiefe nicht überschreiten. Die dynamische Prüfung ermöglicht es, die Ergebnisse der Formeln mit Testeingaben zu überprüfen. Die verschiedenen Prüfungsarten werden im Kapitel 5.1.1 SIF Ist-Zustand (S. 20) weitergehend beschrieben.

Für eine benutzerfreundlichere Darstellung der Fehler wurde die SIF Excel Integration (SIFEI) entwickelt [Dou13] und um eine Benutzerschnittstelle für die dynamischen Prüfungen erweitert [Sch14].

1.2. Ziel

Das Ziel dieser Arbeit ist eine Erweiterung von SIF und SIFEI um einen Mechanismus zur Plausibilitätsprüfung von in Spreadsheets enthaltenen Daten. Es soll eine Möglichkeit geschaffen werden, vorhandene semantische Bezüge zwischen Datenwerten zu definieren und zu prüfen. Für die Endbenutzertauglichkeit soll das SIFEI erweitert werden, um diese Prüfung über Microsoft Excel zu ermöglichen. Der Schwerpunkt dabei liegt auf dem SIF, um weiterhin eine möglichst große Unabhängigkeit gegenüber dem benutzten Spreadsheet Programm zu gewährleisten.

Die Ursachen der zu findenden Fehler gehören zur unzureichenden Domainkenntnis und zu den motorischen Fehlern, Näheres dazu in Kapitel 2.3 Fehler in Spreadsheets (S. 11) .

1.3. Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen: Hier werden die grundliegenden Begriffe und Konzepte dargestellt.

Kapitel 3 – Verwandte Arbeiten fasst einige ähnliche Arbeiten zusammen und grenzt diese gegen die Excel eigene Validierung ab.

Kapitel 4 – Konzept für die Plausibilitätsprüfung stellt das Konzept der Plausibilität und der Umsetzung vor.

Kapitel 5 – Implementierung legt den Zustand vom SIF und SIFEI vor und nach der Änderung dar.

Kapitel 6 – Evaluation beinhaltet die Durchführung und die Ergebnisse der Evaluation.

Kapitel 7 – Zusammenfassung und Ausblick fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

2. Grundlagen

2.1. Taxonomie

Nachdem diese Arbeit auf dem SIF aufbaut, sind die folgenden Definitionen an die der vorherigen Arbeiten [Zit12] angelehnt. Da nicht alle Begriffe benötigt werden, sind einige bewusst ausgelassen.

Spreadsheet (Konzept)

Das Konzept eines Spreadsheets kommt von großen Papierbögen, die in der Buchhaltung benutzt wurden und eine gradlinige, tabellenförmige Darstellung von Zellen beinhaltet haben. Sie wurden anfangs benutzt um Grundkosten auf gewinnbringende Abteilungen umzulegen, jedoch werden sie in der elektronischen Form inzwischen für viel mehr benutzt.

Zelle

Die Zelle ist die atomare Einheit für Benutzereingaben, dessen Inhalt grundsätzlich als Text angesehen werden kann. Dieser kann auf verschiedene Arten formatiert und dargestellt werden. Die Position der Zelle wird in der Regel über den Buchstaben für die Spalte und eine Zahl für die Zeile notiert, zum Beispiel $A2$, die Zelle in der ersten Spalte und zweiten Zeile.

Zellbereich

Es gibt die Möglichkeit, Zellbereiche zu definieren. Dazu gehören die relativen Bereiche mit dem Format A1[:B2]. Die Werte in den eckigen Klammern sind optional; über den zweiten Zellnamen mit dem Doppelpunkt kann man einen Zellbereich definieren, der das Rechteck von links oben nach rechts unten beinhaltet. Alternativ zum relativen Bereich gibt es den absoluten Bereich. Hierfür wird vor die absoluten Bereiche ein '$' vorgestellt, zum Beispiel $\$A\$1:\$A\$3$. Es gibt beliebige Mischformen zwischen absoluten und relativen Bereichen, deren relativen Teile beim Kopieren und Verschieben angepasst werden.

Worksheet

Ein Worksheet (Arbeitsblatt) ist die tabellenförmige Darstellung von Zellen, die als Ganzes einen eindeutigen Namen besitzt. Eine vertikale Reihe von Zellen wird Spalte, eine horizontale Reihe wird Zeile genannt. Die Darstellungsgröße der Spalten und Zeilen kann für den jeweiligen Inhalt angepasst werden.

Spreadsheet

Ein Spreadsheet (Arbeitsmappe), ist eine nicht leere Menge an Worksheets. Sie stellt zusammen mit dem Spreadsheet Programm die grundliegende Interaktionsmöglichkeit für den Benutzer dar, in der Worksheets hinzugefügt, geändert und gelöscht werden können.

Spreadsheet Programm

Ein Spreadsheet Programm stellt die Benutzerschnittstelle zu einem Spreadsheet mit diversen Eingabemöglichkeiten dar. Bekannte Vertreter sind zum Beispiel OpenOffice Calc und Microsoft Excel. Aufgrund der vorhandenen Integration für Excel wird dieses für diese Arbeit synonym als Spreadsheet Programm benutzt.

Überschrift

Die Überschrift, sofern vorhanden, ist in der Regel die erste benutzte Zeile einer Spalte oder Zeile. Somit könnte eine Zelle zwei Überschriften haben, sowohl horizontal, als auch vertikal.

Formel

Eine Formel ist eine besondere Benutzereingabe, die mit einem vorgestellten '=' beginnt und vom Spreadsheet Programm berechnet wird. Sie kann aus Funktionen und Operatoren bestehen. Falls es mehrere Funktionen gibt, müssen sie durch Operatoren verbunden werden. Es gibt einige vorgegebene Funktionen, wie zum Beispiel `SUM` zum Aufsummieren von Werten. Operatoren sind die arithmetischen Zeichen (+, -, *, /), und die boolschen Operatoren `UND` und `ODER`. Neben der direkten Eingabe von Werten in die Funktionen können auch Zellen als Eingabewert referenziert werden.

Referenz

Zellen können im Rahmen von Formeln auch referenziert werden, um den Wert, der in der Zelle steht, zu benutzten. Eine Referenz kann den Namen eines anderen Worksheets enthalten,

der vor dem Zellbereich steht und durch ein '!' getrennt wird. Beim Kopieren oder Verschieben eines relativen Zellbereiches, wird dieser angepasst, beispielsweise beim Kopieren in eine Zeile weiter unten werden die Zeilennummern des Zellbereiches ebenfalls erhöht. Sofern das nicht erwünscht ist, muss man einen absoluten Zellbereich definieren.

Ribbon

Ein Ribbon, oder Multifunktionsleiste, ist die standardmäßige graphische Darstellung der Schaltflächen in Microsoft Excel 2013. Es gibt mehrere Ribbons die in Tabs aufgeteilt sind und Funktionalitäten gruppiert bereitstellen. Die Überschrift des Ribbon bezeichnet die Kategorie, wie zum Beispiel „Einfügen“, das unter anderem Knöpfe für das Kopieren und Einfügen aus der Zwischenablage bereitstellt.

Spreadsheet Format

Ein Spreadsheet Format ist die technische Umsetzung wie das Spreadsheet auf der Festplatte gespeichert wird. Ein Programm unterstützt in der Regel mehrere Formate, jedoch gibt es Einschränkungen durch proprietäre und verschiedene Versionen der Formate.

2.2. Kopplung

In Rahmen dieser Arbeit wird die Kopplung als semantischer Zusammenhang zwischen Datenwerten in verschiedenen Zellen bezeichnet. In einer simplen, tabellenförmigen Anordnung von Werten gibt es in der Regel zwei Achsen der Kopplung, zum Einen in der Vertikalen, zum Anderen in der horizontalen Richtung.

Die Auslegung der Richtungen ist an sich beliebig, für diese Arbeit wird es so ausgelegt: Eine Spalte beinhaltet Datenwerte einer Klasse, die durch eine Überschrift beschrieben werden können, während eine Zeile verschiedene Klassen zueinander in Bezug setzt.

Implizite Kopplung

Als implizite Kopplung wird der vorhandene semantische Zusammenhang bezeichnet, der jedoch nicht explizit definiert ist. Eine Möglichkeit dafür sind zum Beispiel Namenskonventionen bei Überschriften und die Struktur der Daten. Für das Spreadsheet Programm sind dies jedoch nur Werte, die in einer Tabelle stehen und keinen Bezug zueinander haben. Im Kontrast dazu gibt es die explizite Kopplung, zum Beispiel die Referenzierung dieser Werte durch eine Formel.

