Institut für Softwaretechnologie
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart
Deutschland

Bachelorarbeit Nr. 83

Visualisierung von Fehlern in Spreadsheets

Ehssan Doust

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. rer. nat. Stefan Wagner
Betreuer:	M. Sc. Daniel Kulesz Dipl.-Inf. Fabian Beck
begonnen am:	12.06.2013
beendet am:	12.12.2013
CR-Klassifikation:	H.4.1, D.2.5, H.5.2

Kurzfassung

Spreadsheets sind weit verbreitete Dokumente mit Tabellenstruktur, die laut diverser Studien hohe Fehlerquoten aufweisen. An der Universität Stuttgart wurde ab 2012 mit dem Spreadsheet Inspection Framework ein Prüfwerkzeug für statische und dynamische Prüfungen auf Spreadsheets entwickelt. Das Spreadsheet Inspection Framework gibt nach Prüfung eines Spreadsheets einen statischen Bericht im HTML-Format aus. Dieser Bericht ist als Fehlervisualisierung unzureichend, da er separat als unformatierte Tabelle vorliegt und für typische Endanwender kaum verständlich ist.

Im Rahmen dieser Bachelorarbeit wurden Kriterien herausgearbeitet, die eine gute Fehlervisualisierung für Spreadsheet-Fehler erfüllen sollte. Danach wurde ein zu diesen Kriterien konformes Visualisierungskonzept für die aus dem Spreadsheet Inspection Framework stammenden Befunde erarbeitet. Dieses Konzept enthält einerseits die Darstellung eines interaktiven Berichts, und andererseits die Darstellung von Befund-Icons im Kontext der verursachenden Zelle. Diese zweigleisige Darstellung der Befunde soll eine optimale Navigierbarkeit und intuitive Verständlichkeit der Fehlervisualisierung sicherstellen.

Das Konzept wurde anschließend prototypisch als Microsoft Office Excel 2013 Add-In umgesetzt und nach Abschluss der Implementierung mit sieben Probanden evaluiert. Die Evaluation hat kleinere Probleme bei der Umsetzung des Konzepts aufgezeigt. Grundsätzlich hat sich der Prototyp jedoch bei echten Excel-Anwendern als interaktives Fehlervisualisierungs-Werkzeug bewähren können.

Abstract

Spreadsheets are widely used documents with table structure. Recent studies report that spreadsheets have high error rates. In 2012, a spreadsheet auditing tool named Spreadsheet Inspection Framework was developed at the University of Stuttgart. This tool supports the static and dynamic testing of spreadsheets. The Spreadsheet Inspection Framework creates a static HTML report after having scanned a spreadsheet. This report is insufficient as fault visualization, because it is a separate unformatted table that is hardly understandable by typical end users.

Within this thesis, criteria were worked out, which a good fault visualization of spreadsheet errors should meet. Then, a visualization concept compliant to these criteria was developed. This concept includes on the one hand the representation of an interactive report, and on the other hand the representation of finding icons in the context of the causing cell. This two-tier approach of presenting the findings is meant to ensure an optimal navigability and intuitive intelligibility of the fault visualization.

The concept was then implemented as a Microsoft Office Excel 2013 add-in. After the implementation, the add-in was evaluated by seven subjects. The evaluation showed minor problems with the realization of the concept. In general, the prototype has proven its capabilities as interactive fault visualization tool in practice with real Excel users.

Inhaltsverzeichnis

1 Einleitung

Spreadsheets sind weit verbreitete Dokumente mit Tabellenstruktur, die hauptsächlich Daten und Formeln beinhalten. Laut diverser Studien weisen Spreadsheets hohe Fehlerquoten auf (Panko, Facing the Problem of Spreadsheet Errors, 2006). Da Spreadsheets von vielen Unternehmen eingesetzt werden und teilweise Einfluss auf unternehmenskritische Entscheidungen haben, gab es in der Vergangenheit bereits beträchtliche finanzielle Schäden, die sich direkt auf die Nutzung fehlerhafter Spreadsheets zurückführen lassen (European Spreadsheet Risks Interest Group, 2013). Dies hat zur Folge, dass ein erhöhtes Interesse an einer effektiven Senkung der bereits angesprochenen Fehlerquoten besteht. Hierfür wurden bereits Prüfwerkzeuge entwickelt, diese konnten sich in der Praxis allerdings noch nicht durchsetzen.

1.1 Motivation

Am Institut für Softwaretechnologie der Universität Stuttgart wurde ein Prüfwerkzeug für Spreadsheets unter dem Namen *Spreadsheet Inspection Framework* (Zitzelsberger, 2012) entwickelt. Dieses stellt eine Alternative zu bereits existierenden Prüfwerkzeugen dar. Das *Spreadsheet Inspection Framework* prüft Spreadsheets mit Hilfe von Prüfregeln. Nach der Prüfung eines Spreadsheets gibt das *Spreadsheet Inspection Framework* einen HTML-Bericht aus, welcher eine Liste von Befunden (das sind Verstöße gegen Prüfregeln) beinhaltet. Dieser HTML-Bericht liegt in einem statischen, nicht formatierten Tabellenformat vor. Er ist unübersichtlich, und für durchschnittliche, unauffällige Excel-Anwender schwer verständlich, deshalb wird im Rahmen dieser Bachelorarbeit ein geeignetes Visualisierungskonzept für die vom *Spreadsheet Inspection Framework* gemeldeten Befunde entwickelt.

1.2 Ziel

Die Fehlervisualisierungen der bereits existierenden Spreadsheet-Prüfwerkzeuge haben alle spezifische Stärken aber auch Schwächen. Das Ziel dieser Bachelorarbeit liegt in der Herausarbeitung von Eigenschaften, die eine gute Fehlervisualisierung erfüllen sollte, und in der anschließenden Erstellung eines technologieunabhängigen Konzepts, welches diese Kriterien

alle erfüllt. Hierfür werden unterstützend die bereits existierenden Prüfwerkzeuge betrachtet, um aus deren Schwächen zu lernen. Das fertige Konzept wird prototypisch für ein bestimmtes Tabellenkalkulationsprogramm umgesetzt und visualisiert die vom *Spreadsheet Inspection Framework* gemeldeten Befunde. Im Anschluss wird eine Evaluation der Umsetzung durchgeführt, welche eine qualitative Bewertung des erstellten Konzepts ermöglichen soll.

1.3 Kontext

Im Rahmen dieser Bachelorarbeit wird eine *Fehlervisualisierung für Spreadsheets* entwickelt. Die inhaltlich verwandte Bachelorarbeit *Benutzerschnittstelle für einen Spreadsheetprüfstand* (Scheurich, 2013), welche parallel zu dieser Arbeit von Jonas Scheurich erstellt wird, befasst sich mit der Eingabe von Prüfregeln für das *Spreadsheet Inspection Framework* direkt in einem Tabellenkalkulationsprogramm. Es ist geplant, dass für beide Bachelorarbeiten eine Komponente mit gemeinsamer Codebasis für ein bestimmtes Tabellenkalkulationsprogramm entwickelt wird. Diese Komponente trägt den Namen *Spreadsheet Inspection Framework Excel Integration (SIFEI)*. Da sich diese Bachelorarbeit mit der Fehlervisualisierung befasst, erhält die im Rahmen dieser Arbeit entwickelte Komponente den Kurznamen *SIFEI-Visualisierung*, die Arbeit von Jonas Scheurich erhält den Kurznamen *SIFEI-Szenarienerfassung*.

1.4 Aufbau des Dokuments

Zunächst werden in Kapitel 2 die für das generelle Verständnis notwendigen Grundlagen eingeführt. Danach wird in Kapitel 3 das Konzept zur Visualisierung von Fehlern in Spreadsheets erstellt. Im Anschluss wird der implementierte Prototyp in Kapitel 4 vorgestellt und in Kapitel 5 evaluiert. Als Abschluss folgt in Kapitel 6 die Zusammenfassung der gewonnenen Erkenntnisse sowie eine Auflistung der verbleibenden Arbeit.

2 Grundlagen

In diesem Kapitel werden die Grundlagen der Fehlererkennung und -visualisierung in Spreadsheets erläutert, sowie der Bezug zu verwandten Arbeiten hergestellt. Hierfür werden zunächst wichtige Begriffe definiert und die Fehlertaxonomie für Spreadsheets vorgestellt, danach wird das *Spreadsheet Inspection Framework* beschrieben. Im Anschluss wird auf existierende Spreadsheet-Visualisierungen und auf Fehlervisualisierungen im Bereich der Software-Entwicklung eingegangen. Als letztes wird auf Basis der gewonnenen Erkenntnisse zusammengefasst, welche Kriterien eine gute Spreadsheet-Fehlervisualisierung erfüllen sollte.

2.1 Spreadsheets

Ein *Spreadsheet* (Arbeitsmappe) ist ein Dokument zur Speicherung und Verarbeitung von alphanumerischen Daten in Tabellenform. Ein Spreadsheet kann mehrere Tabellen enthalten, welche im Folgenden als *Arbeitsblätter* (Worksheets) bezeichnet werden. In Abbildung 1 ist das Tabellenkalkulationsprogramm *Microsoft Office Excel* (Excel) dargestellt. Tabellenkalkulationsprogramme ermöglichen das Anzeigen, Erstellen und Bearbeiten von Spreadsheets.

