Institut für Softwaretechnologie
Abteilung Software Engineering
Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Diplomarbeit Nr. 3224

Fehlererkennung in Spreadsheets

Sebastian Zitzelsberger

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. rer. nat. Jochen Ludewig
Betreuer:	Daniel Kulesz, M.Sc.
begonnen am:	25. Juli 2011
beendet am:	24. Januar 2012
CR-Klassifikation:	H.4.1, D.2.4

Zusammenfassung

Spreadsheets sind auf Grund ihrer flexiblen Einsetzbarkeit aus der Unternehmenswelt kaum mehr wegzudenken. Es gibt jedoch Belege dafür, dass in vielen von ihnen ernstzunehmende Fehler enthalten sind, von denen einige bereits schwerwiegende Folgen für Unternehmen verursacht haben. In der Praxis wird dieses Risiko jedoch kaum wahrgenommen und Spreadsheets werden zumeist ungeprüft verwendet. Zwar existieren kommerzielle Werkzeuge, die Prüfungen von Spreadsheets anbieten, jedoch sind diese mit konzeptionellen Einschränkungen versehen, die deren Nutzen stark vermindern.

In dieser Arbeit wird das Konzept einer technischen Inspektions-Werkstatt für Spreadsheets entwickelt, wodurch die technische Grundlage geschaffen werden soll, um Spreadsheets mit statischen und dynamischen Mitteln auf Fehler zu prüfen. Die Umsetzung dieses Konzepts erfolgt dabei durch das erweiterbare Java-Framework *Spreadsheet Inspection Framework* (*SIF*), mit dem Prüfzentren für Spreadsheets realisiert werden können. In dieser Arbeit wird es durch die Umsetzung der ersten Ausbaustufe von *SIF* ermöglicht, konfigurierbare, statische Prüfungen von Spreadsheets nach individuellen Richtlinien durchzuführen. Es werden unter Verwendung von *SIF* Prüfverfahren für die drei Vorschriften *Konstanten in Formeln*, *Leserichtung* und *Formelkomplexiät* realisiert, deren Anwendung mit Hilfe des prototypischen Prüfzentrums *Example Testing Center* (*ETC*) demonstriert werden können.

Die anschließende Evaluation von *ETC* mit operativen Spreadsheets hat gezeigt, dass vergleichbare Ergebnisse zu denen der kommerziellen Prüfwerkzeuge *Spreadsheet Professional* und *Rainbow Analyst* erzielt werden konnten. *ETC* unterliegt jedoch durch die Verwendung der Inspektions-Werkstatt nicht den konzeptionellen Einschränkungen mit denen bestehende Prüfwerkzeuge versehen sind.

Abstract

Because of their enormous flexibility, spreadheets play an essential role in business today. However, there is evidence that many spreadsheets contain serious errors, some of which have already caused severe consequences for certain enterprises. Nevertheless, spreadsheet risks are hardly perceived in practice and it is common to use untested spreadsheets. Commercial tools that offer checks for spreadsheets do exist, but they are flawed with conceptual limitations, which strongly reduce their merits.

This paper proposes the concept of an inspection-facility for spreadsheets, laying the technical foundation to check spreadsheets for errors with statical and dynamic means. The implementation of this concept is carried out through the extendable Java-framework *Spreadsheet Inspection Framework* (*SIF*), which enables the creation of testing centers for spreadsheets. The realisation of *SIF*'s first stage in this work makes it possible to execute configurable static tests that check spreadsheets for indiviudal policies. The checking of the three policy rules *Constants In Formulas*, *Reading Direction* and *Formula Complexity* has benn implemented with the use of *SIF* and their application can be demonstrated by using the prototypical *Example Testing Center* (*ETC*).

The subsequent evalution of *ETC* with operational spreadsheets has shown that similiar results to those of the commercial tools *Spreadsheet Professional* and *Rainbow Analyst* could be achieved. However, *ETC* is not afflicted with the conceptual limitations of existing tools, because it is based on the inspection-facility.

Inhaltsverzeichnis

1 Einleitung **7**
- 1.1 Motivation . . . 8
- 1.2 Ziel . . . 9
- 1.3 Übersicht . . . 10

2 Spreadsheet-Grundlagen **11**
- 2.1 Elemente und Konzepte . . . 11
- 2.2 Spreadsheet-Systeme und Endbenutzer . . . 17
- 2.3 Risiken durch Qualitätsmängel . . . 21

3 Software-Qualität **23**
- 3.1 Der Qualitäts-Begriff . . . 23
- 3.2 Taxonomie der Software-Qualitäten . . . 25
- 3.3 Qualitätskosten . . . 26
- 3.4 Software-Qualitätssicherung . . . 28
- 3.5 Software-Prüfung . . . 29

4 Spreadsheet-Qualität **35**
- 4.1 Stand der Forschung . . . 35
- 4.2 Zusammenfassung . . . 38

5 Vorhandene Ansätze zur Erhöhung der Spreadsheet-Qualität **41**
- 5.1 Spreadsheet-Engineering . . . 41
- 5.2 Spreadsheet Prüfung . . . 43
- 5.3 Schlussfolgerung . . . 46

6 Konzept **49**
- 6.1 Technische Grundlage zur Spreadsheet-Prüfung . . . 49
- 6.2 Metapher . . . 51
- 6.3 Verwendung . . . 55

7 Anforderungen **59**
- 7.1 Funktionale Anforderungen . . . 59
- 7.2 Nichtfunktionale Anforderungen . . . 61

8 Umsetzung **63**
8.1 Vorgehen . . . 63
8.2 Design . . . 64
8.3 Implementierung . . . 71

9 Evaluation **73**
9.1 Rahmenbedingungen . . . 73
9.2 Ergebnisse . . . 77
9.3 Analyse . . . 79

10 Fazit **81**
10.1 Zusammenfassung . . . 81
10.2 Ausblick . . . 83
Anhang . . . 85

A Anhang **87**
A.1 Inhalt und Aufbau des beigelegten Datenträgers . . . 87
A.2 Evaluationsergebnisse . . . 87
A.3 Fehler-Taxonomien . . . 89

Literaturverzeichnis **91**

Kapitel 1

Einleitung

Seit dem ersten Erscheinen von Tabellenkalkulationssoftware in den frühen 80er Jahren erfreut sich diese Art von Software großer Beliebtheit. Populäre Tabellenkalkulationsanwednungen, wie beispielsweise Lotus 1-2-3[1], Microsoft Excel[2] oder verschiedene Calc-Derivate[3] der Star Office Familie, sind in der Unternehmenswelt allgegenwärtig und aus dem Arbeitsalltag kaum mehr wegzudenken.

Mit Hilfe dieser Anwendungen lassen sich flexibel einsetzbare Softwareprogramme, sogenannte Spreadsheets, erstellen, modifizieren und ausführen. Zu diesem Zweck können tabellarisch angebrachten Zellen mit numerischen und alphanumerischen Daten gefüllt und in verschiedenen Formaten angezeigt werden. Zusätzlich können Berechnungen und Verarbeitungen dieser Daten über Formeln definiert werden, die mit Hilfe von bereitgestellten Operationen den Wert einer Zelle aus den Werten anderer Zellen berechnen. Besonders wertvoll für Unternehmen sind Spreadsheets, da deren Erstellung, Modifizierung und Erweiterung auch direkt durch deren späteren Nutzer erfolgen kann, ohne dass dieser über nennenswerte Programmierkenntnisse verfügen muss. Daher ist es nicht weiter verwunderlich, dass Tabellenkalkulationssoftware die weitreichende und bereichsübergreifende Verbreitung erreicht haben, die sie heute besitzen.

Als Folge dieser Prävalenz entstehen jährlich mehrere Millionen Spreadsheets [Pan08c] in allen denkbaren Ausprägungen [PBLFJ08, Cro07, JH96]. So unterscheiden sich die entwickelten Spreadsheets etwa in ihrer Größe, Komplexität und ihrem Erstellungsprozess, aber auch in ihrem Einsatzzweck, ihrer Einsatzdauer und ihrer Bedeutung. Dabei entstehen verschiedenste Variationen, angefangen von kleinen skizzenhaften Spreadsheets, die spontan innerhalb weniger Minuten für kleine einmalige Berechnungen entstehen, bis zu großen, komplexen Systemen von Spreadsheets, die mit sorgfältiger Planung entwickelt werden und über Jahre hinweg für missions- oder sogar unternehmenskritische Aufgaben eingesetzt werden [Cro07]. Trotz dieser unterschiedlichen Ausprägungen scheinen jedoch fast alle eingesetzten Spreadsheets eine Eigenschaft zu teilen – die hohe Wahrscheinlichkeit für darin enthaltene Fehler.

[1] http://www-01.ibm.com/software/lotus/products/123/
[2] http://office.microsoft.com/en-us/excel/
[3] Beispielsweise Open Office Calc: http://www.openoffice.org/

1.1 Motivation

Verschiedene Studien [Pan98, PBL09, CHM08, But00] haben in der Praxis eingesetzte oder in Experimenten entstandene Spreadsheets auf Fehler untersucht und das übereinstimmende Resultat dieser Studien ist, dass eine sehr hohe Anzahl aller Spreadsheets ernstzunehmende Fehler enthält, die die Korrektheit der durchgeführten Berechnungen beeinflussen können. Eine überwiegende Anzahl der Studien prognostiziert dabei eine Quote von fehlerhaften Spreadsheets um die 90%.

Zwar müssen nicht alle Spreadsheet-Fehler bemerkbare Auswirkungen oder sogar kritische Konsequenzen zur Folge haben, jedoch existieren zahlreiche Belege für Fälle, in denen verheerende finanzielle oder reputationelle Verluste auf unbemerkte Fehler in Spreadsheets zurückzuführen sind [Cro09, O'B]. In vielen dieser Horrorszenarien begünstigt das Fehlen von Kontrollmechanismen, die Entstehung und Nutzung solcher fehlerhaften Spreadsheets. Dieser Mangel, an von Unternehmen vorgeschriebenen Qualitätssicherungsmaßnahmen für Spreadsheets, ist in der Forschung bekannt und gut dokumentiert [PO08, PHJ96, CMW07, PBLFJ08]. Aber dennoch haben die meisten Unternehmen dieses Wissen bisher ignoriert und der Aussicht auf ernstzunehmende Spreadsheet-Fehler wenig Aufmerksamkeit geschenkt [Pan98].

Neben der fehlenden Kontrolle setzen auch die Spreadsheet-Nutzer selbst zu viel Vertrauen in ihre Spreadsheets und tendieren zu einem übermäßigen Selbstvertrauen bezüglich ihrer Korrektheit und Genauigkeit [Pan08a]. Infolgedessen werden die meisten Spreadsheets auch nicht selbständig von ihren Erstellern auf Fehler überprüft und falls doch eine Prüfung stattfindet, ist diese in ihrem Umfang meist sehr beschränkt und wird informell und unsystematisch durchgeführt [PBLFJ08, CMW07]. Diese vorhandene Tendenz, Spreadsheets nicht oder nicht ausreichend auf deren Qualität zu überprüfen, wird dahingehend noch verstärkt, dass die manuelle Prüfung von Spreadsheets meist sehr mühsam und bei einer Prüfung durch eine Einzelperson häufig wenig effektiv ist [PO08].

Angetrieben durch neue Gesetze wie den Sarbanes-Oxley Act[4] oder Basel II[5], haben sich einige Dienstleister etabliert, die Audits von Spreadsheets und entsprechende Prüfwerkzeuge für Spreadsheets anbieten. Die geringe Nutzung von bestehenden Werkzeuge in der Praxis [CMW07, PBLFJ08] lässt jedoch darauf schließen, dass diese nicht in der Lage sind das Problem von Fehlern in Spreadsheets zu lösen.

Verantwortlich dafür, dass die angebotenen Prüfwerkzeuge keine adäquate Unterstützung für die Prüfung von Spreadsheets bieten können, sind dabei die erheblichen Einschränkungen, mit denen diese Werkzeuge versehen sind. So sind die meisten Werkzeuge für die Verwendung im Finanzsektor konzipiert und können daher meist nur eine begrenzte Anzahl an vordefinierten und sehr spezialisierten Regeln überprüfen. Jedoch besteht bisher weder in der Praxis noch in

[4] http://www.soxlaw.com/index.htm
[5] http://www.bis.org/publ/bcbs107.htm

der Wissenschaft eine Einigung darüber, welche Eigenschaften ein gutes Spreadsheet besitzen muss und wie diese Eigenschaften erreicht werden können. Daher ist die Prüfung von einer scheinbar zufällig ausgewählten Anzahl an Regeln nicht zielführend, um die Qualität von Spreadsheets allgemein zu überprüfen. Es ist jedoch auch nicht möglich die überprüfbaren Regeln zu erweitern, da die Prüfwerkzeuge meist auch auf die Benutzung mit einer einzigen Spreadsheet-Software eingeschränkt sind und keine Möglichkeiten zur Konfiguration oder Erweiterung vorsehen. Zusätzlich werden die meisten Werkzeuge als quell-geschlossene Software kommerziell vertrieben und stellen keine wissenschaftlichen Ansprüche an die eingesetzten Methoden. Als Folge daraus sind die meisten bestehenden Ansätze zur Prüfung von Spreadsheets nicht dokumentiert und deren Nutzen wurde in den seltensten Fällen objektiv evaluiert [AP10, NO10, PB08].

1.2 Ziel

Das Ziel dieser Arbeit ist daher die Konzeption und Entwicklung eines erweiterbaren plattformunabhängigen Java-Frameworks, das die technische Grundlage schaffen soll, um Prüfungen von Spreadsheets durchzuführen. Dadurch soll ein systematisches, reproduzierbares Prüfverfahren für Spreadsheets ermöglicht werden, das von Endbenutzern durchgeführt werden kann. Die unterstützen Prüfungen beschränken sich dabei zunächst auf statische Prüfungen, aber die Erweiterung um dynamische Prüfungen und die Unterstützung von nichtmechanischen Prüfungen soll möglich sein.

Die Aufgabe des Frameworks ist es zunächst die Spreadsheets auf die Einhaltung festlegbarer Qualitätskriterien mit statischen Mitteln zu prüfen. Verstöße von Qualitätskriterien sollen dabei vollautomatisch entdeckt werden, um den Nutzer auf mögliche Mängel des untersuchten Spreadsheet hinzuweisen und ihn so bei der Prüfung von Spreadsheets zu unterstützen. Zu diesem Zweck soll es entdeckte Verstöße automatisch klassifizieren, bewerten, gruppieren und dem Nutzer in geeigneter Form zur Verfügung stellen. Anders als bei bestehenden Prüfwerkzeugen sind die überprüfbaren Qualitätskriterien vom Benutzer frei zu einer Qualitätsrichtlinie zusammenstellbar. Überprüfbare Qualitätskriterien werden mittels Vorschriften definiert, welche individuell konfiguriert werden können. Außerdem soll es ermöglicht werden, mit Hilfe einer Java-API neue Vorschriften auf einfache Weise hinzuzufügen. Zwar muss in der initialen Version die Prüfung einer neuen Vorschrift von einem professionellen Programmierer implementiert werden, jedoch wird es ermöglicht diesen Mechanismus in Zukunft mit einer benutzerfreundlicheren Möglichkeit zu ersetzen.

Um die Brauchbarkeit des Frameworks zu demonstrieren, soll die Prüfung von drei möglichst unterschiedlichen Vorschriften implementiert und ein prototypisches Prüfwerkzeug, das diese Vorschriften verwendet, entwickelt werden. Die implementierten Prüfverfahren für die ausgewählten Vorschriften sollen dann mit der Hilfe von operationalen Spreadsheets aus Real- und Laborumgebungen evaluiert werden, um die entstandene Lösung mit bestehenden Prüfwerkzeugen vergleichen zu können.

1.3 Übersicht

Der Inhalt dieser Diplomarbeit wird in folgenden Kapiteln präsentiert.

Kapitel 2 - Spreadsheet-Grundlagen definiert zentrale Begriffe und Konzepte bezüglich Spreadsheets, wie sie in dieser Arbeit verwendet werden. Zudem werden die Eigenschaften von Spreadsheet-Systemen untersucht und die daraus resultierenden Folgen vorgestellt. Es wird erläutert, dass Spreadsheets in der Praxis häufig qualitative Mängel besitzen und dadurch ein Risiko für die Verwendung in der Unternehmenswelt darstellen können.

Kapitel 3 - Software-Qualität definiert zentrale Begriffe bezüglich der Qualität von Software. Anschließend wird das vorhandene Wissen und die vorhandenen Maßnahmen im Software Engineering vorgestellt, die es erlauben die Risiken, die durch Qualitätsmängel entstehen, beherrschbar zu machen.

Kapitel 4 - Spreadsheet-Qualität stellt den Stand der Forschung bezüglich Spreadsheet-Qualität vor.

Kapitel 5 - Vorhandene Ansätze zur Erhöhung der Spreadsheet-Qualität untersucht ob Prinzipien aus dem Software-Engineering auf Spreadsheets übertragen werden können und beleuchtet vorhandene Ansätze zur Erhöhung der Qualität von Spreadsheets. Speziell werden dabei die vorhandenen Ansätze zur Prüfung von Spreadsheets untersucht.

Kapitel 6 - Konzept präsentiert im Detail die vorgeschlagene Lösung zur Prüfung von operationalen Spreadsheets.

Kapitel 7 - Anforderungen listet die funktionalen und nicht-funktionalen Anforderungen auf, die an die in Kapitel 6 vorgeschlagene Lösung gestellt werden.

Kapitel 8 - Entwicklung beschreibt den gewählten Entwicklungsprozess, sowie Details aus Entwurf und Implementierung.

Kapitel 9 - Evaluation setzt sich kritisch mit der fertiggestellten Umsetzung auseinander und vergleicht diese mit bestehenden Spreadsheet-Prüfwerkzeugen.

Kapitel 10 - Fazit fasst die Ergebnisse dieser Arbeit zusammen und nennt die Vorteile und Einschränkungen der entwickelten Lösung. Im Anschluss daran wird diskutiert, wie sich die entstandene Lösung verbessern lässt und welche weiteren Prüfungsarten für Spreadsheets unterstützt werden können.

Kapitel 2

Spreadsheet-Grundlagen

Um einen effektiven Ansatz zur Prüfung von Spreadsheets zu entwickeln, ist es wichtig zuerst die Grundlagen von Spreadsheets zu kennen und zu verstehen. Aus diesem Grund werden in den folgenden Abschnitten die grundlegenden Elemente und Konzepte von Spreadsheets vorgestellt, um dann die Eigenschaften von Spreadsheets und die daraus resultierenden Folgen beschreiben zu können.

2.1 Elemente und Konzepte

In der Einleitung wurden Spreadsheets als kleine Softwareprogramme präsentiert, mit deren Hilfe Berechnungen getätigt und Daten verarbeitet werden können. Jedoch wurde der Begriff *Spreadsheet* bisher nicht genau definiert und in Übereinstimmung mit dem Gebrauch in der Praxis eher unpräzise verwendet.

Der Grund für diesen unpräzisen Gebrauch ist die ursprüngliche Herkunft des Begriffes *Spreadsheet*. Denn als *spread sheets* wurden bereits in den 50er-Jahren Papierarbeitsblätter bezeichnet, die für eine zweidimensionale Analyse von Bilanzdaten in Tabellenform genutzt wurden. Als Anfang der 80er-Jahre das Prinzip einer tabellenförmigen Anordnung von Daten und Berechnungen mittels Software umgesetzt wurde, übernahm man auch die Bezeichnung *Spreadsheet*. Anders als bei den Papierarbeitsblättern ist jedoch für die Erstellung und Nutzung der elektronischen Form von Spreadsheets ein System aus mehreren Komponenten notwendig. Daher wird der Begriff *Spreadsheet* heutzutage zwar im Allgemeinen mit einer Art Software für die Dateneingabe und -verarbeitung assoziiert, aber meist wird der Begriff als ungenauer Sammelbegriff für alle beteiligten Komponenten verwendet.

Um Missverständnisse bei der Interpretation von Spreadsheet-Begriffen zu vermeiden, werden daher die folgenden Definitionen von Spreadsheet-Elementen und -Konzepten in dieser Arbeit verwendet. Einige dieser Definitionen sind an die in [ACM00] gegebenen Definitionen angelehnt, die meisten wurden aber für die Verwendung in dieser Arbeit angepasst oder erweitert. In Abbildung 2.1 wurde versucht einige der im Folgenden beschriebenen Elemente durch grafische Markierungen zu veranschaulichen.

2.1.1 Das Spreadsheet-Konzept und dessen Basis-Elemente

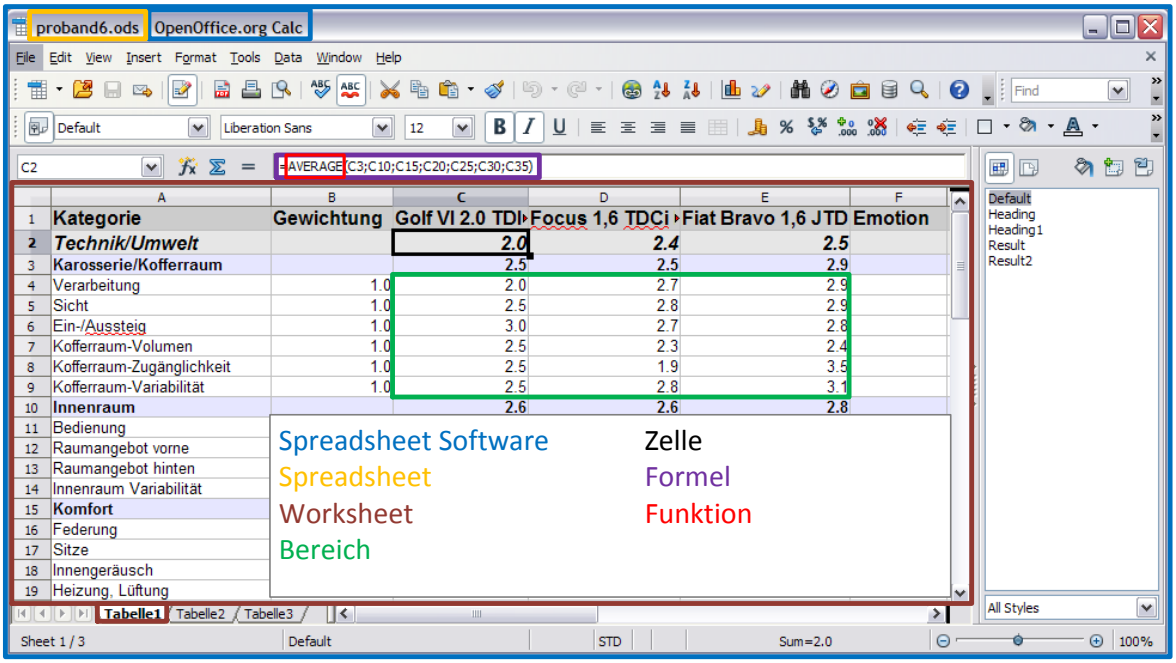

Abbildung 2.1: Veranschaulichung einiger Spreadsheet-Elemente anhand der Spreadsheet-Software *OpenOffice.org Calc*

Spreadsheet (Konzept) bezeichnet das grundlegende Konzept einer zweidimensionalen Tabelle von Zellen, die in einem geradlinigen Gitter von Reihen und Spalten unterschiedlicher Höhe und Breite angebracht sind. Dieses Konzept wird bei Spreadsheet-Programmen angewandt, die eine beliebige Anzahl solcher Tabellen in elektronischer Form, sogenannte Worksheets, enthalten können.

Eine Zelle ist die atomare Einheit eines Spreadsheets. Die Zelle eines Worksheets, also der elektronischen Version eines Spreadsheets, kann sich in einem von fünf verschiedenen Zuständen befinden. Die Zelle kann (a) leer sein, (b) sie kann eine Beschriftung enthalten, die den Inhalt anderer Zellen beschreibt, (c) einen konstanten Wert enthalten, der von anderen Zellen weiterverarbeitet wird und einmalig beim Erstellen des Spreadsheets Spreadsheet-Programms bereitgestellt wird, (d) einen Eingabewert enthalten, der vom Benutzer des Spreadsheet-Programms bei dessen Ausführung bereitgestellt werden muss (Eingabe-Zelle) oder (e) einen Ausgabewert enthalten, der von einer Formel anhand von konstanten und eingegeben Werten berechnet wird (Ausgabe-Zelle).

Eine Zellformat gibt an, wie eine Zelle visuell dargestellt werden soll. Dabei können Eigenschaften, wie etwa das verwendete Schriftbild, die Darstellungsart des Inhalts oder die Umrandung der Zelle, eingestellt werden.

Ein Schreibschutz gibt an, ob der Inhalt einer Zelle oder eines ganzen Worksheets mittels eines Passworts davor geschützt werden soll, überschrieben zu werden.

Eine Reihe / Zeile ist die Menge aller Zellen auf einer horizontalen Linie eines Spreadsheets. Reihen werden dabei mit Ziffern, beginnend bei 0, durchnummeriert.

Eine Spalte ist die Menge aller Zellen auf einer vertikalen Linie eines Spreadsheets. Spalten werden dabei mit Großbuchstaben, beginnend bei A, durchnummeriert.

Eine Zelladresse definiert über ein Koordinatenpaar aus Spalten- und Zeilennummer eindeutig die Position einer Zelle innerhalb eines Spreadsheets. Zelladressen werden in dieser Arbeit im Format *A1* notiert, das heißt zuerst steht die Nummer der zugehörigen Spalte, gefolgt von der Nummer der zugehörigen Reihe. Beispielsweise würde die Adresse *A1* die Zelle in der ersten Spalte, nummeriert mit dem Buchstaben A, und in der ersten Reihe, nummeriert mit der Ziffer 1, eines Spreadsheet bezeichnen.

Ein Bereich ist die Menge aller Zellen innerhalb eines beliebig großen rechteckigen Ausschnitts aus einem Worksheet. Ein Bereich kann dementsprechend aus (a) einer einzelnen Zelle, (b) einer einzelnen Reihe, (c) einer einzelnen Spalte oder (d) einer Menge von Zellen eines rechteckigen Ausschnitts, der sich über mindestens eine Reihe und eine Spalte erstreckt, bestehen.

Eine Bereichsadresse definiert eindeutig die Position eines Bereiches innerhalb eines Spreadsheet über die Zelladressen seiner Zellen in der linken oberen und in der rechten unteren Ecke. Bereichsadressen werden in dieser Arbeit im Format *A1:C4* notiert, das heißt zuerst steht die Zelladresse der Zelle in der linken oberen Ecke, gefolgt von einem Doppelpunkt als Trennzeichen zur Zelladresse der Zelle in der rechten unteren Ecke. Beispielsweise würde die Adresse eines Bereichs, der von der Zelle in der ersten Spalte und Reihe (*A1*) bis zu der Zelle in der dritten Spalte und vierten Reihe (*C4*) reicht, als *A1:C4* notiert werden. Falls ein Bereich nur aus einer einzigen Zelle besteht, kann die Adresse im selben Format angeben werden, sie unterscheidet sich jedoch semantisch nicht von der entsprechenden Zelladresse.

Ein benamter Bereich ist ein Bereich, der vom Bearbeiter eines Spreadsheet-Programms mit einem aussagekräftigen Namen versehen wurde. Dabei müssen Namen innerhalb eines Spreadsheet-Programms eindeutig sein und dürfen nicht so benannt sein, dass der Name mit einer Adresse verwechselt werden kann. Benamte Bereiche können dadurch neben ihrer Bereichsadresse auch mit ihrem zugewiesenen Namen adressiert werden.

Eine Referenz ist der Verweis auf den Wert oder die Werte eines Bereiches. Referenzen können dazu verwendet werden, Werte für Zellen oder Operanden für Formeln bereitzustellen, indem sie als Zeiger auf die Werte anderer Zellen fungieren. Um den sprachlichen Gebrauch von Referenzen zu vereinfachen, wird in dieser Arbeit der Ausdruck Wert weggelassen, wenn von Referenzen gesprochen wird. Eine Zell- bzw. Bereichsreferenz wird also gleichbedeutend verwendet, wie die Referenz auf den oder die Werte einer Zelle oder eines Bereichs. Referenzen können dabei entweder absolut oder relativ festgelegt werden.

Eine absolute Referenz hat eine analoge Funktionsweise zu einer Postanschrift ohne den Namen des Adressaten. So wie Post an diese Anschrift immer an den aktuellen Bewohner unter dieser Adresse zugestellt wird, zeigt eine absolute Referenz immer auf die Werte der Zellen, die sich aktuell an der angegeben Adresse befinden.

Eine relative Referenz hingegen zeigt immer auf den Wert der Zelle, die sich bei der Erstellung der relativen Referenz an der angegeben Adresse befand, auch wenn diese seitdem ihre Position verändert hat.

Innerhalb von Formeln sind Referenzen nicht speziell gekennzeichnet, wenn sie jedoch verwendet werden, um den Inhalt einer Zelle festzulegen, müssen sie durch ein vorangestelltes Gleichheitszeichen *(=)* gekennzeichnet werden.

Die Angabe einer relativen Referenz geschieht über die Adresse des gewünschten Ziels, also entweder über eine Bereichsadresse, eine Zelladresse oder den Namen eines benamten Bereichs. Absolute Referenzen werden wie relative Referenzen angegeben, jedoch wird absoluten Adresselementen ein Dollarzeichen *($)* vorangestellt. Dabei ist es auch möglich nur eine der beiden Zellen einer Bereichsadresse, oder auch nur die Zeile oder Spalte einer Zelle absolut anzugeben. Beispielsweise wird innerhalb einer Formel die absolute Referenz zu dem Bereich *A1:B3* als *$A$1:$B$3* angegeben, die relative Referenz hingegen wie die Adresse selbst als*A1:B3*.

Eine Formel ist eine Operation auf einem oder mehreren Operanden, die als Ergebnis den Wert genau einer Zelle berechnet ohne dabei Seiteneffekte zu verursachen. Eine Formel wird dabei als mathematischer Ausdruck angeben, der aus Operanden, Operatoren und Funktionen besteht. Formeln, die mindestens einen Operator oder eine Funktion und mindestens einen Operanden enthalten, werden in dieser Arbeit als ***berechnende Formeln*** bezeichnet. Formeln, die lediglich den Wert einer anderen Zelle referenzieren, werden in dieser Arbeit als ***referenzierende Formeln*** bezeichnet.

Operanden einer Formel sind entweder konstante Werte, Referenzen oder die Ergebnisse anderer Formeln.

Operatoren in einer Formel sind Berechnungsvorschriften wie aus einem oder mehreren Objekten ein neues Objekt gebildet werden kann. Dabei werden von einer Spreadsheet-Software meist eine Reihe von Arithmetik-, Vergleichs-, Text- und Referenz-Operatoren bereitgestellt. Beispiele sind das Pluszeichen (+) für die arithmetische Addition, das Größerzeichen (>) für den Vergleich von Werten, das kaufmännische Und (*&*) für die Verkettung von Texten oder die Tilde (~) für die Vereinigung von Referenzen.

Funktionen sind von einem Benutzer oder der jeweiligen Spreadsheet-Software vordefinierte Operationen, für die nur noch die Operanden definiert werden müssen. Sie sind dafür zuständig, beliebige Operationen, speziell diese, die nicht rein über Operatoren ausgedrückt werden können, zu ermöglichen. Dabei dürfen sie jedoch ebenfalls nur

genau ein Ergebnis liefern und keine Seiteneffekte produzieren. Verwendet wird eine Funkion, durch die Angabe des Namens, gefolgt von einer öffnenden Klammer, einer durch Strichpunkt getrennten Liste von Operanden und einer abschließenden Klammer angegeben. Daher muss jede Funktion einen eindeutigen Namen.

Formeln können ineinander verschachtelt werden, das heißt die Ergebnisse einer Formel können als Operand der übergeordneten Formel verwendet werden. Jedoch müssen die Formeln dabei korrekt geklammert werden und der äußersten Formel muss ein Gleichheitszeichen (=) vorangehen. Die Angabe der Formeln erfolgt dabei in Infixnotation.

Eine Formel, die alle Werte der Zeilen 1 bis 3 in Spalte A und alle Werte der Zeilen 2 bis 5 der Spalte D addiert, würde beispielsweise folgendermaßen notiert:
=SUMME(SUMME(A1:A3);SUMME(D2:D5))

Ein Worksheet oder auch Arbeitsblatt ist die elektronische Version eines Spreadsheets. Ein oder mehrere Worksheets können in einem Spreadsheet-Programm enthalten sein, müssen aber einen eindeutigen Namen besitzen. Innerhalb eines Spreadsheet-Programms kann problemlos auf die Inhalte von unterschiedlichen Worksheets referenziert werden. Der referenzierten Adresse wird dabei der Name des übergeordneten Worksheets sowie ein Ausrufezeichen (*!*) vorangestellt. Beispielsweise würde eine Referenz eines Worksheets auf die Zelle *A1* eines anderen Worksheets, mit den Namen *Daten*, als *=Daten!A1* angegeben werden.

2.1.2 Der Verbund der Spreadsheet-Komponenten

Ein Spreadsheet(-Programm) ist die Spezifikation von Datenwerten (Zellinhalten), Datenfluss (Referenzen), Datenverarbeitung (Formeln) und Darstellungsinformationen (Anordnung der Zellen und Worksheets, Zellformate) in einer Spreadsheet-Sprache. Zusätzlich können noch Meta-Informationen, wie etwa Bereichs- oder Worksheet-Namen, sowie allgemeine Informationen zum Spreadsheet-Programm spezifiziert werden. Ein Spreadsheet-Programm ist dadurch die konkrete Umsetzung eines Spreadsheet-Modells und kann als Spreadsheet-Datei vom jeweiligen Spreadsheet-System in verschiedenen Formaten gespeichert werden.

Um den sprachlichen Gebrauch dieses Konzeptes für den weiteren Verlauf zu vereinfachen, wird außerhalb dieses Definitionskapitels der Begriff ***Spreadsheet*** als Synonym zum Begriff Spreadsheet-Programm verwendet, sofern keine Gefahr für eine Mehrdeutigkeit besteht. Zwar steht der Begriff Spreadsheet eigentlich für das grundlegende Konzept, jedoch ist dieses im weiteren Verlauf von geringerer Bedeutung als das Konzept eines Spreadsheet-Programms, und die elektronische Umsetzung dieses Konzept hat mit dem Begriff Worksheet ohnehin eine eigene unmissverständliche Bezeichnung.

Eine Spreadsheet-Sprache ist eine Menge von abstrakten Sprachkonstrukten, um ein Spreadsheet-Programm zu definieren. Es existieren verschiedene Spreadsheet-Sprachen, die sich in ihren verfügbaren Konstrukten und syntaktischen Details unterscheiden

können. Jedoch sind in allen Sprachen entsprechende Konstrukte vorhanden, um die oben beschriebenen Elemente und Konzepte auszudrücken.

Ein Spreadsheet-Modell ist das zugrunde liegende Modell eines Spreadsheet-Programms, das mittels abstrakten Spreadsheet-Konzepten ausgedrückt ist. Ein konkretes Spreadsheet-Modell entspricht dabei einem vom Benutzer transformierten konzeptuellen Modell eines Problems aus der realen Welt in Ausdrücke einer bestimmten Spreadsheet-Sprache. Die Umsetzung eines Spreadsheet-Modells erfolgt als Spreadsheet-Programm, welches mittels einer Spreadsheet-Software erstellt wird.

Beispiel: Ein Benutzer steht vor der Aufgabe, die Summe der Zahlen von 1 bis 10 zu berechnen und verknüpft dies mit dem konzeptuellen Modell der kumulativen Addition der Zahlen 1 bis 10. Ein entsprechendes Spreadsheet-Modell dazu besteht aus der Definition von 10 Zellen mit den Zahlen 1 bis 10 als Inhalt. Außerdem muss eine weitere Zelle definiert werden, die die Werte der anderen Zellen über eine Formel addiert. Die Umsetzung dieses Spreadsheet-Modells kann dabei in verschiedenen Spreadsheet-Sprachen und durch verschiedene Spreadsheet-Programme realisiert werden. Ein Spreadsheet-Programm verwendet etwa die Summen-Funktion, ein anderes addiert die Inhalte der Zellen mittels des Plus-Operators.

Eine Spreadsheet-Programminstanz ist ein Spreadsheet-Programm, bei dem alle Zellen, die vom Benutzer mit Eingabewerten befüllt werden müssen, einen gültigen Wert besitzen. Ein Spreadsheet-Programm kann dabei beliebig oft instantiiert werden, ohne dass das zu Grunde liegende Spreadsheet-Modell verändert wird. Durch das Verändern eines Eingabewertes wird die Spreadsheet-Programminstanz in einer andere Instanz des selben Spreadsheet-Programms transformiert.

Eine Spreadsheet-Software ist eine integrierte Entwicklungsumgebung für Spreadsheet-Programme. Mit Hilfe dieser Entwicklungsumgebung ist es möglich Spreadsheet-Programme zu erstellen, zu modifizieren, anzuzeigen und auszuführen. Die Erstellung und Bearbeitung erfolgt entweder per direkter Manipulation über eine graphische Oberfläche oder durch Texteingabe. Eine Spreadsheet-Software interpretiert dabei eine oder mehrere Spreadsheet-Sprachen und kann als Desktop-Anwendung oder Web-Anwendung realisiert sein.

Ein Spreadsheet-System bezeichnet den Verbund der Komponenten, die zur elektronischen Umsetzung des Spreadsheet-Konzepts notwendig sind. Dieser Verbund besteht aus einer Spreadsheet-Software, einer oder mehreren Spreadsheet-Sprachen, die von dieser Software interpretiert werden können, sowie den Spreadsheet-Programmen, die von der Spreadsheet-Software erstellt, modifiziert und als Spreadsheet-Programminstanzen ausgeführt werden können.

2.2 Spreadsheet-Systeme und Endbenutzer

Im vorangegangen Abschnitt wurde ein Spreadsheet-System als der Verbund an Komponenten definiert, der für die elektronische Umsetzung des Spreadsheet-Konzepts notwendig ist. Mit Hilfe eines Spreadsheet-Systems wird es dessen Benutzern ermöglicht, ohne nennenswerte Programmierkenntnisse, neue Spreadsheets zu erstellen oder bestehende Spreadsheets zu erweitern oder verändern.

Ein solches System, das *eine Menge von Methoden, Techniken und Werkzeugen bereitstellt, die es den Benutzern dieses Systems erlauben, auch als nicht-professionelle Software-Entwickler eigenständig Software-Artefakte zu erstellen, zu modifizieren oder zu erweitern* [LPKW06], wird als ***Endbenutzer-Entwicklungssystem*** bezeichnet. Dabei wird der Begriff ***Endbenutzer*** typischerweise verwendet, um die Benutzer von Software und die Entwickler von Software voneinander abzugrenzen. Es wird also zwischen den Personen unterschieden, die Software nur bedienen, und denen, die neue Software schaffen oder bestehende Software verändern. Diese Abgrenzung wird jedoch durch die Endbenutzer-Entwicklung verwaschen, da es eben auch Endbenutzern ermöglicht wird, Software zu erstellen und zu verändern.

Die Eröffnung der Software-Entwicklung für Endbenutzer ist jedoch ein entscheidendes Merkmal von Spreadsheet-Systemen und prägend für deren Beschaffenheit und deren Einsatz in der Praxis. Dabei scheint die Umsetzung der Endbenutzer-Entwicklung in Spreadsheet-Systemen besonders gelungen zu sein, da sie sich zur beliebtesten Art dieser Systeme entwickelt haben. Nardi und Miller [NM90b] haben daher die Eigenschaften von Spreadsheet-Systemen untersucht und führen deren Erfolg auf zwei Haupteigenschaften zurück:

- Die Verfügbarkeit von Programmierkonzepten und -techniken auf einem hohen Abstraktionsniveau, die den Aufgaben des Benutzers angemessen sind und sie vor den technischen Einzelheiten der traditionellen Programmierung abschirmen.
- Die Verwendung einer tabellenförmigen Benutzeroberfläche, die als Modell für die Anwendungen der Benutzer dient.

2.2.1 Vorzüge von Spreadsheet-Systemen

Durch die Kombination dieser Eigenschaften können mit Hilfe von Spreadsheet-Systemen die beiden Basisprobleme des Benutzers gelöst werden: Die Berechnung und die Darstellung von Informationen. Dadurch bieten Spreadsheet-Systeme eine Reihe von Vorzügen. Diese Vorzüge können durch die tatsächliche Verwendung von Spreadsheet-Systemen in der Praxis bestätigt werden [CS96, PBLFJ08, GMz05].

Einfache Nutzbarkeit und Bedienung

Zuallererst benötigen Benutzer von Spreadsheet-Systemen wenig Übung, da sie bereits mit dem Verständnis von zwei Grundkonzepten – Zellen als Variablen und Formeln als Beziehungen zwischen diesen Variablen – in der Lage sind, grundlegende Programmieraufgaben zu lösen. Die Interviews in [NM90b] haben zudem gezeigt, dass die meisten Benutzer weniger als zehn verschiedene Funktionen benötigen, die alle Bezug zu ihrem jeweiligen Fachbereich besitzen. Zwar können fortgeschrittene Konzepte, wie Makros oder die bedingten Formatierung von Zellen, mitverantwortlich für eine schnellere Entwicklung oder eine höhere Qualität des Spreadsheets sein, jedoch werden sie grundsätzlich für die Erstellung von Spreadsheets nicht benötigt. Als Folge daraus werden fortgeschrittene Konzepte und das Wissen über zusätzliche Funktionen meist nur langsam angeeignet. So haben Powell et al. [PBLFJ08] ein unregelmäßiges, selbstständiges Erlernen von Spreadsheet-Systemen mittels Handbüchern, über Kollegen oder die Benutzung anderer Spreadsheets als die gängigste Form der Weiterbildung für den Umgang mit Spreadsheets dokumentiert.

Zusätzlich zu der geringen Einlernzeit, erlaubt die vertraute Tabellenanordnung in Spreadsheets die einfache Strukturierung und Präsentation von deren Inhalten. Neben der Definition von Spreadsheet-Konstrukten mittels der Eingabe von Text, stellt Spreadsheet-Software zu diesem Zweck meist auch direkte Manipulationsmöglichkeiten bereit [HHN85]. So können Referenzen beispielsweise per Maus angegeben werden und existierende Funktionen können aus einer Liste ausgewählt und eingefügt werden, was weiter zur einfachen Nutzung von Spreadsheet-Systemen beiträgt. Darüber hinaus bietet die automatische Neuauswertung von Formeln, bei Änderung einer zugrunde liegender Zelle, dem Nutzer sofortige Rückmeldung während der Entwicklung. Dadurch kann er den Effekt von durchgeführten Änderungen und den Fortschritt seiner Arbeit sofort beurteilen [NM90b]. Diese einfache Verwendung von Spreadsheet-Systemen erlaubt es deren Nutzern sich auf ihre Hauptaufgabe zu konzentrieren – das Lösen von fachspezifischen Problemen.

Flexible Entwicklungs- und Einsatzmöglichkeiten

Durch die einfache Benutzung und die sofortige Rückmeldung von Spreadsheet-Systemen können Spreadsheets mit einer viel höheren Geschwindigkeit erstellt, modifiziert oder erweitert werden als traditionell entwickelte Software. Denn nicht nur der Inhalt der Eingabe-Variablen eines Spreadsheets, sondern auch das zugrunde liegende Spreadsheet-Modell, kann beinahe augenblicklich an veränderte Anforderungen angepasst werden. Eine zusätzliche Zeitersparnis kommt dadurch zustande, dass Anforderungen, nicht wie bei der traditionellen Software-Entwicklung, von Fachexperten an Entwickler übertragen werden müssen, sondern direkt durch die Fachexperten umgesetzt werden können. Diese höhere Entwicklungsgeschwindigkeit ist vor allem in der Unternehmenswelt von enormer strategischer Bedeutung, da sie ausschlaggebend für den Erfolg eines Unternehmens sein kann [Gro07].

Des Weiteren helfen Spreadsheet-Systeme ihren Benutzern dabei, die umzusetzenden Probleme zu erforschen und zu verstehen. Da die Modellierung und Strukturierung des Problems einfach und schnell möglich ist, kann ein konzeptuelles Modell des Problems während der Erstellung des entsprechenden Spreadsheets explorativ entstehen. Dies kann zum Beispiel eine effektive Methode sein, um unklare Anforderungen während des Entwicklungsprozesses zu klären [Gro07]. Zusätzlich eignen sich Spreadsheets-Systeme gut für die kooperative Erstellung, Bearbeitung oder Erweiterung [NM90a] und können daher zur Vermittlung und Verteilung von Wissen verwendet werden.

Eine weitere Eigenschaft von Spreadsheet-Systemen ist deren flexible Einsetzbarkeit, die durch die unterschiedlichen Einsatzzwecke von Spreadsheets in der Praxis dokumentiert ist [GMz05]. So sind Spreadsheet-Systeme weder an einen bestimmten Aufgabentyp noch an einen bestimmten Fachbereich gebunden und werden in Unternehmen jeder Fachrichtung, durch Nutzer in verschiedensten Positionen, für eine Vielfalt von Aufgaben eingesetzt. Typischerweise werden Spreadsheets etwa für die Datenverwaltung, Informationskommunikation, Problemmodellierung und -analyse, sowie für die Prognose von Trends und die Unterstützung bei der Findung von Entscheidungen verwendet [CMW07, PBLFJ08, Cro07].

2.2.2 Gefahren von Spreadsheet-Systemen

Spreadsheet-Systeme bieten jedoch nicht nur Vorzüge für die Verwendung im Unternehmen, sondern sie bergen auch Gefahren. Diese Gefahren werden dabei teilweise durch die selben Eigenschaften der Spreadsheet-Systeme verursacht, die ihnen auch ihre Verzüge verleihen [NM90b]. Sie bestehen zumeist darin, dass Spreadsheets mit mangelhafter Qualität entstehen und ohne Kontrolle verwendet werden.

Gefahren durch den Nutzer

Eine dieser Gefahren sind dabei die menschlichen Fehler, die bei der Erstellung oder Bearbeitung von Spreadsheets begangen werden. So hat die Forschung auf dem Gebiet der menschlichen Fehler ergeben, dass der Mensch durch Einschränkungen in dessen Wahrnehmung nicht in der Lage ist, komplexe kognitive Aufgaben fehlerfrei durchzuführen [Pan08c]. Empirische Studien (zitiert in [Pan98]) haben dabei gezeigt, dass die Quote von unkorrigierten Fehlern bei komplexeren Aufgaben, wie etwa dem Programmieren von Software, auf bis zu fünf Prozent ansteigen kann. Raymond Panko zeigt in [Pan08c] auf, dass in mehreren Studien ähnliche Prozentzahlen an fehlerhaften Zellen in Spreadsheets gefunden wurden. Daher ist davon auszugehen, dass bei der Erstellung und Bearbeitung aller nicht-trivialen Spreadsheets Fehler begangen werden und dass sich diese ohne entsprechende Gegenmaßnahmen auf die Qualität der Spreadsheets auswirken.

Eine weitere Gefahr steckt in dem übermäßigen Selbstvertrauen, das Benutzer in ihre erstellten Spreadsheets setzen. So tendieren die Benutzer in den meisten Fällen dazu die Qualität ihrer

eigenen Arbeit zu überschätzen [Pan08a], was dazu führt, dass keine oder nur sehr oberflächliche Prüfungen nach Vollendung der Entwicklung durchgeführt werden [PBLFJ08, CMW07]. Dies wiederum bewirkt, dass die menschlichen Fehler, die bei der Erstellung von Spreadsheets gemacht wurden, nicht entdeckt werden.

Dieser Effekt tritt besonders in Erscheinung, da der typische Nutzer von Spreadsheets keine entsprechende Ausbildung für die Programmieraufgaben besitzt, die beim Erstellen und Bearbeiten von Spreadsheets nötig ist [PBLFJ08]. Dadurch, dass die Endbenutzer von den Details der Programmierung abgeschirmt werden, sind sich diese oftmals gar nicht darüber bewusst, Programmieraufgaben auszuführen. Daher nehmen sich auch die damit verbundenen Risiken nicht wahr. Foglich werden die meisten Spreadsheets ohne einen geordneten Prozess nach dem Prinzip *Code and Fix* entwickelt, ohne die Anforderungen an das Spreadsheet vorher zu spezifizieren, dessen Struktur zu planen oder das Ergebnis nach Vollendung zu überprüfen [PBLFJ08].

Gefahren durch das Spreadsheet-Konzept

Gefahren entstehen auch durch das Konzept der Spreadsheets selbst und durch die Einschränkungen der bestehenden Spreadsheet-Software [Aya01]. Beispielsweise können Zellinhalte, die entweder als Text oder als Zahl formatiert sind, nicht auf Anhieb voneinander unterschieden werden. Die Formatierung kann jedoch für das Ergebnis einer Formel, die diese Inhalte verwendet, eine große Rolle spielen. Außerdem wird es durch die zweidimensionale Anordnung der Zellen erschwert, die Struktur des Spreadsheets zu erkennen und Referenzen nachzuvollziehen, da die entsprechende Lokalität fehlt. Allgemein sind Spreadsheets nur schwierig in einer für Menschen lesbaren Form zu gestalten, da keine sinnvollen Strukturvorgaben oder Aggregationsmöglichkeiten gegeben werden. Es bleibt ganz dem Ersteller überlassen, wie er die Struktur seines Spreadsheets gestaltet und ob er beispielsweise eine Trennung von Eingabe, Verarbeitung und Ausgabe vorsieht. Und auch die Möglichkeiten zur internen Dokumentation von Spreadsheets sind mit einfachen Zellkommentaren und Textzellen eher begrenzt. Erschwerend kommt hinzu, dass eine Modularisierung von Logik, Präsentation und Datenhaltung im Spreadsheet Konzept nicht vorgesehen ist und die Bearbeitung eines Spreadsheets nicht von der Benutzung abgetrennt ist. So ist es leicht möglich, dass das Model eines Spreadsheets während der Benutzung eines Spreadsheets ungewollt verändert wird, wenn dies nicht über Schreibschutz verhindert wird.

Zusammengefasst erlauben die Eigenschaften von Spreadsheets zwar eine hohe Flexibilität, aber durch die Vereinfachung von Darstellung und Programmierung tragen sie auch zur Entstehung von Fehlern bei.

Gefahren durch das Umfeld

Und auch das Umfeld, in dem Spreadsheets entstehen, begünstigt es, dass neue Fehler und Mängel in Spreadsheets entstehen oder bestehende nicht entdeckt werden. So fehlen etwa effektive Qualitätsrichtlinien für die Steuerung der Entwicklung, Nutzung und Verwaltung von Spreadsheets. Zwar herrscht kein Mangel an Vorschlägen für sogenannte ***Best Practices*** für Spreadsheets

[Raf08, O'B05, EuSa], die von erfahrenen Anwendern oder Forschern verfasste Vorschriften enthalten, um die Qualität von Spreadsheets zu verbessern. Diese sind jedoch oft widersprüchlich. So besteht bereits bei grundlegenden Prinzipien zur Entwicklung von Spreadsheets Uneinigkeit. Beispielsweise wird diskutiert, ob es für die Qualität von Spreadsheets förderlich ist, benamte Bereiche zu verwenden [MMB09]. Als Folge daraus hat sich bis heute kein allgemein anerkannter, wissenschaftlicher Standard für Spreadsheet Best Practices etabliert und es ist auch nicht gesichert, dass der Bedarf nach einem solchen Standard überhaupt erfüllt werden kann [Col10, Dun10, Gro07]. Diese fehlende Gewissheit scheint jedoch die flächendeckende Einführung von Qualitätsrichtlinien für Spreadsheets in Unternehmen stark zu behindern, da in einer Reihe von Interviews das überwiegende Fehlen solcher Richtlinien in Unternehmen festgestellt wurde [CMW07, PBLFJ08, PO08]. Und so tendieren die meisten Unternehmen dazu, wenige bis gar keine formale Qualitätssicherungsmaßnahmen für Spreadsheets vorzuschreiben. Stattdessen verlassen sie sich in hohem Maße auf die Kontrolle der Qualität und Gebrauchstauglichkeit von Spreadsheets durch die jeweiligen Ersteller [MK05]. Da die Ersteller jedoch in den meisten Fällen keine Ausbildung für die Programmieraufgaben besitzen, die sie beim Verwenden von Spreadsheets-System bewältigen müssen [PBL09], und die Prüfung von Spreadsheets durch deren Konzept erschwert wird, bleiben Spreadsheets meist ungeprüft. Ein Blick auf die Ergebnisse durchgeführter Interviews und Studien [JH96, PBLFJ08] zeigt dabei, dass diese Gefahren in der Praxis akut vorhanden sind.

2.3 Risiken durch Qualitätsmängel

Im vorherigen Abschnitt wurden unter anderem Gefahren, die bei der Verwendung von Spreadsheet-Systemen entstehen, aufgezeigt. Diese Gefahren bestehen in der Entstehung und Verwendung von Spreadsheets mit mangelhafter Qualität und werden durch die Eigenschaften von Spreadsheet-Systemen und deren Umfeld begünstigt. Ein Blick auf die Praxis lässt erkennen, dass die meisten Spreadsheets ohne eine ausreichende formale Prüfung verwendet werden und die Wahrscheinlichkeit für ernstzunehmende Fehler in eingesetzten Spreadsheets sehr hoch ist [Pan98, PBL09].

Die vorhandenen Qualitätsmängel in Spreadsheets müssen zwar nicht immer ernsthafte Konsequenzen zur Folge haben, jedoch entstehen durch sie Risiken, die in vielen Firmen weder vom Management noch von den jeweiligen Nutzern wahrgenommen werden [PBL08b, Pan08c, MK05]. Dementsprechend werden diese Risiken auch nicht bekämpft. So werden viele wichtige Entscheidungen im Unternehmen durch Informationen aus Spreadsheets unterstützt, ohne jedoch die Qualität der verwendeten Spreadsheets mit geeigneten Qualitätssicherungsmaßnahmen sicherzustellen [CMW07]. Dies ist besonders verwunderlich, da die folgenschweren Konsequenzen von vielen falsch getroffenen Entscheidungen, die auf mangelhafte Spreadsheets zurückzuführen sind, gut dokumentiert sind [O'B]. Zwar sind die Spreadsheets meist nicht die alleinig treibende oder sogar ausschlaggebende Kraft für diese Entscheidungen [CMW07], jedoch stellt die Beein-

flussung von kritischen Entscheidungen durch mangelhafte Spreadsheets ein ernsthaftes Risiko für die Verwendung von Spreadsheet-Systemen im Unternehmen dar.

Das Risiko von ernsthaften Konsequenzen, die durch den Einsatz qualitativ schlechter Produkte für wichtige Aufgaben im Unternehmen verursacht werden, ist jedoch nicht nur auf Spreadsheets beschränkt. Auch bei traditioneller Software, also Software, die nicht von Endbenutzern sondern von professionellen Softwareentwicklern erstellt wird, besteht dieses Risiko [LL07, S. 62]. So zeigen katastrophale Ereignisse, wie beispielsweise der Absturz der Ariane 5 Rakete[1], welche schwerwiegenden Folgen Fehler in Software verursachen können. Dieses Risiko scheint jedoch bei Software, die nach den Prinzipien des Software Engineering [LL07] entwickelt wird, beherrschbar zu sein, da adäquate Methoden bestehen, um eine angemessene Qualität von Software sicherzustellen.

Bei Spreadsheets hingegen ist die mangelnde Qualität und die daraus resultierenden Risiken das Hauptproblem bei deren Einsatz in der Unternehmenswelt. Es handelt sich bei Spreadsheets jedoch ebenfalls um eine spezielle Art von Software. So kann eine Spreadsheet-Sprache als Programmiersprache der vierten Generation zur raschen Entwicklung von fachspezifischen Anwendungen gesehen werden. Und bei einer Spreadsheet-Software handelt es sich um eine integrierte Entwicklungsumgebung [GO10], wie sie auch bei der Entwicklung traditioneller Software eingesetzt wird. Um zu untersuchen, in wie weit sich die Methoden des Software Engineerings auf Spreadsheets übertragen lassen, soll im nächsten Kapitel das vorhandene Wissen über Software-Qualität aus dem Software Engineering genauer untersucht werden.

[1] http://www.ima.umn.edu/~arnold/disasters/ariane.html

Kapitel 3

Software-Qualität

Im vorherigen Kapitel wurden unter anderem die Vorzüge und Gefahren von Spreadsheet-Systemen analysiert. Dabei wurde gezeigt, dass in der Praxis eingesetzte Spreadsheets häufig Fehler enthalten und eine allgemeine Tendenz zu niedriger Qualität besitzen, wodurch sie ein Risiko für den Einsatz im Unternehmen darstellen können. Dieses Risiko ist dabei im Vergleich zu Software, die von professionellen Entwicklern nach den Prinzipien des Software Engineerings entwickelt werden, ungemein höher. Da es sich bei Spreadsheets auch um Software im weiteren Sinne handelt, sollen daher zunächst allgemeine Qualitätsbegriffe bezüglich Software definiert werden, um dann das vorhandene Wissen im Software Engineering über die Qualität von Software im Detail zu untersuchen.

3.1 Der Qualitäts-Begriff

Entgegen dem allgemeinen Sprachgebrauch impliziert die ursprüngliche Bedeutung von Qualität keinerlei Wertung. Sie bezeichnet lediglich eine Eigenschaft eines Gegenstandes. Im Lauf der Zeit wurde jedoch der Begriff Qualität mit guter Qualität gleichgesetzt. Unabhängig davon existieren verschiedene Definitionen, die unterschiedliche Blickwinkel auf die Qualität zulassen.

3.1.1 Qualitäts-Ansätze

Garvin nennt in [Gar84] fünf unterschiedliche Ansätze für die Definition von Qualität und plädiert für eine ganzheitliche Betrachtung aller Ansätze.

- Die transzendente Sicht entspricht in etwa der umgangssprachlichen Definition von Qualität. So wird Qualität als eine subjektive Eigenschaft gesehen, die zwar wahrnehmbar ist und durch Erfahrung erkannt wird, aber nicht präzise definiert werden kann.
- Die produkt-basierte Sicht hingegen, definiert Qualität als die objektiv eindeutig messbaren inhärenten Eigenschaften eines Produktes. Qualität ist also eine messbare Größe, bei der subjektive Kriterien nicht berücksichtigt werden.

- Bei der benutzer-basierten Sicht wird Qualität allein durch den Benutzer des Produktes definiert und bewertet.
- Bei der hersteller-basierten Sicht bezeichnet Qualität das Ausmaß, in dem die spezifizierten Anforderungen für ein Produkt erfüllt werden.
- Bei der wert-basierten Sicht werden auch die Kosten und der Preis berücksichtigt. Gute Qualität entspricht also einem günstigen Preis-Leistungs-Verhältnis.

3.1.2 Definitionen

In dieser Arbeit soll die Qualitäts-Definition der DIN 55350 (1995) verwendet werden, die unterschiedliche Blickwinkel nach Garvin zulässt.

Qualität ist die Gesamtheit von Eigenschaften und Merkmalen eines Produktes oder einer Tätigkeit, die sich auf die Eignung zur Erfüllung gegebener Erfordernisse beziehen.

Ein Qualitätsmerkmal bezeichnet dabei ergänzend zu der DIN Definition eine einzelne Eigenschaft oder ein einzelnes Merkmal eines Produktes oder einer Tätigkeit.

Dabei beschränkt sich Qualität jedoch nicht nur auf die Qualitätsaspekte nach allgemeinem Sprachgebrauch, sondern sie beinhaltet insbesondere auch alle Eigenschaften bezüglich der Funktionalität [LL07, S.65]. Qualität bestimmt also nach der gegebenen Definition die Eignung eines Produktes oder einer Tätigkeit gegebene Erfordernisse zu erfüllen. Dabei liegt, je nach dem wie diese Erfordernisse bestimmt und in welcher Form sie festgelegt werden, eine der unterschiedlichen Sichtweisen nach Garvin vor. So liegt beispielsweise eine benutzer-basierte Sicht der Qualität vor, wenn sich die Erfordernisse aus den impliziten Erwartungen und Bedürfnissen einzelner Benutzer ableiten. Für die Verwendung in der Praxis ist diese subjektive Sicht jedoch nicht praktikabel und stattdessen wird häufig eine hersteller-basierte Sicht von Qualität verwendet. In diesem Zusammenhang sind auch die drei Begriffe Anforderung, Fehler und Mangel, wie sie in der ISO-Norm 9000 (2005) definiert sind, eng mit der Definition von Qualität verbunden.

Eine Anforderung ist ein Erfordernis oder eine Erwartung, das oder die festgelegt, üblicherweise vorausgesetzt oder verpflichtend ist.

Ein Fehler ist die Nichterfüllung einer Anforderung.

Ein Mangel ist die Nichterfüllung einer Anforderung in Bezug auf einen beabsichtigten oder festgelegten Gebrauch.

Eine Anforderung ist also ein festgelegtes Erfordernis von dem erwartet, in der Regel sogar rechtlich verlangt wird, dass es erfüllt wird. Anforderungen können dabei in unterschiedlicher Form vorliegen, etwa als offene oder latente, als harte oder weiche oder als funktionale oder nichtfunktionale Anforderungen [LL07, S.366-369]. In der Praxis sind jedoch meist nur solche

Anforderungen von Bedeutung, die in überprüfbarer Form in einem Spezifikationsdokument definiert und vom Kunden abgenommen wurden. So ist ein Fehler, also die Nichterfüllung einer spezifizierten Anforderung, auch nur dann feststellbar, wenn die Möglichkeit besteht diese Erfüllung objektiv zu überprüfen. Entsprechend kann ein Fehler nur dann auch als Mangel bezeichnet werden, wenn der beabsichtige Gebrauch des Produkts oder das Ziel der Tätigkeit überprüfbar definiert ist.

Im allgemeinen Sprachgebrauch werden die Begriffe Fehler und Mangel jedoch kaum unterschieden. Sie werden stattdessen allgemein verwendet, um die Abweichung bestimmter Qualitätsmerkmale von den erwarteten Werten zu bezeichnen. In dieser Arbeit sollen jedoch die hier definierte Bedeutung des Begriffes Fehler verwendet werden.

3.2 Taxonomie der Software-Qualitäten

Die Qualität von Software-Produkten setzt sich aus verschiedenen Qualitäten und Qualitätsmerkmalen zusammen, die einander beeinflussen und oft auch miteinander in Konkurrenz stehen (Siehe Abbildung 3.1 und Abbildung 3.2).

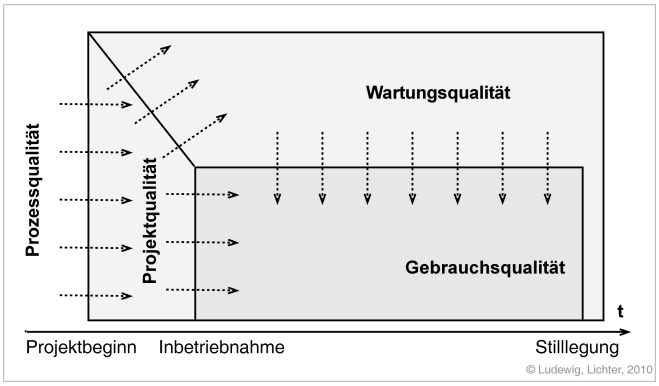

Abbildung 3.1: Bedeutung verschiedener Qualitätsaspekte über der Zeit nach [LL07, S.67]

So setzt sich die Produktqualität, also die Qualität eines Software-Produkts, aus der Wartungsqualität und der Gebrauchsqualität zusammen. Der Benutzer ist in der Regel nur an der Gebrauchsqualität interessiert, während die Wartungsqualität für den Hersteller meist eine große Rolle spielt, da er zumeist auch die Wartung der Software durchführt. Da eine gute Wartungsqualität jedoch auch eine Voraussetzung dafür ist, dass die Gebrauchsqualität des Produktes während der Wartung hoch bleibt, sollte auch der Benutzer diese Qualität beachten. Entsprechend sind für Hersteller und Benutzer die benutzer-basierte Sicht als auch die hersteller-basierte Sicht der Qualität von Bedeutung.

Die Produktqualität wiederum wird stark von der Qualität des Entwicklungsprozesses beeinflusst, die sich aus der Prozess- und Projektqualität zusammensetzt. Auch hier zeigt sich das

Problem einer einseitigen Betrachtungsweise der Qualität. So scheint die Prozessqualität auf den ersten Blick für den Nutzer bzw. Kunden uninteressant, jedoch ist er dem Hersteller bezüglich der Lieferung der Software ausgeliefert und sollte daher Interesse an einer hohen Prozessqualität besitzen.

Ludewig und Lichter sprechen wegen dieser komplexen Beziehung zwischen den Qualitäten von einer Software-bezogenen Qualität und schließen mit diesem Begriff *alles ein, was im weitesten Sinne als Software-Qualität erscheint* [LL07, S. 65-70]. Die einzelnen Qualitätsmerkmale dieser unterschiedlichen Qualitäten können mit Hilfe von Taxonomien gegliedert werden, die für das Verständnis von Qualität äußert nützlich sein können. In dieser Arbeit soll daher die Taxonomie Software-bezogener Qualitäten nach Ludewig und Lichter (Abbildung 3.2) verwendet werden.

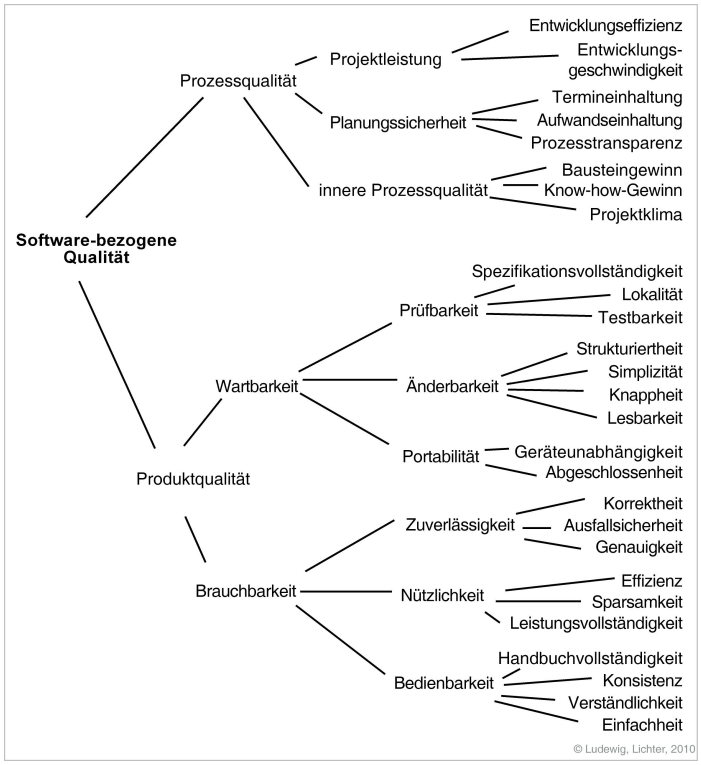

Abbildung 3.2: Der Quälitätenbaum nach Ludewig und Lichter [LL07, S.68]

3.3 Qualitätskosten

Das Streben nach Qualität ist tief im Software-Engineering verwurzelt. Dabei ist das Ziel jedoch nicht die Optimierung einzelner Qualitätsmerkmale, sondern es muss aus wirtschaftlicher Betrachtung ein globales Optimum aller Qualitätsmerkmale, sowie der damit verbundenen Kosten angestrebt werden.

Zu Beachten ist dabei die Zusammensetzung der Kosten für Software. Denn die Kosten bestehen nicht nur allein aus den Netto-Herstellungskosten, sondern beinhalten auch die Qualitäts- und Wartungskosten (Siehe Abbildung 3.3). Ein Hauptbestandteil der Qualitätskosten sind dabei die Kosten für die Qualitätssicherung, die sich aus den Fehlerverhütungskosten und den Prüf- und Nachbesserungskosten zusammensetzen. Es handelt sich also um Kosten, die anfallen um eine gewisse Qualität her- und sicherzustellen. Wenn die Kosten für die Qualitätssicherung minimiert werden, können die Fehlerfolgekosten, also die Kosten die durch den Einsatz von Software mangelhafter Qualität entstehen, die gesamten Herstellungskosten schnell um ein hundertfaches übersteigen.

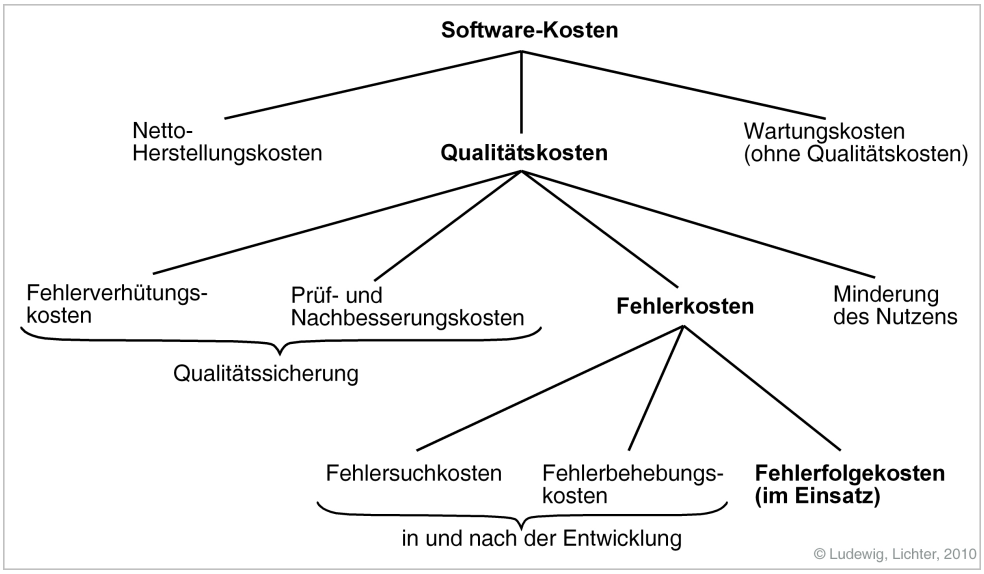

Abbildung 3.3: Kostendifferenzierung in Richtung Fehlerkosten [LL07, S.63]

Daher hat die Vermeidung und die frühzeitige Erkennung von Fehlern höchste Priorität. Zur Entwicklung einer möglichst fehlerfreien Software, also einer nach Herstellersicht qualitativ hochwertigen Software, müssen drei Voraussetzungen erfüllt sein [LL07, S.236].

- Der Entwicklungsprozess muss so gestaltet sein, dass die einheitliche Organisation und Durchführung von Software-Projekten festgelegt ist. Dabei soll er jedoch die nötige Flexibilität besitzen, um projektspezifische Anpassungen durchzuführen, wo diese notwendig sind. Ein Prozess ist dann von Nutzen, wenn er von allen Beteiligten akzeptiert wird und der Nutzen objektiv festgestellt werden kann.
- Die Techniken müssen so vorhanden sein, das die durchzuführenden Tätigkeiten möglichst optimal durch die eingesetzten Sprachen, Technologien, Methoden und Werkzeuge unterstützt werden.
- Die Mitarbeiter müssen über eine entsprechende Ausbildung verfügen, die sie dazu befähigt, den Entwicklungsprozess umzusetzen und die gewählten Techniken anzuwenden.

Um diese Voraussetzungen sicherzustellen, ohne die eine hohe Software-Qualität nicht erreicht werden kann, existieren im Software Engineering eine Reihe von Maßnahmen, die unter dem Begriff Qualitätssicherung zusammengefasst werden können.

3.4 Software-Qualitätssicherung

Software-Qualitätssicherung wird im IEEE-Standard 610.12 (1990) doppeldeutig definiert. So bezeichnet der Begriff einerseits *alle geplanten und systematischen Aktivitäten, die das Vertrauen in die Konformität zu technischen Anforderungen sichern* und andererseits *eine Menge von Aktivitäten, die entworfen wurden, um den Prozess, mit dem eine Produkt gefertigt oder entwickelt wird, zu bewerten.*

Die zweite Bedeutung wird heutzutage meist unter dem Begriff Prozessbewertung geführt und ist für den weiteren Verlauf dieser Arbeit nicht von Bedeutung. Nach der ersten Bedeutung, die für diese Arbeit verwendet werden soll, handelt es sich bei Software-Qualitätssicherung um Maßnahmen, die die Entstehung qualitativ hochwertiger Software fördern oder zeigen, dass eine Software eine hohe Qualität besitzt. *[Sie] hat dabei unmittelbar zum Ziel, das Vertrauen in eine Software zu erhöhen; mittelbar wirkt sie sich [...] auf das Qualitätsniveau aus, steigert also die Fähigkeit, qualitativ hochwertige Produkte zu entwickeln* [LL07, S.270].

Die Schwerpunkte der Software-Qualitätssicherung lassen sich dabei nach Ludewig und Lichter in drei Gruppen unterteilen (Abbildung 3.4):

- Die organisatorischen Maßnahmen umfassen dabei alle Formen der systematischen Planung und Organisation von Entwicklung und Qualitätssicherung.
- Konstruktive Maßnahmen behandeln den Einsatz geeigneter Methoden, Technologien, Sprachen und Werkzeuge, die der hohen Qualität und der Vermeidung von Fehlern dienen. Aber auch der Einsatz geeigneter Prozessmodelle oder die Weiterbildung von Mitarbeitern sind konstruktive Maßnahmen.
- Analytische Maßnahmen verfolgen das Ziel, mittels systematisch durchgeführter Prüfungen Fehler in den Arbeitsergebnissen zu erkennen.

Keine dieser Maßnahmen ist dabei in der Lage allein die Qualität von Software sicherzustellen. Stattdessen komplementieren sich die Maßnahmen und sind zusammen dafür zuständig, dass die Voraussetzungen für Qualität geschaffen und gesichert werden. Zwar wird eine angemessene Software-Qualität vor allem durch konstruktive Maßnahmen sichergestellt und die notwendigen Rahmenbedingungen werden durch organisatorische Maßnahmen geschaffen. Jedoch sind auch die analytischen Maßnahmen ein wichtiger Bestandteil der Qualitätssicherung, da sie Abweichungen von den Prinzipien der anderen Maßnahmen erkennen [FLS04, S.20-21].

Da in dieser Arbeit ein Prüfwerkzeug für Spreadsheets nach dem Vorbild der vorhanden Methoden und Werkzeuge für die traditionelle Software-Entwicklung entstehen soll, werden die

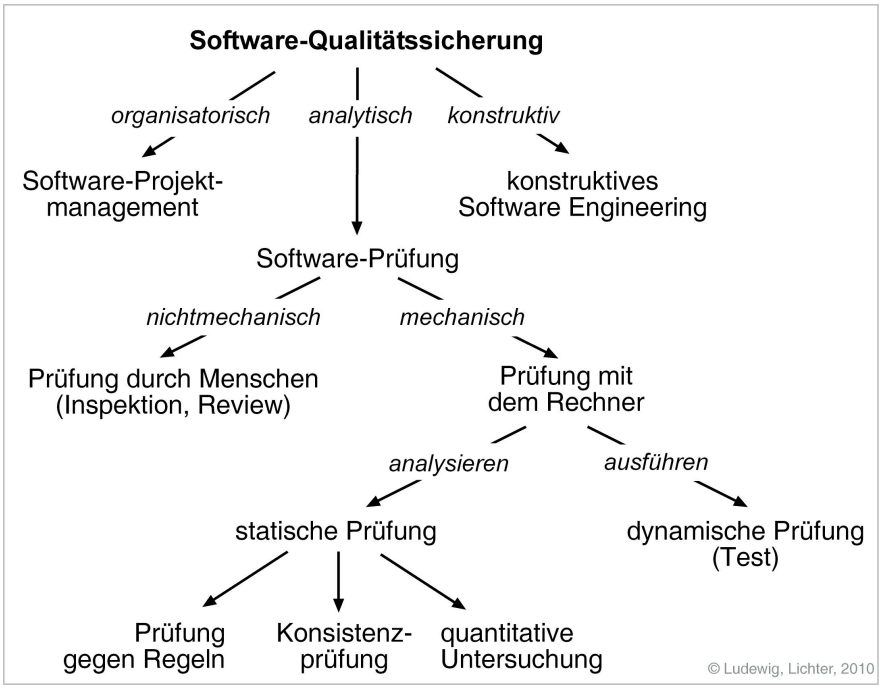

Abbildung 3.4: Gliederung der Software-Qualitätssicherung nach Ludewig und Lichter [LL07, S.271]

verschiedenen Arten Software zu prüfen im nächsten Abschnitt kurz vorgestellt. Speziell soll dabei auf die statischen Prüfungen von Software näher eingegangen werden.

3.5 Software-Prüfung

Software-Prüfungen haben den Zweck mittels einer systematischen Suche nach Fehlern, die Qualität eines Prüflings festzustellen. Als ***Prüfling*** werden dabei Software-Produkte oder Software-Bestandteile bezeichnet, die einer Prüfung unterzogen werden sollen. Prüfungen bieten dabei folgenden Nutzen [FLS04, S.14]:

- Prüfungen liefern eine simple Definition der Qualitätskriterien und ergänzen damit die oft unzureichende Spezifikation der Anforderungen.
- Prüfungen erhöhen die Qualität nicht direkt, aber sie zeigen die Qualität des Prüflings an und bieten den Entwicklern Feedback.
- Prüfungen decken Prüflinge mit besonders guter und besonders schlechter Qualität auf. Dadurch wird einerseits verhindert, dass Prüflinge mit schlechter Qualität verwendet werden und andererseits können Prüflinge mit besonders guter Qualität als Vorbilder verwendet werden.
- Die Erwartung einer Prüfung beeinflusst das Verhalten von Entwicklern positiv, da der Anreiz besteht gute Prüfresultate zu erzielen, wodurch die Qualität direkt erhöht wird.

3.5.1 Prüfungsarten

Die verschiedenen Arten zur Prüfung von Software lassen sich, wie in Abbildung 3.4 dargestellt, in nicht-mechanische und mechanische Prüfungen unterteilen.

- Als nicht-mechanische Prüfungen werden dabei Prüfungen bezeichnet, die vom Menschen ohne einen Rechner durchgeführt werden. Zu den bekannten nicht-mechanischen Prüfungen für Software zählen dabei die Durchsicht, das technische Review, der Walkthrough und die Stellungnahme [LL07, S.281 ff.].
- Mechanische Prüfungen können weiter in statische und dynamische Prüfungen unterteilt werden. Dynamische Prüfungen umfassen alle Arten des Software-Tests, wie etwa in [FLS04] beschrieben. Bei statischen Prüfungen werden die Prüflinge mit Hilfe spezieller Werkzeuge analysiert, ohne die Prüflinge dabei auszuführen.

Die verschiedenen Prüfungsarten sind dabei unterschiedlich geeignet, um bestimmte Fehlerarten im Prüfling zu finden. So können Fehler in Dokumenten nur mit Hilfe von manuellen Prüfungen entdeckt werden, zur Erkennung von Fehlern bezüglich nicht-funktionaler Anforderungen können statische Prüfungen einen Beitrag leisten und die verschiedenen Arten des Tests eignen sich am besten für die Prüfung von funktionalen Anforderungen. Für diese Arbeit sind jedoch speziell die mechanischen statischen Prüfungen von Bedeutung und sollen daher im Folgenden genauer untersucht werden.

3.5.2 Statische Software-Prüfung

Die statische Software-Prüfung ist eine Unterklasse der mechanischen Software-Prüfung. Prüfungen dieser Art werden daher mit Hilfe von sogenannten statischen Analysewerkzeugen (engl. static analysis tools) am Rechner durchgeführt. Die Aufgabe der statischen Analysewerkzeuge ist es, Softwareartefakte ohne deren Ausführung auf die Einhaltung von Regeln und generelle Konsistenz zu prüfen oder quantitative Merkmale über die Artefakte zu erfassen. Eine Prüfung gegen Regeln liefert Aussagen darüber, ob gewisse Normen und Vorschriften eingehalten wurden. Eine Konsistenzprüfung kann beispielsweise zeigen, ob alle definierten Elemente auch verwendet werden und eine quantitative Untersuchung hilft dabei, objektive Metriken zu erfassen, auf Basis derer eine Bewertung des Prüflings durchgeführt werden kann [FLS04, S.21].

Anwendung

Durch den Verzicht auf die Ausführung der Prüflinge bestehen kaum Einschränkungen für Softwareartefakte, um von statischen Analysewerkzeugen geprüft werden zu können. So können Softwareartefakte bereits frühzeitig in der Entwicklung geprüft und Informationen über deren Qualität gewonnen werden. Gleichzeitig beschränken sich die überprüfbaren Vorschriften durch den Verzicht auf die Ausführung jedoch hauptsächlich auf nicht-funktionale Qualitätsmerkmale.

Es werden also hauptsächlich die Qualitätsmerkmale geprüft, die in Abbildung 3.2 unter der Kategorie Wartbarkeit eingegliedert sind.

Die Prüfungen erfolgen außerdem meist rein syntaktisch und sind dadurch anfällig für falsch positive Befunde. Und auch bei richtig positiven Befunden liegen nicht zwangsweise Fehler im Sinne einer nicht erfüllten Anforderung vor. Es handelt sich vielmehr um Warnungen, dass untersuchte Elemente auf bestimmte Art kritisch für die Qualität sein können. Generell greift die gegebene Definition von Fehlern nach einer hersteller-basierten Sicht der Qualität nur bedingt bei Befunden von statischen Analysewerkzeugen. So steht für Kunden die Funktionalität meist im Vordergrund und daher werden nicht-funktionale Anforderungen meist nur ungenügend oder gar nicht festgelegt [LL07, S.369]. Vorschriften, die von statischen Analysewerkzeugen überprüft werden, sind daher überwiegend nicht als Anforderungen durch den Kunden definiert worden und es handelt sich bei Befunden meist nicht um Fehler nach Herstellersicht. Daher wird der Begriff *Defekt* verwendet, um von statischen Analysewerkzeugen entdeckte Befunde zu bezeichnen.

Ein Defekt ist dabei nach IEEE 1044 (2009) eine zu behebende oder zu ersetzende Unvollkommenheit eines Prüflings, auf Grund derer der Prüfling die an ihn gestellten Ansprüche oder spezifizierten Vorschriften nicht erfüllt.

Ein Defekt stellt also ein Defizit oder eine Schwäche eines Prüflings dar und muss nicht zwangsweise einen Fehler im Sinne einer nicht erfüllten Anforderung sein. Trotz dieser Einschränkungen sind statische Analysewerkzeuge eine wichtige Qualitätssicherungsmaßnahme bei der Entwicklung von Software, da deren wiederholte Anwendung nur sehr geringe Kosten verursacht und sich dabei trotzdem ausreichend Informationen über die Qualität des Prüflings gewinnen lassen. Damit dies gewährleistet ist, müssen geeignete Werkzeuge ausgewählt und entsprechend dem jeweiligen Nutzungskontext angepasst werden. Die Konfiguration des Werkzeugs ist dabei essentiell, da sonst falsch-positiv Raten bis zu 96% auftreten können [WJKT05]. Jedoch können mit gut konfigurierten statischen Analysewerkzeugen bis zu 80% aller Defekte in einem Softwareprodukt gefunden werden [Wag06]. Zwar werden durch die statische Analyse hauptsächlich Defekte mit relativ geringer Schwere entdeckt [WDA+08], jedoch weisen die gefundene Defekte häufig auf die Anwesenheit solcher hin. Die Defekte zeigen so die Stellen auf, an denen mit hoher Wahrscheinlichkeit Fehler vorhanden sind oder in Zukunft entstehen könnten [WDA+08, ZWN+06]. Dabei kann auch der parallele Einsatz von verschiedenen Analysewerkzeugen lohnenswert sein, da in der Studie von Rutar et al. [RAF04] gezeigt wurde, dass verschiedene Analysewerkzeuge meist auch unterschiedliche Defekte entdecken.

Techniken

Statische Analysewerkzeuge können dabei eine oder mehrere der folgenden Techniken[1] verwenden, die nach ihrer Fortschrittlichkeit sortiert sind.

- Bei einer Stil-Überprüfung (engl. Style-Checking) wird der Prüfling auf die Konformität mit festgelegten Richtlinien untersucht, die die Lesbarkeit, Verständlichkeit und Wartbarkeit verbessern sollen. So werden beispielsweise Namenskonventionen für Variablen oder die Formatierung des Quelltextes in Programmierrichtlinien festgelegt und können mittels einer Stil-Überprüfung geprüft werden.
- Bei einem syntaktischen Musterabgleich wird der Prüfling nach spezifischen Sprachkonstrukte und -konstellationen durchsucht, die auf die Anwesenheit eines Defekts hinweisen können. Die Sprachkonstrukte und -konstellationen werden dabei durch rein syntaktisch definierte Muster angegeben – eine semantische Analyse findet nicht statt. Diese Muster sind meist sprachspezifisch und werden als Softwarefehler-Muster (engl. Bug-Pattern), Anti-Muster (engl. Anti-Pattern) oder schlechter Code-Geruch (engl. Code-Smell) angegeben. Die Übergänge zwischen diesen unterschiedlichen Musterarten ist dabei fließend, jedoch wird generell folgendes unter den Begriffen verstanden:
  - Ein Bug-Pattern ist eine mögliche Fehlanwendung von Konstrukten einer formalen Sprache, die oft zu einem Fehlverhalten bei der Ausführung des Prüflings führen. Ein typisches Bug-Pattern für die Programmiersprache Java ist beispielsweise der Vergleich von Text-Objekten (Strings) mittels des Vergleichsoperators (==) , anstatt mittels der `equals()`-Methode. Dabei kann die resultierende Semantik bei der Verwendung des Vergleichsoperators, der Vergleich der beiden Objekt-Adressen, in manchen Fällen gewünscht sein. Jedoch deutet die Anwesenheit dieses Patterns darauf hin, dass der Programmierer die Inhalte der beiden Objekte vergleichen wollte und somit ein Defekt vorliegt, der zu einem Fehlverhalten führen kann.
  - Ein Anti-Pattern beschreibt eine zu vermeidende Struktur bei der Lösung eines bestimmten Problemtyps. Die Anti-Patterns bilden damit das Gegenstück zu den Design Patterns, also den bewährten Lösungsschablonen für wiederkehrende Architekturprobleme, wie sie durch das Buch der „Gang of Four“ [GHJV95] bekannt geworden sind. Ein bekanntes Anti-Pattern ist dabei der Spaghetti-Code, also ein Quellcode der komplexe und verworrene Quellstrukturen aufweist und viele Sprunganweisungen enthält.
  - Ein Code-Smell definiert Heuristiken zur Erkennung von unsauberen Abschnitten, also solchen Abschnitten, die besser einer Umstrukturierung (engl. refactoring) unterzogen werden sollten. Der Begriff wurde dabei durch Kent Beck, dem Erfinder des

[1] Basierend auf einer noch unveröffentlichten Publikation von Prof. Dr. rer. nat. Stefan Wagner http://www.iste.uni-stuttgart.de/se2/menschen/wagner.html

Extreme Programming, geprägt und durch Martin Fowler verbreitet. Ein typischer Code-Smell ist beispielsweise duplizierter Quelltext, also Quelltext der an verschiedenen Stellen in identischer Form vorliegt.

Zwar können komplexere Defekte mit einer Mustererkennung nicht erkannt werden, jedoch zeigt die Erfahrung, dass in den meisten Anwendungen Defekte durch einen Musterabgleich aufgedeckt werden und daher ist dies die am häufigsten eingesetzte Methode zur statischen Analyse.

- Bei einfachen Kontroll- und Datenflussanalysen werden Kontroll- und Datenstrukturen des Quelltextes untersucht, um die zeitliche Abfolge von Anweisungen im Quelltext und den Austausch von Daten zwischen Entitäten zu bestimmen. Der Kontrollfluss kann dabei als Kontrollflussgraph ausgedrückt werden, aus dem auch der Datenfluss entnommen werden kann. Der Kontrollflussgraph kann dann dazu verwendet werden um Informationen über die Werte zu sammeln, die zu verschiedenen Zeitpunkten berechnet werden. So können beispielsweise Abhängigkeiten zwischen Variablen eines Programms oder nicht erreichbare Anweisungen eines Quelltexts entdeckt werden.
- Bei der abstrakten Interpretation wird der Kontroll- und Datenfluss abstrahiert, um durch die semantische Analyse der verbliebenen Informationen Aussagen über das Verhalten von Programmen treffen zu können. Es werden also semantische Informationen ausgeblendet, um statisch die Eigenschaften des Programmes bei dessen dynamischer Ausführung bestimmen zu können. So kann durch abstrakte Interpretation der Ergebnistyp der einfachen Java-Anweisung: `3.12 + 2` bestimmt werden. Dazu werden die konkreten Werte abstrahiert, durch ihren Typ ersetzt (`double + int`) und anschließend der Operator ausgewertet (`double + int = double`).

Beispiele

In der Praxis ist eine große Auswahl an statischen Analysewerkzeugen vorhanden, die ihren Nutzern beim Auffinden von Defekten durch unterschiedliche Techniken helfen. Eine kleine Auswahl von Werkzeugen, die Techniken unterschiedlichen Reifegrads verwenden, sei im Folgenden gegeben.

Check-Style[2] ist ein statisches Analysewerkzeug zur Stilüberprüfung von Java-Quellcode. Dabei ist Check-Style vielseitig konfigurierbar und erlaubt die automatische Überprüfung von individuellen Programmierrichtlinien. Zunächst als reines Werkzeug zur Stilüberprüfung konzipiert, verwendet Check-Style inzwischen auch fortgeschrittenere statische Analysetechniken, um weitere Defekte in Java-Quellcode zu entdecken.

[2] http://checkstyle.sourceforge.net/

Find-Bugs[3] ist ein statisches Analysewerkzeug für Java, das mittels syntaktischem Musterabgleich, sowie einfachen Kontroll- und Datenflussanalysen Bug-Patterns entdeckt. Für die Prüfung von Find-Bugs wird dabei der Bytecode der untersuchten Java-Klassen verwendet und dieser wird mittels einer breiten Anzahl an relativ einfachen Techniken auf Defekte untersucht. Find-Bugs bietet zu diesem Zweck eine Vielzahl an vordefinierten Bug-Patterns für Java an, erlaubt es jedoch auch eigene Detektoren für Bug-Patterns zu definieren.

Coverty Static Analysis[4] ist ein hochentwickeltes kommerzielles Werkzeug zur statischen Analyse von C, C++, C# und Java-Quelltexten, das von vielen großen Unternehmen eingesetzt wird und auch für die Analyse vieler Open Source Projekte verwendet wurde [5]. Es verwendet dazu vor allem fortgeschrittene Techniken, wie die abstrakte Interpretation, um beispielsweise Speicherlecks und Race-Conditions zu entdecken.

[3] http://findbugs.sourceforge.net/
[4] http://www.coverity.com/products/static-analysis.html
[5] http://scan.coverity.com/all-projects.html

Kapitel 4

Spreadsheet-Qualität

Im vorherigen Kapitel wurde gezeigt, dass ein umfangreiches Wissen über die Qualität von Software vorhanden ist und dass der Stand der Forschung im Software Engineering es erlaubt, mittels einer Vielzahl an konkreten Maßnahmen eine angemessene Qualität sicherzustellen. Die Qualität von Spreadsheets in der Praxis hingegen ist häufig mangelhaft und die daraus resultierenden Risiken stellen das Hauptproblem bei deren Einsatz in der Unternehmenswelt dar. Daher soll in diesem Kapitel nun der Stand der Forschung bezüglich der Qualität von Spreadsheets untersucht werden.

4.1 Stand der Forschung

Viele Forschungsarbeiten beschäftigen sich mit der Qualität von Spreadsheets, insbesondere mit Spreadsheet-Fehlern und den Techniken mit denen Fehler vermieden, entdeckt und korrigiert werden können. Jedoch ist generell relativ wenig über Spreadsheet-Fehler und den Techniken zu deren Bekämpfung bekannt, wie Powell, Baker und Lawson in ihrem kritischen Review der bestehenden Literatur über Spreadsheet-Fehler feststellen [PBL08b].

4.1.1 Klassifikation von Spreadsheet-Fehlern

Bereits die Definition von Fehlern bei Spreadsheets erweist sich als problematisch, da Spreadsheets meist nicht in einem geregelten Prozess nach festgelegten Anforderungen, sondern ad hoc und nach dem Prinzip *Code and Fix* entwickelt werden. Ohne eine Spezifikation der Anforderungen kann die Korrektheit und die Qualität eines Spreadsheets jedoch nicht objektiv überprüft werden. Da jeder Nutzer seine ganz eigenen Anforderungen an das Spreadsheet stellt, wird die Anwesenheit von Fehlern oder Mängeln in Spreadsheets zu einer subjektiven Streitfrage. In der Forschung wird versucht dieses Problem zu umgehen, indem Fehlerklassifikationen verwendet werden, die eine allgemeine Menge von Anforderungen an Spreadsheets darstellen. Für die Erstellung einer Fehlerklassifikation wird eine Hierarchie von abstrakten Fehlerklassen identifiziert, um konkrete Fehler entsprechend ihrer Eigenschaften eindeutig einer Klasse zuordnen zu können. Dabei können verschiedene Eigenschaften zur Bildung von Fehlerklassen herangezogen werden,

wodurch unterschiedliche hierarchische Strukturen, sogenannte Fehler-Taxonomien, entstehen. Eine gute Fehler-Taxonomie zeichnet sich dabei durch drei Eigenschaften aus [PBL08b]:

- Als erstes sollte die Taxonomie angeben, für welchen Zweck sie erstellt wurde und in welchem Kontext sie verwendet werden soll.
- Zweitens sollte für jede Fehlerklasse klar definiert sein, welche Eigenschaften ein Fehler besitzen muss, um ihr zugeordnet werden zu können. Zu diesem Zweck sollten auch einige Positiv- und Negativbeispiele angeben werden.
- Drittens sollte jede Taxonomie in ihrem spezifizierten Kontext erprobt sein, um sicherzustellen, dass Fehler von unterschiedlichen Personen den Fehlerklassen einheitlich zugeordnet werden.

Da Fehler-Taxonomien für eine bestimmte Verwendung erstellt werden, gibt es keine allgemeine optimale Taxonomie [GO08]. Stattdessen sind Fehler-Taxonomien für bestimmten Zwecke unterschiedlich geeignet. So können Fehler beispielsweise nach deren Ursache, Auswirkungen oder Entstehungsphase klassifiziert werden. Entsprechend sind die daraus entstehenden Taxonomien unterschiedlich nützlich, um beispielsweise Wissen zur Fehlervermeidung oder zur Einschätzung der Risikos aus ihnen zu gewinnen. Folglich existiert eine Vielzahl von Taxonomien, von denen die wichtigsten in [Pan08b, PBL09] zusammengefasst sind. Die bekannte Fehler-Taxonomie für Spreadsheets von Panko und Halverson [Pan08b] kann dem Anhang (Abbildung A.4) entnommen werden. Trotz der relativ zahlreich vorhandenen Fehler-Taxonomien, besteht weiterhin ein großer Verbesserungsbedarf, da bei vielen dieser Taxonomien der gedachte Einsatzzweck und Kontext nicht angegeben ist und Belege fehlen, dass Fehler in diesen Taxonomien einheitlich zugeordnet werden können.

4.1.2 Auswirkungen von schlechter Spreadsheet-Qualität

Trotz ihrer Bedeutung sind die realwirtschaftlichen Auswirkungen, die von Spreadsheet-Fehlern verursacht werden, ein größtenteils unerforschtes Gebiet. Denn um die Auswirkungen von Spreadsheet-Fehlern in der realen Welt durchführen zu können, benötigt man eine lückenlose Verfolgung und Dokumentation der verwendeten Spreadsheets, von ihrem Entstehen bis hin zum Eintreten der Auswirkungen. Dies wäre zwar theoretisch umsetzbar, etwa über entsprechende Versionsverwaltungssysteme, jedoch fehlt die Umsetzung und Unterstützung in der Praxis. Da eine Untersuchung der Auswirkung in einer künstlichen Umgebung wenig Aussagekraft bietet, ist das bisherige Wissen über die Auswirkungen von Spreadsheet-Fehlern beinahe ausschließlich auf Interviews, Audits und Berichte beschränkt [CMW07, O'B, PB08]. Diese geben die Auswirkungen von Spreadsheet-Fehlern durch die prozentuale Abweichung der Ausgabewerte von den Sollwerten an oder zählen die Anzahl der Fehlentscheidungen, die aufgrund der Fehler getroffen wurden. Alternativ wird der finanzielle Schaden geschätzt beziehungsweise gemessen, der durch den Fehler potentiell entstehen könnte ober bereits entstanden ist. Die Ergebnisse dieser Arbeiten zeigen dabei, dass Spreadsheet-Fehler kritische Auswirkungen, mit möglichen

Schäden bis zu 10.000.000$ [PB08], verursachen können. Gleichzeitig ziehen eine Vielzahl von Fehlern jedoch gar keine direkten negativen Auswirkungen nach sich, da die Ergebnisse des Spreadsheets durch sie nicht verändert werden. Erstaunlicherweise scheint kaum ein Zusammenhang zwischen der Qualität eines Spreadsheets und den möglichen Folgen, die ein Fehler in diesem Spreadsheet verursachen könnte, zu bestehen. So fanden Powell et al. [PB08] bei der Inspektion von Spreadsheets aus der Praxis qualitativ hochwertige Spreadsheets, die für unkritische Aufgaben verwendet wurden, aber auch Spreadsheets mit vielen schwerwiegenden Fehlern, die Folgekosten in Millionenhöhe verursachen hätten können. Die Frage, welche Fehlerarten mit hoher Wahrscheinlichkeit schwerwiegende Folgen verursachen, bleibt jedoch ungeklärt.

4.1.3 Häufigkeit von Spreadsheet-Fehlern

Eine weitere wichtige Frage bezüglich Spreadsheet-Fehlern ist, mit welcher Häufigkeit diese in Spreadsheets vorhanden sind. Die Häufigkeit von Fehlern wird dabei meist durch eine Zell-Fehlerquote [Pan98], also der Quote an Zellen in denen ein Fehler entdeckt wurde, ausgedrückt. Diese Fehlerquote hängt jedoch von einer Vielzahl an Variablen ab, wie beispielsweise von der verwendeten Fehlerdefinition, den verwendeten Spreadsheets oder der verwendeten Methode um die Fehler in den Spreadsheets zu entdecken. Da für alle diese Variablen ein entsprechender Standard fehlt, ist es wenig überraschend, dass die Ergebnisse der durchgeführten Studien stark variieren und deren Ergebnisse schwer miteinander vergleichbar sind [PBL09, Pan08c, Iro08]. Daher hat Panko in [PHJ96] versucht durch den gewichten Durchschnitt der Ergebnisse von sieben Studien Antwort darauf zu geben, wie hoch die durchschnittliche Zell-Fehlerquote in Spreadsheets ist. Aber auch die von ihm errechnete durchschnittliche Quote von 5% erhebt keinen Anspruch auf eine allgemeine Gültigkeit. Sie legt jedoch den Schluss nahe, dass Spreadsheets im Vergleich mit anderen Software-Artefakten überproportional viele Fehler enthalten.

4.1.4 Erstellung von Spreadsheets und die Vermeidung von Fehlern

Um die hohen Fehlerquoten zu verringern, ist es wichtig zu verstehen, wie Spreadsheets in der Praxis erstellt werden und wie dabei Fehler entstehen. Nur dann können zielgerichtet Technologien, Techniken und Prozesse entwickelt werden, die der Entstehung von Fehlern bei der Entwicklung und Nutzung von Spreadsheets entgegen wirken. Bisher ist jedoch wenig gesichertes Wissen vorhanden, wie Spreadsheets in der Praxis erstellt werden und wie dabei Fehler entstehen. Dabei kommt erschwerend hinzu, dass durch das Spreadsheet-Konzept keine Struktur vorgegeben wird und Spreadsheets daher selbst innerhalb von Unternehmen meist auf sehr unterschiedliche Art und Weise entwickelt werden [PB08]. Das vorhandene Wissen stammt daher zumeist aus Labor-Experimenten [BG87, ON87, Kru06], in denen Benutzer unter Beobachtung Spreadsheets nach textuellen Aufgabenstellungen erstellen mussten. Die Autoren, der Veröffentlichungen dieser Experimente, geben zwar Ratschläge dafür, wie Fehler in Spreadsheets verringert werden können, jedoch fehlt es an konkreten Belegen, dass diese Ratschläge auch in der Praxis zu geringeren Fehlerquoten führen. Außerdem ist ungeklärt, in wie weit ein Eingriff in den Entstehungsprozess von Spreadsheets, eine Verringerung der Nützlichkeit von Spreadsheets für den Endbenutzer zur Folge haben könnte.

4.1.5 Erkennung von Spreadsheet-Fehlern

Ein komplementärer Ansatz zur Vermeidung von Fehlern bei der Erstellung von Spreadsheets ist das Erkennen von Fehlern nach Abschluss der Entwicklung des Spreadsheets. Bestehende Studien [GHJ+96, AP10, Pan99] untersuchen die Prüfung von Spreadsheets dabei entweder durch Laborexperimente, bei denen Probanden eingebaute Fehler in Spreadsheets erkennen sollen, oder durch die Untersuchung von Spreadsheets in der Praxis durch einen Experten. In der Mehrzahl dieser Studien,werden dabei keine konkreten Verfahren angewendet und den Prüfern werden keine Anweisungen vorgegeben, wie die Spreadsheets überprüft werden sollen. Und auch wenn ein konkretes Verfahren zur Prüfung eingesetzt wird, wird dieses häufig nicht detailliert genug beschrieben. Als Folge daraus ist es größtenteils unbekannt, wie es um die Effektivität und Effizienz der Ansätze im Vergleich untereinander bestellt ist. Existierende Ansätze beanspruchen zwar, Spreadsheet-Fehler effektiv erkennen zu können, jedoch fehlt es an Belegen, dass dies in der Praxis tatsächlich der Fall ist.

4.2 Zusammenfassung

Die kritische Analyse des Forschungsstandes bezüglich der Qualität von Spreadsheets in [PBL09] lässt daher die folgenden alarmierende Schlüsse ziehen:

- Es gibt keinen allgemein akzeptierten Standard, wie Fehler gezählt und klassifiziert werden. Außerdem fehlt es an Taxonomien für konkrete Einsatzzwecke, deren Nützlichkeit objektiv evaluiert wurde.
- Die wirtschaftlichen Auswirkungen, die von Spreadsheet-Fehlern verursacht werden, sind größtenteils unbekannt. Gleichzeitig besteht jedoch Grund zur Annahme, dass bestimmte Spreadsheet-Fehler Schäden in Millionenhöhe anrichten können.
- Bestehende Studien über die Häufigkeit von Fehlern in Spreadsheets können nicht verglichen werden, da entsprechende Standards fehlen. Jedoch besteht Grund zur Annahme, dass Spreadsheets im Vergleich zu anderen Software-Artefakten übermäßig viele Fehler enthalten.
- Es ist wenig darüber bekannt, wie Spreadsheet-Fehler entstehen und wie diese verhindert werden können. Viele Arbeiten, die größtenteils auf Beobachtungen von Laborexperimenten basieren, schlagen zwar Lösungen vor, um die Entstehung von Fehler zu verringern, jedoch fehlen häufig die Belege, dass die vorgeschlagenen Maßnahmen tatsächlich die Häufigkeit von Fehlern in der Praxis reduzieren können.
- Ansätze, die beanspruchen effektiv Fehler in Spreadsheets erkennen zu können, sind nicht detailliert genug beschrieben. Außerdem ist kaum etwas darüber bekannt, wie die verschiedenen Ansätze im Vergleich bezüglich ihrer Effektivität und ihrer Effizienz abschneiden.

Warum aber ist der Stand der Forschung von Spreadsheets und Software so unterschiedlich,wenn es sich doch bei Spreadsheets auch um Software handelt? Können bestehende Ansätze aus dem Software Engineering nicht auf Spreadsheets übertragen werden? Diese Fragen sollen im folgenden Kapitel beleuchtet werden, in dem die vorhandenen Ansätze zur Erhöhung der Spreadsheet-Qualität untersucht werden.

Kapitel 5

Vorhandene Ansätze zur Erhöhung der Spreadsheet-Qualität

Die Analyse des Stands der Forschung bezüglich der Qualität von Spreadsheets hat gezeigt, dass dieser weit von dem Stand im Software Engineering entfernt ist. Da es sich bei Spreadsheets jedoch auch um Software handelt, liegt die Idee nahe, die Qualität von Spreadsheets dadurch zu erhöhen, indem das bestehenden Wissen und die bestehenden Ansätze aus dem Software Engineering auch für Spreadsheets verwendet werden.

5.1 Spreadsheet-Engineering

Thomas Grossman bezeichnet die angesprochene Übertragung von Prinzipien des Software Engineerings auf Spreadsheets in [Gro07] als *Spreadsheet Engineering* und verspricht sich davon die Lösung vieler Probleme beim Einsatz von Spreadsheets. Er vermutet im Spreadsheet Engineering großes Potential, um die Produktivität von Spreadsheet-Entwicklern zu erhöhen, die Häufigkeit und Schwere von Spreadsheet-Fehlern zu verringern und die Wartbarkeit von Spreadsheets zu verbessern.

In der Tat sind viele bestehende Ansätze für Spreadsheets auf Methoden des Software Engineerings zurückzuführen und vieles von dem bestehenden Wissen scheint übertragbar. So ist beispielsweise der Zusammenhang von niedriger Spreadsheet-Qualität und den möglichen hohen Fehlerfolgekosten zumindest in der Forschung bekannt und man ist sich des resultierenden Risikos bewusst. Entsprechend warnt die Spreadsheet-Forschung seit dem ersten Verwenden von Spreadsheet-Systemen vor Gefahren durch endbenutzer-entwickelte Spreadsheets [Dav82]. Als Folge dieser Erkenntnis haben sich Forschungsgemeinschaften gebildet, wie etwa die European Spreadsheet Risks Interest Group (EuSpRIG)[EuSb] oder das EUSES Consortium (*End Users Shaping Effective Software*) [1], die versuchen die Risiken von Spreadsheets zu erforschen, die

[1] http://eusesconsortium.org

Wahrnehmung dieser Risiken zu verbessern und entsprechende Maßnahmen zu deren Bekämpfung zu entwickeln. Dabei orientieren sich die Forschungsgemeinschaften häufig am Software Engineering. So finden sich beispielsweise alle Voraussetzungen für die Qualität von Software auch in den Forschungsbereichen der EuSpRIG wieder [Cha08].

- Das Forschungsgebiet *Werkzeuge* beschäftigt sich etwa mit Methoden und Werkzeugen zum Erstellen von Spreadsheets [Raf08, Gro07, RCK08a].
- Mit dem Bereich *Ausbildung* soll das Bewusstsein von Endbenutzern erhöht, sowie die Lehre und Forschung bezüglich Spreadsheets verbessert werden [Pan08c, RCK08b, ACM00].
- Unter dem Begriff *Audit* werden Methoden und Werkzeuge zur Prüfung von Spreadsheets zusammengefasst [Cro07, CHM08, NO10].
- Der Bereich *Management* beschäftigt sich mit Standards und Kontrollmechanismen zum Lenken der Spreadsheet-Entwicklung [But08, GO10].

Gleichzeitig existieren auch Ansätze in allen Bereichen der Qualitätssicherung von Spreadsheets, angefangen von organisatorischen [But08, GO10], über konstruktive [Raf08, O'B10], bis hin zu analytischen Maßnahmen [Aya01, ACM00, AM08, Pan07].

Trotz der Übertragung der Ansätze aus dem Software-Engineering scheinen diese jedoch ohne ausschlaggebenden Erfolg zu bleiben. So hat sich seit den frühen Warnungen vor den Gefahren von Spreadsheets kaum etwas an den Problemen verändert, obwohl sich die Technologien und Techniken weiterentwickelt haben [CS96, PBLFJ08].

Der Grund dafür, dass die Ansätze aus dem Software-Engineering nicht richtig greifen, liegt unserer Meinung nach vor allem an der Tatsache, dass die dazu notwendigen Voraussetzungen bezüglich der Nutzer von Spreadsheets nicht erfüllt sind. So handelt es sich bei den Erstellern und Bearbeitern von Spreadsheets meist nicht um professionelle Entwickler, sondern um Endbenutzer. Diese sind in der Regel nicht für die Entwicklungsaufgaben ausgebildet, die sie bei der Erstellung, Bearbeitung und Prüfung von Spreadsheets bewältigen müssen. So nehmen sie einerseits die Risiken und Probleme nicht wahr und andererseits sind sie durch die fehlende Ausbildung für die Entwicklung von Software auch meist nicht in der Lage, die vorhandenen Ansätze und Werkzeuge einzusetzen. Gleichzeitig fehlt es in der Spreadsheet-Forschung an bewährten Standards zur Reduzierung und Eindämmung von Spreadsheet-Fehlern. Entsprechend fehlt es in den Unternehmen meist an Richtlinien, die solche Standards umsetzen, um dadurch die Qualität von Spreadsheets zu verbessern. Und auch die bestehenden Werkzeuge und Methoden entfalten ihre Wirkungen nur begrenzt, da sie in den seltensten Fällen das bestehende Wissen der Endnutzer berücksichtigen.

Die Weiterbildung von Spreadsheet-Nutzern und die damit verbundene Erhöhung der Wahrnehmung der Spreadsheet-Risiken ist auf lange Sicht mit Sicherheit der richtige Weg, um das Qualitätsproblem von Spreadsheets zu lösen. Folglich wurde dies auch als eines der Hauptziele festgelegt, die sich die Forschungsgemeinschaft EuSpRIG gesetzt hat. Dennoch wird angesichts

der millionenfachen Verwendung von Spreadsheets deutlich, dass eine schnell wirkende Maßnahme benötigt wird, um das Qualitätsproblem bei Spreadsheets in absehbarer Zeit zu lindern. Konstruktive und organisatorische Maßnahmen benötigen jedoch naturgemäß eine lange Vorlaufzeit bis sie effektiv dazu beitragen können die Entstehung von Fehlern zu verringern. Außerdem schränken diese Maßnahmen häufig die Flexibilität und die Freiheit bei der Verwendung von Spreadsheet-Systemen ein, da etwa Entwicklungsprozesse vorgeschrieben werden und der Entwicklungsaufwand durch zusätzliche Aktivitäten stark erhöht werden kann.

Aus diesen Gründen soll in dieser Arbeit ein Ansatz verfolgt werden, der sich an den analytischen Qualitätssicherungsmaßnahmen des Software Engineerings orientiert, aber dabei die gegebenen Voraussetzungen, wie etwa das Wissen von Endnutzern, berücksichtigt. Zu diesem Zweck sollen nun die bestehenden Ansätze zur Prüfung von Spreadsheets genauer beleuchtet werden.

5.2 Spreadsheet Prüfung

Im Zuge der Anwendung von Prinzipien des Software Engineerings auf Spreadsheets, wurde in Forschung und Praxis versucht die verschiedenen Prüfungsarten für Software auf Spreadsheets zu übertragen. Die bestehenden Ansätze, Spreadsheets zu prüfen, sollen daher ebenfalls nach der in Abbildung 3.4 dargestellten Gliederung der analytischen Qualitätssicherungsmaßnahmen vorgestellt werden.

5.2.1 Nichtmechanische Spreadsheet-Prüfungen

Die einfachste nichtmechanische Prüf-Variante ist die meist informell und unsystematisch durchgeführte Prüfung auf Plausibilität durch den Ersteller selbst oder durch einen Kollegen. Diese Art der Prüfung wird im Software Engineering als Durchsicht beziehungsweise Stellungnahme bezeichnet. Umfragen und Interviews [CS96, PBLFJ08] zeigen, dass Durchsichten und Stellungsnahmen die einzigen Prüfungen für Spreadsheets sind, die mit einer gewissen Verbreitung in der Unternehmenswelt durchgeführt werden. Es ist jedoch wenig darüber bekannt, wie solche Prüfungen in der Praxis ablaufen und ob sie sich dazu eignen, zumindest die schwerwiegendsten Defekte und Fehler, in Spreadsheets zu erkennen. So wurden solche Prüfungen in der Praxis noch nicht untersucht und vergleichsweise wenige Experimente [GHJ+96, Pan99] durchgeführt. Bei diesen Studien konnten jedoch meist weniger als 50% der Fehler durch unstrukturierte Prüfungen durch die Probanden entdeckt werden. Außerdem konnte nicht bestätigt werden, dass dabei zumindest die schwerwiegendsten Fehler entdeckt wurden.

In **[PBL08a]** wird daher ein Protokoll vorgeschlagen, nach dem solche Prüfungen mit der Unterstützung von kommerziellen statischen Analysewerkzeugen in einer strukturierten Form durchgeführt werden sollen. Dabei wird zwar ein Prozess vorgegeben, wie die Prüfung stattfinden soll, wie jedoch einzelne Fehler entdeckt werden sollen, ist nur sehr allgemein beschrieben.

[Pan07] rät stattdessen zu formellen Inspektionen die sich an den *Design and Code Inspections* nach Fagan orientieren. Die Organisation, der Ablauf, sowie die beteiligten Rollen werden dabei eins zu eins von den Code Inspektionen übernommen [LL07, S. 282 ff.]. So soll der Prüfling, ähnlich wie bei den Inspektionen von Quellcode, zeilenweise überprüft werden und die zu überprüfenden Vorgaben müssen zuvor anhand von Richtlinien festgelegt worden sein. Außerdem sollten die zu überprüfenden Komponenten eines Prüflings so gewählt werden, dass eine Prüfungssitzung die Dauer von zwei Stunden nicht überschreitet. Wie bei den Inspektionen von Software wird dabei eine Inspektion durch drei bis vier Gutachter angeraten, da bei Inspektionen durch Einzelperson signifikant weniger Befunde entdeckt wurden [Pan99, GHJ+96]. Panko sieht bei dieser Art der Spreadsheet-Prüfung den Vorteil, dass relativ geringe Anforderungen an die Gutachter gestellt werden und sich die Code Inspektionen bei der Prüfung von Software-Artefakten bewährt haben. Es gibt jedoch keine Belege dafür, dass solche Spreadsheet Inspektionen in der Praxis oder als Experiment erfolgreich durchgeführt wurden, oder ob sie sich überhaupt für die Prüfung von Spreadsheets eignen.

5.2.2 Mechanische Spreadsheet-Prüfungen

Will man dynamische Prüfmethoden auf Spreadsheets anwenden, ist man zunächst mit dem Orakel-Problem konfrontiert [Pan07], das auch beim Test von traditioneller Software auftritt. Denn beim Testen von Software-Programmen werden diese mit verschiedenen Eingaben ausgeführt, um die Korrektheit der produzierten Ergebnisse zu überprüfen. Das Problem liegt dabei darin, dass eine Instanz benötigt wird, die in der Lage ist die produzierten Ergebnisse bezüglich ihrer Korrektheit beurteilen. Bei Spreadsheets ist im Allgemeinen nicht klar, wie ein solches Test-Orakel realisiert werden kann, um die Ergebnisse richtig zu beurteilen. So gibt es bei Spreadsheets keine offensichtlichen Kriterien dafür, dass ein Test fehlgeschlagen ist, wie das bei traditionellen Softwareprogrammen bei einem Absturz des Programms der Fall ist. Auch ein Vergleich von Ist-Ergebnissen mit Soll-Ergebnissen ist in den seltensten Fällen möglich, da die Soll-Ergebnisse aufgrund der meist fehlenden Anforderungen nicht angegeben werden können. Und auch wenn die Anforderungen erfasst wurden, besteht das Problem, dass die Berechnungen des Spreadsheets meist zu komplex sind, um Soll-Ergebnisse mit vertretbarem Aufwand auf eine andere Art zu bestimmen.

[Pry08] gibt dennoch einige Tipps, wie Unit-Tests mit Hilfe von Kontrollrechnungen im zu überprüfenden Spreadsheet selbst durchgeführt werden können, falls die Möglichkeit besteht die Korrektheit der Ergebnisse objektiv zu beurteilen. Zur Unterstützung wird dabei ein selbst entwickeltes Prüfwerkzeug mit Namen *Xlsior*[2] verwendet, welches jedoch inzwischen nicht mehr zur Verfügung gestellt wird.

[ACM00] versucht das Problem eines fehlenden Test-Orakels, sowie der fehlenden Testkenntnisse von Endbenutzern durch symbolisches Testen und Intervall-Tests zu lösen. So können durch

[2] http://www.louisepryor.com/xlsior/

Intervall-Tests die Ergebnisse von Formeln mittels vom Anwender spezifizierter Grenzwerte auf Plausibilität überprüft werden. Der Anwender wird somit zum Test-Orakel, jedoch wird die Beurteilung der Ergebnisse durch die Angabe von korrekten Intervallen erleichtert.

Statische Prüfungen von Spreadsheets haben gegenüber den dynamische Prüfungen den Vorteil, dass allgemeine Vorgaben für Spreadsheets definiert werden können und für deren Anwendung weniger Kenntnisse benötigt werden. Statische Analysetechniken können dabei auf unterschiedliche Weise bei der Prüfung von Spreadsheets nützlich sein. Viele Ansätze verwenden die statische Analyse dazu, um die Struktur von Spreadsheets zu analysieren und diese dann zu visualisieren. Dadurch wird das Verständnis der Struktur eines Spreadsheets für dessen Benutzer erleichtert und er wird dadurch bei der Prüfung des Spreadsheets unterstützt. In anderen Ansätzen wird die statische Analyse stärker automatisiert, indem über Muster-Vergleiche und abstrakte Interpretation automatisch nach Defekten gesucht werden kann.

In **[HPD11]** wurde beispielsweise das Visualisierungswerkzeug *Breviz* entwickelt, das die Struktur von Spreadsheets mittels Datenfluss Diagrammen darstellen kann. Dabei kann interaktiv durch verschiedene Ansichten navigiert werden, um das Verständnis der Funktionalität eines Spreadsheets zu erleichtern. Zudem wird angeregt, dass *Breviz* dazu verwendet werden kann, um Anomalien in Spreadsheets zu entdecken. Die Autoren geben dazu eine Reihe von schlechten Strukturen bei Spreadsheets an, die vermutlich auf die Anwesenheit von Fehlern hinweisen könnten und mit Hilfe des visualisierten Datenflusses vom Anwender erkannt werden können. Eine Evaluierung dieses Ansatzes wurde dabei noch nicht durchgeführt, jedoch wird dies als Ziel für weitere Arbeiten ausgegeben.

Ein ähnlicher Ansatz wird in **[CHM08]** unternommen, bei dem Formelbereiche in einem Spreadsheet in Äquivalenzklassen eingeteilt und dargestellt werden. In separaten Fenstern können dabei Äquivalenzrelationen und Datenabhängigkeiten untersucht werden, um durch Unregelmäßigkeiten im dargestellten geometrischen Muster von ähnlichen Formeln bei der Suche nach Fehlern zu helfen. In einer Evaluation des entwickelten Werkzeugs in einem industriellen Kontext wurden Defekte in 3,03% aller Zellen entdeckt, jedoch konnten keine Fehler gefunden werden, die stark abweichende Ergebnisse verursachten und dadurch schwerwiegende Konsequenzen für das Unternehmen bewirken hätten können.

In **[AE06]** wird das vollautomatische Prüfwerkzeug *UCheck* vorgestellt, das versucht über die vorhanden textuellen Beschriftungen und Überschriften in Spreadsheets Information über die semantischen Einheiten einzelner Zellen zu gewinnen. Es wird also eine Art Typisierung der Zellen anhand der Überschriften vorgenommen und die daraus gewonnen Informationen werden verwendet, um Formeln mittels abstrakter Interpretation auf Plausibilität zu untersuchen. Vorteilhaft ist dabei, dass für den Nutzer kein Aufwand entsteht, da die Prüfung vollautomatisch erfolgt. Jedoch ist die Lösung auf zusätzliche Informationen durch Überschriften angewiesen und kann nicht allgemein dazu verwendet werden um Spreadsheets zu prüfen.

Kommerziell vertriebene Prüfwerkzeuge für Spreadsheets verwenden zumeist eine Kombination aus den verschiedenen Ansätzen, um die Prüfung von Spreadsheets zu unterstützen. Da diese

Werkzeuge zumeist nicht dazu eingesetzt werden, um die Anzahl der Fehler zu reduzieren, sondern nur um die Risiken, die von dem Spreadsheet ausgehen einschätzen zu können, bezeichnet man sie meist als *Audit-Werkzeuge* [AP10].

In **[NO10]** und **[AP10]** wird eine Auswahl solcher Audit-Werkzeuge untersucht. Nixon [NO10] verwendet dazu Spreadsheets, bei denen mittels Error-Seeding Defekte eingebaut wurden, Panko [AP10] hingegen untersucht in Experimenten entstandene Spreadsheets. Außerdem unterscheiden sich die beiden Studien dadurch, dass Nixon nur die automatischen Prüfungen der Werkzeuge verwendet, Panko hingegen nur die Funkionen der Werkzeuge, die auffällige Zellen markieren, um menschliche Prüfer bei der Suche nach Fehlern zu unterstützen. Auch die Ergebnisse der beiden Studien weichen stark voneinander ab. Nixon kommt zu dem Schluss, dass die statischen Prüfwerkzeuge bezüglich ihrer Effektivität mit nichtmechanischen Inspektionen von Spreadsheets, die Zelle für Zelle durchgeführt werden, vergleichbar sind. Da sie jedoch bei weitem geringere Kosten verursachen, sieht er sie als gute Alternative zu nichtmechanischen Prüfungen an, um zumindest alle nicht-funktionalen Defekte in Spreadsheets zu erkennen. In der Studie von Panko schneiden die Werkzeuge im Vergleich zu den nichtmechanischen Inspektionen jedoch ungemein schlechter ab. Dies wird damit begründet, dass in den geprüften Spreadsheets überwiegend funktionale Defekte enthalten waren, die nur schwer mittels statischer Analyse erkannt werden können. Panko zieht in [AP10] daher die plausible Schlussfolgerung, das statische Analysewerkzeuge zwar als initiale Fehlererkennung geeignet sind, jedoch nicht ausreichen, um allein alle Defekte in Spreadsheets zu erkennen.

5.3 Schlussfolgerung

In diesem Kapitel wurde untersucht, ob die bestehenden Ansätze aus dem Software Engineering auf Spreadsheets übertragen werden können, um die Qualität von Spreadsheets zu verbessern. Dabei wurde festgestellt, dass durch unterschiedliche Voraussetzungen für die Verwendung der Ansätze häufig große Probleme bei deren Umsetzung bestehen. So ist bei den meisten Nutzern von Spreadsheets die notwendige Wahrnehmung der Risiken nicht vorhanden und die notwendigen Kenntnisse und Fähigkeiten fehlen, um die vorhandenen Ansätze anzuwenden.

Die bestehenden Ansätze reichen daher nicht aus, um eine angemessene Qualität von Spreadsheets mit vertretbarem Aufwand sicherzustellen. Die Verbesserung der Spreadsheet-Qualität durch organisatorische und konstruktive Maßnahmen verspricht dabei auf lange Sicht den größten Erfolg. Jedoch werden angesichts der vielfachen Verwendung von Spreadsheets, Maßnahmen benötigt, die die Qualität von bestehenden Spreadsheets auf kurz bis mittelfristige Frist erhöhen können. Dazu bieten sich analytische Maßnahmen an.

Nach der Untersuchung der vorhandenen Ansätze zur Spreadsheet-Prüfung wird deutlich, dass viele Ansätze nur bedingt dazu geeignet sind, um Fehler in Spreadsheets durch Endbenutzer zu erkennen. Rein nichtmechanische Prüfungen sind beispielsweise mit einem hohen Zeitaufwand verbunden und werden durch die fehlende Lokalität von Spreadsheets erschwert. Angesichts

der geringen Wahrnehmung von Spreadsheet-Risiken scheint es unwahrscheinlich, dass solche Maßnahmen in der Praxis durchzusetzen sind [AP10]. Bei Dynamischen Prüfungen hingegen, ist das Fehlen eines geeigneten Test-Orakels, sowie die fehlende Ausbildung von Endbenutzern zum Testen von Software eine große Hürde. Ansätze wie in [ACM00] haben jedoch das Potential, dieses Problem zu lösen. Statische Prüfungen haben den Vorteil, dass sie kaum Aufwand verursachen und kaum Voraussetzungen an den Endbenutzer stellen. Zudem besitzen sie das Potential, eine hohe Anzahl an Defekten In Spreadsheets zu entdecken und dadurch bei der Erkennung von Fehlern nützlich zu sein. Dabei muss jedoch beachtet werden, dass nicht alle Fehler in Spreadsheets durch statische Prüfungen erkannt werden können.

Als Schlussfolgerung dieser Erkenntnisse soll im folgenden Kapitel ein Konzept zur Prüfung von Spreadsheet vorgestellt werden, das auf der statischen Analyse basiert, jedoch durch die Integration von anderen Prüfungsarten vorsieht, um die bestehenden Einschränkungen der statischen Prüfung zu kompensieren.

Kapitel 6

Konzept

Basierend auf den Erkenntnissen aus den vorherigen Kapiteln wird in diesem Kapitel ein Konzept entwickelt, mit dem Spreadsheets auf Fehler überprüft werden können. Dabei wird zunächst die Grundidee vorgestellt, sowie erläutert, wie die Umsetzung dieser Idee erfolgen soll. Anschließend wird das erarbeitete Konzept anhand einer Metapher genauer erläutert und zuletzt wird beschrieben, wie die in dieser Arbeit realisierte Lösung verwendet werden kann.

6.1 Technische Grundlage zur Spreadsheet-Prüfung

In dieser Arbeit soll ein Konzept entwickelt und realisiert werden, um die bestehende problematische Situation bei der Qualität von Spreadsheets auf kurz bis mittelfristige Sicht zu verbessern [Kul11]. Die realisierte Lösung soll dabei als technische Grundlage verwendet werden können, um systematische, rechnerunterstützte Prüfungen von Spreadsheets effektiv und kostengünstig durchführen zu können. Dabei soll das vorhandene Wissen über Spreadsheet-Systeme und die bestehenden Voraussetzungen für die Qualität von Spreadsheets berücksichtigt werden. So sollen etwa die Vorzüge von Spreadsheet-Systemen erhalten bleiben und das vorhandene Wissen von typischen Endbenutzern in das Konzept miteinbezogen werden.

Zu diesem Zweck soll das erweiterbare Java-Framework *Spreadsheet Inspection Framework (SIF)* geschaffen werden, das die Durchführung von verschiedenen Prüfungsarten für Spreadsheets ermöglicht. Mit Hilfe von *SIF* sollen Prüfwerkzeuge für Spreadsheets realisiert werden können, die Endbenutzer nach einem konkretem Prüfverfahren bei der Prüfung von Spreadsheets unterstützen. Dabei sollen die verschiedenen Arten der Softwareprüfung in einer verzahnten Weise genutzt werden können, um das Erkennen einer möglichst hohen Anzahl an Defekten in den geprüften Spreadsheets zu ermöglichen. Um den Zeitpunkt der Prüfung möglichst frei zu gestalten, sollfür die Prüfung durch *SIF* nur eine korrekt gespeicherte Spreadsheet-Datei benötigt werden. Die Realisierung von *SIF* soll dabei in mehreren Ausbaustufen geschehen, wovon die erste in dieser Arbeit entstehen soll. Der Fokus dieser ersten Ausbaustufe liegt dabei auf der Durchführung von statischen Prüfungen.

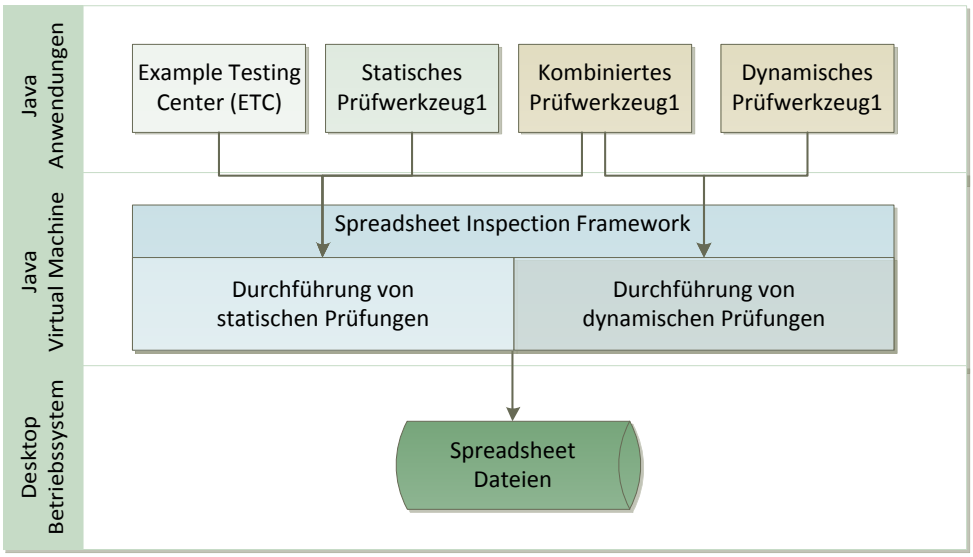

Abbildung 6.1: *SIF* als technische Grundlage für die Prüfung von Spreadsheets

6.1.1 Statische Spreadsheet-Prüfung

Die Hauptaufgabe von *SIF* in der ersten Ausbaustufe ist es, die drei unterschiedlichen Aufgaben der statischen Prüfung zu ermöglichen. So soll erstens die Einhaltung festlegbarer Regeln überprüft werden können, zweitens sollen Konsistenz-Prüfungen ermöglicht werden und drittens soll die Möglichkeit geschaffen werden, quantitative Merkmale des Spreadsheets automatisch zu erfassen. Dabei soll ein ähnlicher Ansatz wie bei dem in Unterabschnitt 3.5.2 vorgestellten *Findbugs* verwendet werden. So soll anstelle einer eingeschränkten Anzahl spezialisierter Techniken, eine breite Anzahl an einfachen Techniken zum Einsatz kommen. In dieser Arbeit sollen dabei die Prüfungen der drei Vorschriften *Konstanten in Formeln*, *Leserichtung* und *Formelkomplexität*, die auch häufig in der bestehenden Literatur empfohlen werden, umgesetzt werden. Eine genaue Definition der umgesetzten Vorschriften wird dabei in Kapitel 7 gegeben.

Wie auch bei Findbugs, ist ein wichtiger Punkt die Konfigurierbarkeit der überprüften Vorschriften, um die Prüfungen an die individuellen Anforderungen des jeweiligen Nutzungskontextes anpassen zu können. Daher können Richtlinien aus bestehenden Vorschriften vom Benutzer frei zusammengestellt und einzeln konfiguriert werden. Mit Hilfe einer Java-API soll es zudem möglich sein, neue Vorschriften hinzuzufügen und deren Überprüfung zu realisieren. Zwar müssen in der initialen Version die Prüfverfahren für neue Vorschriften von einem professionellen Programmierer implementiert werden, jedoch soll es ermöglicht werden, diesen Mechanismus in Zukunft mit einer benutzerfreundlichen Möglichkeit zu ersetzen. Zur Demonstration der Fähigkeiten von *SIF* soll zudem das prototypisches statische Analysewerkzeug *Example Testing Center* (*ETC*) entstehen, das die implementierten Funktionen der ersten Ausbaustufe verwendet.

6.1.2 Erhoffter Nutzen der ersten Ausbaustufe

Wir sind uns bewusst, dass es allein durch die statischen Prüfungen der ersten Ausbaustufe nicht möglich sein wird, alle Fehler in Spreadsheets zu erkennen. So ist davon auszugehen, dass die Prüfung von funktionalen Eigenschaften nur eingeschränkt mit der ersten Ausbaustufe von *SIF* durchführbar sein wird. Jedoch hoffen wir, durch die erste Ausbaustufe einen Ansatzpunkt zu schaffen, um die Suche nach Fehlern in Spreadsheets zu unterstützen. So soll der Nutzer auf Verstöße von Vorschriften, Inkonsistenzen und quantitative Merkmale des geprüften Spreadsheets aufmerksam gemacht werden. Durch diese Anhaltspunkte soll dem Nutzer dabei geholfen werden, Fehler und qualitative Mängel im geprüften Spreadsheet zu entdecken und dessen Qualität zu beurteilen.

In weiteren Ausbaustufen soll diese Suche nach Fehlern in Spreadsheets weiter unterstützt werden. So könnte die manuelle Inspektion in Zukunft durch die Visualisierung des Spreadsheets und der gefundenen Verstöße durch die Prüfanwendungen vereinfacht werden. Und mit der Erweiterung des Frameworks, um die Unterstützung von dynamischen Prüfungen, soll die bestehende Einschränkung der Prüfungen auf überwiegend nicht-funktionale Qualitätsmerkmale behoben werden. Dabei können die Verstöße der statischen Prüfung genutzt werden, um auffällige Zellen des Spreadsheets zu entdecken und diese effektiv auf Fehler zu überprüfen.

6.2 Metapher

Um das Verständnis des gewählten Ansatzes zu erleichtern, wurde eine Metapher für die Veranschaulichung des Konzepts gewählt. Die Wahl fiel dabei auf das Bild einer technischen Inspektions-Werkstatt, wie sie in technischen Prüfzentren, beispielsweise bei solchen für die Zulassung von Kfz-Fahrzeugen, genutzt wird. In diesem Zusammenhang sind viele der Eigenschaften einer solchen Werkstatt und der mit ihr durchführbaren Prüfungen auch auf *SIF* übertragbar. Einige Eigenschaften können bei diesem Bild jedoch nicht übertragen werden. Daher werden wir zuerst die wesentlichen Merkmale einer Inspektions-Werkstatt am Beispiel einer Kfz-Inspektions-Werkstatt nennen. Anschließend werden wir aufzuzeigen, welche Analogien und welche Unterschiede bestehen, wenn man diese Bild auf die Inspektion von Spreadsheets überträgt.

6.2.1 Technisches Kfz-Prüfzentrum

Die Zulassung von Fahrzeugen zum Straßenverkehr ist in den meisten Ländern gesetzlich geregelt. So wird für neue Fahrzeugtypen eine Typ-Genehmigung benötigt und Kfz-Fahrzeuge, die am Straßenverkehr teilnehmen sollen, müssen regelmäßigen Inspektionen unterzogen werden, um sie auf ihre Vorschriftmäßigkeit zu überprüfen. Die Überprüfung der Fahrzeuge erfolgt dabei durch staatlich anerkannte Prüfzentren in Inspektions-Werkstätten. Dabei ist die Inspektion ohne die technischen Mittel, die in den Inspektions-Werkstätten bereitgestellt werden, nicht oder nur sehr schwer durchzuführen. Die Inspektions-Werkstatt und die mit ihr durchgeführte Prüfung zeichnen sich dabei wie folgt aus:

Prüfung

Bei der Prüfung handelt es sich um eine zerlegungsfreie Sicht-, Funktions- und Wirkungsprüfung. Das heißt, das Fahrzeug wird zur Prüfung nicht verändert und im Zusammenspiel von Mensch und Maschine mit statischen und dynamischen Mitteln geprüft. Das Ziel der Prüfung ist es dabei, die Konformität des Fahrzeugs mit gesetzlichen Vorschriften zu überprüfen. Dadurch soll die Verkehrssicherheit erhöht, Unfälle vermieden und entstehende Schäden bei Unfällen verringert werden. Geprüft werden dabei sowohl funktionale, wie nicht-funktionalen Anforderungen. Bei der Prüfung selbst werden Verstöße gegen die Vorschriften jedoch nur erkannt und bewertet, eine Korrektur von Mängeln erfolgt nicht, sondern ist vom Hersteller bzw. Halter des Fahrzeuges durchzuführen. Bei Nichtbestehen der Prüfung wird dem Fahrzeugtyp bzw. Fahrzeug die Teilnahme am Straßenverkehr verweigert, bis die angezeigten Mängel behoben sind.

Inspektions-Werkstatt

Die Inspektions-Werkstatt bietet die technischen Mittel, um die Prüfung des Fahrzeuges durchzuführen. Die Prüfung selbst wird zwar durch einen professionellen Prüfer durchgeführt, dieser kann die Prüfung jedoch ohne die Prüfmittel, die die Inspektions-Werkstatt zur Verfügung stellt, nicht angemessen durchführen. Als Prüfmittel stehen dem Prüfer dabei verschiedene Prüfstände und Werkzeuge zur Verfügung. So gibt es beispielsweise eine Hebebühne, die dem Prüfer erlaubt den Zustand und die einzelnen Teile des Fahrzeuges genauer zu betrachten, als auch spezielle Werkzeuge, wie etwa einen Spannungsmesser, der die anliegende Spannung bei elektronischen Komponenten überprüfen kann. Komplette Prüfstände, wie etwa ein Bremsprüfstand, können dem Prüfer Abweichungen und Auffälligkeiten anzeigen, die auf die Anwesenheit eines Fehler hinweisen könnten. Die Ursache für die Abweichung kann jedoch vom Prüfstand meist nicht selbstständig identifiziert werden, sondern muss vom Prüfer durch manuelle Inspektion oder mit Hilfe von weiteren Prüfungen eingegrenzt werden.

6.2.2 Technisches Spreadsheet-Prüfzentrum

Wie auch bei Kfz-Fahrzeugen, ist die Prüfung von Spreadsheets ohne technische Hilfsmittel nicht oder nur sehr schwer durchzuführen. *SIF* soll daher die Funktionalität einer Inspektions-Werkstatt bereitstellen und damit die technische Grundlage schaffen, um technische Prüfzentren zur Inspektion von Spreadsheets realisieren zu können.

Prüfung

Die Prüfung von Spreadsheets soll nach ähnlichen Grundlagen ablaufen, wie bei Kfz-Fahrzeugen. So soll das Spreadsheet ohne Veränderung durch Mensch und Maschine mit statischen und dynamischen Mitteln geprüft werden. Ziel der Prüfung es auch hier, die Konformität mit Richtlinien zu überprüfen, um eine angemessene Qualität des Spreadsheets sicherstellen zu können. Dadurch können die negativen Konsequenzen, die durch fehlerhafte Spreadsheets verursacht

werden, vermieden oder zumindest reduziert werden. Im Gegensatz zur Kfz-Prüfung existieren dabei jedoch keine gesetzlichen Vorgaben und auch keine allgemeingültigen Standards. Stattdessen müssen die Richtlinien von den Nutzern der Spreadsheet-Inspektions-Werkstatt festgelegt werden. Diese Richtlinien sollten so gewählt werden, dass durch deren Überprüfung Defekte, die bei der Verwendung des Spreadsheets in der Unternehmenswelt Gefahren darstellen könnten, kenntlich gemacht und mangelhafte Spreadsheets vor deren Benutzung aussortiert werden können. Wie bei der Kfz-Prüfung auch, werden Verstöße gegen die Vorschriften jedoch nur erkannt und bewertet. Die Korrektur ist vom Ersteller bzw. Nutzer des Spreadsheets durchzuführen.

Inspektions-Werkstatt

Die Inspektions-Werkstatt für Spreadsheets soll die technischen Mittel bereitstellen, um die Prüfung der Spreadsheets adäquat durchführen zu können. Anders als bei der Kfz-Inspektion wird die Prüfung jedoch nicht zwangsweise von einem professionell ausgebildeten Prüfer durchgeführt. Stattdessen soll es auch dem durchschnittlichen Endbenutzer ermöglicht werden Spreadsheets zu prüfen. Zu diesem Zweck stellt *SIF* eine Reihe von Prüfmitteln bereit. So existieren einerseits Werkzeuge, die das Spreadsheet auf dessen Beschaffenheit untersuchen und eine Inventurliste von dessen Bestandteilen erstellen können, aber auch spezielle Werkzeuge, die bestimmte Eigenschaften bei einzelnen Bestandteilen überprüfen können. Prüfstände, die die einzelnen Vorschriften der vergebenen Richtlinie überprüfen, können automatisch abgefahren werden und liefern dem Prüfer eine Liste mit den gefunden Abweichungen und Auffälligkeiten. Der Prüfer kann dann mittels der gegeben Anhaltspunkte nach Fehlern und deren Ursachen im Spreadsheet suchen oder weitere Prüfungen durchführen.

6.2.3 Bausteine der ersten Ausbaustufe der Spreadsheet-Inspektions-Werkstatt

Zur Umsetzung der ersten Ausbaustufe der Spreadsheet-Inspektions-Werkstatt werden die folgenden Bausteine benötigt:

Vorschrift Eine Vorschrift definiert eine Vorgabe, die von einem Spreadsheet erfüllt werden muss. Bei dieser Vorgabe kann es sich um die Einhaltung von Konsistenz, um die Erfüllung einer quantitativen Eigenschaft oder um die Einhaltung einer generellen Regel handeln. Sinnvolle Vorgaben für Spreadsheets können je nach deren Nutzungskontext variieren und daher können Vorschriften an die überprüften Spreadsheets angepasst werden.

Richtlinie Eine Auswahl an Vorschriften kann dabei zu einer Richtlinie zusammengefasst werden, die dann als Vorgabe für Spreadsheets in einem bestimmten Nutzungskontext verwendet werden kann. Es handelt sich bei einer Richtlinie also um einen Satz von Best Practises, wie etwa in [O'B05, Raf08] beschrieben, jedoch ohne solche Vorschriften, die das Vorgehen bei der Entwicklung von Spreadsheets festgelegen.

Verstoß Die Überprüfung von Vorschriften erfolgt in der ersten Ausbaustufe von *SIF* mittels automatischer statischer Prüfungen. Wenn Abweichungen von den Vorgaben einer Vorschrift erkannt werden, wird ein Verstoß gemeldet. Wie auch bei den statischen Prüfungen von traditioneller Software (Unterabschnitt 3.5.2) handelt es sich bei einem Verstoß nicht immer um einen echten Defekt.

Inspektionsauftrag Ein Inspektionsauftrag wird von einem Nutzer der Inspektions-Werkstatt aufgegeben, um ein Spreadsheet auf die Vorgaben einer Richtlinie zu überprüfen. Der Inspektionsauftrag wird dabei mit Hilfe von *SIF* erstellt, konfiguriert und durchgeführt. Zur vollständigen Konfiguration eines Inspektionsauftrags gibt der Nutzer einen Namen, sowie das zu prüfende Spreadsheet an, wählt eine der vorhandenen Richtlinien aus und passt deren Vorgaben an den Prüfling an.

Spreadsheet-Elemente Ein Spreadsheet besteht im Wesentlichen aus den in Abschnitt 2.1 beschriebenen Elementen. Zwar können alle gängigen Spreadsheet-Sprachen diese grundlegenden Elemente definieren, jedoch existieren Spezialisierungen und Generalisierungen dieser Elemente, die nicht explizit mit den vorhanden Sprachkonstrukten ausgedrückt werden können. So können beispielsweise Eingabe-Zellen, die vom Nutzer des Spreadsheets mit Inhalten gefüllt werden müssen, nicht direkt als solche ausgezeichnet werden, sondern müssen vom Ersteller über Formatierung, Anordnung oder ähnliches kenntlich gemacht werden.

- ***Als Basis-Elemente*** werden daher in in dieser Arbeit Elemente bezeichnet, für die Sprachkonstrukte in allen gängigen Spreadsheet-Sprachen vorhanden sind.
- ***Spezial-Elemente*** sind Spezialisierungen oder Generalisierungen von Basis-Elementen, die individuell festgelegt werden können.

Um diese mangelnde Ausdrucksmöglichkeiten von bestehenden Spreadsheet-Sprachen zu kompensieren, ist es durch eine Schnittstelle in *SIF* möglich beliebige Spezial-Elemente zu definieren, diese automatisch zu erkennen und für die Prüfung zur Verfügung zu stellen.

Element-Scanner Die Erkennung von Spreadsheet-Elementen erfolgt dabei durch sogenannte Element-Scanner. Diese werden vor der Prüfung auf das Spreadsheet angewendet und erstellen eine Inventurliste von dessen Bestandteilen. Zusätzlich sollen die Element-Scanner quantitative und strukturelle Merkmale des Spreadsheets erfassen können, um so die Struktur und die Beschaffenheit des geprüften Spreadsheets aufzuzeigen.

Spreadsheet-Inventar Das Spreadsheet-Inventar ist die Auflistung aller Elemente eines Spreadsheets, die durch Element-Scanner erkannt wurden. Es stellt damit eine Bestandsliste aller Komponenten eines Spreadsheets dar.

Prüfstände sind die Prüfmittel, die die Einhaltung von Vorschriften für alle Bestandteile eines Spreadsheets überprüfen und bei Abweichungen eine Liste von Verstößen erstellen. Prüfstände können dabei entweder modular aus bestehenden Prüfwerkzeugen zusammengestellt werden oder als monolithischer Prüfstand entwickelt werden.

Prüfwerkzeuge Prüfwerkzeuge sind modulare Prüfmittel, die für ein Element einer Klasse von Spreadsheet-Elementen überprüfen können, ob dieses Element eine bestimmte Eigenschaft besitzt oder eine bestimmte Vorschrift einhält. So kann ein Prüfwerkzeug beispielsweise eine Zelle mit numerischen Inhalt, darauf überprüfen, ob deren Inhalt in der richtigen Genauigkeit dargestellt wird. Ein anderes Prüfwerkzeug wiederum kann prüfen, ob eine Zelle mit Textinhalt ein bestimmtes Schriftbild erfüllt. Neue Prüfwerkzeuge, auch für Spezial-Elemente, können entwickelt und als Prüfmittel bereitgestellt werden. Der Zweck dieser Prüfwerkzeuge ist es dabei, generische Prüfstände aus einem oder mehreren dieser Werkzeuge modular zusammenstellen zu können.

Prüffeld Ein Prüffeld ist der Ort an dem Prüfung eines Spreadsheets, die durch einen Inspektionsauftrag definiert ist, durchgeführt wird. Zu diesem Zweck werden die notwendigen Prüfstände im Prüffeld aufgebaut, mit den Vorgaben aus der gewählten Richtlinie des Inspektionsauftrags konfiguriert und dann der Reihe nach ausgeführt.

6.3 Verwendung

Im folgenden Abschnitt soll beschrieben werden, wie die erste Ausbaustufe von *SIF* verwendet werden kann, um Spreadsheets statisch zu prüfen und wie neue Vorschriften zur Überprüfung angeboten werden können.

6.3.1 Ablauf einer Inspektion

Der Ablauf einer Inspektion mit *SIF* gestaltet sich wie in Abbildung 6.2 dargestellt. Dabei wird bereits angedeutet, wie die die Prüfungen der weiteren Ausbaustufen in den Ablauf der statischen Prüfung integriert werden können. Die Details des Ablaufs einer Inspektion in der ersten Ausbaustufe sollen im Folgenden genauer beschrieben werden.

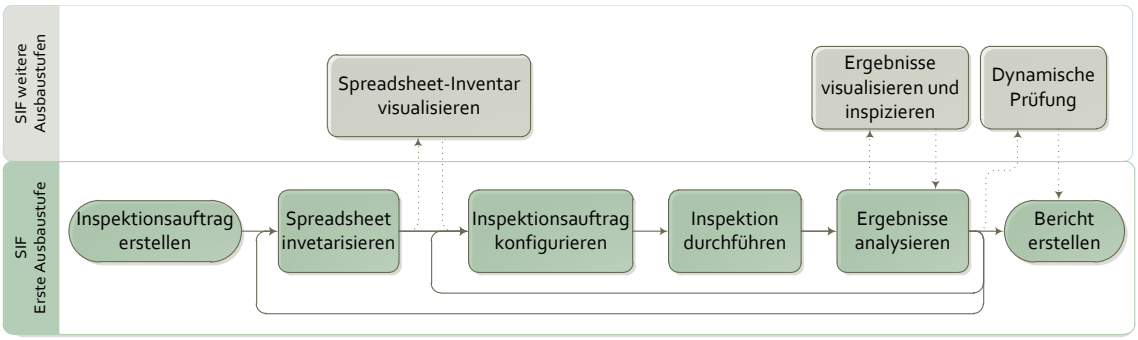

Abbildung 6.2: Ablauf einer Inspektion mit dem *Spreadsheet Inspection Framework*

Inspektionsauftrag erstellen

In der Erstellungsphase wird eine neuer Inspektions-Auftrag angelegt. Dazu spezifiziert der Nutzer welches Spreadsheet inspiziert werden soll, in dem er den Pfad zur entsprechenden Spreadsheet-Datei angibt. Das Spreadsheet wird dann geladen und in ein internes Datenmodell übertragen. Bei der Metapher entspricht dieser Vorgang, dem Vorfahren des Fahrzeugs in die Werkstatt eines Prüfzentrums und dem Beantragen einer Inspektion

Spreadsheet inventarisieren

Im Anschluss an die Erstellung des Inspektionsauftrags wird das Spreadsheet auf dessen Struktur und Beschaffenheit untersucht, wobei eine Inventurliste mit allen Bestandteilen des Spreadsheets erstellt wird. Durch diesen Schritt werden, wie bei einem Fahrzeug auf einer Hebebühne, zunächst die einzelnen Bestandteile des Spreadsheets betrachtet und es wird ein erster Eindruck vom Prüfling gewonnen. Dieser soll dabei helfen, die richtigen Prüfungen für den Prüfling auszuwählen. Zusätzlich können durch diesen Schritt bereits vorab untaugliche Prüflinge, wie beispielsweise ein Spreadsheet ohne Zellinhalte, aussortiert werden.

Inspektionsauftrag konfigurieren

Nach der Inventarisierung des Spreadsheets muss der angelegte Inspektions-Auftrag konfiguriert werden. Dazu muss der Nutzer eine der vom Prüfzentrum angebotenen Richtlinien auswählen, nach deren Vorgaben das Spreadsheet geprüft werden soll. Die einzelnen Vorschriften der gewählten Richtlinie können dann dem Prüfling entsprechend konfiguriert werden. So können beispielsweise einzelne Teile des Spreadsheets von bestimmten Prüfungen ausgeschlossen werden oder die Parameter der Prüfstände eingestellt werden. Die Konfiguration entspricht der Anpassung der Kfz-Prüfung an den jeweiligen Fahrzeugtyp. So muss etwa für Fahrzeuge neueren Baujahres ein Abgasuntersuchung nicht zwingend durchgeführt werden und für landwirtschaftliche genutzte Fahrzeuge gelten anderen Vorgaben als für normale Pkws.

Inspektion durchführen

Zur Durchführung der Inspektion wird von *SIF* ein entsprechendes Prüffeld angelegt. In diesem Prüffeld werden die Prüfstände aufgebaut, die für die Überprüfung der gewählten Richtlinie benötigt werden, und mit den Einstellungen aus dem Inspektion-Auftrag konfiguriert. Anschließend werden die zuvor erkannten Bestandteile der Reihe nach in den Prüfständen geprüft. Von einem Prüfstand erkannte Verstöße werden entsprechend der Vorschrift klassifiziert, bewertet und gruppiert ausgegeben. Die Listen mit den Verstößen der einzelnen Prüfstände werden dann gesammelt und als Ergebnis der Inspektion ausgegeben.

Ergebnisse analysieren

Die resultierende Liste mit allen gefunden Verstößen kann dann vom Prüfer analysiert und exportiert werden. Jeder Verstoß gibt dabei genaue Auskunft darüber, welches Spreadsheet-Element den Verstoß verursacht hat, warum dieser Verstoß auftrat, welche Schwere dem Verstoß zugewiesen wurde und wie man ihn beseitigen könnte. Im Zuge von weiteren Ausbaustufen wäre es vorstellbar, dass die gefundenen Verstöße visualisiert werden können und so dem Benutzer bei der Analyse unterstützen. Außerdem könnten Elemente mit vielen Verstößen im Anschluss an die statische Prüfung mit dynamischen Prüfungen getestet werden, um diese auf bisher unentdeckte Fehler zu überprüfen.

Bericht erstellen

Nach Abschluss der Analyse besteht die Möglichkeit die gefundenen Verstöße zu exportieren, um diese mit Hilfe einer Spreadsheet-Software zu beheben.

6.3.2 Erweiterung der angebotenen Prüfungen

Ein weitere wichtige Funktionalität von *SIF* ist die Erweiterung der angebotenen statischen Prüfungen. So können einerseits Richtlinien aus bestehenden Vorschriften zusammengestellt werden, aber es wird auch die Möglichkeit angeboten, neue Vorschriften hinzufügen und diese überprüfen zu lassen.

Erstellung von Richtlinien

Wie bereits in Abschnitt 2.2 erläutert, existieren keine allgemein anerkannten Best Practices für Spreadsheets, da diese häufig vom Nutzungskontext des jeweiligen Spreadsheets abhängen. Die Erstellung von neuen Richtlinien erlaubt daher, dass die Prüfungen einer Inspektion speziell an den jeweiligen Nutzungskontext angepasst werden können. So kann eine neue Richtlinien frei aus den angebotenen Vorschriften der Inspektions-Werkstatt zusammengestellt werden. Diese Richtlinie kann dann vorkonfiguriert und gespeichert werden, so dass sie für die Verwendung von Inspektionen mehrerer Spreadsheets aus dem selben Nutzungskontext ausgewählt werden kann.

Hinzufügen von neuen Vorschriften

Um das Angebot an überprüfbaren Vorschriften zu erweitern, ist es außerdem möglich neue Prüfstände zu entwickeln. Zu diesem Zweck existiert ein Schnittstelle, mit der neue Prüfstände entwickelt und bei der Inspektions-Werkstatt registriert werden können. Basierend auf diesen Prüfständen können dann neue Vorschriften angeboten und von *SIF* überprüft werden.

Kapitel 7

Anforderungen

Basierend auf dem in Kapitel 6 vorgestellten Konzept, ergeben sich dabei folgende funktionale und nicht-funktionale Anforderungen, die an die erste Ausbaustufe des *Spreadsheet Inspection Framework* gestellt werden.

7.1 Funktionale Anforderungen

Einlesen von Spreadsheets aus Spreadsheet-Dateien unterschiedlichen Formats

Um Spreadsheets zu beliebigen Stadien in deren Lebenszyklus prüfen zu können, muss *SIF* es ermöglichen, gespeicherte Spreadsheets einzulesen. Für die erste Ausbaustufe ist dabei nur die Unterstützung der Microsoft-Excel Dateiformate .xls und .xlsx geplant, aber es soll einfach möglich sein, weitere Spreadsheet-Dateiformate zu unterstützen.

Erstellung und Konfiguration eines Inspektions-Auftrags

Zur Prüfung des eingelesenen Spreadsheets muss ein Inspektions-Auftrag angelegt werden können, um die Prüfung, die für den Prüfling durchgeführt werden soll, zu definieren. Der Umfang und die Konfiguration der Prüfung soll dabei dem geprüften Spreadsheet und dessen Nutzungskontext angepasst werden können.

Inventarisierung: Untersuchung auf Beschaffenheit und Struktur

Um einen ersten Eindruck des zu prüfenden Spreadsheets zu gewinnen, muss es möglich sein eine Inventurliste mit allen Bestandteilen des Spreadsheets zu erstellen. So sollen quantitative Merkmale des Spreadsheets, wie etwa die Anzahl an Zellen mit numerischen Inhalten oder die Anzahl an Formeln, erfasst werden können. Zudem soll auch die Struktur des Spreadsheets, wie zum Beispiel die Abhängigkeiten zwischen den Zellen, erkannt werden können. Da auch Spezial-Elemente erkannt werden sollen, die nicht über spezielle Sprachkonstrukte im Spreadsheet ausgedrückt werden können, soll es möglich sein, diese über eine Schnittstelle zu definieren und in Spreadsheets zu erkennen.

Durchführung von Inspektions-Aufträgen

Zur Prüfung von Spreadsheets muss es möglich sein, dass die erstellten und konfigurierten Inspektions-Aufträge automatisch durchgeführt werden. Dabei sollen die Einstellungen aus der Konfiguration des Inspektionsauftrags und die gewonnen Informationen aus der Inventarisierung verwendet werden, um das Spreadsheet auf die Einhaltung der Vorgaben der ausgewählten Richtlinie zu überprüfen. Gefundene Verstöße sollen entsprechend der zugehörigen Vorschrift klassifiziert, bewertet und gruppiert ausgeben werden.

Analyse der Ergebnisse einer Inspektion

Die Ergebnisse einer Inspektion sollen mit Hilfe von *SIF* analysiert werden können. Zu diesem Zweck müssen die Verstöße angeben, welches Spreadsheet-Element aus welchen Gründen den Verstoß verursacht hat, und wie der Verstoß zu klassifizieren und zu bewerten ist. Zum Abschluss der Analyse soll die Möglichkeit bestehen, die Ergebnisse der Inspektion in einem geeigneten Textformat zu exportieren.

Prüfung von drei ausgewählten Vorschriften

Um die Fähigkeiten der Inspektionswerkstatt zu demonstrieren, sollen die Prüfungen für die drei ausgewählte Vorschriften *Konstanten in Formeln*, *Leserichtung* und *Formelkomplexität* umgesetzt werden. Die drei Vorschriften sollen dabei wie folgt umgesetzt werden:

Konstanten in Formeln Die Vorschrift *Konstanten in Formeln* soll überprüfen, ob konstante Werte in Formeln verwendet werden. Als konstante Werte werden dabei alle Operanden angesehen, bei denen es sich nicht um eine Referenz oder das Ergebnis einer Funktion handelt. Dabei ist es nicht von Bedeutung, um welchen Typ von Konstante es sich dabei handelt, wie beispielsweise eine numerische oder textuelle Konstante. Pro Formel, die mindestens eine Konstante enthält, soll dabei ein Verstoß gemeldet werden. Verstöße von Formeln, die das selbe Formelmuster enthalten, sollen zu einer Gruppe zusammengefasst werden.

Leserichtung Die Vorschrift *Leserichtung* soll prüfen, ob Referenzen nur auf solche Objekte verweisen, die sich links und oberhalb der Zelle befinden, die die Referenz verwendet. Als links von einer Zelle gelten dabei alle Zellen im selben Worksheet, die einen niedrigeren Spaltenindex aufweisen und die Zellen aller Worksheets, die sich vor dem Worksheet der entsprechenden Zelle befinden. Als oberhalb von einer Zelle gelten dabei alle Zellen im selben Worksheet, die einen niedrigeren Reihenindex aufweisen. Pro Referenz, die die Leserichtung nicht einhält, soll dabei ein Verstoß gemeldet werden. Verstöße der selben Formel, sollen zu einer Gruppe zusammengefasst werden.

Formelkomplexität Die Vorschrift *Formelkomplexität* soll überprüfen, ob alle Formeln eines Spreadsheets eine gewisse Komplexität, gemessen an deren Verschachtelungstiefe und deren Anzahl an Operationen, also der Summe von Funktionen und Operatoren, nicht überschreitet. Eine Formel gilt in dieser Arbeit als komplex, wenn sie mehr als fünf Operationen

enthält oder eine Verschachtelungstiefe größer zwei aufweist. Diese Definition wurde dabei jedoch willkürlich gewählt. Pro komplexer Formel soll dabei ein Verstoß gemeldet werden, eine Gruppierung der Verstöße ist dabei nicht vorgesehen.

Umsetzung eines prototypischen Prüfwerkzeugs

Um die Prüfung der drei umzusetzenden Vorschriften einfach zu ermöglichen, soll ein prototypisches Prüfwerkzeug entwickelt werden. Dieses Prüfwerkzeug soll es ermöglichen, dass Spreadsheets über eine grafische Oberfläche mittels *SIF* auf die Einhaltung der drei Vorschriften *Konstanten in Formeln*, *Leserichtung* und *Formelkomplexität* überprüft werden und die Ergebnisse der Prüfungen exportiert werden können.

Erstellung neuer Richtlinien

Damit individuelle Richtlinien mit *SIF* überprüft werden können, soll es ermöglicht werden, dass eigenen Richtlinien aus den angebotenen Vorschriften zusammengestellt werden. Außerdem soll es ermöglicht werden, dass weitere Vorschriften realisiert und durch die Inspektions-Werkstatt überprüft werden können.

7.2 Nichtfunktionale Anforderungen

Unabhängigkeit von verwendetem Betriebssystem und verwendetem Spreadsheet-System

Um die Nutzung von *SIF* einem möglichst breiten Personenkreis zu ermöglichen, soll sich *SIF* nicht auf die Prüfung von Spreadsheets eines bestimmten Spreadsheet-Systems beschränken. Daher soll *SIF* nicht als Erweiterung oder Plug-In einer bestehenden Spreadsheet-Software entwickelt werden. Stattdessen soll *SIF* so konzipiert und entwickelt werden, dass es als plattformunabhängige, abgeschlossene Komponente verwendet werden kann, um Prüfwerkzeuge für Spreadsheets zu realisieren.

Erweiterbarkeit um dynamischen Prüfungsmöglichkeiten

SIF soll in der ersten Ausbaustufe nur die technische Grundlage bieten, um statische Prüfungen von Spreadsheets durchzuführen. Da statische Prüfungen jedoch durch ihre Beschaffenheit nicht dazu geeignet sind, alle Fehlerarten zu erkennen, soll die Architektur von *SIF* es ermöglichen, dass dynamische Prüfungen in das bestehende Framework integriert werden.

Erweiterbarkeit um graphische Darstellung von Spreadsheet und Verstößen

Zusätzlich zur Erweiterbarkeit um dynamische Prüfungen, ist auch die bessere Unterstützung von manuellen Inspektion durch *SIF* vorgesehen. Daher soll die Visualisierung des Spreadsheets und der gefundenen Verstöße in einer weiteren Ausbaustufe ermöglicht werden.

Kapitel 8

Umsetzung

Basierend auf dem in Kapitel 6 vorgestellten Konzept einer Spreadsheet-Inspektions-Werkstatt wird in diesem Kapitel nun beschrieben, wie die in Kapitel 7 gestellten Anforderungen an *SIF* umgesetzt worden sind.

8.1 Vorgehen

Aufgrund dessen, dass es sich bei der Entwicklung von *SIF* um ein Forschungsprojekt handelt, habe ich mich für die Umsetzung von *SIF* für eine evolutionäre Vorgehensweise mit zwei Entwicklungszyklen entschieden. Den Basis-Zyklus habe ich dabei mit zwei Dritteln der verfügbaren Zeit veranschlagt und die verbleibende Zeit habe ich für einen Erweiterungszyklus eingeplant. Innerhalb der Zyklen bin ich dabei nach dem Vorbild des Spiralmodells nach Boehm [LL07, S.177 ff.] vorgegangen, um die größten Risiken zuerst zu beseitigen.

Daher habe ich den Fokus für den ersten Zyklus auf das Übertragen von Spreadsheets in das interne Datenmodell, sowie auf die Konzeption der Infrastruktur für die Fehlererkennung und -verwaltung gelegt. Mit der Umsetzung der drei vorgesehenen Prüfverfahren habe ich dann begonnen, sobald der Fortschritt bei der Entwicklung der Infrastruktur dies zuließ. Gleichzeitig habe ich das prototypische Prüfzentrum, das die wichtigsten Funktionen von *SIF* über eine rudimentäre graphische Oberfläche zugänglich macht, in Grundzügen implementiert, um die Nutzung von *SIF* zu vereinfachen.

Der zweite Zyklus des Projekts sollte dann, je nach Fortschritt und den gewonnen Erkenntnissen aus dem ersten Zyklus, dazu verwendet werden, um die Implementierung der Prüfverfahren abzuschließen und dann das Framework und das prototypische Prüfzentrum zu verbessern und zu erweitern. Da die Umsetzung der Prüfverfahren am Ende des Basis-Zyklus noch nicht so weit vorangeschritten war, wie erhofft, habe ich beschlossen den gesamten Erweiterungszyklus für die Umsetzung von *SIF* zu verwenden und das prototypische Prüfzentrum in seiner rudimentären Fassung zu belassen.

8.2 Design

SIF besteht aus den Komponenten **`FrontOffice`**, **`TechnicalDepartment`**, **`IO`** und **`Model`**. Durch diese Aufteilung werden die Verwaltung der Inspektionsaufträge und der Inspektions-Werkstatt, die Durchführung der Inspektionsaufträge, sowie der Zugriff auf Spreadsheets voneinander getrennt.

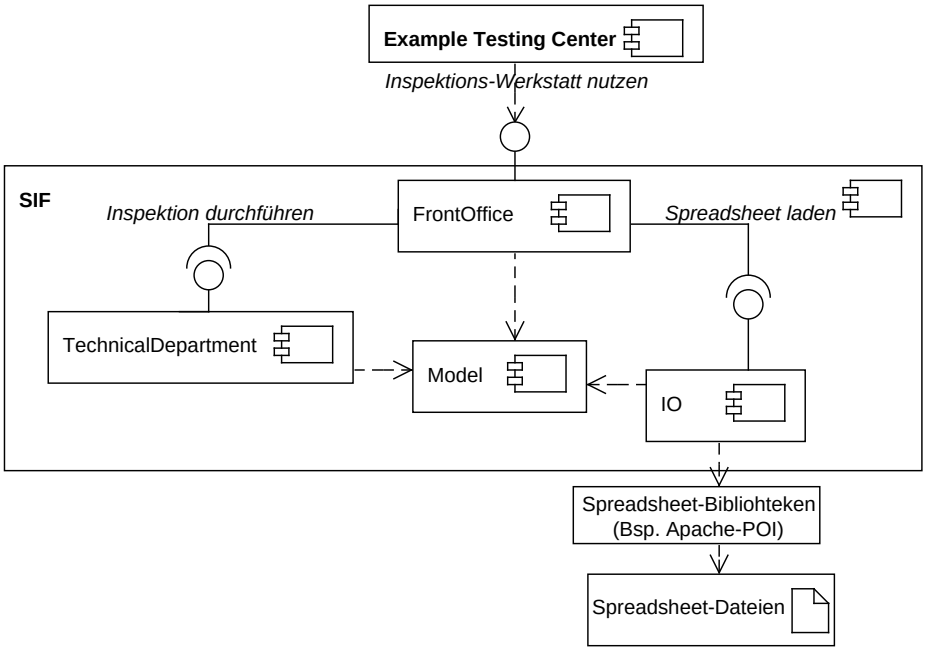

Abbildung 8.1: Architektur von *SIF*

Die einzelnen Komponenten haben dabei folgenden Aufgaben:

- Das **`FrontOffice`** ist der zentrale Zugriffspunkt, um Funktionen von *SIF* zu nutzen. Die Komponente bietet zu diesem Zweck eine Schnittstelle an, um neue Inspektionen in Auftrag zu geben, deren Ablauf zu steuern und die angebotenen Prüfungen der Inspektions-Werkstatt zu verwalten.
- Das **`Model`** bietet Datenstrukturen an, um Spreadsheets zu repräsentieren, Richtlinien für Spreadsheets zu definieren, Inspektions-Aufträge und deren Konfiguration zu erstellen sowie die Ergebnisse von Inspektionen festzuhalten.
- Die **`IO`** ist dafür zuständig, dass Spreadsheets aus Spreadsheet-Dateien geladen und durch Klassen aus dem **`Model`** repräsentiert werden können. Zum Einlesen der Spreadsheet-

Dateien in unterschiedlichen Formaten werden dabei externe Bibliotheken, wie beispielsweise Apache POI-SS [1], verwendet.

- Die Durchführung der Inspektions-Aufträge, die über das **FrontOffice** erstellt und konfiguriert werden, erfolgt dann im **TechnicalDepartment**, das auch die dazu notwendigen Prüfmittel verwaltet.

Die folgenden Abschnitte beschreiben die Funktionen von *SIF* und die jeweils beteiligten Komponenten. Auf eine detaillierte Beschreibung des Designs, die jede Klasse, jede Methode und jedes Attribut umfasst, wurde bewusst verzichtet. Stattdessen wird der Schwerpunkt auf die Zusammenhänge der einzelnen Komponenten und die getroffenen Entscheidungen gelegt.

8.2.1 Repräsentation von Spreadsheets

Um die Unabhängigkeit von *SIF* bezüglich des verwendeten Spreadsheet-Formats zu erreichen, wird für die Repräsentation von Spreadsheets ein eigenes Datenmodell verwendet. Dieses Datenmodell für Spreadsheets gliedert sich dabei in zwei Teile: ein Basis-Modell, repräsentiert durch eine Instanz der Klasse `Spreadsheet` und ein erweitertes Modell – das Spreadsheet-Inventar, repräsentiert durch eine Instanz der Klasse `SpreadsheetInventory`. Das Basis-Modell enthält dabei alle Informationen, die direkt über Sprachkonstrukte von gängigen Spreadsheet-Sprachen ausgedrückt werden können. Es stellt also die Repräsentation eines aus Basis-Elementen bestehenden Spreadsheets dar. Das Spreadsheet-Inventar beinhaltet das Basis-Modell und erweitert dies um Listen mit den Vorkommnissen aller Elemente in dem zu repräsentierenden Spreadsheet. Das Spreadsheet-Inventar beschränkt sich dabei jedoch nicht nur auf Vorkommnisse von Basis-Elementen, sondern kann auch die Erfassung aller Instanzen von benutzerdefinierten Spezial-Elementen beinhalten.

Einlesen von Spreadsheets – Erstellung des Basis-Modells

Das Basis-Modell soll ein Spreadsheet und dessen Bestandteile so darstellen, wie dies in einer Spreadsheet-Datei gespeichert ist. Die Repräsentation eines Spreadsheets erfolgt daher durch eine Instanz der Klasse `Spreadsheet` und die Repräsentation von Basis-Elementen erfolgt als Unterklassen von `BasicAbstractElement`. Zur Erstellung des Basis-Modells muss das zu prüfende Spreadsheet aus einer Spreadsheet-Datei eingelesen und in interne Datenstrukturen umgewandelt werden. Zu diesem Zweck bietet die zuständige **IO**-Komponente über die zentrale Klasse `DataFacade` eine Schnittstelle an, die für eine gegebene Spreadsheet-Datei eine entsprechende Instanz von `Spreadsheet` erstellt. Das Einlesen und Umwandeln der Informationen übernimmt dabei die für das Format der gegebenen Spreadsheet-Datei zuständige `ISpreadsheetIO`. So erfolgt beispielsweise das Einlesen und Umwandeln von Spreadsheet-Dateien in den Excel-Formaten *.xls* oder *.xlsx* durch die `POISpreadsheetIO_HSSF` bzw. die `POISpreadsheetIO_XSSF`.

[1] http://poi.apache.org/spreadsheet/index.html

Diese verwenden zum Einlesen die externe Bibliothek Apache-POI-SS[2] und wandeln die eingelesenen Basis-Elemente in Datenstrukturen von *SIF* um. Die Unterstützung von weiteren Spreadsheet-Dateiformaten ist durch diese Aufgabenverteilung leicht möglich, indem weitere Implementierungen der `ISpreadsheetIO` realisiert werden.

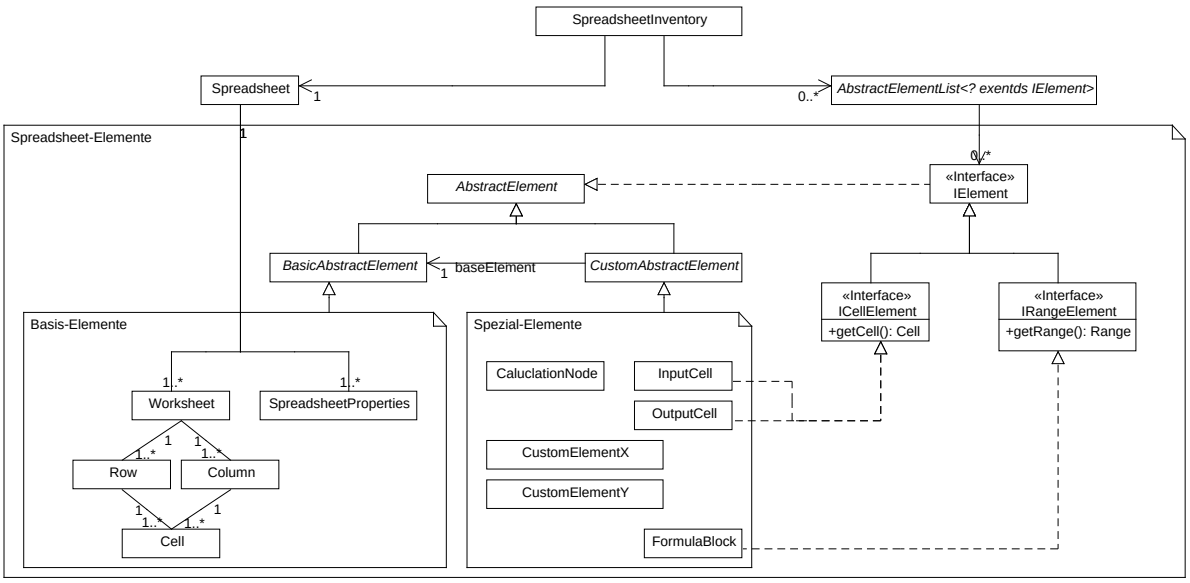

Abbildung 8.2: Darstellung des Spreadsheet-Inventars sowie einiger Basis- und Spezial-Elemente

Inventarisieren von Spreadsheets – Erstellung des erweiterten Modells

Wie in Abschnitt 6.2 beschrieben, sind die Ausdrückmöglichkeiten bestehender Spreadsheet-Sprachen begrenzt und daher wird in dieser Arbeit zwischen Basis- und Spezial-Elementen von Spreadsheets unterschieden. Das erweiterte Modell soll dazu dienen, zusätzlich zu den Basis-Elementen, auch Spezial-Elemente von Spreadsheets zu erfassen und diese für die Prüfung zur Verfügung zu stellen. Zu diesem Zweck können Spezial-Elemente als Unterklassen von `CustomAbstractElement` definiert werden. Dazu müssen diese die Instanz eines entsprechenden Basis-Elements angeben, auf dem sie basieren.

Durch die Implementierung von Unterklassen des Interfaces `IElement` können ähnliche Element-Klassen, beispielsweise die beiden Spezialisierungen einer Zelle `InputCell` und `OutputCell`, gruppiert werden. So kann die Funktionaliät von `Cell` auch bei den Klassen `InputCell` und `OutputCell` genutzt werden, wenn diese das Interface `ICellElement` implementieren und auf die Zelle des Basis-Modells verweisen, auf der sie basieren.

[2] http://poi.apache.org/spreadsheet/index.html

Die Erstellung des erweiterten Modells erfolgt während des Inventarisierung-Schrittes mit Hilfe von `ElementScannern`. So muss für jede Elementklasse, deren Instanzen ins Spreadsheet-Inventar aufgenommen werden sollen, ein `Element-Scanner` definiert und über den `TechnicalManger` beim `ScanningManager` registriert werden. Ein `ElementScanner` erkennt unter Verwendung des Basis-Modells und bereits gescannter Elemente alle Vorkommnisse dieser Elementklasse und fügt diese gesammelt als `AbstractElementList` dem Spreadsheet-Inventar hinzu. Eine `AbstractElementList` stellt eine abstrakten Container für eine bestimmte Elementklasse bereit, der alle Instanzen dieser Klasse aufnehmen kann. *SIF* stellt mit der `SimpleElementList` eine einfache Implementierung dieses Containers bereit, erlaubt jedoch auch spezifische Implementierung dieses Containers für bestimmte Elementklassen.

Das Inventarisieren eines Spreadsheets erfolgt im **`TechnicalDepartment`** durch den `ScanningManager`, welcher alle registrierten `Element-Scanner` ausführt. Da manche Scanner auf den Ergebnissen anderer Scanner aufbauen, ist die Durchführungsreihenfolge entscheidend. Daher können `Element-Scanner` einen Prioritätswert angeben, der die Ausführungsreihenfolge beeinflusst. Für eine korrekte Konfiguration dieser Werte sind jedoch die Entwickler von neuen Scannern verantwortlich, eine Überprüfung durch den `ScanningManager` erfolgt nicht.

Das `SpreadsheetInventory` enthält also nach der Inventarisierung eine Repräsentation des Spreadsheets, wie es in der Spreadsheet-Datei gespeichert ist, und außerdem die Vorkommnisse aller Spreadsheet-Elemente, die bei der Inventarisierung des Spreadsheets entdeckt wurden.

8.2.2 Inspektion von Spreadsheets mit *SIF*

Das **`FrontOffice`** ist die Schnittstelle um Funktionen von *SIF* zu nutzen. Die zentrale Zugriffspunkt für die Nutzer der Inspektions-Werkstatt ist dabei die Klasse `FrontDesk`, die zu diesem Zweck die Enwurfmuster Singleton und Fassade implementiert [GHJV95]. Die Klasse `FrontDesk` soll dabei als Rezeption für die Inspektions-Werkstatt dienen und die zentrale Anlaufstelle für deren Anwender sein. So können über diese Schnittstelle neue Inspektionen in Auftrag gegeben und konfiguriert werden. Und auch die Steuerung der Durchführung und die Abholung der Ergebnisse erfolgt über den `FrontDesk`. Die Abwicklung der beantragten Vorgänge geschieht dabei jedoch nicht direkt im `FrontDesk`, sondern wird an die beiden anderen Klassen des **`FrontOffice`** delegiert. So ist der `InspectionManager` für die Erstellung der Inspektions-Aufträge zuständig und koordiniert deren Durchführung. Der `Policymanager` ist für die Koordination der angebotenen Prüfungen mit den vorhandenen technischen Mittel zuständig und übernimmt daher auch die Konfiguration von Inspektions-Aufträgen.

Erstellung und Konfiguration von Inspektionsaufträgen

Inspektionsaufträge werden durch die Klasse `InspectionRequest` repräsentiert. Zum Erstellen eines neuen Inspektionsauftrags muss der Benutzer dem `FrontDesk` die Spreadsheet-Datei und einen Namen für die Inspektion übergeben. Der `InspectionManager` legt daraufhin den

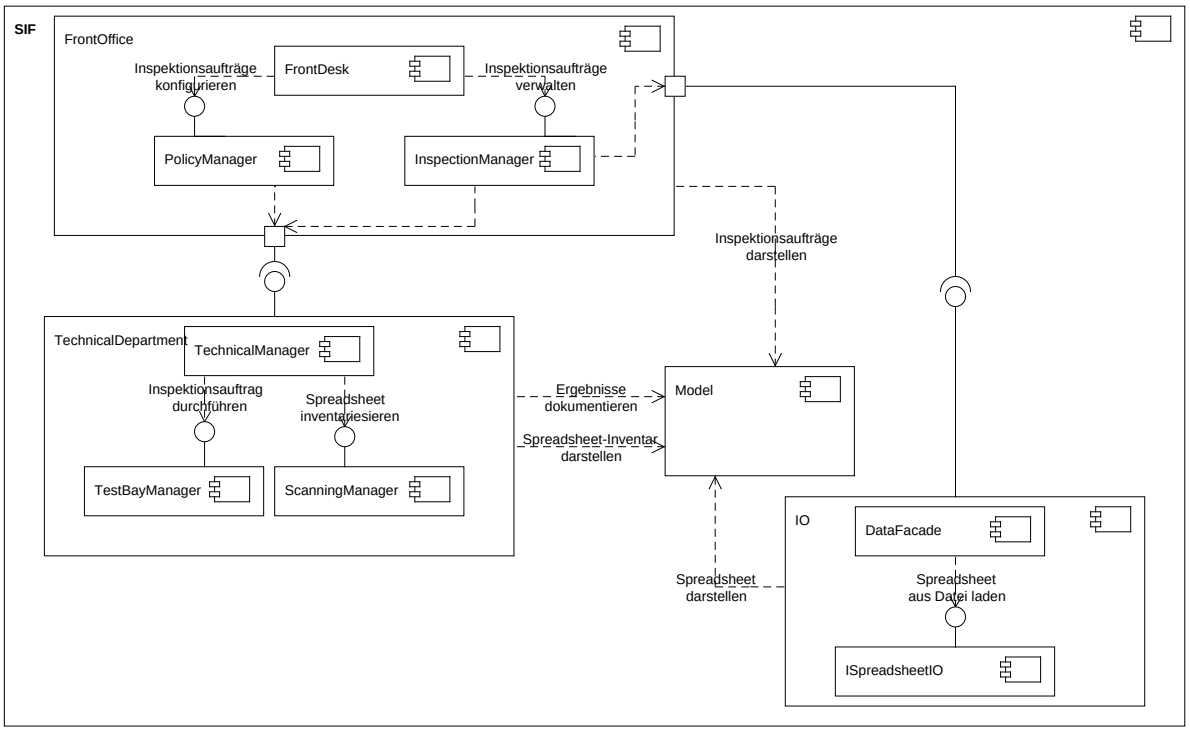

Abbildung 8.3: Die Komponenten von *SIF* im Detail

initialen Inspektionsauftrag an, lässt über die **`IO`** das Basis-Modell für das Spreadsheet erstellen und macht dies als ersten Bestandteil des Spreadsheet-Inventars für den Inspektionsauftrag verfügbar. Anschließend erhält der Aufrufer die Instanz des `InspectionRequest` und kann auf das Basis-Modell zugreifen. Als nächstes erfolgt die Inventarisierung des Spreadsheets, wie oben beschrieben, die ebenfalls über den `FrontDesk` gestartet und im **`TechnicalDepartment`** vom `ScanningManager` durchgeführt wird. Nach der Inventarisierung ist das gesamte Spreadsheet-Inventar über den `InspectionRequest` abrufbar, und ist zur Analyse durch den Nutzer verfügbar.

Zur Konfiguration eines Inspektionsauftrags kann der Benutzer über den `FrontDesk` die Richtlinien erfragen, die beim `PolicyManager` registriert wurden und damit von der Inspektionswerkstatt aktuell angeboten werden. Eine Richtlinie wird dabei durch die Klasse `Policy` dargestellt und die Vorschriften der Richtlinie über eine Liste von `AbstractPolicyRules`. Vorschriften, die über einen monolithischen Prüfstand geprüft werden, werden dabei als `MonotlithicPolicyRule` repräsentiert. Vorschriften, die mit Prüfständen geprüft werden, die aus bestehenden Prüfwerkzeugen zusammengesetzt werden, werden als `CompositePolicyRule` dargestellt. Die Parameter einer `AbstractPolicyRule` werden in dieser deklariert und über die Annotation `ConfigurableParameter` kenntlich gemacht. Eine Konfiguration einer `AbstractPolicyRule` wird bei deren Registrierung im `PolicyManager` erstellt und ist als `PolicyRuleConfiguration` der `AbstractPolicyRule` abrufbar. Die `PolicyRuleConfiguration` erlaubt es die Werte für die

Parameter der Vorschrift, repräsentiert als `ParameterConfigurations`, zu setzen. Nachdem eine `Policy` ausgewählt und deren `AbstractPolicyRules` eventuell konfiguriert wurden, kann die Policy für den Inspektionsauftrag,der nun zur Durchführung bereit ist, gesetzt werden,

Durchführung von Inspektionsaufträgen

Die Durchführung eines Inspektionsauftrags wird wie gewohnt über den `FrontDesk` gestartet. Die Anfrage zur Durchführung wird über den `InspectionManager` und den `TechnicalManager` an den `TestBayManager` weitergeleitet. Dieser ist nun dafür zuständig ein Prüffeld, dargestellt durch die Klasse `TestBay`, aufzubauen, das der konfigurierten `Policy` aus dem `InspectionRequest` entspricht. Der `TestBayManager` kennt die Zuordnung von Vorschriften zu monolithischen Prüfständen und kann für Vorschriften, die nicht über monolithische Prüfstände geprüft werden, den Prüfstand aus den vorhandenen Prüfwerkzeugen zusammensetzen. Für jede Vorschrift wird der entsprechende Prüfstand mit den getätigten Einstellungen konfiguriert und zum Prüffeld hinzugefügt. Die Klasse `TestBay` hält den Inspektionsauftrag und das zugehörige Spreadsheet-Inventar, das für alle Prüfstände zugänglich ist. Sie ist außerdem dafür zuständig die Prüfstände der Reihe nach abzufahren und die gefundenen Verstöße zu sammeln.

Verstöße werden durch Implementierungen der Schnittstelle `IViolation` realisiert. Dabei existieren die Spezialisierungen `ISingleViolation` und `IGroupViolation` um einzelne Verstöße und Gruppen von Verstößen darstellen zu können. Jeder Verstoß enthält dabei die Vorschrift die verletzt wurde, das verursachende Element, sowie eine Beschreibung des Verstoßes. Außerdem wird jeder Verstoß entsprechend der geprüften Vorschrift klassifiziert und unter Berücksichtigung der eingestellten Gewichtung hinsichtlich seiner Schwere bewertet. Eine Verstoß-Gruppe kann dabei beliebig viele Einzelverstöße enthalten und ein Element angeben, das die verursachenden Elemente seiner Mitglieder enthält.

Gesammelt werden die gefundenen Verstöße eines Prüfstands in einer `ViolationList`, die auch die Erstellung von Verstoß-Gruppen übernimmt. So kann einer `ViolationList` ein `ViolationGroupor`-Objekt übergeben werden, der erkennt welche Verstöße zu einer Gruppe zusammengefasst werden müssen und diese Gruppe entsprechend erstellt. Prüfstände geben eine solche `ViolationList` als Ergebnis aus, welche vom Prüffeld in der Container-Klasse `Findings` gesammelt werden. Nachdem alle Prüfstände ihre Prüfungen durchgeführt haben, wird das `Findings`-Objekt dem Inspektionsauftrag hinzugefügt. Der Nutzer der Inspektions-Werkstatt kann nun alle gefundenen Verstöße als Ergebnisse des Inspektionsauftrag abrufen und diese analysieren.

Inspektionsbericht erstellen

Zum Abschluss des Inspektionsauftrags kann für diesen ein Bericht erstellt werden, der die wichtigsten Information zu der Konfiguration des Auftrags zusammenfasst, sowie die Details zu allen gefunden Verstößen enthält. Dazu muss über den `FrontDesk` der Pfad zu dem Ort angegeben werden, an dem der Bericht erstellt werden soll. Daraufhin wird von der `IO`-Komponente eine .hmtl-Datei unter dem Namen des Inspektionsauftrags erstellt, die den Bericht enthält.

8.2.3 Verwaltung der Inspektionswerkstatt

Zur Erweiterung des Prüfungsangebotes können mit *SIF* neue Richtlinien aus bestehenden Vorschriften zusammengestellt werden und über die Implementierung von neuen Prüfständen neue Vorschriften angeboten werden. In diesem Abschnitt wird beschrieben, wie dies technisch abläuft.

Erstellung neuer Richtlinien

Zur Erstellung von neuen Richtlinien muss eine neue Instanz der Klasse `Policy` erstellt werden. Dabei muss für das `Policy`-Objekt ein Name, eine Beschreibung und der Name des Autors gesetzt werden. Über den `Frontdesk` können dann alle registrierten Vorschriften als `AbstractPolicyRule`s abgerufen werden. Diese enthalten bereits eine initiale Konfiguration mit den Werten, die der Ersteller dieser Vorschrift als Standard angegeben hat. Der Ersteller einer neuen Richtlinie wählt eine beliebige Anzahl dieser Vorschriften aus, kann diese nach Bedarf vorkonfigurieren, und fügt sie der neu erstellen `Policy` hinzu. Die neu erstellte `Policy` kann dann über den `FrontDesk` beim `PolicyManager` registriert werden und ist nun für die Prüfung von Spreadsheets verfügbar.

Erstellung neuer Vorschriften

Zusätzlich zur Erstellung neuer Richtlinien können auch neue Vorschriften erstellt und zur Auswahl für Richtlinien verfügbar gemacht werden. Es wird dabei zwischen solchen Vorschriften unterschieden, die von monolithischen Prüfständen überprüft werden, und solchen, die mit zusammengesetzten Prüfständen überprüft werden. Technisch wird das durch die Klassen `MonolithicPolicyRule` und `CompositePolicyRule` gelöst, die beide von der Klasse `AbstractPolicyRule` ableiten. Um das Prüfangebot der Inspektionswerkstatt um eine weitere Prüfung zu erweitern, muss eine Spezialisierung dieser Klassen implementiert werden, sowie geeignete Prüfmittel realisiert werden, die die Einhaltung der dadurch definierten Vorschrift überprüfen.

`MonolithicPolicyRule`s benötigten dabei als Prüfmittel einen monolithischen Prüfstand, umgesetzt als Spezialisierung von `MonolithicTestFacility`, und zur Prüfung von `CompositePolicyRules` werden ein oder mehrere Prüfwerkzeuge, umgesetzt als Spezialisierungen von `AbstractTestInstrument`, benötigt.

Zur Erstellung einer neuen Vorschrift, unabhängig von deren Typ, sollte für die Spezialisierung von `AbstractPolicyRule` ein Name, eine Beschreibung des Hintergrunds, eine Anleitung zur möglichen Behebung von Verstößen gegen diese Vorschrift und der Name des Autors angegeben werden. Außerdem ist es möglich der Vorschrift eine Kategorie zuzuweisen, der die entdeckten Verstöße zugewiesen werden. Zusätzlich besteht die Möglichkeit einen Gewichtungsfaktor anzugeben der für die Bewertung der Schwere von Verstößen verwendet wird.

8.3 Implementierung

In diesem Abschnitt soll beschrieben werden welche Funktionalität bisher umgesetzt wurde, und was aus Zeitgründen noch nicht implementiert werden konnte.

8.3.1 Model

Das Basismodell wurde bisher nur so weit umgesetzt, wie dies für die Umsetzung der geplanten Prüfung notwendig war. Daher sind noch nicht alle Basis-Elemente eines Spreadsheets als Datenstrukturen modelliert, jedoch wurden bereits einige Spezial-Elemente, beispielsweise `InputCell` und `OutputCell` modelliert und über `ElementScanner` erkannt. Außerdem werden bisher nur Spreadsheets in den Dateiformaten .xls und .xlsx unterstützt, da nur eine `ISpreadsheetIO` für das Einlesen dieser Formate implementiert wurde.

8.3.2 Prüfungen

Für die erste Ausbaustufe wurden Prüfungen für die drei Vorschriften *Konstanten in Formeln*, *Leserichtung* und *Formelkomplexität* implementiert. Dabei wurden alle Prüfungen mittels monolithischer Prüfstände als Spezialisierungen von `MonolithicTestFacility` umgesetzt.

Konstanten in Formeln Zur Prüfung dieser Vorschrift werden alle Vorkommnisse des Basis-Elements `Formula`, die über einen entsprechenden Scanner dem Spreadsheet-Inventar hinzugefügt wurden, untersucht. Instanzen von `Formula` enthalten dabei für jeden Bestandteil der Formel ein eigenes Objekt der Schnittstelle `ITokenElement`. So werden konstante Werte als `ScalarConstant` dargestellt und es kann für jede Formel überprüft werden, ob es sich bei einem der enthalten `ITokenElemente` um eine Instanz von `ScalarConstant` handelt. Da jedes `ScalarConstant`-Objekt, den Typ und den Wert der Konstante speichert, ist es möglich bestimmte Werte oder Typen bei der Prüfung zu ignorieren. So kann über den konfigurierbaren Parameter `ignoredConstants` in der zugehörigen Repräsentation der Vorschrift, *NoConstantsInFormulasPolicyRule*, eine Liste von Werten gesetzt werden, die bei der Prüfung durch die *NoConstantsInFormulasTestFacilitiy* ingoriert werden sollen. Zudem kann über `ignoredFunctions` eine Liste mit den Namen der Funktionen angegeben werden, in denen die Verwendung beliebiger Konstanten erlaubt sein soll.

Leserichtung Die Repräsentation dieser Vorschrift erfolgt dabei als `ReadingDirectionPolicyRule`, deren Prüfung in der `ReadingDirectionTestFacility` durchgeführt wird. Die Prüfung wird wiederum auf der Basis des Spreadsheet-Inventars durchgeführt. So werden alle registrierten Referenzen, die als `AbstractReference` ins Inventar aufgenommen wurden, der Reihe nach untersucht. Jede `AbstractReference` enthält dabei die referenzierende Zelle und das referenzierte Element. Von beiden Elementen kann dann ihre Position im Spreadsheet erfragt werden, um anschließend horizontal und vertikal verglichen

zu werden. Dabei ist über zwei Parameter der `ReadingDirectionPolicyRule` separat einstellbar, ob die Leserichtung von links nach rechts und von oben nach unten eingehalten werden muss.

Formelkomplexität Analog zu den anderen Vorschriften erfolgt die Repräsentation der Vorschrift durch die Klasse `FormulaComplexityPolicyRule` und die Prüfung wird mit `FormulaComplexityTestFacility` durchgeführt Über die Repräsentation der Vorschrift kann dabei konfiguriert werden, wie viele Operationen eine Formel maximal enthalten darf und wie hoch die maximale Schachtelungstiefe sein soll. Als Operation werden dabei alle Funktionen und Operanden gezählt und die maximale Schachtelungstiefe, ergibt sich aus der maximalen Anzahl an Funktionen, die einander kaskadierend aufrufen. Geprüft wird dabei wiederum anhand der Liste von `Formula`-Objekten, wobei für jedes Formula-Objekt, die Anzahl an Operationen gezählt wird und über eine Methode der Klasse, die maximale Schachtelungstiefe berechnet wird. Die Werte werden dann mit den Vorgaben der konfigurierten Vorschrift verglichen und bei Abweichungen wird ein entsprechendes Verstoß-Objekt mit der Angabe des Formel Objekts und den gefundenen Werten erstellt.

Kapitel 9

Evaluation

Im vorherigen Kapitel wurde die Umsetzung des Konzepts einer Inspektions-Werkstatt und die Realisierung eines prototypischen Prüfzentrums für die Prüfung von Spreadsheets beschrieben. In diesem Kapitel soll nun beschrieben werden, wie das mit *SIF* entwickelte prototypische Prüfwerkzeug *Example Testing Center* (*ETC*) evaluiert wurde, um dessen Eignung für die Prüfung von Vorschriften zu untersuchen. Dazu wurde *ETC* für die Überprüfung von Spreadsheets bezüglich drei Vorschriften verwendet und anschließend wurden die Ergebnisse mit denen von bestehenden kommerziellen statischen Analysewerkzeugen verglichen.

9.1 Rahmenbedingungen

Im diesem Abschnitt werden die Rahmenbedingungen beschrieben, innerhalb deren die Evaluation durchgeführt wurde. Dazu werden die Eigenschaften und Funktionen der untersuchten Prüfwerkzeuge vorgestellt und miteinander verglichen. Anschließend werden die Spreadsheets kurz vorgestellt, die für die Evaluation verwendet wurden. Zuletzt wird erläutert, wie die Evaluation dabei im Detail ablief.

9.1.1 Prüfwerkzeuge

Die Zahl der kommerziellen Produkte, die versprechen das Qualitäts-Problem von Spreadsheets zu lindern oder sogar zu beseitigen, nahm in den vergangenen Jahren stetig zu. Bei der Mehrheit dieser Produkte handelt es sich dabei um Audit-Werkzeuge, die die Prüfung von Spreadsheets unterstützen (Siehe Kapitel 5). Eine Auswahl an Prüfwerkzeugen, die wie die erste Ausbaustufe von *SIF* statische Prüfungen von Spreadsheets anbieten, wurde im Rahmen dieser Evaluation mit *ETC* verglichen.

Auswahl

Das Hauptkriterium für die Auswahl der untersuchten Prüfwerkzeuge war dabei deren Verfügbarkeit. Das heißt, es wurden nur solche Prüfwerkzeuge in Betracht gezogen für die zumindest eine kostenlose Probeversion verfügbar war, die ohne Registrierung von der Seite des Herstellers bezogen werden konnte. Außerdem wurde bei der Auswahl berücksichtigt, ob die Prüfwerkzeuge vergleichbare Vorschriften zur Überprüfung anbieten, wie sie in dieser Arbeit umgesetzt wurden. Die Wahl der Prüfwerkzeuge fiel zum einen auf das Prüfwerkzeug *Spreadsheet Professional*[1], welches auch in [PBL09] verwendet wird, und zum anderen auf das Prüfwerkzeuge *Rainbow Analyst*[2]. Dabei wurden jeweils die verfügbaren Probeversionen[3] verwendet, die einen vollen Funktionsumfang bieten, aber nur eine eingeschränkte Nutzungsdauer erlauben.

Übersicht

Die Eigenschaften und Funktionen dieser Werkzeuge werden in Abbildung 9.1 präsentiert und mit denen von *ETC* gegenübergestellt. Einträge, die mit einem Stern (*) versehen sind, deuten dabei an, dass diese Funktion oder Eigenschaft von *ETC* zwar noch nicht oder nicht ausreichend erfüllt ist, deren Implementierung jedoch mit Hilfe von *SIF* geplant oder bereits umsetzbar ist.

Rainbow Analyst	Spreadsheet Professional	ETC	
			Eigenschaften
✓	✓	✗*	Visualisierungsunterstützung
✓	✗	✗*	Dynamische Prüfungen
✗	✗	✓	Open Source
✗	✗	✓	Standalone
✗	✗	✓	Erweiterbar
✗	✗	✓	Unterstützt mehrere Datei-Formate
			Funktionen
✓	✓	✗*	Spreadsheet Zusammenfassung
5	25	3*	Anzahl überprüfbarer Vorschriften
✓	✓	✓	Auswahl von Vorschriften
✗	✗	✓	Konfiguration von Vorschriften
✓	✓	✓	Gruppierung von Verstößen
✗	✗	✓	Gewichtung von Verstößen
✗	✗	✓	Detailansicht von Verstößen
✓	✓	✓	Report aller Verstöße

Abbildung 9.1: Eigenschaften und Funktionen der untersuchten Prüfwerkzeuge

[1] http://www.spreadsheetinnovations.com
[2] http://www.themodelanswer.com
[3] Das Werkezug Rainbow Analyst wurde in der Version 5.1 verwendet; das Werkzeug Spreadsheet Professional wird vom Hersteller ohne die Angabe einer Versionsnummer angeboten.

9.1.2 Spreadsheets

Zur Evaluation der untersuchten Werkzeuge wurde eine kleine Auswahl von zwölf unterschiedlichen Spreadsheets aus Real- und Laborumgebungen verwendet. Wie auch bei der Auswahl der untersuchten Werkzeuge, war die Verfügbarkeit der Spreadsheets hierbei ein Hauptkriterium. Gleichzeitig wurde jedoch versucht, Spreadsheets aus möglichst unterschiedlichen Bereichen auszuwählen, die in ihrer Größe und Komplexität dem Durchschnitt der in der Praxis verwendeten Spreadsheets [IR05] entsprechen.

Die Dateien *spreadsheet_sample_01* bis *spreadsheet_sample_05* stammen aus einem Experiment, das in der Abteilung Software Engineering am Institut für Softwaretechnologie der Universität Stuttgart durchgeführt wurde. Dabei mussten die Probanden ausgehend von einer textuellen Aufgabenstellung ein Spreadsheet erstellen, mit dem Pkws bezüglich unterschiedlich gewichtbarer Kriterien miteinander verglichen werden konnten. Bei *spreadsheet_sample_06* handelt es sich um ein frei im Internet verfügbares Spreadsheet zur Analyse von Residualeinkommen[4], das auf der Seite des CPA-Journals[5] zum Herunterladen angeboten wird. Die Datei *spreadsheet_sample_07* ist ein Spreadsheet aus der Praxis zum Vergleich von Reifengrößen, das der Abteilung Software Engineering der Universität Stuttgart zur Verfügung gestellt wurde. Und bei den Spreadsheets *spreadsheet_sample_08* bis *spreadsheet_sample_12* handelt es sich wiederum um frei im Internet verfügbare Spreadsheets, die auch Teil des EUSES Spreadsheet Corpus [IR05] sind.

Dabei wurde für die untersuchten Spreadsheets die Anzahl der Worksheets, Zellen und Formeln gemessen (Siehe Abbildung 9.2). Die verwendeten Spreadsheets sind außerdem auch auf dem beigefügten Datenträger (siehe Abschnitt A.1) zu finden.

spreadsheet_sample_	01	02	03	04	05	06	07	08	09	10	11	12	Ø	Median	Max	Min
Worksheets	3	3	3	5	3	5	1	4	1	4	4	2	3.2	3	5	1
Zellen	397	497	214	741	220	315	170	323	705	884	352	247	422.1	337.5	884	170
Formeln	166	126	27	241	30	74	50	71	80	125	25	40	92.3	72.5	241	25

Abbildung 9.2: Allgemeine Statistiken zu den Spreadsheets der Evaluation

9.1.3 Überprüfte Vorschriften

Für die Evaluation wurden die drei Vorschriften mit den ausgewählten Werkzeugen überprüft, die auch in dieser Arbeit als Prüfverfahren umgesetzt wurden (Siehe Absatz 7.1). Zusätzlich wurde die Vorschrift *Konstanten in Formeln* ebenfalls mittels einer nichtmechanischen Inspektion durch den Autor überprüft. Auf eine Überprüfung aller Spreadsheets auf die beiden anderen Vorschriften wurde aus Aufwandsgründen verzichtet. Alle untersuchten Prüfwerkzeuge bieten

[4] http://www.nysscpa.org/cpajournal/2001/0700/features/f074601.htm
[5] www.cpajournal.com/download/rimodel.xls

die ausgewählten Vorschriften an, jedoch in leichten Abwandlungen und ohne eine genaue Definition der Prüfkriterien. So überprüfen die beiden Prüfwerkzeuge Spreadsheet Professional und Rainbow Analyst Spreadsheets nur auf numerische Konstanten. Zudem meldet Rainbow Analyst Zeilen- und Reihenunterschiede bei Referenzen getrennt. Bezüglich der dritten Vorschrift *Formelkomplexität* gibt leider keines der beiden Werkzeuge an, welche Eigenschaften eine Formel besitzen muss, um nicht als komplexe Formel durch einen Verstoß gemeldet zu werden.

9.1.4 Ablauf

Es wurde vor allem eine quantitative Analyse der gefundenen Verstöße durchgeführt, eine detaillierte Untersuchung von einzelnen Verstößen fand dabei nicht statt. Zu diesem Zweck wurden die ausgewählten Spreadsheets mit den untersuchten Werkzeugen geprüft. Der Ablauf der Vorgänge soll in den folgenden Abschnitten kurz beschrieben werden.

Nichtmechanische Inspektion

Die Inspektion der Spreadsheets durch den Verfasser dieser Arbeit erfolgte am Rechner unter Verwendung der Spreadsheet-Software *OpenOffice.org Calc*. Die Prüfung der Spreadsheets erfolgte dabei nach der in Absatz 7.1 gegebenen Definition und wurde Zelle für Zelle durchgeführt. Da die überprüfte Vorschrift auf Zellen mit Formeln basiert, wurden nur Zellen mit Formelinhalten inspiziert. Daher wurde zur Unterstützung die Funktion *Value Highlighting* der Spreadsheet-Software aktiviert, die Zellen mit Formeln grün und Zellen mit numerischen Werten blau einfärbt. Die Zellen wurden dabei pro Reihe von links nach rechts auf Verstöße untersucht und die Befunde pro Worksheet festgehalten.

Prüfung durch die statischen Analysewerkzeuge

Die Evaluation der untersuchten Werkzeuge verlief jeweils auf ähnliche Art und Weise. Jedoch wurden für ETC zwei Evaluationsdurchgänge ausgeführt, da es als einziges der untersuchten Werkzeuge eine Konfiguration der Vorschriften erlaubt. Die Durchgänge beliefen sich auf die Aufgaben **Konstanten in Formeln** und *Formelkomplexität*.

Jedoch wurden für *ETC* zwei Evaluationsdurchgänge ausgeführt, da es als einziges der untersuchten Werkzeuge eine Konfiguration der Vorschriften erlaubt. Die Durchgänge beliefen sich auf die Aufgaben *Konstanten in Formeln* und *Formelkomplexität*. Ein Durchgang mit den Standardeinstellung von *ETC* und ein weiterer mit benutzerdefinierten Einstellungen. Die benutzerdefinierten Einstellungen wurden dabei durch den Autor wie folgt festgelegt: Für die Vorschrift *Konstanten in Formeln* wurden Vorkommnisse der beiden numerischen Konstanten 0 und 1 ignoriert und außerdem durften die Funktionen *INDEX* und *INDIRECT* beliebige Konstanten verwenden. Für die Vorschrift *Formelkomplexität* wurden die Anzahl an erlaubten Operationen auf acht und die erlaubte Verschachtelungstiefe auf drei angehoben. Die Vorschrift *Leserichtung* wurde unverändert verwendet. Ein Durchgang gestaltete sich dabei für ein Werkzeug wie folgt:

Vorbereitung Zunächst wurde das Werkzeug für die Durchführung der Evaluation vorbereitet. So wurden die beiden Werkzeuge Rainbow Analyst und Spreadsheet Professional zunächst nach den Anweisungen der Hersteller als Plugins für die Spreadsheet-Software *Microsoft Office Excel 2003* installiert. *ETC* hingegen musste nicht installiert werden, da es sich um eine alleinstehende Anwendung handelt, die auch ohen eine Installation ausgeführt werden kann. Jedoch musste eine Richtlinie erstellt werden, die die drei untersuchten Vorschriften enthält.

Konfiguration Im Anschluss an die Vorbereitung wurden alle untersuchten Spreadsheets der Reihe nach jeweils mit einem Werkzeug überprüft. *ETC* wurde dazu gestartet und es wurde ein neuer Inspektionsauftrag mit dem zu prüfenden Spreadsheet angelegt. Für die beiden anderen Werkzeuge wurde dazu *Microsoft Office Excel 2003* gestartet und das jeweilige Spreadsheet geöffnet. Anschließend wurden die Prüfwerkzeuge für die Durchführung konfiguriert. Bei *ETC* wurde dazu die Inventarisierung durchgeführt und anschließend wurde die vorhandene Richtlinie mit den drei Vorschriften ausgewählt. Im zweiten Durchgang für *ETC* wurden zudem die benutzerdefinierte Einstellungen vorgenommen. Für die beiden anderen Werkzeuge wurden über die zur Verfügung gestellten Symbolleisten die drei zu überprüfenden Vorschriften ausgewählt, beziehungsweise andere verfügbare Vorschriften abgewählt.

Durchführung und Analyse Anschließend wurde die Durchführung der Prüfung gestartet. Dabei wurde jeweils die Variante gewählt, die einen Bericht über die gefundenen Verstöße liefert. Im Fall von *ETC* liegt dieser Bericht als .html-Datei vor. Die beiden anderen Werkzeuge, erzeugen ein weiteres Spreadsheet. Die Berichte zu den Prüfungen der Spreadsheets wurden dabei gespeichert und anschließend durch den Verfasser dieser Arbeit analysiert. Die dabei entstandenen Berichte sind auf dem beigefügten Datenträger gespeichert (Siehe Abschnitt A.1). Bei der Analyse wurde dabei die Anzahl der einzelnen Verstöße als auch die Anzahl der gefundenen Gruppierung notiert. Die quantitativen Ergebnisse für einzelne Verstöße werden dabei im nächsten Abschnitt dargestellt; die Darstellung der Ergebnisse für gefundene Gruppierung wurde in Abschnitt A.2 verlagert.

9.2 Ergebnisse

In den folgenden Abschnitten werden einige der Ergebnisse der Evaluation in Tabellenform präsentiert. Für die Präsentation der Ergebnisse werden dabei folgende Abkürzung verwendet:

SP bezeichnet das Prüfwerkzeug Spreadsheet Professional.

RA bezeichnet das Prüfwerkzeug Rainbow Analyst.

ETC bezeichnet das Prüfwerkzeug *Example Testing Center*

ETC(C) bezeichnet das Prüfwerkzeug *Example Testing Center* bei dem die Vorschriften mit den benutzerdefinierten Einstellungen konfiguriert wurden.

NMI bezeichnet die nichtmechanische Inspektion der Spreadsheets durch den Verfasser dieser Arbeit.

Dabei zeigen die Tabellen die Anzahl der einzelnen Verstößen an, die bei den unterschiedlichen Prüfungen der einzelnen Vorschriften gefunden wurden. Außerdem wird der jeweilige Durchschnitts-, Median-, Minimal- und Maximalwert über alle durchgeführten Prüfungen pro Spreadsheet angegeben. Wurden bei Prüfungen eines Spreadsheets die gleiche Anzahl an Verstößen gefunden, wurden die Zellen der Tabelle mit der höchsten Anzahl an Übereinstimmungen grün markiert.

Konstanten in Formeln	SP	RA	ETC	ETC(C)	NMI	Ø	Median	Max	Min
spreadsheet_sample_01	67	0	67	0	67	40	67	67	0
spreadsheet_sample_02	126	126	126	126	126	126	126	126	126
spreadsheet_sample_03	27	0	27	0	27	16	27	27	0
spreadsheet_sample_04	0	0	0	0	0	0	0	0	0
spreadsheet_sample_05	30	30	30	0	30	24	30	30	0
spreadsheet_sample_06	18	6	18	6	18	13	18	18	6
spreadsheet_sample_07	20	20	20	20	20	20	20	20	20
spreadsheet_sample_08	33	0	34	34	34	27	34	34	0
spreadsheet_sample_09	5	2	5	5	5	4	5	5	2
spreadsheet_sample_10	4	0	2	0	0	1	0	4	0
spreadsheet_sample_11	9	0	9	9	9	7	9	9	0
spreadsheet_sample_12	16	5	16	15	15	13	15	16	5

Abbildung 9.3: Evaluationsergebnisse der Vorschrift *Konstanten in Formeln*

Leserichtung	SP	RA	ETC	Ø	Median	Max	Min
spreadsheet_sample_01	27	27	117	57	27	117	27
spreadsheet_sample_02	27	27	45	33	27	45	27
spreadsheet_sample_03	27	27	135	63	27	135	27
spreadsheet_sample_04	219	119	396	245	219	396	119
spreadsheet_sample_05	30	3	120	51	30	120	3
spreadsheet_sample_06	6	46	12	21	12	46	6
spreadsheet_sample_07	5	30	5	13	5	30	5
spreadsheet_sample_08	37	34	55	42	37	55	34
spreadsheet_sample_09	0	6	0	2	0	6	0
spreadsheet_sample_10	7	16	7	10	7	16	7
spreadsheet_sample_11	0	4	0	1	0	4	0
spreadsheet_sample_12	9	17	22	16	17	22	9

Abbildung 9.4: Evaluationsergebnisse der Vorschrift *Leserichtung*

Formelkomplexität	SP	RA	ETC	ETC(C)	Ø	Median	Max	Min
spreadsheet_sample_01	67	0	54	0	30	27	67	0
spreadsheet_sample_02	3	0	6	3	3	3	6	0
spreadsheet_sample_03	27	0	30	6	16	17	30	0
spreadsheet_sample_04	29	0	48	48	31	39	48	0
spreadsheet_sample_05	30	24	60	60	44	45	60	24
spreadsheet_sample_06	7	1	14	6	7	7	14	1
spreadsheet_sample_07	0	0	0	0	0	0	0	0
spreadsheet_sample_08	36	13	16	1	17	15	36	1
spreadsheet_sample_09	0	0	0	0	0	0	0	0
spreadsheet_sample_10	3	0	18	12	8	8	18	0
spreadsheet_sample_11	0	2	2	2	2	2	2	0
spreadsheet_sample_12	5	0	8	4	4	5	8	0

Abbildung 9.5: Evaluationsergebnisse der Vorschrift *Formelkomplexität*

9.3 Analyse

In diesem Abschnitt sollen nun die Ergebnisse der Evaluation analysiert werden. Dabei wird auf einzelne interessante Ergebniswerte eingegangen und anschließend ein allgemeines Fazit gezogen.

9.3.1 Auffällige Werte

Bei der Betrachtung der Ergebnisse stechen einige Werte besonders heraus, diese sollen nun nach den einzelnen Vorschriften geordnet vorgestellt und kommentiert werden.

Formelkomplexiät Bei dieser Vorschrift ergaben sich sehr viele Übereinstimmungen zwischen den Ergebnissen der einzelnen Prüfungen. So ist auch die Anzahl der gefundenen Verstöße, die bei der nichtmechanischen Inspektion gefunden wurden, fast immer mit denen der Prüfwerkzeuge identisch. Auffällig ist dabei außerdem die hohe Anzahl an Übereinstimmungen zwischen der konfigurierten Prüfung von *ETC* und der Prüfung durch das Werkzeug Rainbow Analyst. Das liegt mit hoher Wahrscheinlichkeit daran, dass das Prüfwerkzeug Rainbow Analyst standardmäßig die Konstanten 0 und 1 ignoriert, ohne jedoch darauf hinzuweisen. Dennoch existieren Unterschiede zwischen den Ergebnissen, da die konfigurierte Prüfung von *ETC* außerdem noch Konstanten in den Formeln *INDEX* und *INDIRECT* zulässt.

Leserichtung Die Prüfung durch *ETC* lieferte bei der Vorschrift *Leserichtung* grundsätzlich höhere Werte als die beiden anderen Werkzeuge. Dies ist wohl damit zu begründen, dass die anderen Werkzeuge scheinbar Referenzen auf Worksheets, die vor dem Worksheet

der referenzierenden Zelle liegen, nicht als Verstoß melden. Weiterhin ist die Anzahl der Verstöße, die durch Rainbow Analyst gefunden wurden, meist höher, als die von Spreadsheet Professional. Eine Begründung dafür konnte nicht gefunden werden, da die Kriterien entsprechenden Vorschriften zur *Leserichtung Row differences* und *Column differences* des Werkzeugs Rainbow Analyst nicht definiert werden.

Formelkomplexität Bei der Analyse der Ergebnisse der Vorschrift *Formelkomplexität* fällt auf, dass die einzelnen Definitionen dieser Vorschrift sich recht stark unterscheiden müssen, da bei fast allen Spreadsheets ein Prüfwerkzeug keine oder nur eine sehr geringe Anzahl an Verstößen meldete. Dennoch gibt es auch hier Spreadsheets bei denen die gleiche Anzahl an Verstößen von unterschiedlichen Werkzeugen gemeldet wurde. Auffällig ist außerdem, dass die Ergebnisse bei der konfigurierten Prüfung von *ETC* teilweise recht stark von denen der unkonfigurierten Prüfung abweichen, obwohl die Einstellungen nur in geringem Umfang verändert wurden. Dies zeigt, wie schnell hohe falsch positiv Raten entstehen können, wenn die Vorschriften nicht optimal an den jeweiligen Nutzungskontext angepasst sind, oder gar nicht angepasst werden können.

9.3.2 Zusammenfassung

Die Ergebnisse der Evaluation zeigen, dass die Prüfwerkzeuge in den meisten Fällen eine vergleichbare Anzahl an Verstößen finden. Des Weiteren stimmt diese Anzahl auch häufig mit der Anzahl der Verstöße überein, die in der nichtmechanischen Inspektion gefunden wurden. Gleichzeitig zeigen die Ergebnisse jedoch auf, dass bei den schwieriger zu überprüfenden Vorschriften *Leserichtung* und *Formelkomplexität* die Übereinstimmungen geringer ausfallen als bei der einfachen Vorschrift *Konstanten in Formeln*. Dies bestätigt auch für Spreadsheets die Erkenntnis aus Unterabschnitt 3.5.2, dass unterschiedliche statische Analysewerkzeugen häufig unterschiedliche Verstöße melden.

Natürlich kann eine reine quantitative Analyse keinen Aufschluss darüber geben, ob bei den gefundenen Verstößen, tatsächlich Abweichungen von den Vorgaben vorliegen. Jedoch lässt die relativ hohe Übereinstimmung zwischen den Ergebnissen aller Prüfungen vermuten, dass insgesamt eine geringe Anzahl an unzutreffenden Befunden bezüglich der oben definierten Vorschriften vorliegt. Weiterhin wird gezeigt, dass eine unterschiedliche Konfiguration einer Vorschrift, die Ergebnisse stark beeinflussen kann. Dadurch wird die Gefahr verdeutlicht, dass bei nicht konfigurierbaren Werkzeugen hohe falsch positiv Raten auftreten können, wenn sich die gewünschten Vorgaben von denen, die vom Werkzeug überprüft werden, unterscheiden. Es kann durch diese Evaluation jedoch nicht geklärt werden, ob die gefundenen Verstöße dabei helfen können, Fehler zu entdecken, die zu falschen Ergebnissen führen. Zu diesem Zweck müssten weitere Auswertungen stattfinden, bei denen die Anforderungen an die untersuchten Spreadsheets bekannt sind und die Auswirkungen gemeldete Verstöße im Detail untersucht werden. Das Fazit dieser Evaluation ist jedoch, dass die Ergebnisse des prototypischen Werkzeugs *ETC* mit denen von kommerziellen Prüfwerkzeugen vergleichbar scheinen und für die überprüfte Vorschrift *Konstanten in Formeln* sogar die höchste Übereinstimmung mit der Anzahl der gefunden Verstöße durch eine menschliche Prüfung bieten.

Kapitel 10

Fazit

In diesem Kapitel soll eine Zusammenfassung dieser Arbeit und speziell der entstandenen Lösung, dem *Spreadsheet Inspection Framework*, gegeben werden. Dazu werden die konzeptionellen Vorteile von *SIF* präsentiert, aber auch dessen bestehenden Einschränkungen aufgelistet. Anschließend wird diskutiert, wie die Spreadsheet Inspektions-Werkstatt weiter verbessert werden kann und welche Erweiterungsmöglichkeiten für sie bestehen.

10.1 Zusammenfassung

In dieser Arbeit wurde das Konzept einer Inspektions-Werkstatt für Spreadsheets erarbeitet, das als technische Grundlage dienen soll, um systematische, werkzeug-unterstütze Prüfungen von Spreadsheets durch Endbenutzer durchführen zu können. Dabei sollen die Vorteile von verschiedenen Prüfungsarten genutzt werden, indem diese in einer verzahnten Art und Weise angewendet werden. Gleichzeitig wurden die Fähigkeiten und das Wissen von Endbenutzern für das Konzept berücksichtigt, in dem die angebotenen Prüfungen in einem hohen Maße automatisch ablaufen und zu ihrer Durchführung kaum Prüfwissen notwendig ist.

Die Umsetzung der Inspektions-Werkstatt wurde dabei als plattformunabhängiges Java-Framework mit dem Namen *Spreadsheet Inspection Framework* in mehreren Ausbaustufen geplant. Mittels der Realisierung der ersten Ausbaustufe in dieser Arbeit wurde der Grundstein der Inspektions-Werkstatt gelegt, indem es ermöglicht wird, Spreadsheets mittels statischer Prüfungen auf festlegbare und konfigurierbare Vorschriften zu überprüfen. Ähnlich wie bei Findbugs wurde eine breite Anzahl an recht einfachen Techniken für die statische Analyse verwendet und es wurden die Erkenntnisse aus dem Einsatz von statischen Analysewerkzeugen in der traditionellen Software-Entwicklung in die Konzeption miteinbezogen. So wurde großer Wert darauf gelegt, dass die durchführbaren Prüfungen an den jeweiligen Nutzungskontext angepasst werden können, indem eigene Richtlinien aus bestehenden Vorschriften zusammengestellt, die Vorgaben von Vorschriften konfiguriert und neue Vorschriften hinzugefügt werden können. In dieser Arbeit wurde dabei die Architektur von *SIF* entworfen, sowie die notwendige Basis implementiert, um die Prüfungen für drei ausgewählte Vorschriften realisieren zu können. Zudem wurde die prototypische Anwendung *Example Testing Center* entwickelt, die als erstes

Prüfzentrum für Spreadsheets fungiert und einige Funktionen der Inspektions-Werkstatt über eine rudimentäre grafische Oberfläche zugänglich macht. Bei der Evaluation von *ETC* mittels einer kleinen Auswahl an Spreadsheets aus Real- und Laborumgebungen wurde gezeigt, dass *ETC* in der Lage ist, vergleichbare Ergebnisse zu denen bestehender kommerzieller statischer Analysewerkzeuge zu erzielen. Gleichzeitig bestehen durch die Verwendung von *SIF* , als Basis für die durchgeführten Prüfungen die im folgenden Abschnitt beschriebenen konzeptionellen Vorteile gegenüber bestehenden statischen Prüfwerkzeugen.

10.1.1 Konzeptionelle Vorteile

- Durch die Konzeption von *SIF* als Java-Framework, sowie die gewählte Architektur besteht keine Bindung an ein bestimmtes Betriebssystem, eine bestimmte Spreadsheet-Software oder ein bestimmtes Spreadsheet-Dateiformat. Als Folge daraus bestehen kaum Einschränkungen für die Verwendung von *SIF* , so dass ein breites Spektrum an Einsatzmöglichkeiten gegeben ist.
- Auch für die zu prüfenden Spreadsheets bestehen geringe Voraussetzungen, um einer Prüfung mit *SIF* unterzogen zu werden, da nur eine korrekt gespeicherte Spreadsheet-Datei benötigt wird. Außerdem erfordert eine statische Prüfung durch *SIF* nur einen geringen Aufwand, da die Durchführung der statischen Prüfungen überwiegend automatisch erfolgt.
- Das Framework bietet ein hohes Maß an Konfigurierbarkeit, so dass die durchgeführten Prüfungen an den jeweiligen Nutzungskontext angepasst werden können. So besteht die Möglichkeit, dass in Unternehmen eigene Richtlinien für Spreadsheets erstellt werden können, deren Überprüfung dann mittels der Inspektions-Werkstatt erfolgt. Dabei sind die Vorschriften der erstellten Richtlinien jedoch nicht nur auf die bereits implementierten Vorschriften beschränkt, da neue Vorschriften der Inspektionswerkstatt hinzugefügt und deren Prüfung über eine Schnittstelle realisiert werden können. Außerdem besteht die Möglichkeit, die einzelnen Vorschriften einer Richtlinie an das jeweils zu überprüfenden Spreadsheet anzupassen.
- Diese mögliche Erweiterung des angebotenen Prüfungsumfangs ist unter anderem auch dadurch möglich, da es sich bei dem *Spreadsheet Inspection Framework* um ein Open-Source-Projekt handelt.
- *SIF* ist nicht nur für die Entdeckung von Fehlern mit statischen Fehlern konzipiert, sondern bietet Erweiterungsmöglichkeiten für dynamische Prüfungen und die Unterstützung von nichtmechanischen Prüfungen an. So kann beispielsweise durch die Visualisierung des Spreadsheets und der gefundenen Verstöße, sowie durch ein geeignetes Testverfahren die Fehlerentdeckung in Spreadsheets weiter verbessert werden.

10.1.2 Einschränkungen

Neben den bestehenden Vorteilen unterliegt die erste Ausbaustufe von *SIF* und das prototypische *ETC* jedoch auch einigen Einschränkungen, die durch den bisherigen Fortschritt bei der Umsetzung des Konzept bedingt sind.

- Bisher wurde nur eine geringe Anzahl an Vorschriften und den zugehörigen Prüfungen realisiert.
- Es werden bislang nur die Excel-Dateiformate .xls und .xlsx zur Prüfung unterstützt und die Repräsentation von Spreadsheets und dessen Basiselementen ist noch unvollständig.
- Die Klassifikation von Fehlern bedarf einer Verbesserung. Es wurde zwar eine technische Lösung konzipiert, um die Verstöße von Vorschriften in einer Fehlertaxonomie einzuordnen, jedoch wurde bisher noch kein Klassifikationsschema implementiert. Der Grund dafür ist das Fehlen einer entsprechenden Taxonomie, die sich zur Klassifikation von Befunden eines statischen Analysewerkzeugs eignet. Zwar wurde eine mögliche eigene Taxonomie für Verstöße aufgestellt (Siehe Abbildung A.5), diese ist jedoch noch vollkommen unerprobt und daher für diesen Zweck ungeeignet.
- Verbesserungsbedarf besteht außerdem bei der Erhebung von geeigneten Metriken, die bei der Beurteilung der Qualität des geprüften Spreadsheets behilflich sein könnten. So werden zwar die Grunddaten für solche Metriken in Form des Spreadsheet-Inventars bereits erfasst, sinnvolle Metriken, wie etwa die Rate aller Zellen für die ein Verstoß gemeldet wurde, werden jedoch noch nicht erfasst.
- Ein weiterer Kritikpunkt ist sicherlich die geringe empirische Basis auf der die Evaluation von *ETC* durchgeführt wurde. Zwar zeigen die Ergebnisse der Evaluation, dass die gefundene Anzahl der Verstöße mit denen kommerzieller Prüfwerkzeuge vergleichbar ist, jedoch wurden die Verstöße nicht bezüglich ihrer Eignung untersucht, funktionale Fehler aufzudecken. So ist eine weitere Evaluation notwendig, um die Effektivität und Effizienz von *SIF* und den damit realisierbaren Prüfzentren zu belegen.
- Das prototypisch umgesetzte Prüfzentrum *ETC* ist bislang noch ungeeignet, um von Endbenutzern verwendet zu werden. So fehlt es an einer geeigneten Oberfläche, die alle Funktionen von *SIF* für den Benutzer verfügbar macht, genauso wie an einer geeigneten Fehlerbehandlung. Außerdem entfaltet das Prüfzentrum seine vollen Nutzen erst, wenn die gefundenen Verstöße entsprechend visualisiert werden.

10.2 Ausblick

In dieser Arbeit ist die erste Ausbaustufe des *Spreadsheet Inspection Framework* sowie des prototypischen Prüfwerkzeugs *Example Testing Center* entstanden. Aufgrund der begrenzten Zeit

wurden nur einige Aspekte des Gesamtkonzepts einer Spreadsheet-Inspektions-Werkstatt für die erste Ausbaustufe umgesetzt. So handelt es sich bei *SIF* und *ETC* jeweils um einen Prototyp und nicht um fertige Produkte. Daher besteht noch eine Menge von Punkten, deren Umsetzung oder Erweiterung zur Verbesserung der mit *SIF* und *ETC* durchführbaren Prüfungen beitragen könnten.

10.2.1 Erweiterung und Verbesserung der bestehenden Funktionaliät

Um das Potential einer Inspektions-Werkstatt für Spreadsheets zu zeigen, wurden in dieser Arbeit die Prüfverfahren für drei Vorschriften implementiert. Dabei wurde auch bei der Implementierung des Models von *SIF* so vorgegangen, dass zuerst nur die Spreadsheet-Elemente modelliert und umgesetzt wurden, die für die Realisierung dieser Prüfverfahren notwendig waren. Eine Erweiterung des Modells, der angebotenen Prüfungen und der unterstützen Dateiformate wäre daher mit Sicherheit lohnenswert, um das mögliche Einsatzgebiet von *SIF* zu erweitern. Gleichzeitig besteht auch ein Verbesserungspotential bei den Prüfverfahren für bestehende Vorschriften. So können etwa weitere Konfigurationsmöglichkeiten für die Vorschriften angeboten und die Gruppierung von Verstößen verbessert werden.

10.2.2 Erweiterung um zusätzliche Prüfungsarten

Die erste Ausbaustufen der Inspektions-Werkstatt bietet bisher nur statische Mittel zur Prüfung von Spreadsheets an und die Ergebnisse von Prüfungen können bisher nur in Form von textuellen Beschreibungen in einem Html-Bericht ausgegeben werden. Dies ist jedoch nicht ausreichend, um Spreadsheets auf alle verschiedenen Qualitätsmerkmale zu prüfen, da die statische Prüfung durch ihr Konzept nicht in der Lage ist bestimmte Anforderungen zu überprüfen. Um dies zu Verbessern wäre die Erweiterung um weitere Prüfungsarten sinnvoll.

Nicht-mechanische Prüfungen

Die nicht-mechanischen Prüfungen stellen bei traditioneller Software einen wichtigen Bestandteil der Qualitätssicherung dar, da sie verhältnismäßig leicht umzusetzen sind und mit ihnen Fehler gefunden werden, die mit anderen Prüfungsarten nicht entdeckt werden können. Spreadsheets hingegen sind ohne Hilfsmittel nur recht schwer vom Menschen zu inspizieren, da es durch die Referenzen zwischen den Zellen und der zweidimensionale Anordnung von Zellen an der nötigen Lokalität fehlt. Die Visualisierung der Struktur von Spreadsheets sowie von gefundenen Verstößen aus der statischen Analyse, könnte jedoch eine entscheidende Unterstützung für die Inspektion von Spreadsheets durch den Menschen leisten. So könnte eine Visualisierung in *SIF* integriert werden, die es den Benutzern erlaubt, Zellen, die in der statischen Analyse viele Verstöße verursacht haben genauer auf Fehler zu inspizieren.

Dynamische Prüfungen

Eine weitere Verbesserung der Prüfung durch *SIF* wäre durch die Integration von dynamischen Prüfungen möglich. Diese könnten etwa in Form von Intervall-Tests, wie in [Aya01] beschrieben, in *SIF* integriert werden und speziell dazu verwendet werden, um auffällige Zellen und Formeln aus den Ergebnissen der anderen Prüfungsarten auf ihre Korrektheit zu überprüfen.

Anhang A

Anhang

A.1 Inhalt und Aufbau des beigelegten Datenträgers

Der beiliegende Datenträger ist wie folgt aufgebaut:

Ausarbeitung Der Ordner *ausarbeitung* enthält dieses Dokument im PDF-Format.

Implementierung Der Ordner *Implementierung* enthält den Quell-Code des *Spreadsheet Inspection Framework* und des *Example Testing Center* .

Evaluation Der Ordner *evaluation* enthält die Ordner *spreadsheets* und *ergebnisse*. Im Ordner *spreadsheets* sind dabei die verwendeten Spreadsheets der Evaluation enthalten und im Ordner *ergebnisse* befinden sich die Berichte der einzelnen Prüfwerkzeuge, die bei der Evaluation generiert wurden.

A.2 Evaluationsergebnisse

Im Folgenden werden die Ergebnisse der gefundenen Verstoß-Gruppen aus der Evaluation aufgelistet:

Konstanten in Formeln	SP	RA	ETC	ETC(C)	NMI	Ø	Median	Max	Min
spreadsheet_sample_01	26	0	26	0	29	16	26	29	0
spreadsheet_sample_02	3	17	48	48	17	27	17	48	3
spreadsheet_sample_03	8	0	9	0	9	5	8	9	0
spreadsheet_sample_04	0	0	0	0	0	0	0	0	0
spreadsheet_sample_05	9	10	9	0	10	8	9	10	0
spreadsheet_sample_06	3	6	3	1	3	3	3	6	1
spreadsheet_sample_07	2	4	4	4	4	4	4	4	2
spreadsheet_sample_08	15	0	20	20	7	12	15	20	0
spreadsheet_sample_09	5	2	5	5	5	4	5	5	2
spreadsheet_sample_10	2	0	1	0	0	1	0	2	0
spreadsheet_sample_11	6	0	9	9	9	7	9	9	0
spreadsheet_sample_12	7	5	16	15	3	9	7	16	3

Abbildung A.1: Evaluationsergebnisse der Vorschrift *Konstanten in Formeln* (Gruppiert)

Leserichtung	SP	RA	ETC	Ø	Median	Max	Min
spreadsheet_sample_01	24	27	27	26	27	27	24
spreadsheet_sample_02	24	27	27	26	27	27	24
spreadsheet_sample_03	8	21	27	19	21	27	8
spreadsheet_sample_04	53	119	198	123	119	198	53
spreadsheet_sample_05	9	3	27	13	9	27	3
spreadsheet_sample_06	1	46	6	18	6	46	1
spreadsheet_sample_07	1	30	5	12	5	30	1
spreadsheet_sample_08	22	34	27	28	27	34	22
spreadsheet_sample_09	0	6	0	2	0	6	0
spreadsheet_sample_10	3	16	7	9	7	16	3
spreadsheet_sample_11	0	4	0	1	0	4	0
spreadsheet_sample_12	7	17	9	11	9	17	7

Abbildung A.2: Evaluationsergebnisse der Vorschrift *Leserichtung* (Gruppiert)

Formelkomplexität	SP	RA	ETC	ETC(C)	Ø	Median	Max	Min
spreadsheet_sample_01	26	0	27	0	13	13	27	0
spreadsheet_sample_02	3	0	3	3	2	3	3	0
spreadsheet_sample_03	8	0	27	3	10	6	27	0
spreadsheet_sample_04	26	0	24	24	19	24	26	0
spreadsheet_sample_05	9	8	30	30	19	20	30	8
spreadsheet_sample_06	1	1	7	6	4	4	7	1
spreadsheet_sample_07	0	0	0	0	0	0	0	0
spreadsheet_sample_08	17	8	15	1	10	12	17	1
spreadsheet_sample_09	0	0	0	0	0	0	0	0
spreadsheet_sample_10	2	0	12	12	7	7	12	0
spreadsheet_sample_11	0	2	2	2	2	2	2	0
spreadsheet_sample_12	5	0	5	3	3	4	5	0

Abbildung A.3: Evaluationsergebnisse der Vorschrift *Formelkomplexität* (Gruppiert)

A.3 Fehler-Taxonomien

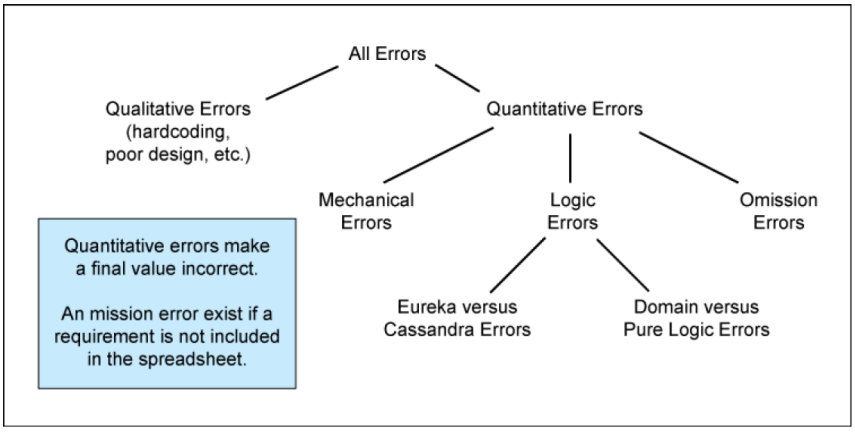

Abbildung A.4: Die bekannte Fehler-Taxonomie für Spreadsheets nach Panko und Halverson [Pan08b]

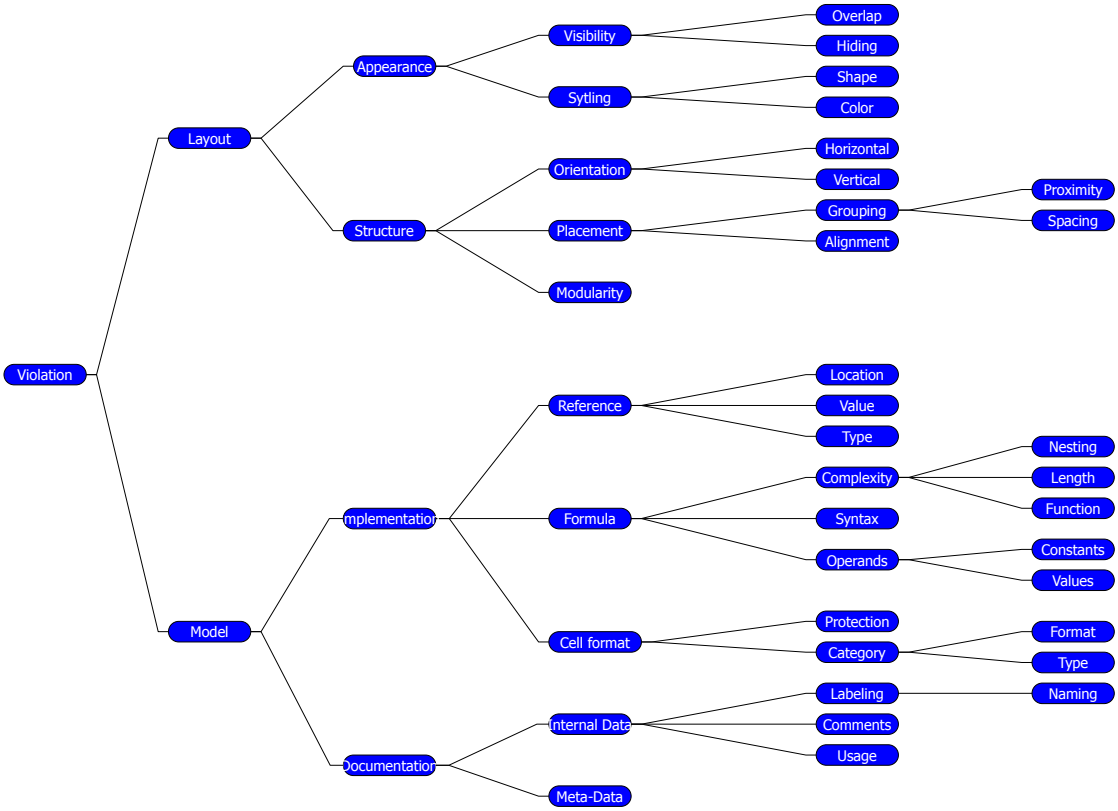

Abbildung A.5: Die prototypische Taxonomie für Verstöße, die in dieser Arbeit konzipiert wurde

Literaturverzeichnis

[ACM00] Y. Ayalew, M. Clermont, R. T. Mittermeir. Detecting Errors in Spreadsheets. In *In Proceedings of EuSpRIG 2000 Symposium: Spreadsheet Risks, Audit and Development Methods*, pp. 51–62. 2000.

[AE06] R. Abraham, M. Erwig. UCheck: A Spreadsheet Type Checker for End Users*, 2006.

[AM08] Y. Ayalew, R. Mittermeir. Spreadsheet Debugging. *CoRR*, abs/0801.4280, 2008.

[AP10] S. Aurigemma, R. R. Panko. The Detection of Human Spreadsheet Errors by Humans versus Inspection (Auditing) Software. *CoRR*, abs/1009.2785, 2010.

[Aya01] Y. Ayalew. *Spreadsheet Testing Using Interval Analysis*. Ph.D. thesis, Universität Klagenfurt, 2001.

[BG87] P. S. Brown, J. D. Gould. An experimental study of people creating spreadsheets. *ACM Trans. Inf. Syst.*, 5:258–272, 1987. doi:http://doi.acm.org/10.1145/27641.28058. URL http://doi.acm.org/10.1145/27641.28058.

[But00] R. J. Butler. Is This Spreadsheet a Tax Evader? How H.M. Customs and Excise Test Spreadsheet Applications. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 4 - Volume 4*, HICSS '00, pp. 4007–. IEEE Computer Society, Washington, DC, USA, 2000. URL http://dl.acm.org/citation.cfm?id=795711.799150.

[But08] R. J. Butler. Applying the CobiT Control Framework to Spreadsheet Developments. *CoRR*, abs/0801.0609, 2008.

[Cha08] D. Chadwick. EuSpRIG TEAM work:Tools, Education, Audit, Management. *CoRR*, abs/0806.0172, 2008.

[CHM08] M. Clermont, C. Hanin, R. T. Mittermeir. A Spreadsheet Auditing Tool Evaluated in an Industrial Context. *CoRR*, abs/0805.1741, 2008.

[CMW07] J. P. Caulkins, E. L. Morrison, T. Weidemann. Spreadsheet Errors and Decision Making: Evidence from Field Interviews. *JOEUC*, 19(3):1–23, 2007.

[Col10] D. Colver. Spreadsheet good practice: is there any such thing? *CoRR*, abs/1001.3967, 2010.

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

[Cro09] G. J. Croll. Spreadsheets and the Financial Collapse. *CoRR*, abs/0908.4420, 2009.

[CS96] Y. E. Chan, V. C. Storey. The use of spreadsheets in organizations: Determinants and consequences. *Information & Management*, 31(3):119 – 134, 1996. doi:DOI:10.1016/S0378-7206(96)00008-0. URL http://www.sciencedirect.com/science/article/pii/S0378720696000080.

[Dav82] G. B. Davis. Caution: User-Developed Systems Can be Hazardous to Your Organization. In *Hawaii International Conference on System Sciences*. Honolulu, Hawaii., 1982.

[Dun10] A. Dunn. Spreadsheets - the Good, the Bad and the Downright Ugly. *CoRR*, abs/1009.5705, 2010.

[EuSa] EuSpRIG. Best Practice. World Wide Web. URL http://www.eusprig.org/best-practice.htm.

[EuSb] EuSpRIG. Website of the European Spreadsheet Risks Interest Group. World Wide Web. URL http://www.eusprig.org/index.htm.

[FLS04] K. Frühauf, J. Ludewig, H. Sandmayr. *Software-Prüfung - eine Anleitung zum Test und zur Inspektion (5. Aufl.)*. vdf, 2004.

[Gar84] D. Garvin. What does product quality really mean? *Sloan Management Review*, 26:25–45, 1984. URL http://doku.iab.de/externe/2006/k060210f02.pdf.

[GHJ+96] D. F. Galletta, K. S. Hartzel, S. E. Johnson, J. L. Joseph, S. Rustagi. Spreadsheet presentation and error detection: an experimental study. *J. Manage. Inf. Syst.*, 13:45–63, 1996. URL http://dl.acm.org/citation.cfm?id=1189548.1189552.

[GHJV95] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns*. Addison-Wesley, Boston, MA, 1995. URL http://www.amazon.co.uk/exec/obidos/ASIN/0201633612/citeulike-21.

[GMz05] T. A. Grossman, V. Mehrotra, Özgür Özlük. Spreadsheet Information Systems are essential to business. Working Paper, 2005.

[GO08] T. A. Grossman, O. Ozluk. Research Strategy and Scoping Survey on Spreadsheet Practices. *CoRR*, abs/0807.3184, 2008.

[GO10] T. A. Grossman, O. Ozluk. Spreadsheets Grow Up: Three Spreadsheet Engineering Methodologies for Large Financial Planning Models. *CoRR*, abs/1008.4174, 2010.

[Gro07] T. A. Grossman. Spreadsheet Engineering: A Research Framework. *CoRR*, abs/0711.0538, 2007.

[HHN85] E. L. Hutchins, J. D. Hollan, D. A. Norman. Direct manipulation interfaces. *Hum.-Comput. Interact.*, 1:311–338, 1985. doi:http://dx.doi.org/10.1207/s15327051hci0104_2. URL http://dx.doi.org/10.1207/s15327051hci0104_2.

[HPD11] F. Hermans, M. Pinzger, A. van Deursen. Breviz: Visualizing Spreadsheets using Dataflow Diagrams. *CoRR*, abs/1111.6895, 2011.

[IR05] M. F. Ii, G. Rothermel. The EUSES Spreadsheet Corpus: A Shared Resource for Supporting Experimentation with Spreadsheet Dependability Mechanisms. In *In 1st Workshop on End-User Software Engineering*, pp. 47–51. 2005.

[Iro08] R. J. Irons. The Wall and The Ball: A Study of Domain Referent Spreadsheet Errors. *CoRR*, abs/0804.0943, 2008.

[JH96] M. Jean, J. Hall. A Risk and Control-Oriented Study of the Practices of Spreadsheet Application Developers. In *In Proceedings of the 29th Hawaii International Conference on System Sciences*, pp. 364–373. 1996.

[Kru06] S. E. Kruck. Testing spreadsheet accuracy theory. *Inf. Softw. Technol.*, 48:204–213, 2006. doi:http://dx.doi.org/10.1016/j.infsof.2005.04.005. URL http://dx.doi.org/10.1016/j.infsof.2005.04.005.

[Kul11] D. Kulesz. From Good Practices to Effective Policies for Preventing Errors in Spreadsheets. 2011.

[LL07] J. Ludewig, H. Lichter. *Software Engineering - Grundlagen, Menschen, Prozesse, Techniken*. dpunkt.verlag, 2007.

[LPKW06] H. Lieberman, F. Paternò, M. Klann, V. Wulf. End-User Development: An Emerging Paradigm. In H. Lieberman, F. Paternò, V. Wulf, editors, *End User Development*, volume 9 of *Human-Computer Interaction Series*, chapter 1, pp. 1–8. Springer Netherlands, Dordrecht, 2006. doi:10.1007/1-4020-5386-X_1. URL http://dx.doi.org/10.1007/1-4020-5386-X_1.

[MK05] T. J. McGill, J. E. Klobas. The role of spreadsheet knowledge in user-developed application success. *Decis. Support Syst.*, 39:355–369, 2005. doi:10.1016/j.dss.2004.01.002. URL http://dl.acm.org/citation.cfm?id=1196235.1196242.

[MMB09] R. McKeever, K. McDaid, B. Bishop. An Exploratory Analysis of the Impact of Named Ranges on the Debugging Performance of Novice Users. *CoRR*, abs/0908.0935, 2009.

[NM90a] B. A. Nardi, J. R. Miller. An ethnographic study of distributed problem solving in spreadsheet development. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, CSCW '90, pp. 197–208. ACM, New York, NY, USA, 1990. doi:http://doi.acm.org/10.1145/99332.99355. URL http://doi.acm.org/10.1145/99332.99355.

[NM90b] B. A. Nardi, J. R. Miller. The spreadsheet interface: A basis for end user programming. In *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*, INTERACT '90, pp. 977–983. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 1990. URL http://portal.acm.org/citation.cfm?id=647402.725609.

[NO10] D. Nixon, M. O'Hara. Spreadsheet Auditing Software. *CoRR*, abs/1001.4293, 2010.

[O'B] P. O'Beirne. European Spreadsheet Risks Interest Group: Horror Stories. World Wide Web. URL http://www.eusprig.org/horror-stories.htm.

[O'B05] P. O'Beirne. *Spreadsheet Check and Control*. Systems Publishing Corporation, 2005.

[O'B10] P. O'Beirne. Spreadsheet Refactoring. *CoRR*, abs/1009.1412, 2010.

[ON87] J. R. Olson, E. Nilsen. Analysis of the cognition involved in spreadsheet software interaction. *Hum.-Comput. Interact.*, 3(4):309–349, 1987. doi:10.1207/s15327051hci0304_1. URL http://dx.doi.org/10.1207/s15327051hci0304_1.

[Pan98] R. R. Panko. What we know about spreadsheet errors. *J. End User Comput.*, 10:15–21, 1998. URL http://portal.acm.org/citation.cfm?id=287893.287899.

[Pan99] R. R. Panko. Applying code inspection to spreadsheet testing. *J. Manage. Inf. Syst.*, 16:159–176, 1999. URL http://dl.acm.org/citation.cfm?id=1189438.1189448.

[Pan07] R. R. Panko. Recommended Practices for Spreadsheet Testing. *CoRR*, abs/0712.0109, 2007.

[Pan08a] R. R. Panko. Reducing Overconfidence in Spreadsheet Development. *CoRR*, abs/0804.0941, 2008.

[Pan08b] R. R. Panko. Revisiting the Panko-Halverson Taxonomy of Spreadsheet Errors. *CoRR*, abs/0809.3613, 2008.

[Pan08c] R. R. Panko. Spreadsheet Errors: What We Know. What We Think We Can Do. *CoRR*, abs/0802.3457, 2008.

[PB08] S. G. Powell, B. L. 0002, K. R. Baker. Impact of Errors in Operational Spreadsheets. *CoRR*, abs/0801.0715, 2008.

[PBL08a] S. G. Powell, K. R. Baker, B. Lawson. An auditing protocol for spreadsheet models. *Inf. Manage.*, 45:312–320, 2008. doi:10.1016/j.im.2008.03.004. URL http://dl.acm.org/citation.cfm?id=1379454.1379496.

[PBL08b] S. G. Powell, K. R. Baker, B. Lawson. A critical review of the literature on spreadsheet errors. *Decis. Support Syst.*, 46:128–138, 2008. doi:http://dx.doi.org/10.1016/j.dss.2008.06.001. URL http://dx.doi.org/10.1016/j.dss.2008.06.001.

[PBL09] S. G. Powell, K. R. Baker, B. Lawson. Errors in operational spreadsheets. *Organizational and End User Computing*, 21(3):24–36, 2009.

[PBLFJ08] S. G. Powell, K. R. Baker, B. Lawson, L. Foster-Johnson. Comparison of Characteristics and Practices amongst Spreadsheet Users with Different Levels of Experience. *CoRR*, abs/0803.0168, 2008.

[PHJ96] R. R. Panko, R. P. Halverson Jr. Spreadsheets on Trial: A Survey of Research on Spreadsheet Risks. In *Proceedings of the 29th Hawaii International Conference on System Sciences Volume 2: Decision Support and Knowledge-Based Systems*, HICSS '96, pp. 326–. IEEE Computer Society, Washington, DC, USA, 1996. URL http://portal.acm.org/citation.cfm?id=795699.798442.

[PO08] R. R. Panko, N. Ordway. Sarbanes-Oxley: What About all the Spreadsheets? *CoRR*, abs/0804.0797, 2008.

[Pry08] L. Pryor. When, why and how to test spreadsheets. *CoRR*, abs/0807.3187, 2008.

[RAF04] N. Rutar, C. B. Almazan, J. S. Foster. A Comparison of Bug Finding Tools for Java. In *Proceedings of the 15th International Symposium on Software Reliability Engineering*, pp. 245–256. IEEE Computer Society, Washington, DC, USA, 2004. doi:10.1109/ISSRE.2004.1. URL http://dl.acm.org/citation.cfm?id=1032654.1033833.

[Raf08] J. F. Raffensperger. New Guidelines For Spreadsheets. *CoRR*, abs/0807.3186, 2008.

[RCK08a] K. Rajalingham, D. Chadwick, B. Knight. An Evaluation of a Structured Spreadsheet Development Methodology. *CoRR*, abs/0801.1516, 2008.

[RCK08b] K. Rajalingham, D. R. Chadwick, B. Knight. Classification of Spreadsheet Errors. *CoRR*, abs/0805.4224, 2008. URL http://arxiv.org/abs/0805.4224.

[Wag06] S. Wagner. A literature survey of the quality economics of defect-detection techniques. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, ISESE '06, pp. 194–203. ACM, New York, NY, USA, 2006. doi:http://doi.acm.org/10.1145/1159733.1159763. URL http://doi.acm.org/10.1145/1159733.1159763.

[WDA+08] S. Wagner, F. Deissenboeck, M. Aichner, J. Wimmer, M. Schwalb. An Evaluation of Two Bug Pattern Tools for Java. In *Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation*, pp. 248–257. IEEE Computer Society, Washington, DC, USA, 2008. doi:10.1109/ICST.2008.63. URL http://dl.acm.org/citation.cfm?id=1381305.1382088.

[WJKT05] S. Wagner, J. Jürjens, C. Koller, P. Trischberger. Comparing Bug Finding Tools with Reviews and Tests. In *IN PROC. 17TH INTERNATIONAL CONFERENCE ON TESTING OF COMMUNICATING SYSTEMS (TESTCOM'05), VOLUME 3502 OF LNCS*, pp. 40–55. Springer, 2005.

[ZWN+06] J. Zheng, L. Williams, N. Nagappan, W. Snipes, J. P. Hudepohl, M. A. Vouk. On the Value of Static Analysis for Fault Detection in Software. *IEEE Trans. Softw. Eng.*, 32:240–253, 2006. doi:10.1109/TSE.2006.38. URL http://dl.acm.org/citation.cfm?id=1435724.1437767.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

_____

(Sebastian Zitzelsberger)