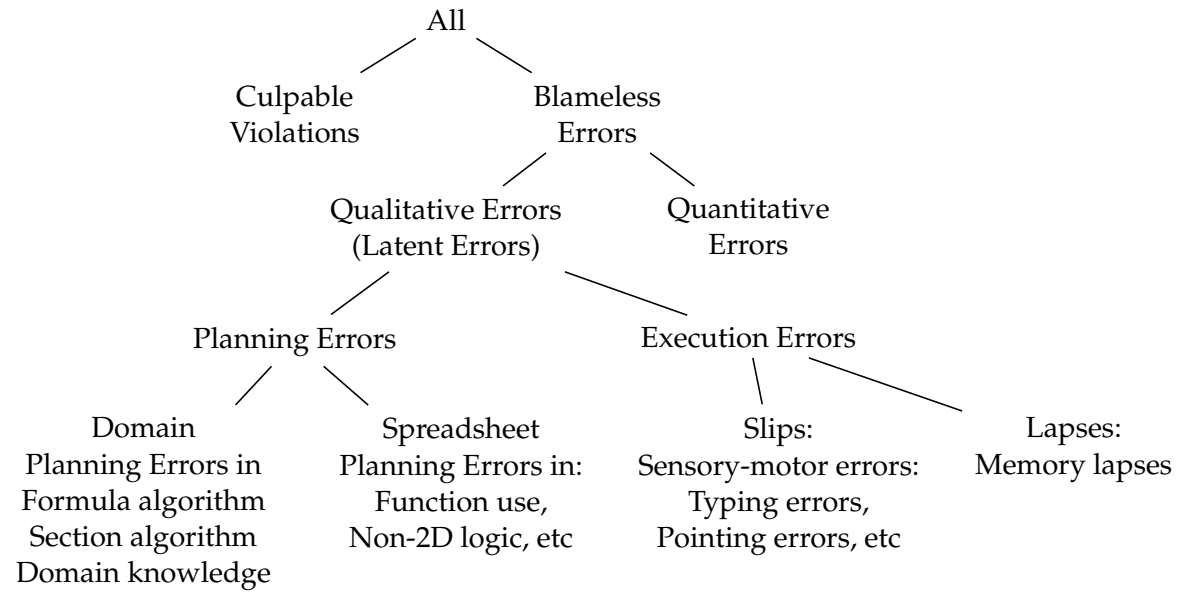

Abbildung 2.1.: Fehlertaxonomiebaum aus [PA10], S. 241

2.3. Fehler in Spreadsheets

Zum besseren Verständnis für die auftretenden Fehler wurden auch für diese einige Taxonomien erstellt und veröffentlicht. Die wahrscheinlich verbreitetste [KZ12] ist die „Revised Panko and Halverson Taxonomy of Spreadsheet Errors“, die im Folgenden kurz vorgestellt wird.

In Abbildung 2.1 wird die überarbeitete Fehlertaxonomie für Spreadsheets von Panko dargestellt. Das erste linke Blatt, die „Culpable Violation“ sind bewuste Fehlinformationen, die den Betroffenen zu einer schlechten Entscheidung verleiten sollen. Nachdem bei einer böswilligen Manipulation diese auch bei einer automatisierten Prüfung verschleiert würden, sind im Rahmen dieser Arbeit nur die „Blameless Errors“ interessant.

Eine Unterkategorie davon, die quantitativen Fehler, führen sofort zu einem falsches Resultat, das angezeigt wird. So zum Beispiel, wenn man bei einer Formel durch Null teilen möchte, wird sofort „`#DIV/0`“ als Fehler angezeigt.

Da sich diese Arbeit nicht mit Formeln beschäftigt, lässt das die „Qualitative Errors“, übrig, die ein Fehlerpotenzial besitzen, jedoch nicht sofort ersichtliche Folgen haben. Eine der Unterklassen hiervon sind die „Execution Errors“, die Fehler bei der Ausführung, die beim Eintippen der Werte in das Spreadsheet, entstehen. Dazu gehören Fehler beim Zeigen oder Klicken („Pointing errors“) und Tippen („Typing errors“) sowie kurze Gedächtnisaussetzer

(„Memory lapses“). Diese Fehler können während der Eingabe jederzeit auftreten und das Finden von Einigen führt dazu, die Restlichen leichter zu übersehen [Pan08].

Die andere Unterklasse der qualitativen Fehler beinhaltet die Planungsfehler allgemein, Fehler in der Auswahl und Benutzung der Funktionen und Formeln, welche für uns jedoch nicht wichtig sind. Der Einzige für uns übertragbare Teil aus dieser Klasse ist das fehlende Domainwissen, so zum Beispiel, wenn die Person, die für die Dateneingabe zuständig ist, nicht mit den Bezeichnungen vertraut ist.

Auf diese Arbeit übertragen, haben wir die folgenden Fehler, die wir automatisiert finden können und wollen:

- Motorische Fehler
  - Verklicken
  - Vertippen
- Gedächtnisaussetzer
  - Nicht vorhandene Werte
  - Leere bzw. vergessene Zellen
- Unzureichendes Domainwissen
  - Unplausible Kombinationen
  - Fehlende Werte

3. Verwandte Arbeiten

Es gibt eine Vielzahl von wissenschaftlichen Artikeln, die sich mit der automatisierten Qualitätssicherung beschäftigen. In „Avoiding, finding and fixing spreadsheet errors – A survey of automated approaches for spreadsheet QA“ wurden zwischen 400 bis 500 Kurzfassungen durchgearbeitet [JSHW14] (S. 5), von denen 158 als relevante Quelle weiter benutzt wurden. Die darin behandelten Methoden für die Datenqualität gehören zur Visualisierung der ähnlichen Datenbereiche, des Datenflusses und die Einheiten- und Typinferenz. Es wurden leider keine automatisierten Prüfmöglichkeiten für implizit gekoppelte Daten vorgestellt.

3.1. Adding Apples and Oranges

In „Adding Apples and Oranges“[EB02] stellen Erwig und Burnett eine einheitenbasierte Betrachtungsweise von Datenwerten vor: Eine Einheit ist eine Klassifizierung der Datenwerte und wird durch Überschriften und Rückfragen an den Benutzer definiert. Sie können nach bestimmten Regeln in Bezug gesetzt werden, wodurch die Einheiten vererbt oder generalisiert werden. Die Bezüge sind Formeln, die mehrere Zellen referenzieren und somit Datenwerte in Verbindung zueinander setzen.

Wie der Titel vermuten lässt, werden Äpfel und Orangen in Bezug gesetzt, die im Beispiel für den Ertrag einer Plantage benutzt werden. Eine Möglichkeit, wie das Spreadsheet aussehen kann, sieht man in Abbildung 3.1. Es werden in einer Tabelle das geerntete Obst gegen die Monate eingetragen und es gibt verschiedene Formeln, die den Gesamtertrag verschiedener

	A	B	C	D
1		Fruit		
2		**Apple**	**Orange**	**Total**
3	**May**	8	11	=B3+C3
4	**June**	20	50	=B4+C4
5			=B3+C4	

Abbildung 3.1.: Beispiel für „Adding Apples and Oranges“, aus [JSHW14], (S. 17).

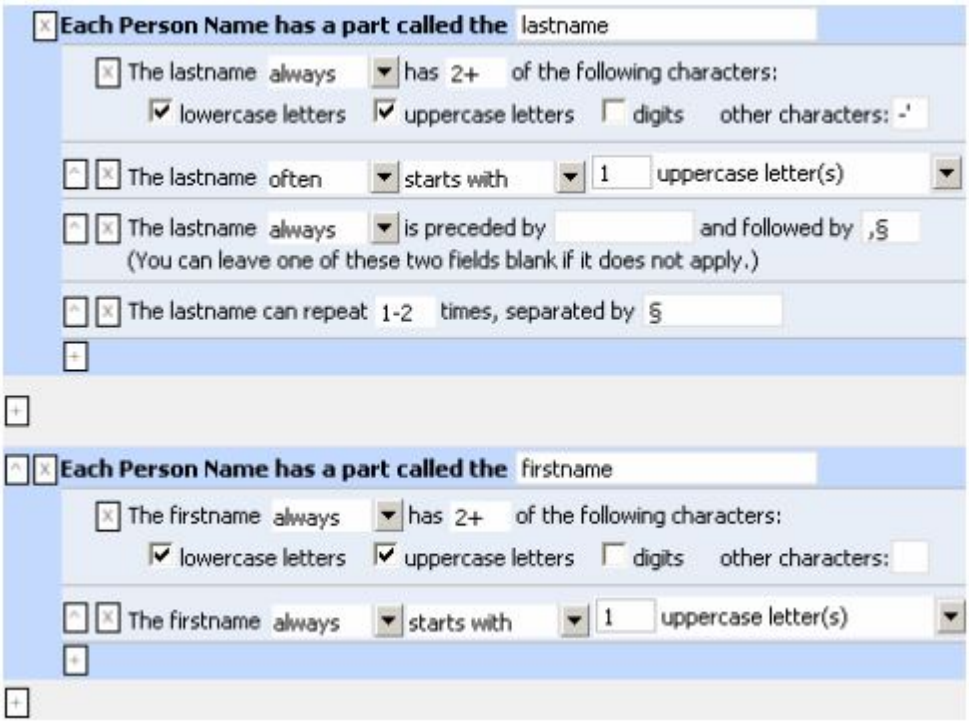

Abbildung 3.2.: Definition vom Format "Nachname, Vorname" im Tope Developement Environment, aus [SMS08] (S. 5)

Obstsorten eines Monats berechnen und Formeln die den Jahresertrag einer Sorte berechnen. Die plausiblen Generalisierungen sind somit verschiedenes Obst im gleichen Monat oder gleiches Obst über verschiedene Monate. Wenn man nun einen Fehler in einer Formel hätte, zum Beispiel eine falsche Zeile in der richtigen Spalte referenziert, gibt es keine valide Generalisierung und es liegt ein Konflikt vor.