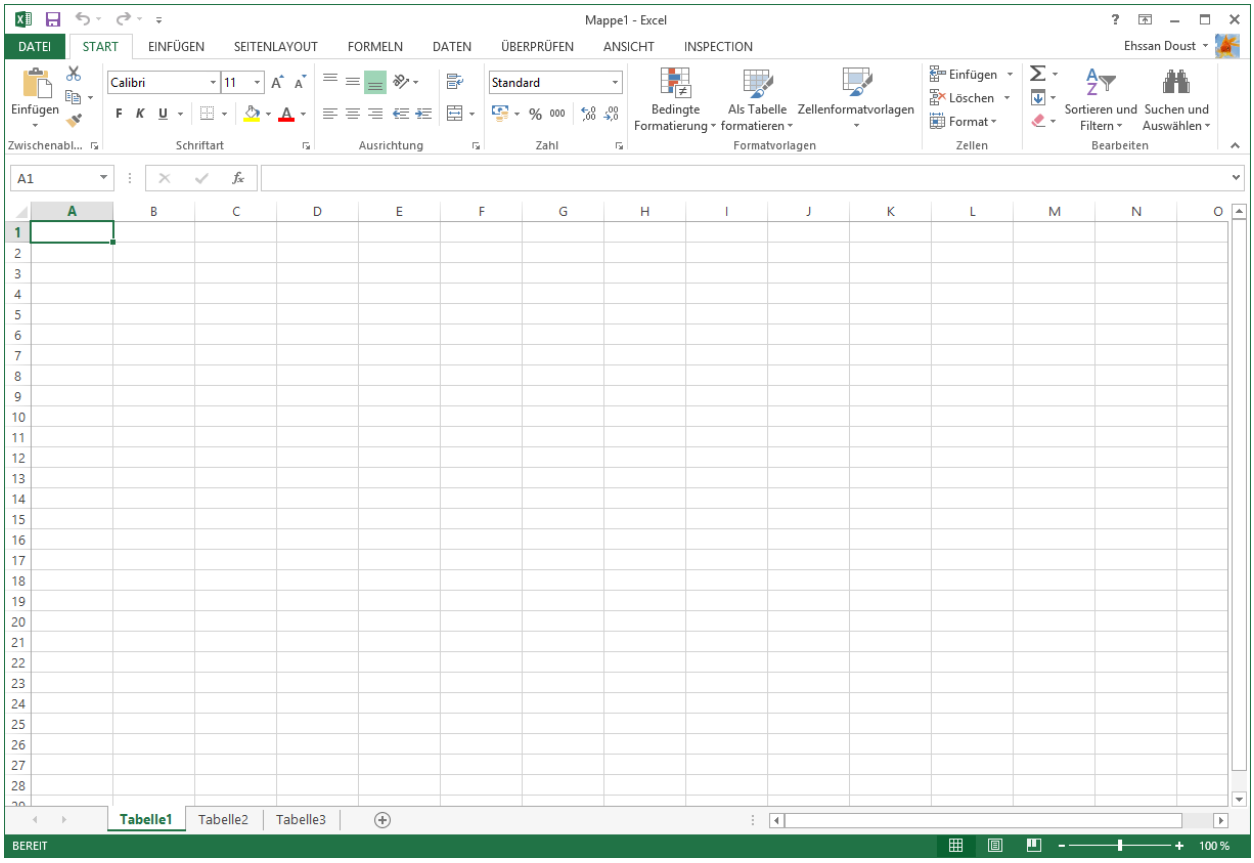

Abbildung 1: Leeres Spreadsheet im Tabellenkalkulationsprogramm Microsoft Office Excel 2013

Jede Tabellenzelle eines Spreadsheets kann eindeutig über Angaben zu dem beinhaltenden Arbeitsblatt, sowie der Spalte und Zeile referenziert werden. Eine solche Referenz wird üblicherweise in der sogenannten A1-Notation angegeben. Die in Abbildung 1 ausgewählte Zelle hätte als Referenz in A1-Notation den Wert "=Tabelle1!$A$1", oder in Kurzform den Wert "A1". Es können sowohl Zellreferenzen innerhalb eines Arbeitsblatts als auch über mehrere Arbeitsblätter hinweg erstellt werden.

In jeder Tabellenzelle kann entweder ein numerischer Wert (eine Zahl), ein alphanumerischer Wert (zum Beispiel ein Wort) oder eine Formel eingetragen werden. Formeln können simple Berechnungen ausführen, und auf einen Katalog an vordefinierten Funktionen zurückgreifen. Die Eingabewerte für Formeln können entweder Konstanten oder Referenzen auf andere Zellen sein. In Listing 1 ist beispielhaft eine Formel dargestellt, welche die Zahl aus Zelle A1 referenziert und auf die Konstante 24 addiert.

```
=Tabelle1!$A$1+24
```

Listing 1: Formel für die Berechnung einer Summe

In vielen Tabellenkalkulationsprogrammen können Zellen beliebig formatiert werden, zum Beispiel können die Schriftart und -größe, sowie die Hintergrundfarbe, Rahmenfarbe und -dicke angepasst werden. Zusätzlich bieten aktuelle Tabellenkalkulationsprogramme weitere Möglichkeiten zur Datenvisualisierung an, zum Beispiel können diverse Diagramme eingefügt werden Im Rahmen dieser Arbeit sind hauptsächlich Daten und Formeln als Zellinhalte von Belang.

2.2 Fehlertaxonomie

Zum besseren Verständnis der in Spreadsheets auftretenden Fehler ist es hilfreich, diese zu klassifizieren. Aktuell gibt es mehrere Fehlertaxonomien für Spreadsheet-Fehler, welche sich teilweise ähneln und aufeinander aufbauen. Die erste formale Fehlertaxonomie für Spreadsheet-Fehler wurde von (Panko & Halverson Jr., Spreadsheets on Trial: A Survey of Research on Spreadsheet Risks, 1996) veröffentlicht. In ihrer Arbeit unterteilen sie die Spreadsheet-Fehler auf oberster Ebene in qualitative und quantitative Fehler. *Quantitative Fehler* führen zu einem numerisch falschen Ergebnis. Hierzu zählen unter anderem fehlerhafte Formeln oder Konstanten. *Qualitative Fehler* hingegen verursachen kein falsches Ergebnis, aber sie verringern die Qualität des Spreadsheets und können bei künftigen Änderungen im Spreadsheet zu quantitativen Fehlern führen. Zu ihnen gehören beispielweise komplexe Formeln, welche nur schwer verständlich und somit bei einer erneuten Bearbeitung oder der Bearbeitung durch einen anderen Anwender fehleranfällig sind.

Viele aktuelle Prüfwerkzeuge für Spreadsheets suchen und finden nur qualitative Fehler. Qualitative Fehler sind im Gegensatz zu quantitativen Fehlern genau lokalisierbar, da jede Zelle unabhängig von allen anderen betrachtet wird. Bei quantitativen Fehlern ist nicht sicher, ob die Fehlerursache (das verursachende Element, zum Beispiel eine fehlerhafte Formel) und die Fehlerauswirkung (ein falsches Ergebnis) in derselben Zelle liegen. Eine Zelle kann beispielsweise ein falsches Ergebnis anzeigen, die eigentliche Fehlerursache (zum Beispiel ein Formelfehler) kann jedoch in einer oder mehreren der referenzierten Zellen liegen.

2.3 Spreadsheet Inspection Framework

Das Spreadsheet Inspection Framework ist ein in Java implementiertes Prüfwerkzeug, welches Spreadsheets mit Hilfe von statischen und dynamischen Prüfungen nach qualitativen beziehungsweise quantitativen Fehlern durchsucht. Es wurde von (Zitzelsberger, 2012) entwickelt und von (Lemcke, 2013) erweitert. Das Spreadsheet Inspection Framework prüft Spreadsheets mit Hilfe von Prüfregeln, welche über XML-Dateien konfiguriert werden. Ein beispielhafter Prüfvorgang ist in Abbildung 2 dargestellt.

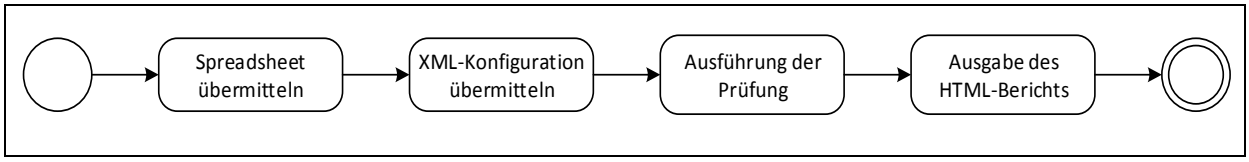

Abbildung 2: Darstellung eines Prüfvorgangs im Spreadsheet Inspection Framework

Zunächst muss ein Spreadsheet an das Spreadsheet Inspection Framework übermittelt werden, und im Anschluss die XML-Datei für die Prüfregeln geladen werden. Danach kann das Spreadsheet Inspection Framework die Prüfung des übermittelten Spreadsheets durchführen. Nach Ausführung der Prüfung gibt das Spreadsheet Inspection Framework einen HTML-Bericht aus, welcher die Befunde enthält.

Es ist zu beachten, dass mit *Befund* jeder vom Spreadsheet Inspection Framework gemeldete Verstoß gegen ausgeführte Prüfregeln gemeint ist. Nicht jeder Befund muss zwangsläufig einen Fehler anzeigen, es kann sich auch um falsch positiv erkannte Befunde handeln. Eine *Fundstelle* gibt den möglichen Ort der Fehlerursache eines Befunds an. Jeder Befund hat eine Fundstelle, die tatsächliche Fehlerursache kann jedoch sowohl in der Fundstelle als auch in einer der referenzierten Zellen liegen. Die *SIFEI-Visualisierung* verwendet das Spreadsheet Inspection Framework zum Prüfen von Spreadsheets und visualisiert im Anschluss die Befunde.

2.3.1 Statische Prüfungen

Die sogenannten statischen Prüfungen zielen auf das Finden qualitativer Fehler ab. Die Zellinhalte werden nicht ausgewertet, sondern lediglich anhand von syntaktischen Merkmalen geprüft. Gegenwärtig unterstützt das Spreadsheet Inspection Framework drei statische Prüfregeln, in Zukunft kann das Spreadsheet Inspection Framework um weitere statische Prüfregeln sowie Konfigurationsmöglichkeiten für diese Prüfregeln erweitert werden. Die Regel

Keine Konstanten in Formeln besagt, dass in Formeln keine konstanten Werte vorkommen sollen, diese sollen in eigenen Zellen gespeichert werden, damit sie an zentraler Stelle modifiziert werden können. Die Regel *Leserichtung* besagt, dass in Formeln lediglich Zellen links und oberhalb der aktuellen Zelle referenziert werden sollen. Die Regel *Formelkomplexität* besagt, dass die Länge und Schachtelungstiefe von Formeln ein bestimmtes, vom Anwender konfigurierbares Maß nicht überschreiten sollte.

2.3.2 Dynamische Prüfungen

Die dynamischen Prüfungen zielen auf das Finden quantitativer Fehler ab. Für das Auffinden von quantitativen Fehlern werden die Formeln in den einzelnen Zellen ausgewertet und die Ergebnisse mit vom Anwender definierten Werten verglichen. Hierfür müssen Zellreferenzen mit einbezogen werden, was dazu führt, dass ein Verstoß gegen eine dynamische Prüfregel zwar einer Zelle (nämlich derjenigen mit falschem numerischen Ergebnis) zugeordnet werden kann, allerdings kann keine Aussage über die tatsächliche Fehlerursache getroffen werden, da der eigentliche Fehler in einer beliebigen referenzierten Zelle aufgetreten sein kann.

2.3.3 Bericht

Nach dem Abschluss eines Prüfvorgangs gibt das Spreadsheet Inspection Framework einen HTML-Bericht aus. Dieser Bericht enthält eine Liste mit Befunden, welche sowohl qualitative als auch quantitative Fehler repräsentieren können. Der Bericht liegt in unformatiertem, statischem HTML-Format vor (siehe Abbildung 3).

Findings

greaterThan0 (1 compounds; 1 single violations)

Author: ML
Descritption: "Executes" the spreadsheet and checks for several possible conditions.
Background: A testscenario which is defined by used defined conditions. While the process of checking the spreadsheet is run i.e. formulae of the spreadsheet are evaluated.
Chosen Severity Weight: 1.0
Possible Solution: Check the formulae in the causing cell or region.

Number	Causing element	Location	Description	Severity
1	Numeric-Cell [Formula]: 3544	[bafoeg-rueckzahlung_mod] Rechner!B36	The expected value was equal 3360 but the found value was 3544.0	60.0

Abbildung 3: Ausschnitt aus einem HTML-Bericht des Spreadsheet Inspection Frameworks

Zusätzlich ist im Bericht eine Unterteilung der relevanten Zellen in Eingabe-, Zwischenergebnis- und Ausgabezellen enthalten. Ein Befund besteht aus Informationen zum Ort des Verstoßes, einer Beschreibung des Verstoßes und einer Angabe seines Schweregrades, sowie einer Beschreibung des verursachenden Elements. Mehrere Verstöße gegen dieselbe Prüfregel, welche in benachbarten Zellen liegen, werden zu sogenannten Verstoßgruppen zusammengefasst.

2.3.4 Einschränkungen

Das Spreadsheet Inspection Framework kann keine in Spreadsheets eingebetteten Makros ausführen, und deshalb keine Spreadsheets prüfen, welche Makros in ihren Berechnungen verwenden. Eine weitere Einschränkung des Spreadsheet Inspection Frameworks in seiner gegenwärtig vorliegenden Version ist, dass es beim Prüfen von Spreadsheets mit mehreren Arbeitsblättern zu Fehlern kommt, da Referenzen auf unterschiedliche Arbeitsblätter nicht unterschieden werden können.

Darüber hinaus kann es vorkommen, dass das Spreadsheet Inspection Framework bei mehrmaligem Prüfen desselben Spreadsheets geringfügig unterschiedliche Fehlermeldungen für dieselben Befunde ausgibt (zum Beispiel wird die Sortierung in einer Aufzählung von Zelltypen nicht beibehalten, sondern die Sortierung der Aufzählung wird zufällig permutiert). Dies erschwert eine eindeutige Zuordnung der Befunde, insbesondere auf programmatischer Ebene.

2.4 Spreadsheet-Visualisierungen

Das Auffinden von Fehlern in Spreadsheets ist seit geraumer Zeit der Gegenstand von Forschungen. So wurden bereits von (Davis, 1996) zwei Visualisierungs-Werkzeuge für Spreadsheets unter der Annahme entwickelt, dass sich die Fehlerquote in Spreadsheets durch ein gutes Verständnis der Abhängigkeiten zwischen Formeln und Zellbezügen senken ließe.

Es gibt eine Reihe weiterer Visualisierungs-Werkzeuge für Spreadsheets, welche größtenteils versuchen, die Struktur und die Beziehungen innerhalb von Spreadsheets darzustellen, um durch ein so erhöhtes Verständnis des Anwenders die vorhandenen Fehler offensichtlich werden zu lassen. Teilweise verwenden diese Visualisierungs-Werkzeuge Konzepte aus der Software-Entwicklung. Ein interessanter Ansatz hierzu findet sich in der Dissertation von

(Hermans, 2012). Sie verwendet unter anderem Klassendiagramme, um die Struktur von Spreadsheets darzustellen. Diese Diagrammtypen werden typischerweise auf einer abstrakten Ebene der Software-Entwicklung eingesetzt.

2.5 Fehlervisualisierungen

Leider wurden selbst nach einer ausführlichen Recherche keine Arbeiten gefunden, welche sich speziell mit der Fehlervisualisierung in Spreadsheets befassen, deshalb wurden verwandten Arbeiten zur Fehlervisualisierung in der Software-Entwicklung auf Ebene des Quellcodes betrachtet. Interessanterweise existieren viele Arbeiten über die Fehlererkennung im Quellcode von Programmen, aber wenige Arbeiten, die sich mit der anschließenden Fehlervisualisierung auseinandersetzen. Dies hat vermutlich damit zu tun, dass Quellcode hauptsächlich in speziellen Anwendungen (Entwicklungsumgebungen) geschrieben wird, und diese Anwendungen bereits Möglichkeiten zur Fehlervisualisierung bereitstellen.

Das von (Slinger, 2005) entwickelte Plugin *CodeNose* für die Entwicklungsumgebung *Eclipse* (eclipse.org) sucht mit Hilfe von statischen Prüfungen nach qualitativen Fehlern im Quellcode. Die Befunde werden im Anschluss mit den Eclipse eigenen Visualisierungsmöglichkeiten dargestellt. Diese werden im folgenden Abschnitt näher betrachtet.

Die *Problemliste* zeigt einen Bericht der aktuell im Quellcode befindlichen Probleme. Hierzu zählen unter anderem Warnungen und Fehler. Jede Warnung und jeder Fehler haben ein bestimmtes Symbol, dieses ist für Warnungen gelb und für Fehler rot. Zusätzlich zu einer Problembeschreibung wird noch der Ort des verursachenden Elements angegeben (Abbildung 4).

Description	Location
◢ Errors (1 item)	
The local variable s may not have been initialized	line 16
◢ Warnings (1 item)	
The method nonStaticMethod() from the type Application can be declared as static	line 20

Abbildung 4: Beispiel einer Problemliste in der Entwicklungsumgebung Eclipse

Neben der Codezeile, welche das Problem verursacht hat, wird das gleiche Symbol angezeigt wie in der Problemliste. Diese Symbole werden durch sogenannte *Marker* links neben den Codezeilen dargestellt. In Abbildung 5 ist beispielhaft ein Marker zu sehen, welcher einer

Warnung in der Problemliste entspricht. Zusätzlich zum Marker ist der Quellcode unterstrichen, welcher innerhalb der Codezeile als problemverursachend festgestellt wurde.

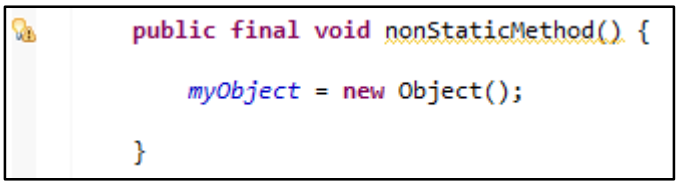

Abbildung 5: Icon an der Codestelle der Problemursache in Eclipse

Zusammenfassend lässt sich zur Fehlervisualisierung in Eclipse sagen, dass es sowohl eine Art Bericht in Form der Problemliste gibt, als auch eine Fehlervisualisierung direkt im Quellcode durch die Marker.

Ein weiteres Praxisbeispiel zur Fehlervisualisierung findet sich in Microsoft Office Excel selbst. Hier werden Fehler, welche nicht automatisch gelöst werden können, über ein kleines Icon (siehe Abbildung 6) neben der betroffenen Zelle dargestellt. Das Icon wird nur dann angezeigt, wenn die betroffene Zelle ausgewählt ist. Ein Klick auf das Icon öffnet ein Kontextmenü, welches Informationen über den Fehler bereitstellt, und Hilfestellungen zur Fehlerbehebung anbietet.

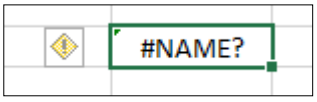

Abbildung 6: Fehlervisualisierung in Microsoft Office Excel 2013

Allgemein ergibt sich der Eindruck, dass sich die aktuelle Forschung mehr auf Methoden zur besseren Fehlererkennung in Spreadsheets und Spreadsheet-Visualisierungen konzentriert, nicht jedoch auf Fehlervisualisierungen in Spreadsheets.

2.6 Spreadsheet-Prüfwerkzeuge

Bei den meisten der aktuell verfügbaren Spreadsheet-Prüfwerkzeuge handelt es sich um proprietäre Anwendungen, welche kommerziell vertrieben werden. In einer Studie von (Nixon & O'Hara, 2010) wurden vier Spreadsheet-Prüfwerkzeuge sowie die in Microsoft Office Excel integrierte Prüffunktionalität auf ihre Nützlichkeit und Effektivität hin untersucht. Hierzu wurden diese einzeln betrachtet und anhand eines exemplarischen Spreadsheets

getestet. Im Anschluss werden die von den betrachteten Prüfwerkzeugen gefundenen und nicht gefundenen Fehler, sowie die Art der gewählten Fehlervisualisierung beschrieben.

Darüber hinaus betrachtete (Howard, 2007) eine Reihe von Werkzeugen für den Umgang mit Spreadsheet, darunter auch einige Spreadsheet-Prüfwerkzeuge. In dieser Studie wird allerdings nur am Rande auf die Visualisierungen der Spreadsheet-Prüfwerkzeuge eingegangen, größtenteils werden die untersuchten Werkzeuge und deren Features betrachtet.

Eine ähnliche Studie von (Berberich, Nguyen, & Vetter, 2012) betrachtet 14 Spreadsheet-Prüfwerkzeuge. Die Autoren beschreiben unter anderem ebenfalls die von den Prüfwerkzeugen gewählten Visualisierungsformen. Die betrachteten Prüfwerkzeuge werden unterteilt in Werkzeuge, welche einen Bericht generieren, und in Werkzeuge, welche Zellen hervorheben, sowie in Werkzeuge, die beides tun. Werkzeuge die sowohl einen Bericht generieren als auch Zellen hervorheben werden als besser eingestuft, als Werkzeuge die entweder einen Bericht generieren oder Zellen hervorheben.

Auf Basis der Studie von (Berberich, Nguyen, & Vetter, 2012) haben (Kulesz & Ostberg, 2013) eine Zusammenfassung der Herausforderungen und Probleme der aktuellen Spreadsheet-Prüfwerkzeuge erstellt. In dieser Zusammenfassung gehen (Kulesz & Ostberg, 2013) insbesondere auch auf die Herausforderungen einer guten Fehlervisualisierung ein, die genannten Aspekte sind in die folgenden Abschnitte mit eingeflossen.

Nach Betrachtung der vier genannten Arbeiten wird ersichtlich, dass sich die Visualisierungsformen aller betrachteten Spreadsheet-Prüfwerkzeuge ähneln und dass diese sich nicht in das Tabellenkalkulationsprogramm integrieren, sondern entweder einen Bericht generieren, oder Inhalte zu dem geprüften Spreadsheet hinzufügen oder modifizieren. Im Folgenden werden die Fehlervisualisierungsansätze der betrachteten Spreadsheet-Prüfwerkzeuge beschrieben.

Das **Generieren eines Berichts** scheint die beliebteste Form der Fehlervisualisierung zu sein. Hierbei kann der Bericht entweder extern als separates Dokument oder intern, als zusätzliches Arbeitsblatt im Spreadsheet angelegt werden. Der externe Bericht ist die aktuelle Art der Fehlervisualisierung des Spreadsheet Inspection Frameworks. Der interne Bericht ist nachteilig, weil dadurch das geprüfte Spreadsheet deutlich geändert

wird. Das geprüfte Spreadsheet kann nicht unmittelbar vom Anwender weiterverwendet werden, da vorher noch der Prüfbericht entfernt werden muss.

Das **Einfärben von Zellen** (oder allgemeiner: das Formatieren von Zellen), zu denen Befunde vorliegen ist eine andere Art der Fehlervisualisierung, auch diese modifiziert das Spreadsheet und verhindert dessen unmittelbare Weiterverwendung. Diese Art der Visualisierung hat den Vorteil, dass der Anwender unmittelbar die Problemzonen des geprüften Spreadsheets sieht, und intuitiv anhand der Farben die schwerwiegendsten Verstöße ermitteln kann. Sollte das Spreadsheet bereits vor der Prüfung Farben verwendet haben, um Zellen hervorzuheben, so ist eine zusätzliche Einfärbung aber störend, da sie die vorhandenen Farbinformationen entfernt. In einem solchen Fall ist das geprüfte Spreadsheet nicht weiterverwendbar, sondern es muss auf eine frühere Version zurückgegangen werden.