3.2. Topes

In „Using Topes to Validate and Reformat Data in End-User Programming Tools " [AAS11] wird eine Möglichkeit vorgestellt, wie man eine reale Entität auf verschiedene Schreibweisen darstellen und übertragen kann. So kann man beispielsweise eine Firma über den „umgangssprachlichen" Namen benennen, über den offiziellen Namen, wie er in Verträgen benutzt wird, oder über die Abkürzung, die an der Börse Anwendung findet.

Um diese verschiedenen Formate zu definieren wird eine eigene Entwicklungsumgebung vorgestellt, die man in Abbildung 3.2 sehen kann. Darin kann man das Format in einer Satz-

ähnlichen Darstellung beschreiben, die intern beispielsweise in eine Reihe von regulären Ausdrücken übersetzt wird.

3.3. Self-Checking Spreadsheets

In „Self-Checking Spreadsheets: Recognition of Semantics“ [Ste13] wird eine serverseitige Validierungsplattform vorgestellt, die bei Berechnungen aus der Physik unterstützen soll. Die benutzten Formeln sollen nach den bekannten, in der Physik erlaubten, Regeln korrekt sein. Die Einheiten der physikalischen Größen, wie z.B. Druck, wird als Konzept angesehen, und eine Formel mit mehreren Konzepten überführt diese in ein Anderes. So kann z.B. das Konzept Kraft aus der Multiplikation von den Konzepten Masse und Beschleunigung bestehen.

Diese Konzepte, also die Einheiten, werden als Tokens extrahiert, auf denen ein Parser arbeitet. Jede Physik-Formel die unterstützt werden soll muss als Regel in einer formalen Grammatik definiert werden, die durch einen Parser-Generator, hier yacc, zu einem funktionierenden Parser übersetzt wird. Ausgehend von der Kombination in der Excel-Formel wird dadurch festgestellt, welche Einheit der Wert hat.

3.4. Abgrenzung

Excel bietet bereits eigene Möglichkeiten, Zellen ein bestimmtes Format vorzugeben, diese sind allerdings auf eine Zelle, beziehungsweise einen Zellbereich, beschränkt und bieten keine Unterstützung, auf den Wert in einer anderen Zelle zu reagieren. Die eine Möglichkeit ist die Formatierungsvorgabe, bei der man über ein benutzerdefiniertes Format die Struktur der Datenwerte vorgeben kann, mit Platzhaltern für Ziffern oder Buchstaben. Eine andere Möglichkeit ist die Dropdown-Auswahlliste, bei der man für eine Zelle eine Reihe von Werten definieren kann, die in einem Dropdown Menü ausgewählt werden können.

Um eine zellübergreifende Prüfung in Excel durchzuführen, müssten zur Zeit Formeln benutzt werden, die über ein `VLOOKUP`, oder eine vergleichbare Funktion, die Einschränkungen nachschlagen. Dieser Ansatz ist allerdings invasiver für den Endbenutzer, da bei dem zu prüfenden Arbeitsblatt mindestens eine weitere Spalte mit diesen Prüfformeln erstellt werden muss und das Feedback, welche Zelle den nicht plausiblen Wert hat, wird auch komplizierter. Außerdem müssten die Formeln zusätzlich geprüft werden und angepasst werden, falls sich die Reihenfolge oder die Anzahl der Spalten ändert.

4. Konzept für die Plausibilitätsprüfung

Im Gegensatz zum Großteil der Literatur (siehe Kapitel 2) konzentrieren wir uns auf Daten, die nicht durch Formeln in Bezug gesetzt werden. Obwohl für die Datenhaltung üblicherweise Datenbanken benutzt werden, kann diese auch durch Spreadsheets durchgeführt werden.

Diese Daten stellen in der Regel Gegenstände aus der Realität dar, für die auch gewisse Einschränkungen gelten. Das führt dazu, dass es meistens eine echte Teilmenge von Kombinationen gibt, die plausibel ist. Beispielsweise passen bei Automobilen nicht alle Motoren in jedes Grundmodell. Intuitiv kann diese Einschränkung als „`Teil A` schränkt die Auswahl von `Teil B` ein“ beschrieben werden, worauf dieses Konzept aufbaut.

Falls es eine Kombination gibt, die nicht plausibel ist, soll diese gefunden werden. Die Kombination kann zum Beispiel durch Unwissen, Vertippen oder ein Verrutschen in einer Zeile entstehen.

Um diese unplausiblen Kombinationen zu finden, müssen Einschränkungen definiert werden, welche Teile zueinander passen. Die Härte der Einschränkung ist jedoch vom Anwendungsfall und den Umständen abhängig und muss für jeden Einsatz selbst definiert werden. So könnte beispielsweise ein Motor mit einem preislich festgehaltenen Mehraufwand in weiteren Grundmodellen verbaut werden.

Um etwas prüfen zu können, muss es natürlich auch die Kombinationen geben. Diese werden zum Beispiel in einer einfachen tabellarischen Datenstruktur zeilenweise aufgeschrieben, bei der idealerweise für jeden Gegenstand eine eigene Spalte benutzt wird.

Folglich braucht man eine zeilenweise Betrachtung bestimmter Bereiche zur Prüfung, sowie die Möglichkeit, Einschränkungen zu definieren und gegebenenfalls zu erklären.

4.1. Beispiel

Man kann sich eine Firma in der Automobilbranche vorstellen, welche verschiedene Bauteile zusammensetzt, zur besseren Veranschaulichung werden „grobe“ Baugruppen verwendet. In diesem Sinne gibt eine Menge an angebotenen Grundmodellen für Autos und deren Motoren, die zusammen bestellt werden. Sowohl die Datenhaltung, welche Modelle existieren, als auch die Bestellung wird durch Spreadsheets realisiert.

Motor

	A	B	C
1	Art.Nr.	Grundmodell	Erklärung
2	1111	2323, 3434	
3	2222	3434	Bei anderen Grundmodellen Preisaufschlag

Grundmodell

	A	B
1	Art.Nr.	Name
2	2323	Flitzer, Sport
3	3434	Ruhe, Sanft

Bestellung

	A	B
1	Motor	Grundmodell
2	2222	2323
3	1111	2332

Abbildung 4.1.: Minimalbeispiel für die Arbeitsblätter

Da nicht jeder Motor in jedes Grundmodell passt, gibt es hierbei gewisse Einschränkungen, die dem Fachpersonal für die Montage bestens bekannt sind. In Bezug auf das hier vorzustellende Konzept sind diese Einschränkungen bereits vermerkt und erklärt.

Die Bestellung ist eine einfache Auflistung der gewünschten Kombinationen, welche dann an die Einkaufs- beziehungsweise die Montageabteilung weitergeleitet wird. In Abbildung 4.1 sieht man, wie die Arbeitsblätter aussehen könnten.

Um zu prüfen, ob die Kombinationen plausibel sind, betrachtet man jede Zeile der Bestellung: Für die Erste wäre das der Motor 2222 und das Grundmodell 2323. Mit diesen Informationen kann man nun nachschlagen, ob Einschränkungen ausgeübt werden. Der Motor 2222 übt eine Einschränkung gegenüber dem Grundmodell aus; mit diesem Typ ist nur 3434 plausibel. Das Grundmodell ist 2323, passt also nicht zusammen und man sollte diese Zeile anschließend gegebenenfalls korrigieren. Nachdem das Grundmodell keine weiteren Einschränkungen ausübt ist damit die Prüfung dieser Zeile fertig.

Bei der zweiten Zeile gibt es den Motor 1111 und das Grundmodell 2332, bei dem es einen „Zahlendreher“ gab. Für den Motor 1111 passen die Grundmodelle 2323 und 3434, zwar ist das Grundmodell 2332 ein nicht plausibler Wert, da es diesen Wert nicht gibt, kann man unter anderem durch die Einschränkung einfach sehen, dass es 2323 seien sollte.

4.2. Benötigte Definitionen und Daten für die Einschränkungen

Die benötigten Daten werden über das Spreadsheet definiert, um sowohl eine einfache Methode der Eingabe, als auch eine gewisse Unabhängigkeit von dem benutzten Spreadsheet Programm zu haben, falls weitere Frontends für das SIF entwickelt werden. Wenn man die Einschränkungen

über Seitenleisten oder andere Möglichkeiten im Spreadsheet Programm definieren würde, gäbe es einen größeren Aufwand, die Informationen zu entkoppeln und zusätzlich einen höheren Kommunikationsaufwand zwischen unseren Komponenten fürs Front- und Backend.