Eine weniger aufdringliche Art der Fehlervisualisierung ist das **Einfügen von Kommentaren** in Zellen, zu denen Befunde vorliegen. Dies ist von Nachteil, falls das geprüfte Spreadsheet bereits Kommentare vor der Prüfung enthielt und verhindert ebenfalls die direkte Weiterverwendung des Spreadsheets (die Kommentare müssen entfernt werden, bevor es weitergegeben werden kann).

2.7 Kriterien

In diesem Kapitel wurden Fehlervisualisierungsarten von verwandten Anwendungen besprochen. Diese haben jeweils ihre Vor- und Nachteile. Das Generieren eines Berichts ist mit Abstand die häufigste Visualisierungsform für Fehler in Spreadsheets. Darüber hinaus wurde die in Microsoft Office Excel integrierte Fehlervisualisierung in Abbildung 6 gezeigt, und es wurde die Parallele zur Fehlervisualisierung in Entwicklungsumgebungen auf der Abstraktionsebene des Quellcodes beispielhaft anhand der Entwicklungsumgbung Eclipse aufgezeigt. Im Folgenden werden die Erkenntnisse, welche aus den gezeigten Fehlervisualisierungsmöglichkeiten gezogen wurden, beschrieben und es werden Kriterien entwickelt, welche eine gute Fehlervisualisierung für Endanwender erfüllen sollte.

Eine gute Fehlervisualisierung sollte zwei Ansätze verfolgen. Zum einen sollte sie einen guten Überblick über alle Befunde in Form eines Berichts geben, und zum anderen sollte jeder Befund im Kontext seiner Zelle dargestellt werden. Der Bericht sollte nicht extern vorliegen, sondern in das Tabellenkalkulationsprogramm integriert sein. Darüber hinaus sollte der Bericht gruppiert nach der Fehlerart und sortiert nach der Fehlerschwere angezeigt werden, die genauen Sortiermöglichkeiten können dabei dem Anwender überlassen werden. Zu jedem Befund sollte direkt ein passender Behebungsvorschlag angezeigt werden können.

Die Darstellung der Befunde im Kontext von Zellen, zum Beispiel über Icons, ermöglicht dem Anwender – ebenso wie das Hervorheben von Zellen durch Einfärben – die intuitive Wahrnehmung von Problembereichen im Spreadsheet und auch die intuitive Wahrnehmung des Schweregrads eines Befunds. Dies setzt voraus, dass eine intuitiv verständliche Farbskala für die Darstellung der Befunde verwendet wird, zum Beispiel eine Skala zwischen den Farben gelb und rot. Die Darstellung der Befunde im Kontext von Zellen muss mit bereits eingefärbten Zellen umgehen können, und auch auf starken Hintergrundfarben gut erkennbar sein.

Da insbesondere in umfangreichen Spreadsheets viele Befunde ermittelt werden können, sollten einzelne Befunde ausgeblendet werden können. Dies ermöglicht dem Anwender eine bessere Fokussierung auf die aktuell wichtigen Befunde. Zusätzlich zu den bereits genannten Aspekten sollte die Fehlervisualisierung das Spreadsheet nicht modifizieren. Falls dies technisch nicht realisierbar ist, so sollte das Spreadsheet zumindest nicht irreversibel geändert werden.

3 Visualisierungskonzept

Im folgenden Kapitel wird das Visualisierungs- und Interaktionskonzept beschrieben, welches im Rahmen dieser Arbeit entwickelt wird. Das Ziel der *SIFEI-Visualisierung* ist es, dem Anwender zu helfen, die Befunde zu verstehen und deren Fundstellen im Spreadsheet zu lokalisieren. Im Anschluss wird der Anwender bei der Behebung der Fehler unterstützt.

3.1 Allgemeines

Die Fehlervisualisierung im Spreadsheet enthält einen Bericht, ähnlich zu dem vom Spreadsheet Inspection Framework ausgegebenen HTML-Bericht. Der Bericht im Spreadsheet soll die Befunde nach Prüfregel gruppieren und nach Fehlerschwere sortieren. Er wird in der Seitenleiste des Tabellenkalkulationsprogramms dargestellt (siehe Abbildung 7).

Damit die Fehlervisualisierung nah an den eigentlichen Befunden erfolgt, werden in Zellen, zu denen Befunde vorliegen, kleine Befund-Icons eingeblendet (siehe Abbildung 7), welche zum einen kontextabhängige Informationen liefern und zum anderen eine einfache Navigations- und Zuordnungsmöglichkeit zwischen der betroffenen Zelle und dem zugehörigen Befund in Seitenleiste ermöglichen.

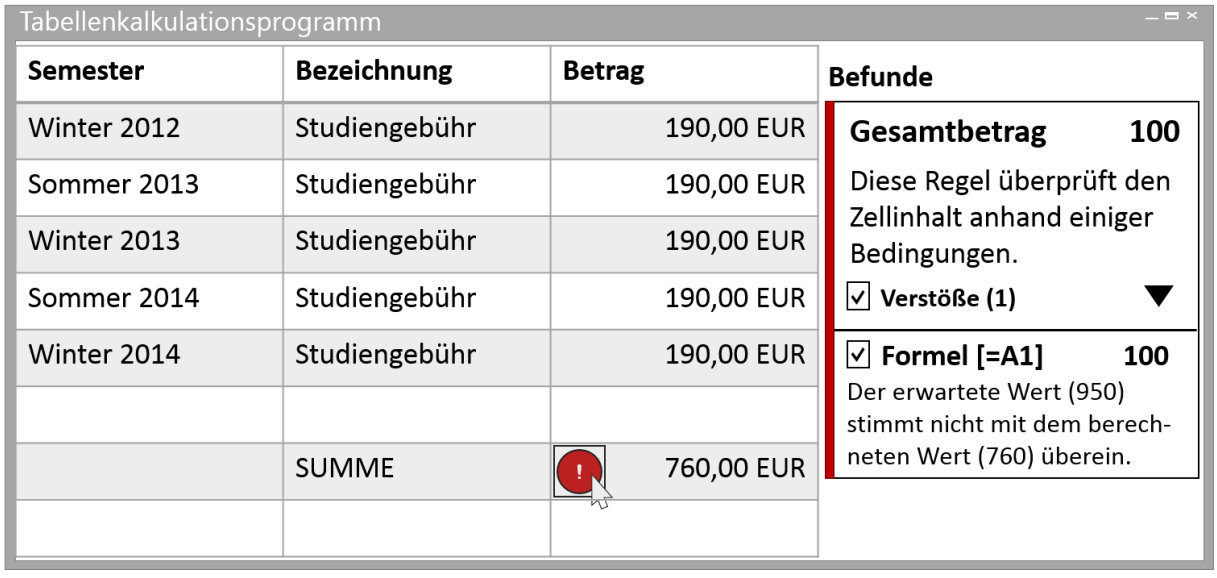

Abbildung 7: Tabellenkalkulationsprogramm mit Befund-Icon und Seitenleiste mit der Prüfregel Gesamtbetrag

Somit hat der Anwender zwei Möglichkeiten, durch die Befunde zu navigieren: Er kann direkt im Spreadsheet in den betroffenen Zellen die Befund-Icons auswählen, oder er kann in der Seitenleiste die gruppierten und sortierten Befunde betrachten.

3.2 Globale Befehle

Die globalen, kontextunabhängigen Befehle, welche jederzeit aufgerufen werden können, werden in das Menü des Tabellenkalkulationsprogramms integriert. Zu diesen Befehlen zählen das Starten von Prüfvorgängen, das Einblenden der Seitenleiste sowie das Zurücksetzen des Dokuments auf den Stand vor der Prüfung.

3.3 Seitenleiste

In der Seitenleiste werden nach ausgeführter Prüfung des aktuellen Spreadsheets alle Prüfregeln sowie die Befunde angezeigt. Die Prüfregeln sind untereinander nach Schwere der aufsummierten Befunde sortiert. Die Befunde werden gruppiert nach den zu Grunde liegenden Prüfregeln und sortiert nach der jeweiligen Schwere innerhalb der zugehörigen Prüfregel angezeigt.

Jede Prüfregel erhält für die aufsummierte Schwere aller ihrer Verstöße ein Farbmapping auf einer Skala zwischen Gelb und Rot. Die Farbskala ist relativ zu sehen, die Prüfregel mit der geringsten Schwere erhält die Farbe Gelb, die Prüfregel mit dem höchsten Schweregrad erhält die Farbe Rot. Diese Skala sollte von allen Menschen intuitiv verstanden werden können. In Abbildung 8 ist beispielhaft die statische Prüfregel *Leserichtung* mit einem einzigen Befund dargestellt, so wie sie in der Seitenleiste angezeigt werden soll.

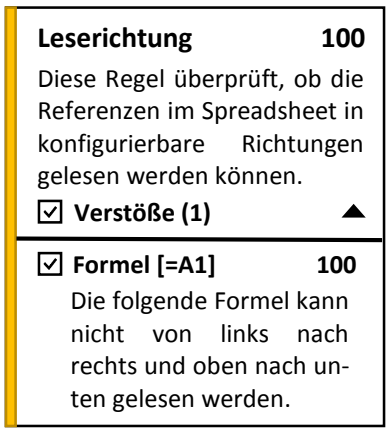

Abbildung 8: Darstellung einer Prüfregel mit Verstoß in der Seitenleiste

Wenn ein Befund in der Seitenleiste ausgewählt wird, so wird das zugehörige Befund-Icon im Spreadsheet ebenfalls ausgewählt und in den sichtbaren Bereich gescrollt. Die kleinen Häkchen links neben den Befunden ermöglichen das Ein- und Ausblenden von Befunden.

3.4 Befund-Icons

Im Spreadsheet selbst werden Befunde in den betroffenen Zellen über ein Befund-Icon dargestellt (siehe Abbildung 9). Die Farbe des Befund-Icons lässt den Rückschluss auf die zugehörige Prüfregel sowie die Schwere des Befunds zu. Bei Auswahl eines Befund-Icons im Spreadsheet öffnet sich die Seitenleiste, und der entsprechende Befund wird ausgewählt.

Abbildung 9: Befund-Icon

Die Darstellung der Befund-Icons hat den Nachteil, dass nur ein Befund pro Zelle angezeigt werden kann. Sobald es mehrere Befunde in derselben Zelle gibt, stößt diese Form der Visualisierung an ihre Grenzen, und es wird zusätzlich in derselben Zelle eine Navigationsmöglichkeit auf der rechten Seite des Befund-Icons eingeblendet (siehe Abbildung 10).