Im Rahmen von einer Art Modularisierung und für die Übersichtlichkeit der Informationen im Spreadsheet gehe ich davon aus, dass für jeden Gegenstand ein eigenes Arbeitsblatt existiert, das nach diesem benannt ist. Für die Unterscheidung sollte jeder Eintrag einen eindeutigen Namen besitzen, die in einer Spalte aufgeschrieben sind, wie die Artikelnummern im obigen Beispiel. Sofern diese nicht mit allen anderen Gegenständen kombiniert werden können, kann man zusätzlich Spalten für Einschränkungen und Erklärungen definieren. Im Detail gibt es folglich vier verschiedene Spalten:

Eine Spalte mit Werten: In dieser Spalte müssen die Bezeichner für den Gegenstand stehen, beispielsweise Artikelnummern, die in dem zu prüfenden Kontext benutzt werden. Da die Überschrift von einer Art Primärschlüssel wahrscheinlich bereits existiert, aber nicht zwingend der Name des Gegenstandes ist, wird stattdessen der Name vom Arbeitsblatt benutzt. Sofern es nur diese Spalte gibt, kann mit dieser automatischen Prüfung nur nach unbekannten Werten gesucht werden, die zu einer Warnung führen.

Eine oder mehrere Spalten mit Einschränkungen: Damit kann eine zellübergreifende Überprüfung durchgeführt werden. Die Spalte muss den Namen der einzuschränkenden Menge als Überschrift enthalten (Arbeitsblattname des einzuschränkender Gegenstand = diese Überschrift). In dieser Spalte wird, für den Wert in der Zeile, eine Aufzählung plausibler Werte erwartet.

Da es bei solchen Einschränkungen nicht nur Eins zu Eins Beziehungen gibt, sollten mehrere Werte definiert werden können. Die Trennung der Werte erfolgt durch ein ';', da es auch aus dem normalen Schriftgebrauch eine Trennung symbolisiert, jedoch seltener benutzt wird als ein Punkt oder ein Komma.

Falls man das ';' im Wert benötigt, kann es mit einem führenden '\' als Zeichen benutzt werden.

Zudem besteht die Möglichkeit reguläre Ausdrücke in der Java Syntax als plausiblen Wert anzugeben, was Power Usern eine Möglichkeit bietet, diese Definition kompakter zu halten oder Wertebereiche anzugeben.

Eine Spalte mit Erklärungen: Sofern es einen Verstoß gibt, wird diese Erklärung in der Warnung mit angegeben. Über diese kann man die Härte der Einschränkung erklären, die

Identifikation von falsch positiven Befunden vereinfachen oder gegebenenfalls benutzte reguläre Ausdrücke erläutern.

Zu prüfende Spalten: Es müssen auch die zu prüfenden Spalten definiert werden. Die Überschrift beinhaltet was in dieser Spalte steht.

Das kann zum einen der Name eines Gegenstandes sein, der eine Einschränkung ausübt. In diesem Falle muss die Überschrift der Spalte gleich dem Namen des Arbeitsblattes sein, in dem dieser Gegenstand aufgezählt ist.

Zum anderen kann es ein eingeschränkter Gegenstand sein. Dies ist der Fall, wenn eine andere zu prüfende Spalte eine Einschränkung besitzt mit dem gleichen Namen wie die Überschrift dieser Spalte.

4.3. Prüfung

Aus den Definitionen kann man zwei verschiedene Datentypen herausarbeiten. Zum einen ist das die Repräsentation der zu prüfenden Werte, die durch die Überschrift und den Ist-Wert als 2-Tupel vorliegen. Zum anderen sind das die Einschränkungen. Um diese vom Spreadsheet separiert betrachten zu können, benötigt man den 2-Tupel, der die Einschränkung ausübt und den einzuschränkenden 2-Tupel.

Während der Prüfung wird für jedes Arbeitsblatt, das zu prüfende Spalten enthält, folgendes für jede Zeile durchgeführt:

- Gegebene Daten der zu prüfenden Spalten als 2-Tupel extrahieren.
- Mit jedem 2-Tupel nachschauen, ob dadurch Einschränkungen ausgeübt werden.
- Überprüfen, ob die gefunden Einschränkungen durch alle Tupel erfüllt werden.

5. Implementierung

In diesem Kapitel werden das SIF und das SIFEI zunächst vorgestellt und anschließend werden die von mir durchgeführte Änderungen dargestellt.

5.1. Spreadsheet Inspection Framework (SIF)

Das SIF ist das Backend, das die eigentliche Prüfung durchführt. Es ist nach der Metapher einer Kfz-Werkstatt aufgebaut und in Java geschrieben. Es benutzt eine Reihe von Bibliotheken, darunter die Apache commons für das Escapen bei der XML Übertragung und das Aufzeichnen der internen Meldungen, und JAXB (Java Architecture for XML Binding) für die Deserialiserung der XML-Dateien.

5.1.1. SIF Ist-Zustand

Von der Metapher ausgehend, wird die Prüfung explizit von außerhalb (durch den Benutzer) angestoßen. Für diese wird ein Prüfling, in diesem Fall ein Spreadsheet, benötigt, an dem eine Reihe von Prüfungen durchgeführt werden soll. Den groben Ablauf der Prüfung kann man in Abbildung 5.1 Ablauf einer Inspektion sehen. Der Aufbau vom SIF und der Kontroll- und Datenfluss wird in der Abbildung 5.2 Aufbau des SIF (S. 21) veranschaulicht. Zunächst wird der Ablauf mit den Aufgabenbereichen der Pakete und Klassen erklärt, anschließend wird kurz der Inhalt und die Definition der Prüfungen erläutert.

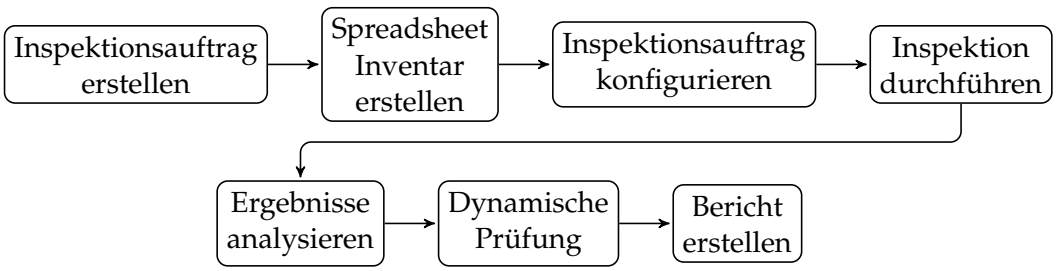

Abbildung 5.1.: Ablauf einer Inspektion, angelehnt an [Lem13], S. 24

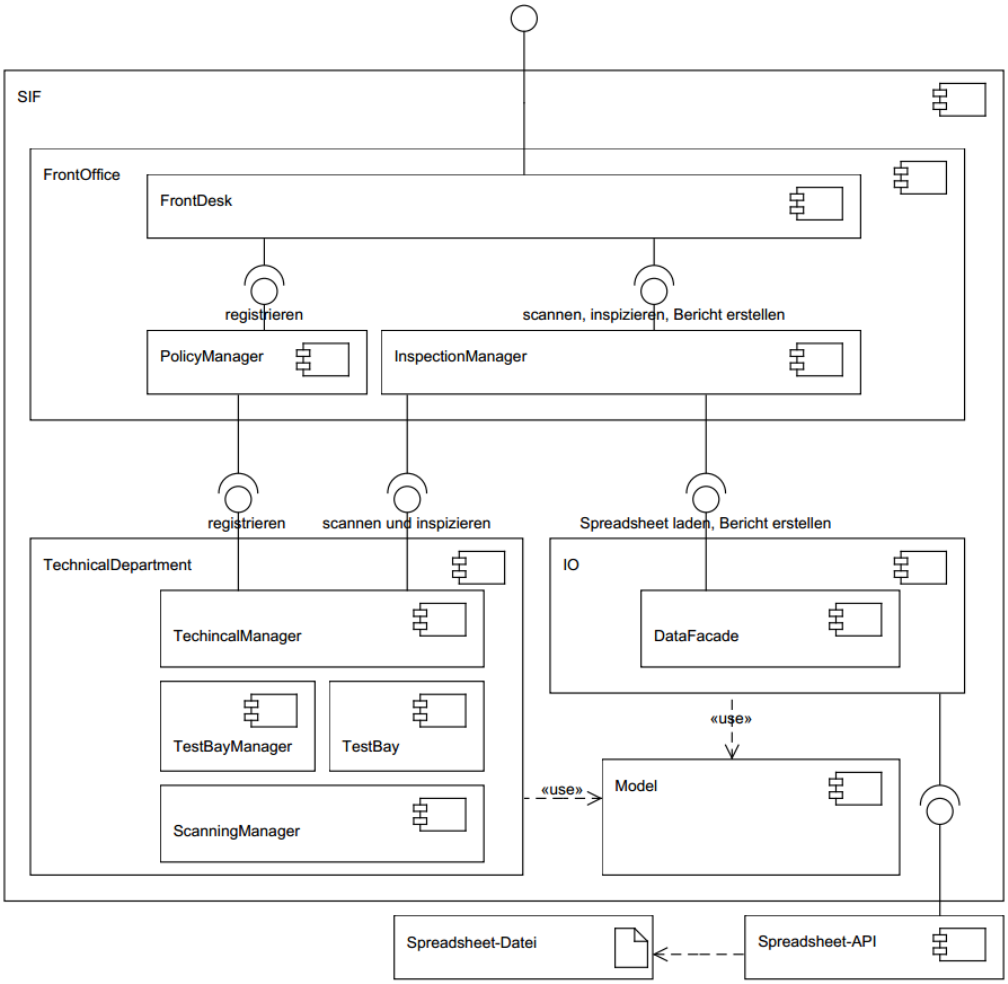

Abbildung 5.2.: Aufbau des SIF, aus [Lem13], S.31

Ablauf:

Die Prüfungen werden als Inspektionsauftrag durch eine XML-Datei definiert, die für die Benutzung durch das Frontend über einen lokalen Port übertragen wird. Die Datei beinhaltet die Informationen, wo das Spreadsheet abgespeichert ist und gegebenenfalls die Parameter für die verschiedenen `Policys` der statischen und der dynamischen Prüfungen.