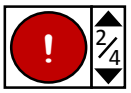

Abbildung 10: Befund-Icon mit zyklischer Navigationsmöglichkeit

Die Navigationsmöglichkeit auf der rechten Seite des Befund-Icons besteht aus zwei kleinen Zahlen, welche den Index des aktuell angezeigten Befunds sowie die Anzahl der insgesamt zu dieser Zelle vorhandenen Befunde angeben. Über und unter diesen Zahlen werden Pfeile eingeblendet, welche eine zyklische Navigation durch alle Befunde in der betroffenen Zelle ermöglichen.

Wenn der Anwender mit der Maus über dem Befund-Icon schwebt, wird ein Hinweis sichtbar, welcher ihn darauf hinweist, dass das Kontextmenü des Befund-Icons weitere Informationen bereithält. Das Kontextmenü (siehe Abbildung 11) des Befund-Icons stellt alle Informationen dar, welche zu dem Befund vorliegen, unter anderem auch einen Lösungsvorschlag.

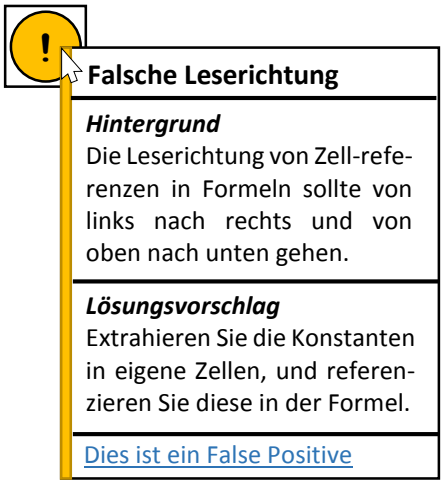

Abbildung 11: Kontextmenü eines Befund-Icons

3.5 Falsch positive Befunde

Die Prüfmechanismen des Spreadsheet Inspection Frameworks sind leider nicht perfekt, weshalb es zu Befunden kommen kann, deren Ursache bei genauer Betrachtung als irrelevant oder sogar als vom Benutzer beabsichtigt einzustufen ist. Solche Befunde werden als falsch positive Befunde bezeichnet.

Es sei an dieser Stelle angemerkt, dass ein Verstoß gegen eine statische Prüfregel jederzeit als falsch positiver Befund definiert werden kann, da es sich bei solchen Verstößen um qualitative Fehler handelt, die aktuell keine numerischen Auswirkungen haben.

Ein Verstoß gegen eine dynamische Prüfregel stellt jedoch einen quantitativen Fehler dar, welcher nur nach reiflicher Überlegung als falsch positiver Befund definiert werden sollte. Es wäre auch denkbar, dem Anwender das Definieren von falsch positiven Befunden nur für qualitative Fehler zu erlauben, und das Definieren für quantitative Fehler zu verbieten, da diese ja vom Anwender selbst verursacht wurden, und entweder ein Fehler in der Definition der dynamischen Prüfregel oder im Spreadsheet selbst vorliegen muss. Um den Anwender nicht zu verwirren, wird ihm die Definition von falsch positiven Befunden sowohl auf Verstößen gegen statische als auch dynamische Prüfregeln erlaubt.

Da jeder Befund als falsch positiver Befund definiert werden kann, wird die Definition von falsch positiven Befunden über ein Kontextmenü sowohl auf den Befund-Icons in der Zelle als auch auf den Verstößen in der Seitenleiste möglich sein. Das Kontextmenü kann für die Definition und das Entfernen eines falsch positiven Befunds verwendet werden. Wenn der Befehl

im Kontextmenü versteckt ist, besteht die Gefahr, dass der Anwender den Befehl nicht findet. Ob dies der Fall ist, wird im Rahmen der Evaluation ermittelt. Sobald ein Befund als falsch positiver Befund definiert wurde, wird dies in der Seitenleiste dargestellt, und das zugehörige Befund-Icon in der Zelle wird ausgeblendet. Dies hat zur Folge, dass die Entfernung eines falsch positiven Befunds in der Seitenleiste erfolgen muss, da das zugehörige Befund-Icon ja nicht mehr vorhanden ist.

4 Implementierung des Prototyps

Dieses Kapitel beschreibt die Umsetzung des in Kapitel 3 vorgestellten Visualisierungskonzepts unter Verwendung eines konkreten Tabellenkalkulationsprogramms. Hierbei werden wichtige Architektur- und Implementierungsdetails berücksichtigt.

Für die Umsetzung des zu entwickelnden Visualisierungsprototyps musste zunächst ein geeignetes Tabellenkalkulationsprogramm ausgewählt werden. Hierfür standen auf Grund ihrer Verbreitung die beiden Anwendungen *Microsoft Office Excel* (Excel) und *LibreOffice Calc* zur Auswahl. Die außerordentlich große Marktdominanz von Excel stellt einen großen Vorteil gegenüber LibreOffice Calc dar. Ein Großteil der Spreadsheet-Benutzer verwendet Excel und ist an dessen Bedienstruktur gewöhnt. Darüber hinaus liegen die meisten Spreadsheets heutzutage als Excel Dateien vor, und werden in anderen Tabellenkalkulationsprogrammen teilweise fehlerhaft dargestellt. Aus diesen Gründen wurde für die Umsetzung des zu entwickelnden Visualisierungsprototyps Excel 2013 ausgewählt.

In Abbildung 12 ist die Umsetzung des Visualisierungsprototyps nach Abschluss der Implementierungsarbeiten zu sehen. Hierbei sind globalen Befehle in der Multifunktionsleiste (Ribbon), die Prüfregeln und Verstöße in der Seitenleiste, sowie die Befund-Icons in den Zellen zu sehen. Es ist zu beachten, dass von den globalen Befehlen, welche in der Multifunktionsleiste sichtbar sind, lediglich die Befehle der Gruppen *Test* und *View* des Abschnitts INSPECTION zur *SIFEI-Visualisierung* gehören.

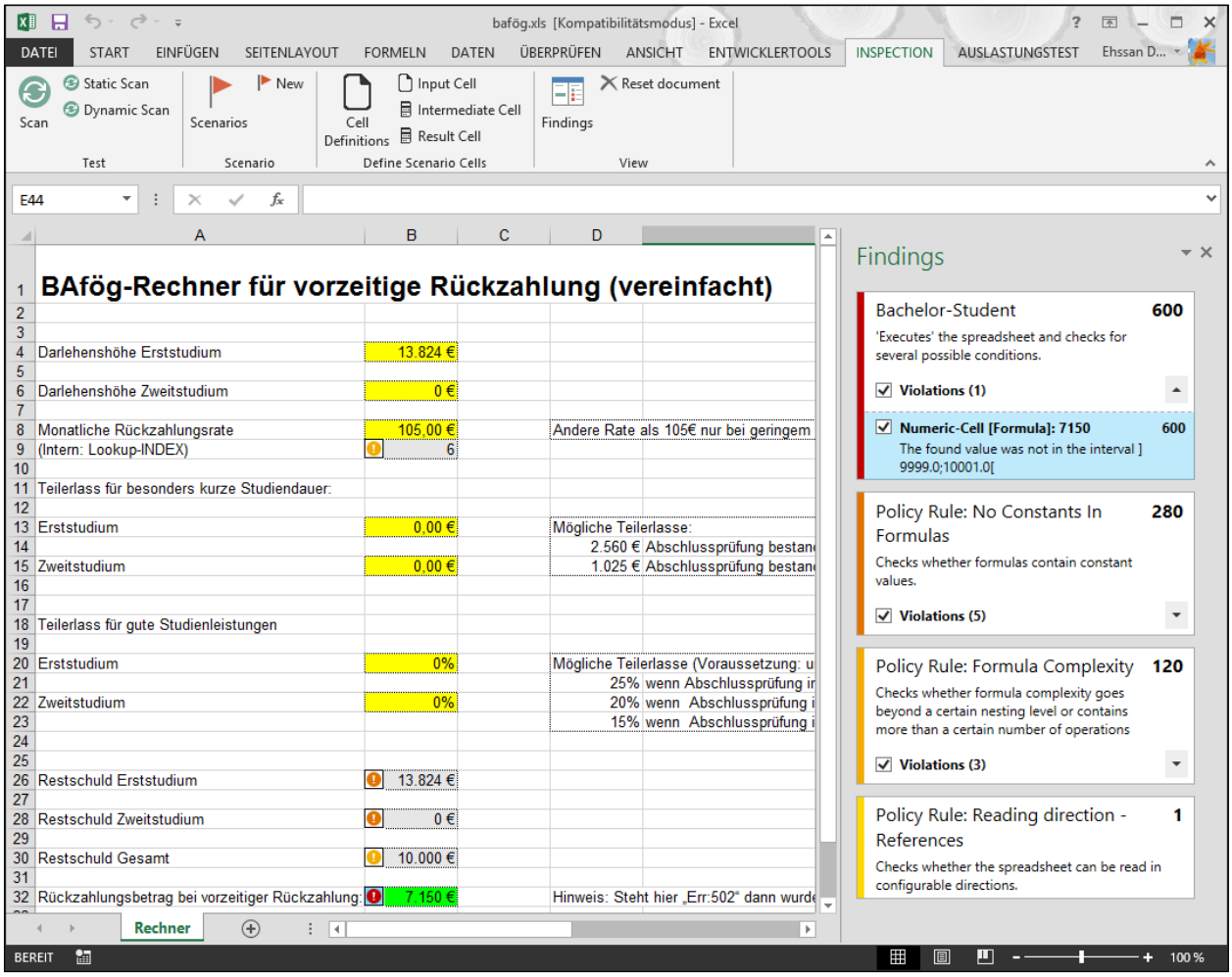

Abbildung 12: Visualisierung der Befunde nach erfolgter Prüfung des Spreadsheets

4.1 Allgemeines

Das zu entwickelnde Excel Add-In ist ein *Add-In auf Anwendungsebene*, welches sich am Lebenszyklus von Excel und nicht am Lebenszyklus von geöffneten Dokumenten orientiert. Dies hat zur Folge, dass das Add-In unmittelbar beim Anwendungsstart von Excel geladen wird, oder sobald es manuell vom Anwender installiert oder aktiviert wird. Von diesem Zeitpunkt an läuft das Add-In kontinuierlich im Hintergrund mit, solange Excel läuft oder es manuell deaktiviert wird, und zwar auch dann wenn keine Dokumente geöffnet sind. Dies hat zur Folge, dass das Add-In für jedes geöffnete Dokument ein eigenes Datenmodell anlegen und verwalten muss, um zum Beispiel die Verstöße zwischenspeichern zu können und um für jede geöffnete Arbeitsmappe eine eigene Seitenleiste mit individuellen Daten anzeigen zu können.

Für die Implementierung des Add-Ins wird das *Microsoft .NET Framework* verwendet. Hierbei kommt die *Windows Presentation Foundation* (WPF) für die Entwicklung der Benutzerschnittstelle zum Einsatz, für die Entwicklung des Programmcodes wird die Programmiersprache *C#* verwendet.

4.2 Multifunktionsleiste

Die Multifunktionsleiste (Ribbon) von Excel wird verwendet, um die globalen, kontextunabhängigen Befehle bereitzustellen. In Abbildung 13 wird der Abschnitt INSPECTION der Multifunktionsleiste dargestellt, welcher die globalen Befehle für *SIFEI* enthält.