Dieser Prüfling wird am `FrontDesk` entgegen genommen und über die Klassen im IO Paket eingelesen und in das eigene Datenmodell übertragen. Die Prüfungen müssen als `Policy` vom `PolicyManager` bei den Managern im Paket `TechnicalDepartment` registriert werden. Die registrierten Prüfungen werden anschließend im `TestBay` durch die, der `Policy` entsprechenden, `Facility` durchgeführt.

Nach der Prüfung wird ein abschließender Bericht erstellt, der die charakteristischen Zellen für die dynamische Prüfung aufzählt und vorhandene `Violations` auflistet. Der Bericht wird von „Hand“ wahlweise als HTML oder XML in der `DataFacade` erstellt.

Policys:
Die in der `Policy` als `ConfigurableParameter` annotierten Attribute, können durch die XML-Datei definiert werden. Die Deserialisierung wird durch die JAXB Bibliothek durchgeführt. Die Übertragung der `ConfigurableParameter` von der `Policy` an die `Facility` wird mit Reflection durchgeführt, deren Routine in der `AbstractTestFacility`, der Vaterklasse aller `Facilitys`, definiert ist.

Bei der dynamischen Prüfung können zum einen Testwerte für die Zellen angegeben werden, die in einer Formel benutzt werden, dazu gehören die `InputCell`, in der referenzierte Daten stehen, die `IntermediateCell`, die Berechnungen durchführt und deren Ergebnis auch referenziert wird, und die `ResultCell`, die nur Berechnungen durchführt und darstellt.

Zum anderen können für beliebige Zellen Vor- und Nachbedingungen und Invarianten definiert werden. Die Vorbedingung muss vor dem Berechnen der Formeln erfüllt sein, die Nachbedingung muss danach erfüllt sein. Eine Invariante muss sowohl davor, als auch danach erfüllt sein.

Die Einschränkungen können als `BinaryRelation` angegeben werden (größer/kleiner als X, gleich Y) oder als `TernaryRelation` (enthalten im Intervall $X - Y$).
Zusätzlich gibt es bereits 3 Policies für statische Prüfungen:

ReadingDirectionPolicyRule Überprüft die Referenzierungsrichtung der Formeln, ob sich diese nur nach Links und Oben beziehen.

FormulaComplexityPolicyRule Überprüft die Anzahl der Operatoren einer Formel und die maximale Verschachtlung der Funktionen.

NoConstantsInFormulasPolicyRule Überprüft, ob Konstanten „hardcoded“ in der Formel sind.

5.1.2. SIF Erweiterung

Um in der Metapher zu bleiben, wurde ein neuer Prüfstand hinzugefügt, der die Plausibilitätsprüfung zellübergreifend realisieren kann. Dies ist die `SanityTestFacility` deren Parameter, also die markierten Zellen, über die `SanityPolicyRule` definiert werden.

Die Deserialisierung der `SanityPolicyRule` wird ebenfalls durch JAXB übernommen, und beinhaltet Listen für die Überschriften der Werte, Einschränkungen, Erklärungen und zu prüfenden Zellen.

1: Einlesen der Einschränkungen
2: Für jedes Arbeitsblatt mit zu prüfenden Spalten:
3: Für jede Zeile mit mindestens einem Wert:
4: Werte als Überschrift - Werte Tupel extrahieren
5: Für jeden Tupel nachschauen, ob er Einschränkungen ausübt
6: Für jede Einschränkung prüfen, ob die Tupel sie erfüllen
7: Für jede nicht erfüllte Einschränkung einen Verstoß erstellen
8: Falls Warnungen aktiviert sind, für jeden unbekannten Wert
↳ einen Verstoß erstellen
9: Alle gefundenen Verstöße an den `FrontDesk` weitergeben

Abbildung 5.3.: Pseudocode der Prüfung

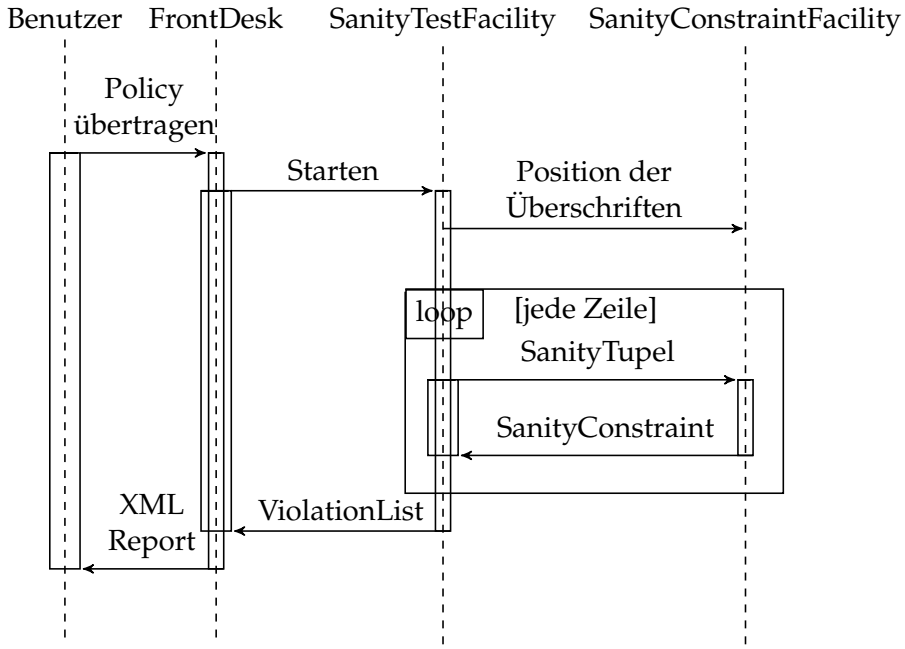

Abbildung 5.4.: Sequenzdiagramm der Prüfung

SanityConstraint
definedFrom : String
value : String
explanation : String
definedFor : String
constraints : String[]

SanityTuple
header : String
value : String
parent : `Cell`

Abbildung 5.5.: Datenstrukturen für die Plausibilitätsprüfung

In den Abbildungen 5.3 Pseudocode der Prüfung (S. 23) und 5.4 Sequenzdiagramm der Prüfung (S. 23) wird der Ablauf verkürzt dargestellt und für die Übersichtlichkeit wurden einige Klassen in der Hierarchie bis zum Start der Prüfung ausgelassen. Im Detail wird dieser so in der Umsetzung durchgeführt:

Über mehrere Ebenen wird vom `FrontDesk` die Hauptroutine von der `SanityTestFacility` aufgerufen. Hier werden zuerst die Informationen, die von der `SanityPolicyRule` übertragen wurden, auf die vorhandenen Arbeitsblätter bezogen und sortiert. Anschließend werden die Informationen, welche Arbeitsblätter Werte, Einschränkungen und Erklärungen besitzt, an die `SanityConstraintFacility` weitergegeben, die diese in das eigene Datenmodell überträgt. Die für die Plausibilitätsprüfung benötigten Datenstrukturen werden in Abbildung 5.5 dargestellt.

Nach dem Einlesen der Einschränkungen werden in der `SanityTestFacility` die zu prüfenden Spalten gesammelt und pro Zeile werden die Zelleninhalte als `SanityTuple` extrahiert. Mit dem Tupel kann man mit der `SanityConstraintFacility` nachschlagen, ob er eine Einschränkung, einen `SanityConstraint`, ausübt und/oder ein unbekannter Wert ist. Sofern die Warnungen aktiviert sind, wird an dieser Stelle für die unbekannten Werte ein Verstoß generiert.

Wenn man für jeden Tupel der Zeile nachgeschlagen hat, ob es eine Einschränkung ausübt, wird mit der gefundenen Menge der `SanityContraints` geprüft, ob die Tupel diese erfüllen. Wenn der `header` des Tupels dem `definedFor` des `SanityConstraint` entspricht, muss der `value` des Tupel einer der Werte sein, die in `constraints` aufgeführt sind, ansonsten wird ein Verstoß generiert.

Abschließend werden die gefundenen Verstöße als `ViolationList` über das Return Statement von der Hauptroutine der `SanityTestFacility` an den `FrontDesk` weitergegeben.

Die Datenhaltung und das Einlesen der Einschränkungen wird separat vom Datenmodell des SIF gehalten, was einen kleinen Bruch mit der aktuellen Durchführung darstellt, in der das „Erkennen“ von besonderen Zellen bisher beim Einlesen in das Datenmodell vom SIF stattfindet. Allerdings werden Informationen aus der `Policy` benötigt, die zu dem Zeitpunkt noch nicht

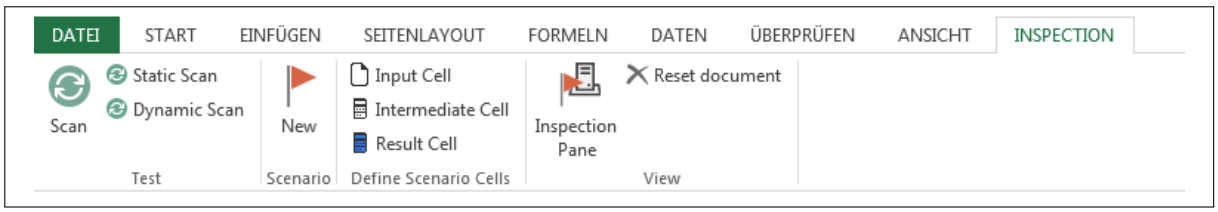

Abbildung 5.6.: Ribbon des SIFEI

(einfach) bereitstehen und nicht, wie die Zellen für die dynamische Prüfung, durch (nicht) referenziert werden erkannt werden können.

Ebenso werden die Zelldefinitionen im Datenmodell vom SIF selbst gehalten, jedoch gab es für mich keinen plausiblen Weg meine Definitionen dort sauber zu integrieren, abgesehen davon, dass es bei einer möglichen Erweiterung der Prüfung von explizit gekoppelten Daten beliebige Kombinationen aus den Zelltypen für die dynamische Prüfung und der Plausibilitätsprüfung entstehen können.

Auch bei der Art der Prüfung entfernt sich diese Arbeit etwas von dem gegebenen Konzept, da sie bisher durch `Conditions` definiert wurden, die eine `Relation` auf eine Zelle bezogen. Um in diesem Ansatz zu bleiben, hätte man nach dem Nachschlagen der Einschränkungen für jeden Teil der Einschränkung eine `Condition` erstellen müssen, von denen nur eine erfüllt sein müsste. Alternativ hätte man im Frontend die Berechnung durchführen können, welche Zelle durch welchen Wert wie eingeschränkt wird und diese als `Invarianten` anzugeben, jedoch war es meines Erachtens sowohl praktischer als auch portabler, diese Prüfung im Spreadsheet Programm unabhängigen SIF durchzuführen.

5.2. SIF Excel Integration (SIFEI)

Das SIFEI wurde geschaffen um eine endbenutzerfreundliche Benutzung des SIF zu ermöglichen. Es bietet die Möglichkeit die dynamischen Prüfungen als sogenanntes Szenario zu definieren, die statischen Prüfungen sind aufgrund der Übertragung der Prüfaufträge immer vorhanden. Als Plattform wurde Microsoft Excel gewählt, da es eine höhere Verbreitung besitzt [Dou13]. Von daher ist es ein in C# entwickeltes Office Addin.

5.2.1. SIFEI Ist-Zustand

In Abbildung 5.6 Ribbon des SIFEI sieht man das Ribbon, das von SIFEI bereitgestellt wird. Per Scan wird eine Kopie des Spreadsheets im temporären Verzeichnis erstellt und ein Prüfauftrag

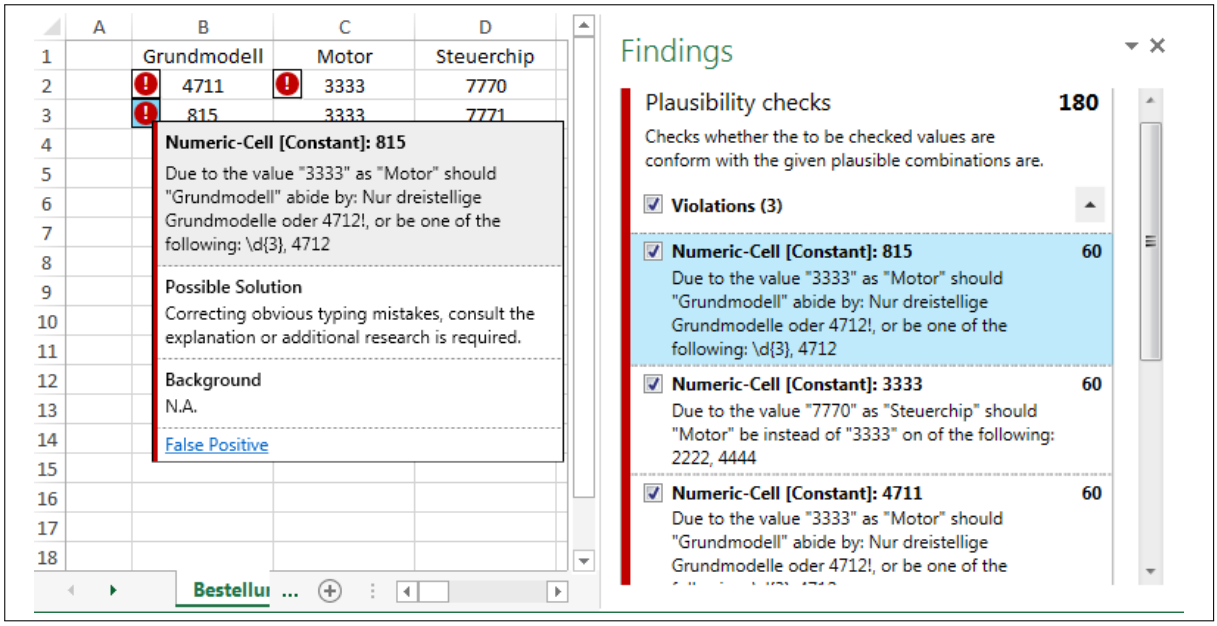

Abbildung 5.7.: Unplausible Kombinationen

wird für diese Kopie gestartet. Die Verstöße werden danach über XML vom SIF an das SIFEI übertragen und im eigentlichen Spreadsheet dargestellt.

Es bietet zudem die Möglichkeit die dynamische Prüfung, hier Szenario genannt, zu definieren. Dafür muss man zuerst die relevanten Zellen als Eingabe-, Zwischen- oder Ergebniszelle definieren. Danach kann man über `New` bei `Scenario` direkt in den Zellen des Spreadsheets die Testwerte eingeben.

In Abbildung 5.7 Unplausible Kombinationen sieht man, wie die Verstöße dargestellt werden. Sie werden zum Einen über ein Ausrufezeichen in der verursachenden Zelle dargestellt, bei dem über einen Rechtsklick die Details angezeigt werden. Zum Anderen kann die gesamte Liste der Verstöße über einen Linksklick auf das Ausrufezeichen in einer Seitenleiste angezeigt werden. Die Darstellung ist bereits so vorhanden gewesen, jedoch ist der Inhalt der Verstöße bereits von meiner Erweiterung.

5.2.2. SIFEI Erweiterung

Wie in Abbildung 5.8 Erweitertes Ribbon des SIFEI (S. 27) zu sehen ist, wurde es um vier Zelldefinitionen erweitert, die für die Plausibilitätsprüfung benötigt sind. An sich ist die gesamte Spalte für uns interessant, aufgrund der variablen Bereiche in der eine Tabelle erweitert oder gelöscht werden kann, ist das Markieren des gesamten Bereiches nicht sinnvoll.

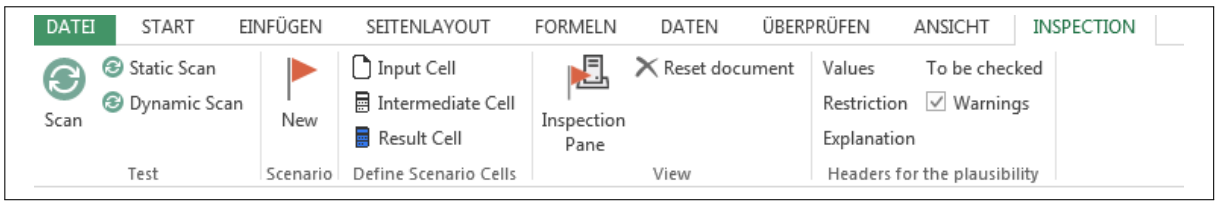

Abbildung 5.8.: Erweitertes Ribbon des SIFEI

Aufgrund des erhöhten Rechenaufwandes beim Markieren einer kompletten Spalte, konzentrieren wir uns von daher auf eine Zelle pro Spalte, die Überschrift. Es macht in der Umsetzung im Backend zwar keinen Unterschied, welche Zelle in der Spalte markiert wurde, doch die Überschrift bietet hierbei die beste Möglichkeit, die markierte Zelle wiederzufinden.