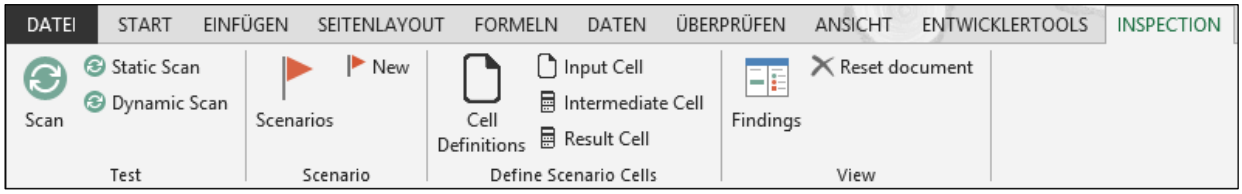

Abbildung 13: Multifunktionsleiste von Excel im Abschnitt INSPECTION mit den Gruppen Test und View

Die folgenden drei Befehle werden im Rahmen der *SIFEI-Visualisierung* bereitgestellt, der Rest der sichtbaren Befehle gehört zur *SIFEI-Szenarienerfassung*.

Scan – Startet einen neuen Prüfvorgang

Findings – Öffnet die Seitenleiste und zeigt die Befunde an

Reset document – Entfernt alle sichtbaren Steuerelemente aus dem Dokument

4.3 Ablauf einer Prüfung

Sobald der Anwender im INSPECTION-Abschnitt der Multifunktionsleiste auf den Scan-Button klickt, wird intern ein neuer *InspectionJob* angelegt und in eine globale Warteschlange eingefügt. Die *InspectionEngine* überwacht die globale Warteschlange und leitet eintreffende InspectionJobs sequenziell an das Spreadsheet Inspection Framework zur Prüfung weiter. Nach erfolgreicher Prüfung durch das Spreadsheet Inspection Framework aktualisiert der InspectionJob das Datenmodell der zugehörigen Arbeitsmappe. Nach Änderung des Datenmodells aktualisiert die Arbeitsmappe die Benutzerschnittstelle und erzeugt gegebenenfalls im Spreadsheet neue Steuerelemente für die empfangenen Befunde.

4.4 Änderungen des Spreadsheet Inspection Frameworks

Für die Übermittlung von Prüfaufträgen an das Spreadsheet Inspection Framework müssen zunächst einige Anpassungen im Spreadsheet Inspection Framework vorgenommen werden, da es in seiner aktuellen Form nicht mit auf Microsoft .NET basierenden Anwendungen kommunizieren kann. Darüber hinaus kann das Spreadsheet Inspection Framework gegenwärtig ausschließlich HTML-Berichte ausgeben, das HTML-Format ist zur Datenübermittlung zwischen Anwendungen aber nur bedingt geeignet.

Zunächst wurde das Spreadsheet Inspection Framework um die Möglichkeit erweitert, anstelle eines HTML-Berichts einen Bericht im XML-Format ausgeben zu können. Im Anschluss wurde das Spreadsheet Inspection Framework als eigenständige Konsolenanwendung lauffähig gemacht, da es bis zum damaligen Zeitpunkt lediglich als Java-Bibliothek in andere Java-Anwendungen eingebunden werden konnte. Als nächstes wurde in der nun entstandenen Java-Anwendung eine Socket-Schnittstelle implementiert, welche Spreadsheets und Prüfregeln empfangen und als Antwort einen Bericht senden kann. Diese Form der Interprozesskommunikation wurde gewählt, weil sie einerseits von allen Computern mit Windows-Betriebssystem unterstützt wird, und andererseits einen geringen Implementierungsaufwand erfordert. Zudem können über Socket-Verbindungen große Datenmengen mit sehr geringem Overhead übertragen werden.

Der Start der Java-Anwendung und die Konfiguration der Socket-Verbindung erfolgt über den in Listing 2 angegebenen Aufruf, es muss lediglich ein lokaler Port übergeben werden.

```
java -jar sif.jar port
```

Listing 2: Kommandozeilenparameter des Spreadsheet Inspection Frameworks nach der Anpassung

Das Excel Add-In stellt einen Socket-Server bereit, welcher auf einem bestimmten Port Verbindungen entgegennimmt. Dieser Socket-Server läuft während der gesamten Ausführungszeit des Excel Add-Ins im Hintergrund mit. Das Excel Add-In wurde als Socket-Server gewählt, da die Lebensdauer des Microsoft Office Excel Add-Ins entscheidend ist, die Funktionalität des Spreadsheet Inspection Frameworks jedoch nur benötigt wird, solange das Microsoft Office Excel Add-In läuft.

Das Spreadsheet Inspection Framework ist der Socket-Client, welcher vom Excel Add-In bei Bedarf gestartet wird und sich direkt mit dem per Kommandozeile übergebenen Port auf dem

lokalen Host verbindet. Sobald die Socket-Verbindung hergestellt ist, kann das Spreadsheet Inspection Framework eine beliebige Anzahl an InspectionJobs abarbeiten.

Die nötige Datenübertragung für die Prüfung eines InspectionJobs, welche über die Socket-Verbindung läuft, wird in Abbildung 14 dargestellt.

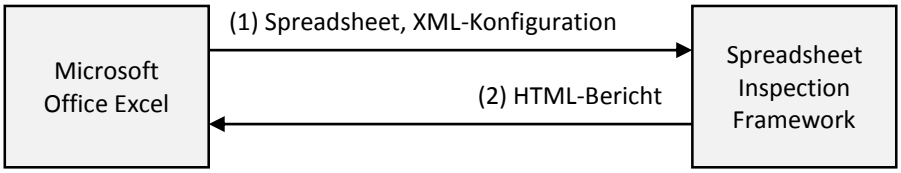

Abbildung 14: Datenübertragung zwischen dem Excel Add-In und dem SIF

Das Spreadsheet Inspection Framework bleibt mit dem Excel Add-In verbunden, bis die Socket-Verbindung vom Excel Add-In geschlossen wird. Sobald die Socket-Verbindung geschlossen wird, beendet sich das Spreadsheet Inspection Framework automatisch.

In dieser Konstellation wird davon ausgegangen, dass sich das Excel Add-In und das Spreadsheet Inspection Framework auf demselben Computer befinden, es wäre jedoch nach kleinen Anpassungen auch möglich, das Excel Add-in und das Spreadsheet Inspection Framework auf separaten, mit dem selben Netzwerk verbundenen Computern auszuführen.

Abgesehen davon wurde das Spreadsheet Inspection Framework mit der neuesten Version des Apache POI (poi.apache.org) aktualisiert, welche für das Lesen von Spreadsheets verwendet wird und in der aktuellen Version Stabilitäts- und Kompatibilitätsverbesserungen, sowie neue Features enthält.

4.5 Abspeichern der falsch positiven Befunde

Der Prototyp unterstützt die Definition und das Entfernen von falsch positiven Befunden. Wenn ein Befund als falsch positiv definiert wird, müssen der Befund und dessen Ort (Zelle) im Spreadsheet eindeutig abgespeichert werden, damit selbst bei einer zwischenzeitlichen Änderung des Spreadsheets der Befund nach einem erneuten Prüfvorgang noch als falsch positiv erkannt wird. Diese Daten werden als benutzerdefinierte XML-Daten im Spreadsheet abgelegt.

Da sich das Spreadsheet nach dem Speichervorgang ändern und die Orte von Befunden verschoben werden können, werden für die betroffenen Zellen für den Anwender unsichtbare

Namen angelegt. Namen sind relative Referenzen auf Zellen, welche von Excel auf dem aktuellen Stand gehalten werden, falls sich der absolute Ort einer Zelle ändert. Somit können falsch positive Befunde auch dann noch korrekt zugeordnet werden, falls sich deren Ort zwischenzeitlich geändert hat.

4.6 Zurücksetzen des Spreadsheets

Die im Spreadsheet eingefügten Steuerelemente werden, falls der Anwender das Spreadsheet speichert, mit abgespeichert und erscheinen beim nächsten Öffnen des Dokuments als unscharfe Grafiken (Artefakte), nicht mehr als interaktive Steuerelemente. Um diese Artefakte entfernen zu können, gibt es den Befehl *Reset document*, welcher das Spreadsheet auf den Stand vor dem Prüfvorgang zurücksetzt. Es ist zu beachten, dass ein Spreadsheet, welches die abgespeicherten Artefakte enthält, nur in einem Microsoft Office Excel mit dem Add-In der *SIFEI-Visualisierung* über den Befehl *Reset document* zurückgesetzt werden kann.

Abgesehen von den Artefakten kann das Spreadsheet benutzerdefinierte XML-Daten durch das Abspeichern von falsch positiven Befunden besitzen, diese können in Excel mit der Dokumentprüfung entfernt werden. In Abbildung 15 ist ein Screenshot der Dokumentprüfung zu sehen, welcher benutzerdefinierte XML-Daten gefunden hat. Diese können über den Button *Alle entfernen* aus dem Spreadsheet entfernt werden.

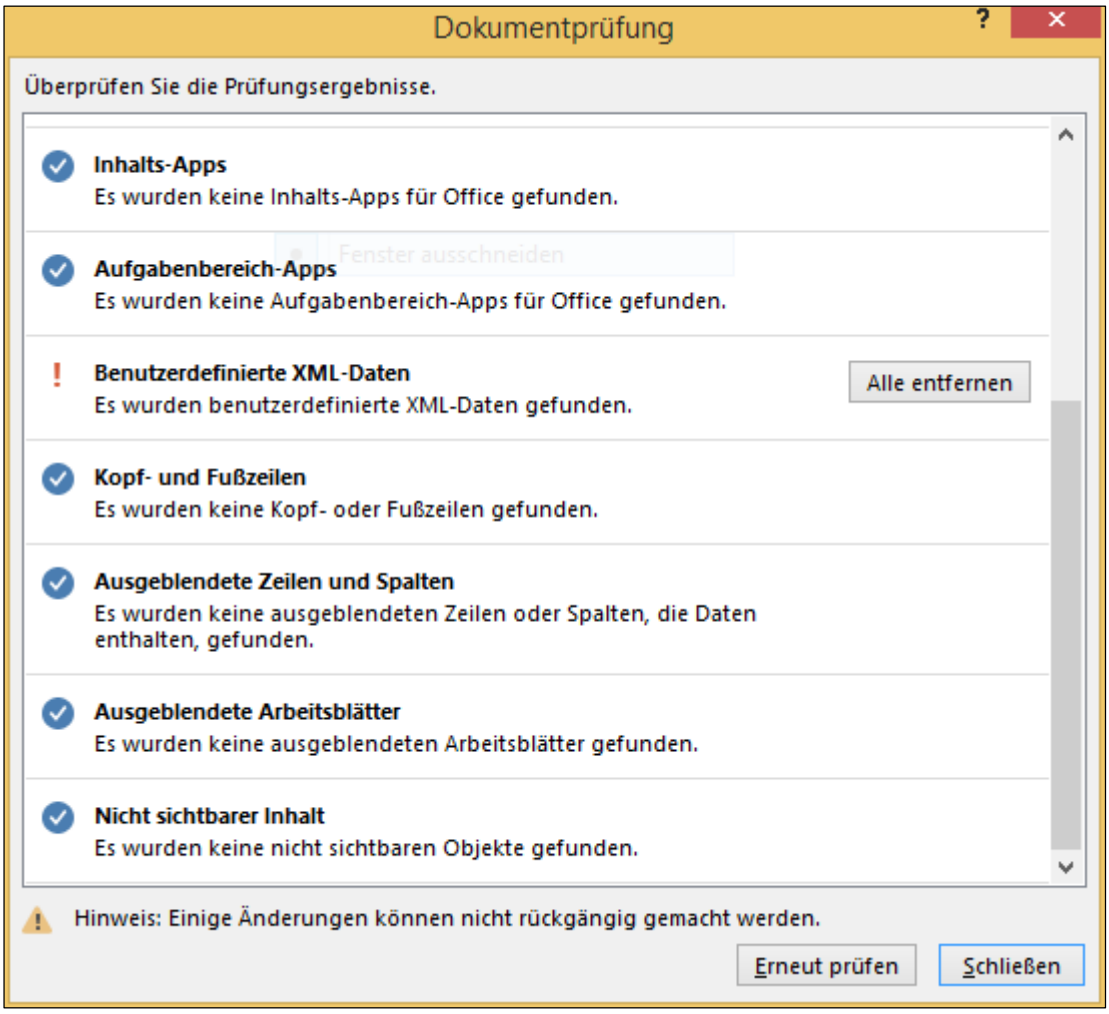

Abbildung 15: Dokumentprüfung in Microsoft Excel

5 Evaluation

Nach Abschluss der Implementierung wurde eine Evaluation mit typischen Excel-Anwendern durchgeführt, um die Intuitivität und generelle Verständlichkeit des entwickelten Prototyps zu prüfen. In diesem Kapitel wird zunächst die durchgeführte Evaluation beschrieben, im Anschluss werden die gewonnen Ergebnisse und Beobachtungen ausgewertet. Da die Evaluation mit sieben Probanden durchgeführt wurde, liegt der Fokus bei der Auswertung der Evaluation eher auf den qualitativen Aspekten als auf quantitativen Auswertungen.

Die Evaluation wurde zunächst im Rahmen einer Pilotstudie an drei Probanden auf die allgemeine Verständlichkeit der Aufgaben hin getestet, im Anschluss folgte dann die Hauptstudie mit vier weiteren Probanden. Da sich die Umgebungsbedingungen und Aufgabenstellungen zwischen der Pilot- und der Hauptstudie nur minimal unterschieden haben (es wurde ein Rechtschreibfehler korrigiert), werden im Folgenden die Probanden der Pilotstudie und die Probanden der Hauptstudie nicht unterschieden.

Mit Hilfe der Evaluation soll herausgefunden werden, ob der umgesetzte Prototyp für unauffällige, in etwa normale Excel-Anwender intuitiv verständlich ist, und ob sich diese in die Struktur der Fehlervisualisierung hineindenken können. Bei den Probanden handelte es sich größtenteils um Studenten aus Ingenieursstudiengängen der Universität Stuttgart. Konkret gab es zwei Probanden aus dem Master-Studiengang *Energietechnik*, einen Probanden aus dem Bachelor-Studiengang *Erneuerbare Energien*, zwei Probanden aus dem Bachelor-Studiengang *Softwaretechnik*, einen Probanden aus dem Master-Studiengang *Maschinenbau* sowie einen Probanden aus dem Diplom-Studiengang *Luft- und Raumfahrttechnik*. Es wurden bewusst nur zwei Softwaretechnik-Studenten als Probanden verwendet, da spekuliert wurde, dass diese Studenten sich einfacher in die *SIFEI-Visualisierung* hineindenken können und deshalb einen Vorteil hätten.

5.1 Versuchsaufbau

Die Evaluation wurde in Kooperation mit Jonas Scheurich durchgeführt. Sie bestand aus zwei Teilen. Im ersten Teil ging es um den in dieser Arbeit entwickelten Prototypen zur *Fehlervisualisierung in Spreadsheets*, im zweiten Teil ging es um den von Jonas Scheurich entwickelten Prototypen zur *Benutzerschnittstelle für einen Spreadsheetprüfstand* (Scheurich, 2013). Beide Teile enthielten unterschiedliche Aufgabenstellungen und waren inhaltlich entkoppelt.

Für die Durchführung der Studie wurde den Probanden ein Fragebogen mit sechs Aufgaben vorgelegt, den diese unter Verwendung der *SIFEI-Visualisierung* und Excel ausfüllen sollten. Die ersten drei Aufgaben bezogen sich inhaltlich auf die *SIFEI-Visualisierung*. Zusätzlich stand den Probanden zu jeder Aufgabe ein Blatt mit einer kleinen Hilfestellung (Hilfeblatt) zur Verfügung, welches bei Problemen verwendet werden durfte. Das Hilfeblatt beinhaltete jeweils einen Lösungshinweis, welcher sich auf den schwierigsten Teil der gestellten Aufgabe bezog. Darüber hinaus durften bei ernsthaften Problemen jederzeit Fragen gestellt werden, allerdings nur nach vorheriger Konsultation des Hilfeblatts.

Für die Aufgaben wurde ein einfaches und für Studenten interessantes Spreadsheet verwendet, welches in Abbildung 16 dargestellt ist. Es handelte sich um einen BAföG-Rückzahlungsrechner, welcher die Höhe des nach Studienende zurückzuzahlenden BAföG-Betrags errechnet. Das Spreadsheet wurde bereits für frühere Evaluationen und Tests des Spreadsheet Inspection Frameworks verwendet. Es wurde darauf geachtet, dass keiner der Probanden bereits an einer früheren Evaluation des Spreadsheet Inspection Framework teilgenommen hatte.

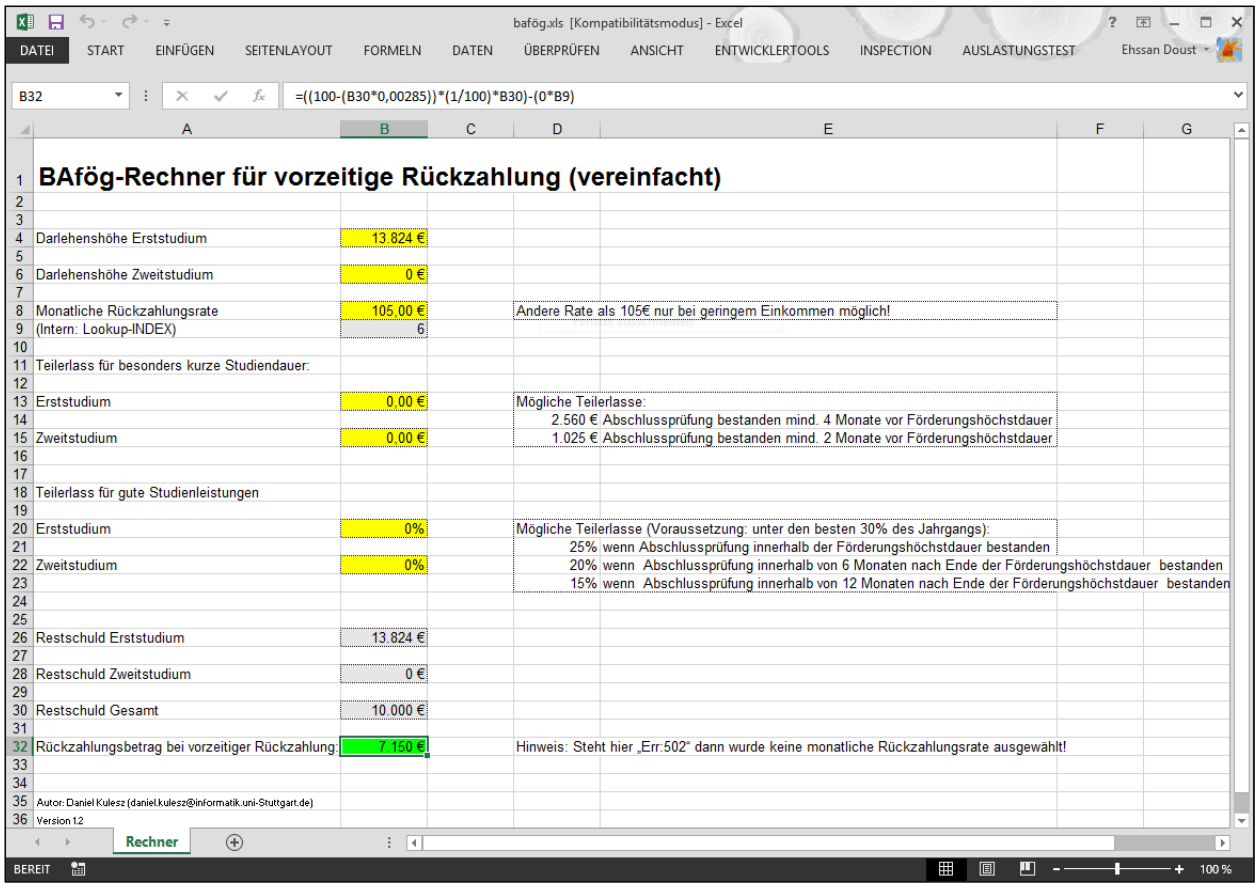

Abbildung 16: Testdokument für die Evaluation

Vor dem Start der eigentlichen Aufgaben wurde den Probanden eine kleine Übersicht mit je einem bis zwei Sätze zur Multifunktionsleiste, zur Seitenleiste und zu den Befund-Icons gegeben. Diese Übersicht beinhaltete allgemeine Informationen zur SIFEI-Visualisierung.

Nachfolgend werden die drei gestellten Aufgaben für den ersten Teil der Evaluation beschrieben. In Aufgabe 1 waren einige einfache Fragestellungen zu beantworten, welche das Herauslesen von Informationen aus der Fehlervisualisierung erforderten. Es handelte sich hauptsächlich um Multiple-Choice-Aufgaben, unter anderem sollte die Anzahl geprüfter Regeln, die Zelle mit dem schwersten Befund, die am häufigsten verletzte Regel und die am schwersten verletzte Regel genannt werden.

Die zweite Aufgabe führte das Konzept der falsch positiven Befunde ein. Der Proband sollte Befunde ein- und ausblenden, einige falsch positive Befunde definieren, sowie die Definition als falsch positiver Befund für einen Befund wieder rückgängig machen.

In der letzten Aufgabe sollte der Proband erweiterte Informationen zu einem Fehler sowie einen möglichen Lösungsvorschlag zu einem bestimmten Befund abrufen. Der Lösungsvorschlag ließ sich ausschließlich über das Kontextmenü des zugehörigen Befund-Icons im Spreadsheet anzeigen.

5.2 Beschreibung des Ablaufs

Die Probanden hatten allgemein kleinere bis mittelschwere Probleme mit der Bedienung des Prototyps. Die genauen Problembereiche der einzelnen Probanden waren breit gestreut, einige Probanden hatten Probleme mit Dingen, die andere Probanden auf Anhieb verstanden.

In der ersten Aufgabe wurde die Anzahl der geprüften Regeln von den meisten Probanden korrekt erfasst, lediglich ein Proband konnte die richtige Anzahl nicht feststellen. Die nächste Teilaufgabe, in der die konkrete Anzahl an Verstößen gegen die einzelnen Prüfregeln gefragt war, konnten fünf von sieben Probanden fehlerfrei lösen können.

Das Auslesen der korrekten Anzahl an Befunden in einer Zelle mit drei Befunden gelang leider keinem der Probanden, da jeweils nur ein Befund-Icon sichtbar war. Dieses Befund-Icon hatte keine Navigationsmöglichkeit. Die einzige Möglichkeit, alle Befunde in der Zelle (nacheinander) zu sehen wäre das Ausblenden der jeweils anderen Befunde gewesen. Hier wäre eine Implementierung der zyklischen Navigationsmöglichkeit (vergleiche Kapitel 3) sicherlich hilfreich gewesen.

Sehr gut abgeschnitten haben alle Probanden beim Nennen der am schwersten verletzten Prüfregel, des schwersten Einzelverstoßes und dessen Zelle sowie der am häufigsten verletzten Prüfregel. Hier gab es keinen einzigen Probanden mit Schwierigkeiten, dies liegt vermutlich an der intensiven roten Hervorhebung der betreffenden Befunde.

Die zweite Aufgabe war für die Probanden einfacher lösbar als die erste Aufgabe, das Ein- und Ausblenden von Befunden war für fast alle Probanden intuitiv möglich. Die Definition von falsch positiven Befunden war fünf von sieben Probanden ohne Hilfeblatt möglich, dennoch haben die Probanden unterschiedlich lange nach dem Kontextmenü, über welches die Definition als falsch positiver Befund möglich ist, suchen müssen.

Die dritte Aufgabe fiel den Probanden insgesamt am schwersten, vier von sieben Probanden mussten hier das Hilfeblatt zu Rate ziehen. Das Kontextmenü auf den Befund-Icons war trotz des unterstützenden Tooltips nicht intuitiv auffindbar, jedoch war der anschließend im Kontextmenü angezeigte Lösungshinweis für alle Probanden verständlich. Es kam sogar vor, dass Probanden den Tooltip des Befund-Icons gelesen haben (der Tooltip weist auf das Kontextmenü auf dem Befund-Icon hin), aber dennoch das Kontextmenü nicht gefunden haben.

5.3 Erkenntnisse

Nach Abschluss der Evaluation ist aufgefallen, dass es zwei Probleme mit dem Prototypen gibt. Das erste Problem ist, dass kein Proband die korrekte Anzahl vorhandener Befunde aus einer Zelle mit drei Befunden auslesen konnte. Dieses Problem kann durch eine Implementierung der zyklischen Navigationsmöglichkeit (siehe Kapitel 3) behoben werden.

Das zweite Problem ist, dass vier von sieben Probanden Probleme mit der intuitiven Benutzung der Befund-Icons hatten, insbesondere ist den Probanden das Kontextmenü der Befund-Icons erst nach Konsultation des Hilfeblatts aufgefallen. Nachdem die Probanden das Hilfeblatt zu Rate gezogen hatten, konnten sie das Kontextmenü finden und so unter anderem auch den Behebungsvorschlag zum Befund anzeigen lassen. Es ist davon auszugehen, dass die Anwender im Allgemeinen lediglich einmal auf die Existenz dieses Kontextmenüs hingewiesen werden müssen.

Abgesehen von diesen beiden Problemen hat sich der Prototyp im Praxistest bewähren können. Die Probanden konnten den Großteil der Aufgaben selbstständig und zielgerichtet lösen. Die intuitive Verständlichkeit des Prototyps kann stellenweise sicherlich noch verbessert werden, nach einer kurzen Einweisung in die Verwendung sollten sich die meisten Anwender jedoch zurechtfinden können.

5.4 Gültigkeit der Ergebnisse

Die Studie wurde mit lediglich sieben Probanden durchgeführt, welche zudem alle ein ähnliches Studienfach belegten. Die gewonnenen Erkenntnisse können somit nicht für eine Mehrheit der Excel-Anwender verallgemeinert werden.

6 Zusammenfassung und Ausblick

Trotz existierender Spreadsheet-Prüfwerkzeuge bleiben Spreadsheet-Fehler weiterhin ein großes Problem, dessen Bedeutung auch in jüngster Vergangenheit nicht an Relevanz verliert, wie die finanziellen Verluste, welche durch fehlerhafte Spreadsheets verursacht werden, belegen. Die Fehlersuche in Spreadsheets und die Visualisierung von Spreadsheet-Strukturen ist ein reges Forschungsgebiet (vgl. Kapitel 2). Dennoch existieren aktuell keine Arbeiten, welche sich dediziert mit der Fehlervisualisierung in Spreadsheets befassen.

Im Verlauf dieser Arbeit wurde ein Konzept zur Fehlervisualisierung in Spreadsheets erstellt, welches auf den in Kapitel 2.7 erarbeiteten Kriterien basiert. Das Konzept umfasst die Darstellung der Befunde in einer sortierten und gruppierten Liste in der Seitenleiste des Tabellenkalkulationsprogramms, und die Darstellung der einzelnen Befunde durch Befund-Icons direkt in den verursachenden Zellen. Jede Prüfregel erhält ein Farbmapping passend zu der aufsummierten Schwere aller zugehörigen Befunde, um intuitiv die schwersten Befunde (und deren Befund-Icons) zuordnen zu können. Darüber hinaus können einzelne Befunde als falsch positiv definiert werden.

Das erstellte Konzept wurde prototypisch für Microsoft Office Excel 2013 implementiert, alle wichtigen Funktionen des Konzepts mit Ausnahme der zyklischen Navigationsmöglichkeit zwischen mehreren Befund-Icons in derselben Zelle wurden umgesetzt.

Im Gegensatz zu den in Kapitel 2 betrachteten, aktuell auf dem Markt vertretenen Prüfwerkzeugen setzt die *SIFEI-Visualisierung* als einzige die interaktiven und kontextabhängigen Fehlervisualisierungen (in Form der Seitenleiste und mit Hilfe von Befund-Icons) um. Die bereits existierenden Prüfwerkzeuge generieren im Gegensatz dazu größtenteils statische Berichte und verwenden statische Zellhervorhebungen (zum Beispiel Kommentare, oder Einfärben von Zellen) als kontextabhängige, jedoch nicht interaktive Fehlervisualisierung.

Die anschließende Evaluation des Prototyps hat gezeigt, dass das Konzept vielversprechend ist, es allerdings noch Probleme in Hinsicht auf die intuitive Verständlichkeit der zugrunde

liegenden Konzepte (Prüfregel, Befund, Fehler) gibt. Hier müssten die Anwender gegebenenfalls vor Verwendung der *SIFEI-Visualisierung* in das Thema Spreadsheet-Fehlersuche und die Bedienung der *SIFEI-Visualisierung* eingeführt werden, außerdem sollte die *SIFEI-Visualisierung* in Hinsicht auf ihre Benutzerfreundlichkeit verbessert werden.

Abgesehen davon konnte sich der Prototyp der *SIFEI-Visualisierung* im Praxistest bewähren, nach einer Eingewöhnungsphase war den Anwendern die Benutzung des Prototyps bekannt und dieser konnte effektiv eingesetzt werden.

6.1 Ausblick

Für die Zukunft ist es wünschenswert, die Aspekte des Prototyps, welche von einem Großteil der Probanden nicht intuitiv verstanden wurden, einfacher umzusetzen. So sollte beispielsweise der Lösungsvorschlag für einen Befund nicht ausschließlich im Kontextmenü des Befund-Icons, sondern ebenfalls im Kontextmenü des Befunds in der Seitenleiste dargestellt werden. Zudem wäre eine deutsche Variante der Benutzerschnittstelle sinnvoll.

Die zyklische Navigationsmöglichkeit zwischen mehreren Befund-Icons würde das Problem der Visualisierung von mehreren Befunden in derselben Zelle vermutlich lösen. Eine weitere Überlegung wäre, eine grundlegende Unterscheidung der Befund-Icons für quantitative Fehler und qualitative Fehler einzuführen, zum Beispiel durch eine andere geometrische Basisform des Befund-Icons (gegenwärtig haben alle Befund-Icons als geometrische Basisform einen Kreis). Nach diesen Änderungen sollte dann gegebenenfalls eine weitere Evaluation durchgeführt werden.

Zum aktuellen Zeitpunkt laufen weitere Bachelorarbeiten an der Universität Stuttgart, welche sich mit der Erweiterung des Spreadsheet Inspection Frameworks beschäftigen. Es ist zu erwarten, dass sich somit auch neue Anforderungen an die Fehlervisualisierung der Befunde ergeben, welche eventuell im Rahmen einer zukünftigen Bachelorarbeit umgesetzt werden.

7 Literaturverzeichnis

Berberich, T., Nguyen, A. B., & Vetter, M. (2012). *Audit-Werkzeuge für Spreadsheets.* Fachstudie, University of Stuttgart, DE.

Davis, J. S. (1996). Tools for spreadsheet auditing. *International Journal of Human-Computer Studies*(45), S. 429-442.

European Spreadsheet Risks Interest Group. (2013). *Spreadsheet mistakes*. Abgerufen am 12. November 2013 von http://www.eusprig.org/horror-stories.htm

Hermans, F. F. (2012). *Analyzing and visualizing Spreadsheets.* Doktorarbeit, Delft University of Technology, NL.

Howard, P. (2007). *Spreadsheet Management.* Northhamptonshire, UK: Bloor Research.

Kulesz, D., & Ostberg, J.-P. (2013). *Practical Challenges with Spreadsheet Auditing Tools.* University of Stuttgart, DE.

Lemcke, M. (2013). *Dynamische Prüfung von Spreadsheets.* Diplomarbeit, University of Stuttgart, DE.

Nixon, D., & O'Hara, M. (2010). *Spreadsheet Auditing Software.* University of Salford, Information Systems Institute, UK.

Panko, R. (2006). Facing the Problem of Spreadsheet Errors. *Decision Line*(37.5), S. 8-10.

Panko, R., & Halverson Jr., R. P. (1996). Spreadsheets on Trial: A Survey of Research on Spreadsheet Risks. *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*, (S. 326-335). Hawaii.

Scheurich, J. (2013). *Benutzerschnittstelle für einen Spreadsheet-Prüfstand.* Bachelorarbeit (noch nicht fertiggestellt), University of Stuttgart, DE.

Slinger, S. (2005). *Code Smell Detection in Eclipse.* Thesis, Delft University of Technology, Department of Software Technology, NL.

Zitzelsberger, S. (2012). *Fehlererkennung in Spreadsheets*. Diplomarbeit, University of Stuttgart, DE.

Erklärung

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

_____

(Ehssan Doust)