Somit war es nicht nötig, von den existierenden Designpatterns abzuweichen und die vier neuen Zellen konnten analog zu den vorhandenen Datenstrukturen der dynamischen Prüfung implementieren. Die neuen Definitionsmöglichkeiten sind:

Values: Die Überschrift der Spalte mit den eindeutigen Bezeichnungen.

Restriction: Die Überschrift der Spalte mit den plausiblen Werten.

Explanation: Die Überschrift der Spalte mit den Erklärungen.

To be checked: Die Überschriften der zu prüfenden Spalten.

Warnings: Ob unbekannte Werte eine Warnung verursachen sollen.

Entsprechend mussten die Routinen für das Serialisieren und Deserialisieren der XML Daten erweitert werden, die sowohl für das Abspeichern in der Excel Datei benötigt werden als auch für das Übertragen zum SIF. Aufgrund der umliegenden Einschränkungen bei der vorhandenen Übertragung der XML Daten an das SIF werden die Informationen für die Plausibilitätsprüfung an das Ende der dynamische Prüfung abgespeichert.

6. Evaluation

Um hauptsächlich die Umsetzung auf Tauglichkeit zu prüfen, wurde eine Evaluation durchgeführt. Im Besonderen sollte überprüft werden, ob das grundliegende Konzept zum einen verständlich und anwendbar ist, zum anderen ob die Umsetzung dem Konzept gerecht wird und dem Benutzer in der Qualitätssicherung hilft.

Es gab einen Pilotprobanden, der einige ungünstige Textstellen erkannte, die anschließend geändert wurden. Bei zwei Aufgaben wurde, von den Anmerkungen des Piloten ausgehend, nachgebessert. Die anderen beiden Aufgaben blieben praktisch gleich und der Pilot wird bei diesen mit ausgewertet. Nach der Korrektur der Aufgaben gab es fünf reguläre Probanden. Die Probanden waren überwiegend männlich, zwischen 20 und 25 Jahren waren hauptsächlich in den Studiengängen Informatik und Softwaretechnik eingeschrieben, sowie einen aus dem Studiengang Luft- und Raumfahrttechnik.

6.1. Durchführung

Für das Experiment wurde ein Rechner mit Windows 8 benutzt, auf dem Excel 2013 installiert war. Bis auf das Ribbon „Start“ wurden die anderen Excel eigenen Ribbons ausgeblendet,um der Übersichtlichkeit beizutragen, da die Usability des zu Grunde liegende Ribbons bereits in anderen Arbeiten evaluiert wurde [Dou13, Sch14].

Den Probanden wurde zuerst an einem Benutzungsbeispiel das Konzept und die Benutzung des SIFEI veranschaulicht, das in vier Aufgaben umgesetzt werden sollte. Abschließend gab es einen speziellen Fragebogen zum Konzept und der Umsetzung, und der allgemeinen Benutzung von Excel.

Die Aufgaben waren in sich abgeschlossen und es gab keine direkte Abhängigkeit zwischen den verschiedenen Dateien. Von den Aufgaben der Evaluation werden in 6.3 Beobachtungen (S. 30) zunächst die Ziele und anschließend die Beobachtungen bei der Durchführung erläutert.

Frage	∅	Beste	Schlechteste
Die Aufgaben waren verständlich formuliert	1.5	1	2
Das Konzept der Plausibilität war verständlich	1.3	1	2
Das Markieren der Überschriften war für mich leicht	1.5	1	3
Die Strukturierung der Datenwerte war für mich leicht	1.6	1	3
Die Definitionsmöglichkeit für die Einschränkungen ist gut gelöst	2.3	1	4
Ich würde SIFEI für die Prüfung eigener Spreadsheets benutzen	1.8	1	3

Tabelle 6.1.: Ergebnisse des speziellen Fragebogen

6.2. Fragebogen

In der Tabelle 6.1 Ergebnisse des speziellen Fragebogen werden die Ergebnisse dargestellt. Es gab fünf Auswahlmöglichkeiten, es wird Eins für die beste Wertung und Fünf als Schlechteste angesehen. Im Folgenden werden die Fragen des speziellen Fragebogens und die zugehörigen Antworten wiedergegeben.

Kommt Ihnen das grundliegende Konzept der Einschränkungen („`Gegenstand A` schränkt die Auswahl von `Gegenstand B`“) sinnvoll vor?

Alle Probanden gaben Ja an, Anmerkungen gab es keine.

Haben Sie bereits nicht plausible Daten in einem Spreadsheet vorgefunden?

Zwei Probanden gaben Ja an.

Benutzen Sie bereits eigene Methoden oder andere Werkzeuge, um Daten plausibel zu halten?

Ein Proband gab an, gesunden Menschenverstand in einer manuellen Prüfung zu benutzen.

Könnten Sie sich grundsätzlich vorstellen die Grundlagen von regulären Ausdrücken zu erlernen oder würden Sie die Muster lieber wie in einer Excel-Formel definieren?

Nachdem es durchaus Personen gibt, denen reguläre Ausdrücke nicht bekannt sind, gab es zunächst eine kurze Erklärung:

Sofern es Muster in den Datenwerten gibt, kann man diese aktuell durch einen regulären Ausdruck beschreiben. Reguläre Ausdrücke haben den Vorteil, dass sie vieles beschreiben

können und dabei sehr kompakt sind, worin jedoch auch ein Nachteil liegt, da man diese „Sprache“ erlernen müsste.

Einige relevante „Abkürzungen“ wären [a-z] für einen Buchstaben zwischen a und z; \d für eine Zahl; und \s für Leerzeichen, sogenannte Whitespaces. Über geschweifte Klammern kann man angeben, wie oft ein Zeichen wiederholt werden soll, als Abkürzungen gibt es das + für mindestens einmal und * für beliebig oft, auch Nullmal. So „erkennt“ zum Beispiel „[WZ]ahl\s\d+“ entweder Wahl oder Zahl, gefolgt von einem Leerzeichen und mindestens einer Zahl.

Fünf der sechs Probanden gaben an, dass sie sich die regulären Ausdrücke vorstellen können.

Allgemeiner Fragebogen:

Es wurde zudem ein Fragebogen zur allgemeinen Benutzung von Excel, den es am Institut für Softwaretechnologie bereits gab, ausgefüllt. Die eine Hälfte der Probanden waren erfahrene Excel-Benutzer, deren Anwendungsfälle von wissenschaftlichen Auswertungen zu einfacher Datenhaltung reichten. Die andere Hälfte hatte noch nicht viel mit Excel gearbeitet. Das äußerte sich jedoch nur bei der Geschwindigkeit der Dateneingabe und der Navigation in Excel sowie der Anpassung der Spaltenbreiten für die Eingaben.

6.3. Beobachtungen

Die Ziele und Beobachtungen bei den Aufgaben waren:

Aufgabe 1 war die grundliegende Struktur für die Datenhaltung, damit die Einschränkungen über das Worksheet definiert werden können. Dafür sollten die Arbeitsblätter und die Überschriften richtig benannt und mit dem SIFEI markiert werden.

Beim Pilotprobanden ist aufgefallen, dass die Daten in der Aufgabenstellung schlecht formatiert waren. Nachdem dies behoben wurde, machten die weiteren Probanden keine Fehler in der Struktur.

Es wurden von allen zu viele Spalten als Werte Spalte markiert. Es wäre nur die Bestellnummer nötig gewesen, aber es wurden bei der Karosserie zusätzlich der Name und bei den Reifen die Größe markiert. Zudem wurde die Werte Spalte bei den Reifen von drei Probanden vergessen, vermutlich, da sie keine Einschränkung ausübten.

Aufgabe 2 war das Übertragen von Daten mit einigen simulierten Tippfehlern, die mit dem SIF geprüft und korrigiert werden sollten.

Beim Pilotprobanden gab es eine Kombination aus der Zahl 1 und dem Buchstaben l, um Tippfehler zu provozieren. Aufgrund der verursachten Verwirrung wurde das zu simulierten Tippfehlern geändert und der Pilot wird bei dieser Aufgabe nicht weiter ausgewertet.

Drei der fünf Probanden hatten die Überschriften zuerst fälschlicherweise als Werte Spalte markiert, nachdem die Prüfung nichts ergab wurden sie, nach Nachschlagen in der Beschreibung, als zu prüfende Spalte markiert.

Drei Probanden korrigierten alle Tippfehler auf die gleiche Art, einer änderte den Zahlendreher von 505 $\leftrightarrow$ 550 zu 500 geändert (beide waren in den plausiblen Werten). Der Letzte übersprang das Korrigieren vollständig.

Ein Proband schrieb die Spalten in einer geänderten Reihenfolge ab, was über den Bezug durch die Überschrift bei der Prüfung keinen Unterschied machte.

Aufgabe 3 war das Prüfen einer größeren Ansammlung von teils fehlerhaften Daten, die korrigiert werden sollte. Es gab mehrere Tippfehler, einige Zeilen bei denen nur ein Wert unplausibel war, eine Zeile, die nur unbekannte Werte enthielt und bei der letzten Zeile war ein Wert in darunterliegende Zeile verrutscht.

Fünf der sechs Probanden ersetzten die Werte bei einer eindeutigen Alternative durch die Einschränkung, einer davon hatte durch verkette Abhängigkeiten[1] an einem Wert mehrfach Änderungen durchgeführt.

Das Verrutschen wurde nur von zwei Probanden als solches bearbeitet.

Aufgabe 4 war das Erweitern einer Einschränkung.

Die Hälfte der Probanden ersetzte die Einschränkung durch den genannten Wert, einer von diesen entfernte zudem den plausiblen Wert aus einer anderen Zeile. Die andere Hälfte fügte den neuen Wert der bestehenden Einschränkung hinzu.

6.4. Fazit

Das grundliegende Konzept wurde positiv aufgenommen, jedoch ist die Anwendungsmöglichkeit in der aktuellen Form eingeschränkt.

Das Definieren der Einschränkungen ist nicht ideal gelöst, konkrete Verbesserungsmöglichkeiten wurden keine genannt. Dazu kommt, dass die Aufzählungen der Daten nicht immer in

[1] A schränkt B ein, B schränkt C ein und nur B ist falsch, aber es zeigt einen Verstoß bei B und C an

einer Arbeitsmappe vorhanden sind, oder, wenn sie es sind, einen höheren Wartungsaufwand durch die Redundanz mit sich führen. Somit wäre es praktischer, wenn die Einschränkungen außerhalb des aktuellen Spreadsheets liegen könnetn.

Bei der Darstellung der Verstöße könnte man die farbige Markierung benutzen um die Anzahl der Verstöße darzustellen, zum Beispiel wenn ein Wert sowohl unbekannt ist, als auch eine Einschränkung nicht erfüllt.

Um das „falsch korrigieren“ zu verhindern, könnte man sich eine bessere visuelle Darstellung der Abhängigkeiten vorstellen, ähnlich der Visualisierung von Excel für den Datenfluss von Formeln. Zudem könnte man das Dokument erneut automatisch prüfen, sobald etwas geändert wurde.

Hinzu kommt, dass die schriftliche Form der Bedienungsanleitung als nicht ideal aufgefasst wurde, ein kurzer Workshop oder ein Einführungsvideo wäre zwei Probanden lieber gewesen, die anderen machten dazu keine Aussage.

7. Zusammenfassung und Ausblick

Nachdem es zu der genauen Problemstellung, der Plausibilitätsprüfung implizit gekoppelter Spreadsheet-Daten keine, für mich auffindbaren, Artikel existierten, wurde das intuitive Konzept, „`Teil A` schränkt die Auswahl von `Teil B` ein“, für die Erweiterung benutzt.

Die Umsetzung von diesem Konzept konzentrierte sich auf das SIF, nachdem es sowohl Plattform- als auch Spreadsheet Programm unabhängig ist. Die Erweiterung des SIFEI war vergleichsweise einfach gehalten und wurde im vorhandenen Look and Feel der Integrationskomponente gehalten.

Die Evaluation zeigte einige Schwächen bei der Definitionsart der Einschränkungen auf, sowohl was die Verständlichkeit angeht als auch für die Verfügbarkeit der Daten in einem “normalen“ Gebrauch von Spreadsheets für die Datenhaltung, erweckte trotzdem einen positiven Eindruck, so dass es bei der Qualitätssicherung nützlich sein kann.

Ausblick

Von der Evaluation ausgehend kann man sich die folgenden Erweiterungen vorstellen:

- „Auslagern“ der Einschränkungen in separate Spreadsheets oder XML-Dateien
- Eine visuelle Darstellung der Abhängigkeiten der Einschränkungen
- Die farbige Markierung über die `Severity` an die Anzahl der Verstöße anpassen
- Eine Art „Echtzeitprüfung“, die nach jeder Änderung durchgeführt wird

Von der Aufgabe zur grundliegenden Datenhaltung könnte man sich eine Erweiterung der Datenstruktur, zu etwas den Topes [AAS11] ähnlichem, für die Gegenstände überlegen. So könnte man sowohl über einen Primärschlüssel als auch über eine Kombination von anderen Namen, einen Gegenstand identifizieren. Wenn man dies über etwas, dem Tope Developement Environment ähnlichem, definieren kann, würde das zudem für das Spreadsheet weniger instrusiv sein.

Um die Fehlerkorrektur bei verketteten Einschränkungen zu unterstützen, könnte man, zusätzlich zur allgemeinen Darstellung, die Kombinationen vorschlagen, die mit den wenigsten Änderungen den größten Teil der Verstöße korrigiert.

Weitergehend könnte man die Prüfung auf explizit gekoppelte Daten erweitern, bei der die Parameter als Gegenstände angesehen werden, die durch die Formel kombiniert werden.

A. Anhang

A.1. Inhalt der Daten-CD

Die Ergebnisse dieser Arbeit sind auf einer CD dokumentiert, die folgenden Inhalt besitzt:

- Aufgabenstellung dieser Bachelorarbeit im PDF-Format
- Kurzfassung und Abstract als Textdatei
- Diese Ausarbeitung im PDF-Format
- Das Projekt für das SIF
- Das Projekt für das SIFEI
- Installationsdateien
- Schnellstart Anleitung
- Aufgabenstellung und Fragebogen der Evaluation

Literaturverzeichnis

[AAS11] A. Asavametha, P. Ayyavu, C. Scaffidi. No Application Is an Island: Using Topes to Transform Strings during Data Transfer. In *Information Science and Applications (ICISA), 2011 International Conference on*, S. 1–10. 2011. doi:10.1109/ICISA.2011.5772325. (Zitiert auf den Seiten 14 und 33)

[Cro07] G. J. Croll. The Importance and Criticality of Spreadsheets in the City of London. *CoRR*, abs/0709.4063, 2007. (Zitiert auf Seite 6)

[Dou13] E. Doust. *Visualisierung von Fehlern in Spreadsheets*. Bachelorarbeit, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart, 2013. (Zitiert auf den Seiten 6, 25 und 28)

[EB02] M. Erwig, M. Burnett. Adding Apples and Oranges. In *Practical Aspects of Declarative Languages, 4th International Symposium, PADL 2002*, S. 173–191. 2002. (Zitiert auf Seite 13)

[JSHW14] D. Jannach, T. Schmitz, B. Hofer, F. Wotawa. Avoiding, finding and fixing spreadsheet errors – A survey of automated approaches for spreadsheet QA. *Journal of Systems and Software*, 2014. doi:http://dx.doi.org/10.1016/j.jss.2014.03.058. URL http://www.sciencedirect.com/science/article/pii/S0164121214000788. (Zitiert auf Seite 13)

[KZ12] D. Kulesz, S. Zitzelsberger. Investigating Effects of Common Spreadsheet Design Practices on Correctness and Maintainability. 2012. (Zitiert auf Seite 11)

[Lem13] M. Lemcke. *Dynamische Prüfung von Spreadsheets*. Diplomarbeit, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart, 2013. URL http://elib.uni-stuttgart.de/opus/volltexte/2013/8722. (Zitiert auf den Seiten 6, 20 und 21)

[PA10] R. R. Panko, S. Aurigemma. Revising the Panko–Halverson taxonomy of spreadsheet errors. *Decision Support Systems*, 49(2):235 – 244, 2010. doi:http://dx.doi.org/10.1016/j.dss.2010.02.009. URL http://www.sciencedirect.com/science/article/pii/S0167923610000461. (Zitiert auf Seite 11)

[Pan08] R. R. Panko. Spreadsheet Errors: What We Know. What We Think We Can Do. *CoRR*, abs/0802.3457, 2008. (Zitiert auf den Seiten 6 und 12)

[PBL09] S. G. Powell, K. R. Baker, B. Lawson. Impact of errors in operational spreadsheets. *Decision Support Systems*, 47(2):126 – 132, 2009. doi:http://dx.doi.org/10.1016/j.dss.2009.02.002. URL http://www.sciencedirect.com/science/article/pii/S0167923609000335. (Zitiert auf Seite 6)

[Sch14] J. Scheurich. *Benutzerschnittstelle für einen Spreadsheet-Prüfstand*. Bachelorarbeit, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart, 2014. (Zitiert auf den Seiten 6 und 28)

[SMS08] C. Scaffidi, B. Myers, M. Shaw. Topes: Reusable Abstractions for Validating Data. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE '08, S. 1–10. ACM, New York, NY, USA, 2008. doi:10.1145/1368088.1368090. URL http://doi.acm.org/10.1145/1368088.1368090. (Zitiert auf Seite 14)

[Ste13] M. Stewart. Self-checking Spreadsheets: Recognition of Semantics. *Procedia Computer Science*, 18(0):199 – 207, 2013. doi:http://dx.doi.org/10.1016/j.procs.2013.05.183. URL http://www.sciencedirect.com/science/article/pii/S1877050913003268. 2013 International Conference on Computational Science. (Zitiert auf Seite 15)

[Zit12] S. Zitzelsberger. *Fehlererkennung in Spreadsheets*. Diplomarbeit, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart, 2012. URL http://elib.uni-stuttgart.de/opus/volltexte/2012/7056. (Zitiert auf den Seiten 6 und 8)

Alle URLs wurden zuletzt am 10. 05. 2014 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift