Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 322

Reengineering und Erweiterung einer Prüfungsumgebung für Spreadsheets

Nicolas Mauch

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Stefan Wagner
Betreuer/in:	M.Sc. Daniel Kulesz
Beginn am:	12. April 2016
Beendet am:	11. Oktober 2016
CR-Nummer:	C.4, D.2.3, D.2.4, D.2.5, D.2.7, D.2.8, J.1, K.8.1

Kurzfassung

Spreadsheets werden in vielen Bereichen eingesetzt. Obwohl sie wichtige Daten enthalten und Entscheidungen auf diesen Daten basieren, befinden sich in vielen Spreadsheets Fehler. Daher wurde am Institut für Softwaretechnologie der Universität Stuttgart das Werkzeug Spreadsheet Inspection Framework entwickelt, um Spreadsheets halbautomatisch auf Fehler zu überprüfen. Es führt statische und dynamische Prüfungen eines Spreadsheets durch und zeigt gefundene Befunde an. Es besteht aus zwei Komponenten, einem Excel-Plugin zur Verwaltung von Prüfungen und Anzeigen von Resultaten sowie einer Java-Anwendung, welche die Prüfungen durchführt.
In dieser Arbeit sollte das Excel-Plugin bezüglich folgender Aspekte erweitert und verbessert werden: Usability, Wartbarkeit, Zuverlässigkeit und Korrektheit. Um Schwächen im Excel-Plugin zu finden, musste es genauer untersucht werden. Hierzu wurden zehn Bewertungsverfahren in Erwägung gezogen und sechs davon angewendet. So wurden die Usability mittels Usability-Patterns und einer Nutzerstudie, die Wartbarkeit mit Teamscale, die Zuverlässigkeit mit Lasttests und die Korrektheit mit Unit-Tests und einem Systemtest überprüft. Da mehr Schwächen gefunden wurden als im Rahmen der Bachelorarbeit behoben werden konnten, mussten diese priorisiert werden. Für zwölf der dreizehn ausgewählten Schwächen konnten Änderungen implementiert werden, die zu Verbesserungen führen sollten. Da das Excel-Plugin vor Beginn der Bachelorarbeit korrekt arbeitete, wurden dahingehend keine Änderungen durchgeführt.
Nachdem die ausgewählten Änderungen durchgeführt wurden, wurde das Excel-Plugin erneut bewertet. Hier wurden die zuvor verwendeten Bewertungsverfahren erneut angewandt. Außerdem wurde eine Studie mit vier Probanden durchgeführt, um das Spreadsheet Inspection Framework aus Sicht der Anwender zu bewerten. Bei dieser mussten die Probanden, mit Hilfe des Excel-Plugins in bestehenden Spreadsheets Fehler finden und korrigieren. Die erneute Bewertung der Zuverlässigkeit ergab, dass sie gestiegen ist aber in Zukunft noch weiter verbessert werden sollte. Eine erneute Bewertung der Wartbarkeit ergab, dass 30% der zuvor gefundenen Verstöße die auf eine schlechte Wartbarkeit hinweisen, gelöst wurden. Es konnten zwei neue Usability-Patterns umgesetzt werden und die Probanden der Studie bewerteten die Usability als gut.
Zusammenfassen lässt sich sagen, dass das verbesserte Excel-Plugin geeignet für den produktiven Einsatz ist.

Abstract

Nowadays, spreadsheets are used in multiple fields. They contain important data and important decisions are made based on this data. However many spreadsheets contain mistakes. The Institute of Software Technology from the University of Stuttgart developed the tool Spreadsheet Inspection Framework. This tool scans spreadsheets semi-automatically for mistakes. It executes static and dynamic scans and displays the results of these. The tool is composed of two components; an excel plug-in used to manage scans and display the results of these scans, as well as a Java-application which executes the scans.
The goal of this thesis was to extend and reengineer the plug-in regarding usability, maintainability, reliability and correctness. To find weaknesses in the plug-in, it needed to be assessed first. For this purpose ten procedures were considered. Six of these were used to actually evaluate the plug-in. The usability was assessed via usability patterns and a user study, the maintainability via Teamscale, the reliability via load tests and the correctness via unit tests and a system test. Since there were more weaknesses found than could have been fixed during this thesis, the weaknesses needed to be prioritized. The plan was to improve thirteen weaknesses out of which for twelve changes could be made. Since the plug-in worked correctly before the start of the thesis, no changes were made regarding correctness.
After implementing the improvements, the plug-in was evaluated again. The reliability got increased, but further improvements are required. Also more then 20% of priorly found maintainability issues could be fixed. An extra user study with four participants was performed to better assess the usability. In this study, the participants needed to find and fix mistakes in a spreadsheet using the plug-in. The participants evaluated the usability positively . Furthermore two new usability patterns could be implemented.
In summary, it can be said that the improved plug-in is fit for productive use.

Inhaltsverzeichnis

1. Einleitung **11**
- 1.1. Motivation . . . 11
- 1.2. Ziel . . . 11
- 1.3. Aufbau der Bachelorarbeit . . . 12

2. Hintergrund **13**
- 2.1. SIFCore und SIFEI . . . 13
- 2.2. Entwicklung . . . 14

3. Bewertungsverfahren **15**
- 3.1. Usability . . . 15
- 3.2. Wartbarkeit . . . 17
- 3.3. Zuverlässigkeit . . . 19
- 3.4. Korrektheit . . . 20

4. Ergebnisse der Bewertung **23**
- 4.1. Usability . . . 23
- 4.2. Wartbarkeit . . . 23
- 4.3. Zuverlässigkeit . . . 24
- 4.4. Korrektheit . . . 25

5. Priorisierung **27**
- 5.1. Überblick über verschiedene Priorisierungsarten . . . 27
- 5.2. Verwendete Priorisierung . . . 28

6. Vorgenommene Änderungen und Ergänzungen **29**
- 6.1. Änderungen an der graphischen Benutzeroberfläche . . . 29
- 6.2. Änderungen am Code . . . 34
- 6.3. Hinzugefügte Funktionen . . . 35

7. Benutzerstudie **39**
- 7.1. Einleitung . . . 39
- 7.2. Hintergrund . . . 40
- 7.3. Vorbereitung . . . 40
- 7.4. Durchführung . . . 42

7.5. Analyse der Ergebnisse 43
7.6. Diskussion 44

8. Erneute Bewertung von SIFEI **45**
8.1. Vergleich der Resultate 45

9. Zusammenfassung und Ausblick **47**
9.1. Zusammenfassung 47
9.2. Ausblick 48

A. Schwächen von SIFEI **53**
A.1. Im Issue Tracker vorher eingetragene Schwächen 53
A.2. Neu gefundene Schwächen 55

Abbildungsverzeichnis

2.1. Architektur des Spreadsheet Inspection Frameworks 13

6.1. Rückmeldung bei Prüfungen 30
6.2. Änderungen des Szenarieneditors 31
6.3. Varianten des alten Kontextmenüs 33
6.4. Geändertes Kontextmenü 33
6.5. Entwicklung der Methodenlänge 35
6.6. Befund-Icon mit verschiedenen Befundentypen 36

7.1. Befunde in der ersten Aufgabe 42
7.2. Fragen bezüglich der Usability 43
7.3. Bewertung der Zuverlässigkeit durch Probanden 44

8.1. Entwicklung der Wartbarkeit gemessen von Teamscale 46

A.1. Teamscale-Bewertung von SIFEI vor Beginn der Bachelorarbeit 57

1. Einleitung

1.1. Motivation

Spreadsheets werden in vielen Unternehmen in unterschiedlichsten Einsatzbereichen verwendet und herangezogen, um Entscheidungen zu fällen [eus]. Ein Großteil der verwendeten Spreadsheets enthält Fehler [But00, Pan98]. Wie viele Fehler in einem Spreadsheet vorhanden sind und wie schwerwiegend diese sind, lässt sich nicht einheitlich sagen. Die Ergebnisse variieren je nach Studie. Man kann aber davon ausgehen, dass zwischen ein und zwei Prozent der Zellen in Spreadsheets Fehler enthalten. [Pan08].
Diese Fehler wirken sich unter Umständen auf Entscheidungen aus, die für Unternehmen weitreichende Folgen haben können [Pan98]. Aus diesem Grund sollten Spreadsheets regelmäßig auf Fehler analysiert werden. Um dies zu bewerkstelligen, können Spreadsheets durch Software analysiert werden.
An der Universität Stuttgart forscht das Institut für Softwaretechnologie an der halbautomatischen Prüfung von Spreadsheets. Hierzu wurde das Spreadsheet Inspection Framework (SIF) entwickelt. Das Werkzeug besteht aus zwei Komponenten. SIFCore prüft ein Spreadsheet auf die Einhaltung von statischen und dynamischen Regeln. Spreadsheet Inspection Framework Excel Integration (SIFEI) ist ein Plugin für Microsoft Excel. Es visualisiert die von SIFCore erstellten Daten im zugehörigen Spreadsheet. Das Excel Plugin SIFEI war für den produktiven Einsatz in der Praxis bisher nicht geeignet. Zu oft arbeitete SIF nicht zuverlässig oder es kam zu unbehandelten Ausnahmen. Außerdem waren manche Funktionen nicht intuitiv zu bedienen. Deshalb wurden im Rahmen dieser Bachelorarbeit systematische Bewertungsverfahren gesucht. Mit diesen sollten in SIFEI enthaltene Fehler gefunden werden, um sie dann gegebenenfalls zu beheben.

1.2. Ziel

SIFEI soll in der Praxis von Nutzern problemlos verwenden werden können. Hierfür soll SIFEI zuverlässig und korrekt arbeiten. Außerdem soll es einfach zu bedienen sein und den Arbeitsfluss des Nutzers kaum einschränken. Das Ziel dieser Bachelorarbeit ist es, SIFEI dahingehend zu erweitern und zu verbessern.

Außerdem soll durch eine erneute Bewertung überprüft werden, ob diese Änderungen zu Verbesserungen geführt haben.

1.3. Aufbau der Bachelorarbeit

Nach einer kleinen Einführung in Spreadsheetprüfungen im Kapitel 1, wird im Kapitel 2 ein Überblick über SIFEI und SIFCore gegeben. Daraufhin werden in Kapitel 3 verschiedene Bewertungsverfahren vorgestellt und kritisiert. Einige dieser Bewertungsverfahren wurden angewendet, um SIFEI zu evaluiert. Die Ergebnisse dieser Bewertung sind im Kapitel 4 zusammengefasst. Wie diese Ergebnisse dieser Bewertung priorisiert wurden, findet sich in Kapitel 5. Im Kapitel 6 werden vorgenommene Änderungen vorgestellt und im Kapitel 7 eine Benutzerstudie beschrieben. Diese wurde durchgeführt, um die Änderungen von SIFEI besser evaluieren zu können. Die Ergebnisse dieser erneuten Bewertung werden im Kapitel 8 vorgestellt. Kapitel 9 enthält die Zusammenfassung und den Ausblick für künftige Arbeiten. Im Anhang A finden sich eine Auflistung aller gefundenen Schwächen, die in SIFEI enthalten sind und wie diese priorisiert wurden.

2. Hintergrund

Das Spreadsheet Inspection Framwork besteht aus den zwei Komponenten SIFEI und SIFCore. Diese beiden Komponenten werden im folgenden Kapitel kurz vorgestellt.

2.1. SIFCore und SIFEI

SIFCore wurde entwickelt, um Spreadsheets auf Mängel zu untersuchen. Dabei wurden Konzepte aus der Softwareprüfung verwendet. Eines dieser Konzepte sind statische Prüfungen, die das Spreadsheet auf gewisse Regeln prüfen. Hierzu gehören z.B. *"Konstanten in Formeln"*, *"Komplexe Formeln"* und *"Konstanten in Formeln"*. Außerdem werden dynamische Prüfungen unterstützt, die für jedes Spreadsheet separat definiert werden müssen [Zit12].
SIFCore erhält ein Spreadsheet, zu prüfende Regeln und definierte dynamischen Prüfungen. Danach führt SIFCore die Prüfung durch und speichert einen Bericht des Spreadsheets im XML-Format [Sch14].
SIFEI bietet die Möglichkeit zukünftige Prüfungen zu konfigurieren und zu starten. Nachdem eine Prüfung gestartet wird, schickt SIFEI die Daten über eine Socket-Schnittstelle an SIFCore [Sch14]. Dann kann es den von SIFCore erstellen Bericht einlesen und die gefundenen Befunde visualisieren.

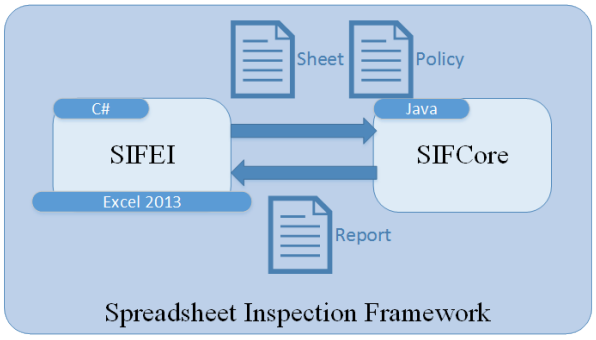


Abbildung 2.1.: Architektur des Spreadsheet Inspection Frameworks

2.2. Entwicklung

Das Spreadsheet Inspection Framework wurde an der Universität Stuttgart am Institut für Softwaretechnologie von Studenten im Rahmen von Diplom- und Bachelorarbeiten entwickelt. Betreut wurden die Arbeiten von Daniel Kulesz.
Zuerst wurde die Java-Anwendung SIFCore entwickelt. In der ersten Version konnte SIFCore nur statische Prüfungen durchführen. Von Zitselsberger wurden die drei Regeln *Konstanten in Formeln, Leserichtung* und *Formelkomplexität* umgesetzt [Zit12]. Anschließend erweiterte Lemcke SIFCore so, dass dynamische Prüfungen möglich waren [Lem13].
Danach wurde die Entwicklung des Excel-Plugins SIFEI begonnen. Doust entwickelte das Anzeigen von Befunden in Excel-Zellen [Dou13]. Gleichzeitig wurde die Konfiguration und Ausführung von dynamischen Prüfungen von Scheurich in SIFEI integriert [Sch14]. Anschließend wurden weitere statische Prüfmethoden von Beck integriert [Bec14] und Echtzeit-Prüfungen der Spreadsheets von Toth ermöglicht [Tot14].

3. Bewertungsverfahren

Um vorhandene Bugs, fehlende Funktionen oder ähnliches zu finden, musste SIFEI genauer untersucht werden. Es wurden dabei vier Aspekte genauer betrachtet. Diese Aspekte, Usability, Wartbarkeit, Zuverlässigkeit und Korrektheit, werden in diesem Kapitel kurz vorgestellt. Für jeden dieser Aspekte kamen mehrere Verfahren infrage. Schienen sie im Rahmen der Bachelorarbeit realisierbar, wurden sie genauer untersucht. Diese genauer untersuchten Verfahren werden in diesem Kapitel vorgestellt. Von jedem Verfahren werden Vor- und Nachteile hervorgehoben und dann die letztendlich Verwendeten genannt.

3.1. Usability

3.1.1. Überblick

Da der englische Begriff “Usability“ im deutschen Sprachraum etablierter ist als die deutsche “Gebrauchstauglichkeit“, wird in dieser Arbeit der englische Begriff verwendet. Die “Usability ist das Ausmaß, in dem ein [Programm] durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effzient und zufriedenstellend zu erreichen.“ [Röd12].
Bei der Bewertung der Usability ging es vor allem darum, wie schnell die Nutzer von SIFEI Funktionen finden und wie gut sie mit diesen arbeiten können. Für die Usability ist es außerdem wichtig, dass sich SIFEI gut in den Arbeitsablauf des Nutzers integriert.
All dies führt dazu, dass der Nutzer das System effizienter nutzen kann und sich dabei wohlfühlt. Für die Bewertung der Usability wurden die folgenden Verfahren gefunden, die im Rahmen einer Bachelorarbeit geeignet schienen:

Usability-Patterns

Es gibt eine Reihe von Usability-Patterns, welche von Röder in [Röd12] untersucht und vorgestellt wurden. Sie versprechen eine erhebliche Steigerung der Usability. Es handelt sich hier nicht um starre Vorschriften, sondern eher um Vorschläge wie die Usability verbessert werden kann. Nicht bei allen Programmen lassen sich jedoch alle Usability-Patterns umsetzen.

Ein sehr bekanntes und weitverbreitetes Usability-Pattern ist das “Globale Undo“. Ein Programm sollte die Funktionalität bieten “ausgeführte Aktionen rückgängig zu machen und den vorherigen Zustand des Systems wiederherzustellen.“[Röd12]

Analyse einer Benutzerstudie

Vor dem Start der Bachelorarbeit wurde am Institut für Softwaretechnologie im Bereich Software Engineering von Daniel Kulesz eine Studie zu SIF durchgeführt. Bei dieser mussten Spreadsheets bearbeitet werden. Beobachtet wurde unter anderem, wie die Probanden SIFEI verwendet haben. Hierbei wurden Bildschirmaufzeichnungen erstellt, die zur Auswertung zur Verfügung standen. Anhand der Videos konnte das Verhalten der Probanden analysiert werden. Vor allem die Momente, in denen sie Probleme mit der Usability hatten, konnten identifiziert werden.
Nachdem die Probanden den praktischen Teil beendet hatten, mussten sie einen Fragebogen ausfüllen. In diesem wurde unter anderem SIFEI bewertet. Auch hatten die Probanden Gelegenheit, ihre Meinung über SIFEI zu äußern. Es kann davon ausgegangen werden, dass hier erwähnte Probleme besonders gravierend für die Probanden waren.

Heuristische Evaluation

Bei einer heuristischen Evaluation [JN94] beurteilt eine Gruppe von Usability Experten anhand bestimmter Heuristiken, die Usability von Software. Dabei kann es je nach Programmtyp verschiedene Heuristiken geben. Die Experten bewerten die Software einzeln, um dann ihre Befunde zusammenzutragen und zu bewerten.

Cognitive Walkthrough

Bei einem Cognitive Walkthrough [JN94] versetzt sich ein Usability-Experte in den Nutzer hinein. Er löst dabei verschiedene Aufgaben, die ein typischer Nutzer mit dem Programm durchführen würde. Anders als bei der heuristischen Evaluation gibt es ein klares Vorgehen, welche Teile des Programms in welcher Reihenfolge abgearbeitet werden. Es wird versucht einen Nutzer zu simulieren. Hierbei wird notiert, auf welche Hindernisse der Nutzer stoßen könnte, wie sehr ihn das Programm an die Hand nimmt und wie intuitiv das Programm zu bedienen ist.

3.1.2. Kritik

Sowohl bei der heuristischen Evaluation, wie auch beim Cognitive Walkthrough braucht man mindestens einen, wenn möglich sogar mehrere Usability-Experten und sinnvolle Heuristiken. Sinnvolle Heuristiken sind im Rahmen der Bachelorarbeit schon schwer zu entwickeln, diese Verfahren scheitern aber am Mangel von Usability-Experten.
Bei der Überprüfung mittels Usability-Patterns prüft man das Programm nur gegen abstrakte Konzepte, bei der keine Abstufung innerhalb eines Patterns möglich ist. Außerdem lässt sich von der Abwesenheit eines Patterns nicht darauf schließen, dass das Programm mit dem Pattern eine bessere Usability hätte. So bringt z.B. eine Autovervollständigung bei Eingaben nichts, wenn sie die Verwendung nicht vereinfacht. Wenn also keine sinnvolle Vervollständigung möglich ist, sollte diese weggelassen werden. Jedoch wurde SIFEI trotzdem auf verwendete Usability-Patterns untersucht, da die Patterns eine schnelle Übersicht über fehlende Usability-Funktionen ermöglichen.
Durch die Länge der Videos ist eine Auswertung dieser sehr zeitintensiv. Trotzdem lohnt sich eine Analyse, da Probleme der Nutzer mit dem Programm erkannt werden.
Als die Videos analysiert wurden, wurde die Wiedergabe teilweise beschleunigt. Dies geschah dann, wenn die Nutzer keine Probleme hatten. Außerdem wurden Freitextkommentare der Studie ausgewertet.

3.2. Wartbarkeit

Die Wartbarkeit wird vom IEEE wie folgt beschrieben: "[Wartbarkeit ist,] die Möglichkeit der Software verändert zu werden. Änderungen können Korrekturen, Verbesserungen oder Anpassungen enthalten, um auf Veränderungen der Umgebung, den Anforderungen und der Spezifikation einzugehen"[IEE04]. Eine gute Wartbarkeit ist nötig, da SIFEI an der Universität Stuttgart von Studenten entwickelt wurde und weiterentwickelt werden soll.
Für die Bewertung der Wartbarkeit wurden nur automatische Verfahren betrachtet, da im Rahmen der Bachelorarbeit eine manuelle Durchsicht des Codes, mit anschließender Bewertung, zeitlich nicht möglich war.

Maintainability Index

Dieses Bewertungsverfahren wurde 1991 eingeführt[OHA91] und über die Jahre verändert und verbessert [Wel01]. Es verwendet die folgenden Metriken:

- Zyklomatische Komplexität: Sie wurde in [McC76] von McCabe eingeführt. Sie beschreibt die Anzahl von linear unabhängigen Pfaden in dem Kontrollflussgraphen.
- "Lines of Code": Enthält die Anzahl an Codezeilen oder auch Anzahl an Anweisungen, die ein Programm oder Modul beschreiben.

- Halstead-Volumen: Wurde 1977 in [Hal77] vorgestellt und versucht die Größe von Algorithmen und Programm zu berechnen.

Diese Werte werden zusammengerechnet und ergeben eine Zahl. Dieser Maintainability Index liegt zwischen 0 und 100. Dabei gilt ein Index unter 20 als schlecht, alles unter 10 als sehr schlecht wartbar.
Dieser Maintainablity Index kann auch in Visual Studio überprüft werden, wodurch Quellcode mit nur wenigen Klicks analysiert werden kann.

C.R.A.P (Change Risk Anti-Patterns)

Bei C.R.A.P handelt es sich um ein ähnliches Prinzip wie beim Maintainability Index. Es werden Metriken gemessen und zu einer Zahl verrechnet [Sav07]. Diese Zahl gibt die Wartbarkeit an. Bei C.R.A.P werden die zyklomatische Komplexität und die Testüberdeckung zur Bewertung verwendet. Die Testüberdeckung beschreibt, welcher Anteil des Codes durch Test überprüft wurde. Dies ist aber problematisch, da eine hohe Testüberdeckung keine Tests erfordert, die die Komponente ausreichend testen.

Analyse einzelner Metriken

Für jedes Programm kann eine Vielzahl von Metriken berechnet werden. Was diese Metriken im einzelnen aussagen und implizieren, ist nicht immer eindeutig. Gleiche Werte einer Metrik können, je nach Programm, auf eine gute oder eine schlechte Wartbarkeit hinweisen. Oder sie sagen über die Wartbarkeit nichts aus. Noch unübersichtlicher wird es, wenn diese Metriken zusammengerechnet werden. Daher sollten Metriken einzeln betrachtet werden. Dann können für jede Metrik Grenzwerte bestimmt werden.
Dies kann durch die Verwendung bestimmter Werkzeuge vereinfacht werden. Ein Beispiel für ein solches Werkzeug ist Teamscale. Es ist ein Werkzeug für die statische Codeanalyse, das Metriken misst, analysiert und visualisiert [cqs]. Hierbei wird am Ende der Analyse keine Punktzahl über alle Metriken zusammen angegeben, sondern jede Metrik einzeln aufgelistet. Teamscale verbindet alle Befunde der Analyse mit dem zugehörigen Code und gibt Verbesserungsvorschläge, damit die Befunde gezielt gelöst werden können.
Durch eine Historie ist auch zu einem späteren Zeitpunkt noch ersichtlich, wie sich die Wartbarkeit entwickelt hat. Dies ist selbst dann möglich, wenn man Grenzwerte für Metriken oder ganze Metriken verändert.

3.2.1. Kritik

Da sich C.R.A.P und der Maintainability Index sehr ähnlich sind, haben sie auch die gleichen Schwächen. So kann es bei beiden Metriken vorkommen, dass ein Teil des Codes eine sehr

schlechte Bewertung erhält. Wenn aber dieser Teil prozentual nur wenig vom Code ausmacht, fließt dieser auch wenig in die Gesamtbewertung ein. Sollte aber genau dieser Teil des Codes besonders kritisch oder wichtig für die Anwendung sein, würde eine schlechte Wartbarkeit eines wichtigen Codestücks verschleiert werden. Plant man ein Softwareprojekt, sollten diese Teile gesondert und besonders genau betrachtet werden. Verlassen sich die Entwickler aber auf die Bewertung, kann dies eventuell komplett übersehen werden.
Außerdem treffen die Endbewertungen beider Verfahren keine Aussagen darüber, was verändert werden muss, um die Wartbarkeit zu erhöhen. Man weiß vor einer erneuten Analyse nicht, welche Veränderungen sich auf die Wartbarkeit ausgewirkt haben. Dies liegt daran, dass die Bewertung von vielen Faktoren abhängt und diese Abhängigkeiten nicht klar aufgezeigt werden. Dies ist für die Überprüfung von SIFEI von großem Nachteil.
Bei der Analyse einzelner Metriken lässt sich genau entscheiden, welche Metriken untersucht werden. Außerdem lassen sich Grenzwerte für diese Metriken festlegen, die auf eine schlechte Wartbarkeit hinweisen. Aber die manuelle Analyse ist umständlich und umfangreich. Außerdem ist es hilfreich, wenn man sich an gewissen Grenzwerten orientieren kann. Daher wurde Teamscale verwendet. Es bietet viele Möglichkeiten, den Code zu analysieren. Es können viele verschiedene Metriken untersucht werden. Außerdem werden Grenzwerte vorgeschlagen, die aber individuell angepasst werden können.
Dies ermöglicht umfangreiche Analysen, birgt aber auch Risiken. So kann die Bewertung zurechtgebogen werden, ohne die Wartbarkeit zu verbessern. Jeder Verstoß kann theoretisch ignoriert werden, sodass er nicht mehr in die Bewertung mit einfließt. Es müssen also die richtigen Parameter gewählt werden, um die Wartbarkeit zu beurteilen.

3.3. Zuverlässigkeit

3.3.1. Überblick

Die Zuverlässigkeit wird definiert, als die “Fähigkeit, eines Systems oder einer Komponente, die Anforderungen in festgelegten Zuständen für einen festgelegten Zeitraum, zu erfüllen“ [RGK90]. Bei Zuverlässigkeit-Tests wird überprüft, ob ein Programm über einen langen Zeitraum funktioniert und wie es mit einer großen Arbeitslast zurechtkommt.

Lasttest

Beim Lasttest wird getestet, wie das Programm unter einer hohen Arbeitslast reagiert. Führt dies zu einer nicht spezifizierten Leistungsreduktion oder einer Erhöhung des Fehlverhaltens, sollte die Zuverlässigkeit des Programms verbessert werden. Hierbei können sowohl einzelne Komponenten, als auch das gesamte System getestet werden. Die Tests können auch automatisiert werden, was vor allem bei einer regelmäßigen Ausführung von Vorteil ist.

3.3.2. Kritik

Besteht ein System aus verschiedenen Komponenten, ist es schwierig zu entscheiden, welche genauer betrachtet werden sollten. Hierzu braucht man Daten aus der Praxis, welche Komponenten das System am stärksten ausbremsen. Man muss außerdem bedenken, dass manche Komponenten eventuell unter Last sehr schlecht reagieren, jedoch nur sehr selten gebraucht werden. Für diese lohnt sich eine Beschleunigung eventuell nicht.
Bei einem Excel Plugin kann es unter Umständen dazu kommen, dass nicht ganz klar zuzuordnen ist, ob Excel selbst oder das Plugin langsam reagiert. Dies sollte bei der Durchführung des Lasttest bedacht werden. Außerdem lohnen sich bei wenigen Testdurchführungen automatische Tests nicht. Manuelle Tests haben jedoch den Nachteil, dass die einzelnen Ausführungen eventuell nicht unter gleichen Bedingungen durchgeführt werden.
Trotzdem wurde für die Bewertung von SIFEI manuelle Lasttests verwendet. Es wurden Formeln erstellt, die Regeln bewusst brechen und diese im Dokument vervielfacht. Hierdurch wurde die Prüfung von statischen Prüfkriterien durchgeführt und SIFEI konnte darauf untersucht werden, wie es mit über 150 Befunden reagiert.
Außerdem wurden Szenarios erstellt, welche besonders viele Eingabe-, Zwischenergebnis- und Ergebniszellen enthielten.

3.4. Korrektheit

3.4.1. Überblick

Die Korrektheit wird vom IEEE, unter Anderem, wie folgt definiert:

(1) “Der Grad bis zu dem ein System oder eine Komponente bezüglich Spezifikation, Design und Implementierung, frei von Fehlern ist.“ [RGK90]

(2) “Der Grad bis zu dem Software, Dokumentation und ähnliche Dokumente, die Erwartungen und Bedürfnisse der Nutzer erfüllen; egal ob spezifiziert oder nicht.“ [RGK90]

Im Rahmen dieser Bachelorarbeit sollte die Korrektheit hinsichtlich beider Definitionen getestet werden. SIFEI soll sowohl so funktionieren wie spezifiziert, als auch so wie von den Nutzern erwartet. Im Folgenden werden zwei Verfahren vorgestellt, mit denen SIFEI auf Korrektheit getestet wurde.

Modultests

Modultests testen kleine Module des Programms auf Korrektheit. Diese können einzelne Methoden, als auch ganze Klassen sein. Während der Erstellung der Testfälle hat der Tester

Einblick in den Quellcode.
Für SIFCore waren bereits Modultests vorhanden. Es handelt sich dabei um JUnit-Tests. Mit ihnen kann SIFEI auch indirekt mitgetestet werde. So kann mit SIFEI mit wenig Aufwand ein Regresstionstest durchgeführt werden. Er überprüft, ob durch Änderungen neue Fehler eingeführt wurden [LL12].

Manueller Systemtest

Bei einem Systemtest wird das ganze Programm auf seine Funktionalität überprüft. Wenn er manuell durchgeführt wird, steigt der Testaufwand pro Test. Man spart dafür aber viel Aufwand für die Automatisierung. Dieser Test ist gut geeignet, um zu überprüfen, ob ein Programm so funktioniert, wie es die Nutzer erwarten.

3.4.2. Kritik

Bei den Modultests muss bei der initialen Bewertung darauf vertraut werden, dass in den bisherigen Projekten ausreichend gute Modultests für SIFEI und SIFCore erstellt wurden. Um die Modultests zu erweitern, wäre eine komplette Einarbeitung in SIFCore nötig gewesen. Das war jedoch aufwandsbedingt nicht möglich. Trotzdem wurden die Modultests verwendet, da mit ihnen ein Regressionstest einfach umzusetzen war.
Die manuellen Systemtests haben den Nachteil, dass zwei identische Tests beinahe unmöglich sind. Außerdem musste erst herausgefunden werden, welche Anforderungen die Nutzer an SIFEI haben. Dann können aber eventuell Fehler erkannt werden, die nicht von den Modultests gefunden wurden und getestet werden, ob sich SIFEI so verhält, wie es der Nutzer erwartet.

4. Ergebnisse der Bewertung

Eine genaue Auflistung aller Befunde findet sich im Anhang A. Im Folgenden wird ein grober Überblick über die Ergebnisse der Bewertung gegeben.

4.1. Usability

Insgesamt wurden SIFEI auf 19 Usability-Patterns untersucht. Von diesen waren 10 in SIFEI nicht umgesetzt. Die Restlichen waren entweder bereits umgesetzt oder in SIFEI nicht anwendbar. Die Auswertung der Nutzerstudie ergab, dass die Usability von SIFEI generell als ausreichend bewertet wurde. Es wurden dennoch 30 Verbesserungen vorgeschlagen. Diese bezogen sich auf die Usability, Funktionswünsche und kleinere Bugs. Eine genaue Auflistung finden sich im Unterabschnitt A.2.4
Besonders deutlich wurde, dass die Nutzer nach Prüfungen mehr Rückmeldung benötigen. Außerdem waren einige Funktionen, wie die Erstellung von Szenarios, für den Nutzer nicht intuitiv. Sobald SIFEI eine Änderung an einem Spreadsheet vornimmt, gibt es keine Möglichkeit mehr, diese rückgängig zu machen. Dies schränkte die Probanden ein und war auch dementsprechend in der Benutzerstudie sichtbar.

4.2. Wartbarkeit

Bei einer Analyse durch Teamscale wurden insgesamt 536 Verstöße gemeldet. Eine genaue Aufteilung der gefundenen Verstöße findet sich in Tabelle 4.1. Man kann erkennen, dass vor allem die Anzahl an Kommentaren nicht ausreichend war.
Die anderen Verstöße verteilen sich relativ gleichmäßig auf die restlichen Kategorien. Hervorgehoben werden sollte die Kategorie *Struktur*. Sie weist auf zu lange Methoden und Klassen hin. Lange Methoden können ein Hinweis auf schlechten Code sein [BFB99]. Daher sollte bei diesen Methoden überprüft werden, ob sie verständlich sind oder ob sie gekürzt besser lesbar wären. Gerade in Verbindung mit fehlenden Kommentaren sind lange Methoden und Klassen oft schlechter lesbar und wartbar.

Tabelle 4.1.: Übersicht der von Teamscale gefundenen Befunde

Kategorie	Unterkategorie	Anzahl Befunde
Code Anomalien		**64**
	Bad practice	2
	Qualität der Kommentare	13
	Generelle Checks	29
	Wartbarkeitsregeln von Stylecop	3
	Nicht verwendeter Code	2
Code Duplikate		**27**
Dokumentation		**320**
	Vollständigkeit der Kommentare	306
	Dokumentationregeln	2
	TODO Tags	12
Code-Formatierung		**22**
Nomenklatur		**38**
Struktur		**48**
	Dateigröße	8
	Methodenlänge	24
	Schachtelungstiefe	16

4.3. Zuverlässigkeit

Um die Zuverlässigkeit zu testen, wurde ein großes Spreadsheet erstellt und befüllt. Es wurden Formeln eingefügt, bei denen ein Verstoß vorlag und dann viele Prüfungen in kurzer Zeit angestoßen.
Wurden einzelne Prüfungen mit vielen Verstößen gestartet, stieg die Ausführungszeit der Prüfung gleich mit der Anzahl an Verstößen. Die Anzahl der fehlgeschlagenen Prüfungen ist im gleichen Maß gestiegen.
Die Anzahl der in kurzer Zeit gestarteten Prüfungen war deutlich bedeutender für die Dauer der einzelnen Prüfungen. Vor allem wenn Prüfungen gleichzeitig durchgeführt wurden, ist die Dauer der einzelnen Prüfungen stark angestiegen. Außerdem haben sich die Ausfälle stark gehäuft, wenn mehrere Prüfungen gleichzeitig durchgeführt wurden.
Somit lag der Schluss nahe, dass vor allem viele gleichzeitige Prüfungen die Zuverlässigkeit vermindern.

4.4. Korrektheit

Bei dem Test der Korrektheit sind alle in SIFCore enthaltenen Unit-Tests fehlerfrei beendet worden. Der manuelle Systemtest hat einige Fehler aufgezeigt. Diese konnten aber komplett den Aspekten Usability und Zuverlässigkeit zugeordnet werden. Die so gefundenen Schwächen finden sich im Unterabschnitt A.2.6.
Durch diese Tests wurden keine neuen Schwächen bezüglich der Korrektheit gefunden. Die Modultests ermöglichten aber einen späteren Regressionstest und der Systemtest fand neue Schwächen bezüglich der Usability und der Zuverlässigkeit.

5. Priorisierung

Da nicht alle gefunden Schwächen im Laufe der Bachelorarbeit hätten behoben werden können, mussten diese Schwächen priorisiert werden. Bei der Priorisierung der Schwächen mussten verschiedene Aspekte betrachtet werden. Dazu gehörte, wie sehr die Verwendung von SIFEI durch die Schwächen eingeschränkt wurde und wie hoch der Zeitaufwand für die Behebung der Schwäche wäre. Außerdem wurden alle Schwächen gleichzeitig priorisiert und nicht nach Aspekten getrennt.
Es werden im Folgenden erst einige Priorisierungsverfahren vorgestellt. Sie wurden ausgewählt, da sie auf den ersten Blick geeignet schienen [BA05]. Danach wurden sie genauer betrachtet und verglichen, um dann das geeignetste auszuwählen.

5.1. Überblick über verschiedene Priorisierungsarten

- **Analytical Hierarchy Process**: Entwickelt und erweitert wurde das Verfahren von Thomas L. Saaty [Saa08] [WS80]. Es ist vor allem für Gruppenentscheidungen geeignet. Bei dieser Priorisierungsart wird jeder zu priorisierende Punkt mit jedem anderen verglichen. Dies bedeutet, dass $\approx \frac{n^2}{2}$ Vergleiche gemacht werden müssen, wobei n die Anzahl der zu priorisierenden Anforderungen ist. Es werden immer zwei Anforderungen miteinander verglichen. Es wird dann entschieden, wie viel höher eine Anforderung gegenüber der anderen priorisiert wird.
- **Cumulative Voting**: Beim Cumulative Voting - oder auch 100 Dollar Test - können 100 Bewertungseinheiten auf die Anforderungen verteilt werden. Durch diese Art der Bewertung sind die Anforderungen priorisiert. Außerdem befinden sie sich auf einer Kardinalskala, wodurch sich bei mehreren Interessensgruppen die Ergebnisse zusammenrechnen lassen, um eine Priorisierung über alle Interessensgruppen zu erhalten.
- **Gruppierung**: Bei dieser Art der Priorisierung werden die Anforderungen auf bestimmte Gruppen verteilt. Die Anzahl dieser Gruppen kann variieren, jedoch wird meistens eine Aufteilung in 3 Gruppen vorgenommen [LW00] [SS97]. Es ist außerdem hilfreich, wenn die einzelnen Gruppen aussagekräftige Namen haben wie z.B. “Critical Feature“, “Nice to have“ oder Ähnliches. Innerhalb jeder dieser Gruppen haben alle Elemente die gleiche Priorität.

- **Ranking**: Die Anforderungen werden priorisiert, indem sie geordnet werden. Es gibt aber hierdurch keine Abstufung, wie viel wichtiger eine Anforderung einer anderen gegenüber ist. Daher ist es auch schwierig, Priorisierungen verschiedener Interessengruppen zusammenzuführen.
- **Top 10**: Bei diesem Verfahren wählt jede Interessengruppe ihre zehn wichtigsten Anforderungen. Hierbei wird die Priorität der gewählten Anforderungen nicht weiter definiert. So lassen sich die zu realisierenden Funktionen einfacher auswählen [Lau02].

5.2. Verwendete Priorisierung

Um die Anforderungen an SIFEI zu priorisieren, wurden die Anforderungen in verschiedene Gruppen eingeordnet. Es wurden insgesamt vier Gruppen für die Einteilung verwendet. Bei den vorher bekannten Schwächen geschah die Einordnung, vor allem durch Gespräche mit dem Betreuer der Bachelorarbeit. Er hatte die initialen Anforderungen an SIF gestellt und hat die Entwicklung seither begleitet.
Bei den neu gefundenen Schwächen wurde abgewogen, wie stark sie den produktiven Einsatz von SIF einschränken. Hierzu wurde die Nutzerstudie ausgewertet und auch wieder der Betreuer befragt. Die Ergebnisse der Priorisierung sind im Unterabschnitt A.2.7 dargestellt.

6. Vorgenommene Änderungen und Ergänzungen

6.1. Änderungen an der graphischen Benutzeroberfläche

6.1.1. Befunde auf “später“ verschieben

Beschreibung und Auswirkungen

In SIFEI ist es vorhergesehen, dass gefundene Befunde in verschiedene Kategorien eingeteilt werden können. Befunde in der Kategorie “später“ wurden aber in die Kategorie “offen“ verschoben. Dies geschah jedes mal, wenn eine neue Prüfung gestartet wurde.

Änderungen

Um das Problem zu lösen, wurde der Quellcode untersucht, um die Fehlerursache zu finden. Besonders genau wurde dabei betrachtet, was während und nach einer Prüfung geschieht. War ein Befund in der Kategorie “später“, wurde diese Kategorie nach jeder Prüfung überschrieben. Nur die Befunde in der Kategorie “später“ wurden wieder in die Kategorie “offen verschoben“. Um das Problem zu lösen, wurde dieser Teil des Codes entfernt. Nun wird die Kategorie eines Befundes nach einer Prüfung nicht mehr überschrieben.

6.1.2. Rückmeldung bei Prüfungen

Beschreibung und Auswirkungen

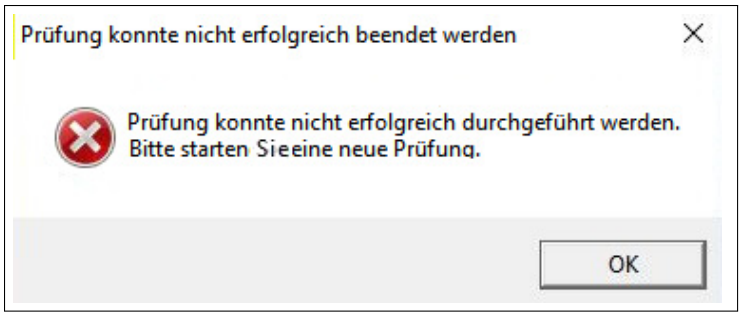
Wurde in SIFEI eine Prüfung angestoßen, so erhielt der Nutzer keine Rückmeldung, ob diese stattfindet bzw. ob sie beendet wurde. Hierdurch haben Nutzer teilweise mehrere Prüfungen angestoßen. Dies führte zu Kommunikationsproblemen zwischen SIFEI und SIFCore. Diese Kommunikationsprobleme verminderten die Zuverlässigkeit, da sie Abstürze provozierten. Wurden außerdem bei einer Prüfung keine neuen Befunde gefunden oder gelöscht, hatten

Nutzer ein weiteres Problem. Sie wussten nicht, ob die Prüfung erfolgreich abgeschlossen wurde.

Änderung

Zum einen musste dem Nutzer mitgeteilt werden, dass eine Prüfung durchgeführt wird. Zum anderen musste verhindert werden, dass der Nutzer eine zweite Prüfung starten kann. Außerdem sollte an den Nutzer eine Rückmeldung über durchgeführte Prüfungen erfolgen. Um Ersteres umzusetzen, wird bei einer Prüfung der Prüfungsbutton ausgegraut, deaktiviert und seine Beschriftung geändert (Abbildung 6.1a). Da der Button hierdurch nicht mehr geklickt werden kann, wird auch vermieden, dass der Nutzer selbst eine weitere Prüfung über den Button anstößt. Außerdem wird bei automatisch durchgeführten Prüfungen überprüft, ob schon eine Prüfung durchgeführt wird. Ist dies der Fall, wird keine zweite Prüfung gestartet, sondern der neue Prüfauftrag verworfen.
Um dem Nutzer zurückzumelden, dass die Prüfung beendet wurde, wird dem Nutzer in der Statusleiste von Excel ein Text angezeigt. Bei einer gescheiterten Prüfung erscheint außerdem eine Fehlermeldung (Abbildung 6.1b).
Ein Problem bei der Umsetzung dieser Änderungen war, dass Prüfungen durch verschiedene Ursachen abgebrochen bzw. scheitern können. Daher mussten alle Stellen gefunden werden, an denen eine Prüfung abgebrochen wurde.

(a) Ausgegrauter Prüfbutton



(b) Fehlermeldung bei gescheiterter Prüfung

Abbildung 6.1.: Rückmeldung bei Prüfungen

6.1.3. Szenarieneditor

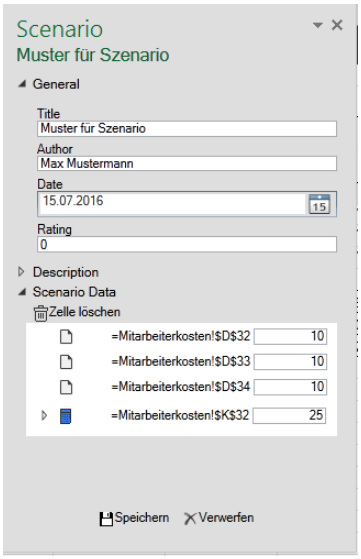
Beschreibung und Auswirkungen

Um Szenarien zu editieren, müssen Nutzer eine Seitenleiste öffnen. Wurde ein Szenario zum Editieren geöffnet, wurde eine zweite Seitenleiste geöffnet. In dieser war dann das Editieren möglich.

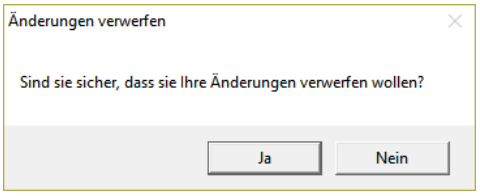
Es gab keinen expliziten Speicher-Button, daher wusste der Nutzer nicht ob und wann die vorgenommenen Änderungen übernommen wurden. Außerdem wurden Änderungen an Szenarien auch beim erfolgreichem Speichern des Szenarios nicht in die Datei übernommen. Die Änderungen gingen beim Schließen des Dokuments verloren, wenn das Dokument nicht zuvor gespeichert wurde.

Änderungen

Der Szenarieneditor wurde so verändert, dass nun keine zweite Seitenleiste angezeigt wird. Stattdessen wird der Szenarieneditor als extra Fenster geöffnet. Auch wurden die gewohnten "Speichern" und "Verwerfen" Buttons hinzugefügt, damit der Nutzer gezielt Änderungen an Szenarios speichern kann (Abbildung 6.2a).
Außerdem wird nachgefragt, ob Änderungen an Szenarien verworfen werden sollen, wenn das Fenster geschlossen wird oder der "Verwerfen" Button gedrückt wird (Abbildung 6.2b). Im Gegensatz zu früher werden Änderungen nur dann gespeichert, wenn dies explizit erwünscht ist.



(a) Szenarieneditor



(b) Absicherung gegen Verwerfen von Änderungen

Abbildung 6.2.: Änderungen des Szenarieneditors

6.1.4. Befund-Icons

Beschreibung und Auswirkung

Wird ein Befund in einer Zelle gefunden, wird in diese ein Button gezeichnet, der ein Icon enthält. Dieses Zeichnen dauerte bei vielen Befunden relativ lange. Dies führte gerade bei automatischen Prüfungen zu häufigen Wartezeiten, in denen das Spreadsheet nicht verwendet werden konnte.
Diese Buttons waren außerdem so implementiert, dass sie in die Datei gespeichert wurden. Das heißt, sie waren bei einem erneuten Öffnen der Datei immer noch vorhanden. Sie wurden aber als Bilder ohne Funktionalität in die Zellen geladen, wenn die Datei geöffnet wurde. Daher musste in jedem Fall eine neue Prüfung gestartet werden, um die Details der Szenarios zu betrachten.
Bei Benutzern die SIFEI nicht installiert hatten, gab es noch weitere Probleme. Sie konnten keine neue Prüfung starten und die Bilder auch nicht entfernen.

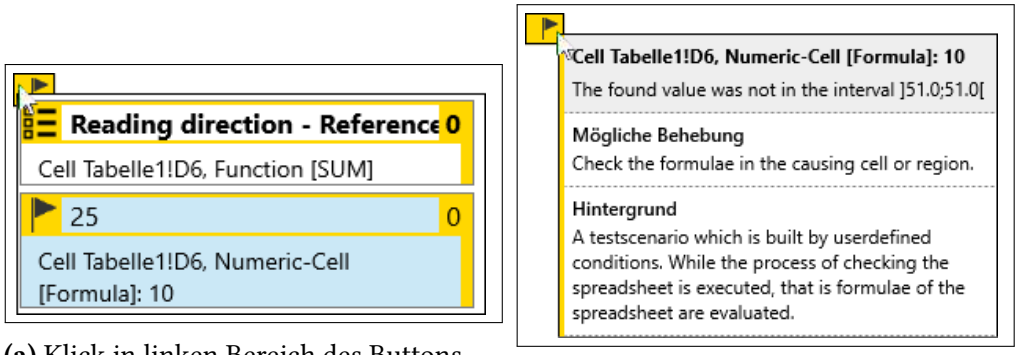
Änderungen

Der Zeichenvorgang wurde beschleunigt, indem durchgeführte Berechnungen vor das eigentliche Zeichnen verschoben wurden und die Reihenfolge der Zeichenvorgänge optimiert wurde.
Die Buttons hätten so abgespeichert werden können, dass sie mit Funktionalität in das Spreadsheet geladen werden. Dies hätte jedoch dazu geführt, dass Nutzer ohne SIFEI in der Verwendung der Spreadsheets eingeschränkt werden. Außerdem wären die Ladezeiten beim Öffnen des Spreadsheets verlängert worden. Daher wurde SIFEI so abgeändert, dass die Buttons nicht in das Spreadsheet gespeichert werden. Sie werden nun beim Schließen der Datei aus den Zellen gelöscht.

6.1.5. Kontextmenü

Beschreibung und Auswirkungen

Das Kontextmenü eines Befund-Icons war bisher relativ kompliziert umgesetzt, da die Bedienung davon abhing, ob nur einer oder mehrere Befunde in der Zelle gefunden wurden. Wurden mehrere Befunde gefunden, so wurden unterschiedliche Kontextmenüs aufgerufen, je nach dem, wo genau in das Befund-Icon geklickt wurde. Weiter erschwert wurde die Bedienung dadurch, dass die beiden Bereiche visuell nicht unterschieden werden konnten (Abbildung 6.3).



(a) Klick in linken Bereich des Buttons

(b) Klick in rechten Bereich des Buttons

Abbildung 6.3.: Varianten des alten Kontextmenüs

Änderungen

Das Kontextmenü von Befund-Icons wurde so geändert, dass es nur noch ein Kontextmenü pro Icon gibt. Dieses zeigt eine Liste aller Befunde in der Zelle an. Neben dieser Liste wird nun eine Detailansicht des ausgewählten Befundes angezeigt. (Abbildung 6.4)

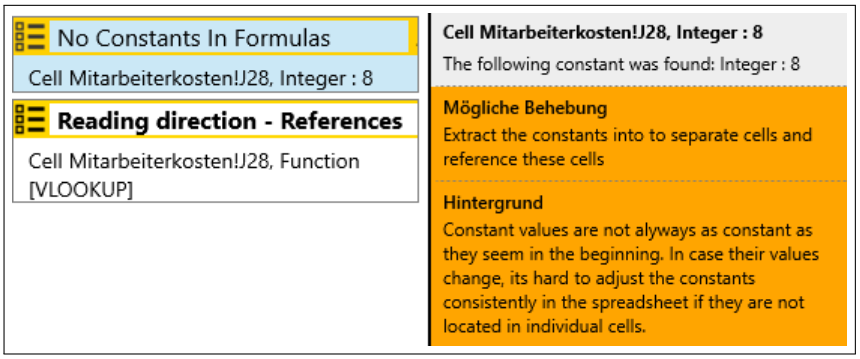


Abbildung 6.4.: Geändertes Kontextmenü

6.1.6. Fehlermeldungen

Beschreibung und Auswirkungen

Bisher wurden kaum Fehlermeldungen angezeigt. Der Nutzer wurde unter Anderem nicht informiert wenn Prüfungen abgebrochen wurden, Prüfungen nicht gestartet werden konnten und die Datei nicht gespeichert war. Dies schränkte die produktive Verwendung von SIFEI ein.
Außerdem sind viele der Fehlermeldung in verschiedenen Versionen entstanden. Dies führte

dazu, dass die Fehlermeldungen unterschiedlich aussahen. Ein einheitliches Aussehen führt meistens zu einer besseren Usability.

Änderungen

Es wurden an vielen Stellen neue Fehlermeldungen hinzugefügt. Außerdem wurden die vorhandenen Fehlermeldungen geändert, sodass alle ein einheitliches Aussehen haben.

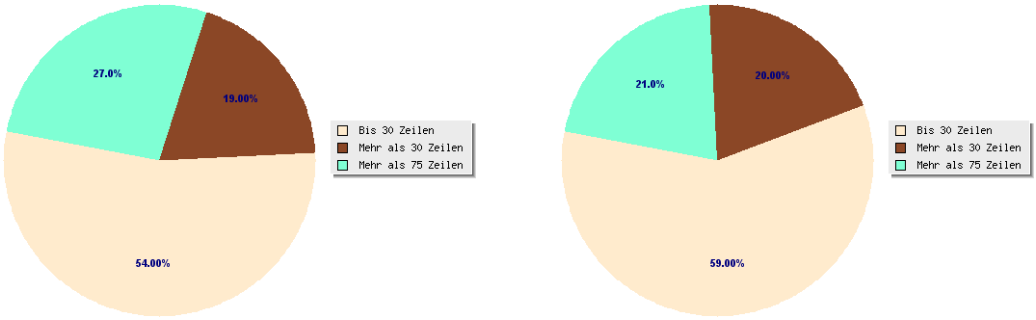
6.2. Änderungen am Code

6.2.1. Dokumentation

Große Teile von SIFEI enthielten kaum oder keine Kommentare. Es fehlten sowohl Klassen- und Methodenkommentare, wie auch Kommentare im Code. Da auch sonst kaum Dokumentation zu SIFEI existiert, wurden im Laufe der Bachelorarbeit Kommentare nachgeführt. Vor dem Start der Bachelorarbeit waren beinahe 60% der Klassen und Methoden unkommentiert, jetzt sind es 43%.

6.2.2. Methodenlänge und Schachtelungstiefe

Viele der in SIFEI verwendeten Methoden waren sehr lang. Lange Methoden können dazu führen, dass sie schwer zu lesen und zu verstehen sind [BFB99]. Daher wurden Methoden aufgeteilt und gekürzt. Dies wurde gemacht, um das Leseverständnis des Codes zu verbessern. Wenn dies zu einer Vereinfachung führt, sind auch fehlende Kommentare für zukünftige Programmierer weniger gravierend. In Abbildung 6.5 kann man erkennen, wie sich die maximale Methodenlänge pro Datei verändert hat. Es wird angegeben in wie vielen Dateien zu lange Methoden gefunden wurden. Hierbei wird immer nur die längste Methode betrachtet. Da immer noch 21% der Dateien und Methoden mit über 75 Zeilen haben, sollten diese in Zukunft weiter untersucht und gegebenenfalls gekürzt werden.



(a) SIFEI vor den Änderungen **(b)** SIFEI nach den Änderungen

Abbildung 6.5.: Entwicklung der Methodenlänge

6.2.3. Weitere Änderungen

Weitere kleinere Änderungen am Code waren die Verminderung von Code-Duplikaten, Anpassen des Codes an Namenskonventionen und Löschen von redundantem und unnötigem Code.

6.3. Hinzugefügte Funktionen

Um das produktive Arbeiten mit SIFEI zu verbessern und zu erweitern, wurden auch Funktionen hinzugefügt. Diese zielten darauf ab, die Usability zu verbessern.

6.3.1. Befund-Icons

Problembeschreibung

Bisher waren die Befund-Icons so umgesetzt, dass immer nur ein Befundtyp angezeigt wird, das heißt, es war nicht ersichtlich, welche Art von Befunden in der Zelle gefunden wurden. Hierdurch war es wahrscheinlicher, dass Nutzer Befunde übersehen, da das entsprechende Icon nicht angezeigt wurde.

Erweiterung

Die Befund-Icons wurden so verändert, dass sie nun alle enthaltenen Befundtypen anzeigen. Hierzu werden die verschiedenen entsprechenden Icons nebeneinander platziert (Abbildung 6.6). Hierdurch wurde darauf abgezielt, dass Nutzer deutlich schneller erkennen, welche Befunde in welchen Zellen vorliegen.

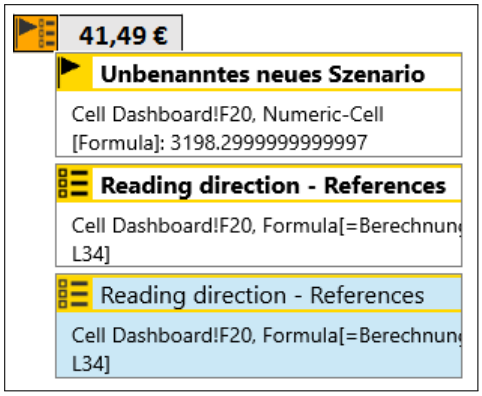


Abbildung 6.6.: Befund-Icon mit verschiedenen Befundentypen

6.3.2. Automatisches Speichern

Problembeschreibung

War die Excel-Datei nicht im richtigen Format oder noch gar nicht gespeichert, war bisher keine Prüfung möglich. Dies lag daran, dass SIFCore die Datei nicht finden konnte. Daher konnte SIFCore sie nicht analysieren und hat eine Fehlermeldung ausgegeben.

Erweiterung

Um dies zu verhindern, wird nun vor einer Prüfung sichergestellt, dass die Datei im “.xls“- oder im “.xlsx“-Format gespeichert ist. Sollte dies nicht der Fall sein, so wird versucht die Datei zu speichern und erst dann eine Prüfung ausgeführt.

6.3.3. Wiederausführung einer Prüfung

Problembeschreibung

Konnte SIFEI oder SIFCore bisher eine Prüfung nicht erfolgreich beenden, wurde in manchen Fällen eine Fehlermeldung angezeigt und die Prüfung nicht weiter fortgesetzt. In anderen Fällen wurde den Nutzern nicht mitgeteilt, dass die Prüfung gescheitert ist.

Erweiterung

In der jetzigen Version wird eine gescheiterte Prüfung bis zu drei mal wiederholt. Dies geschieht dann, wenn bei SIFEI Ausnahmen auftreten. Meistens liegen diesen Ausnahmen Fehler beim Laden der von SIFCore generierten Ergebnisse zugrunde.
Stürzt hingegen SIFCore beim Analysieren eines Dokumentes ab, wird im Moment noch keine automatische Wiederholung der Prüfung angestoßen.

7. Benutzerstudie

7.1. Einleitung

Nach dem Ende der Implementierungsphase wurde eine Studie durchgeführt. Die Studie basiert auf einer von Daniel Kulesz durchgeführten Studie vor Beginn der Bachelorarbeit.

7.1.1. Problemstellung

Im Rahmen der Bachelorarbeit wurde SIFEI in vielerlei Hinsicht verändert. Ohne eine objektive Bewertung, kann jedoch nicht festgestellt werden, ob diese Änderungen zu Verbesserungen geführt haben.

7.1.2. Ziele der Studie

Da Änderungen an der Usability nicht notwendigerweise zu Verbesserungen führen müssen, sollten diese Änderungen überprüft werden. Nur so konnte sichergestellt werden, dass die Änderungen ihr Ziel erreicht haben. Zudem sollte untersucht werden, wie sich die Zuverlässigkeit verändert hat. Um diese Problemstellung zu beantworten, wurden folgende zwei Fragen herausgearbeitet:

- Können die Nutzer SIFEI schnell und zielgerichtet verwenden?
- Vertrauen die Nutzer den Prüfungen von SIFEI?

7.1.3. Probanden

Die Studie wurde mit vier Bachelorstudenten der Universität Stuttgart durchgeführt. Von den vier Studenten haben drei Softwaretechnik studiert, der vierte Informatik. Alle vier waren Mitte 20, männlich und waren mindestens im achten Fachsemester ihres Studiums. Sie wurden ausgewählt, da sie am einfachsten zu rekrutieren waren und etwas Erfahrung mit Excel hatten. Die Termine zur Durchführung der Studie wurden individuell mit den Probanden vereinbart, sodass diese ausreichend Zeit für die Studie hatten. Alle Probanden

wurden gebeten, nicht mit anderen Probanden über die Studie zu reden. So sollte sichergestellt werden, dass keiner der Probanden wusste, welche Aufgaben zu lösen waren. Auch hatte jeder Proband jederzeit die Möglichkeit, die Studie abzubrechen .
Bei der Rekrutierung der Probanden wurden sie darüber aufgeklärt, dass die Studie ca. 2 Stunden dauern würde und sie mit Excel arbeiten würden. Jedoch hatten alle Probanden nur wenig Erfahrung mit Excel und mussten bisher keine umfangreichen Aufgaben damit lösen. Da die Studenten schon fortgeschritten in ihrem Studium waren, kann davon ausgegangen werden, dass alle ein überdurchschnittlich hohes Verständnis für Softwaresysteme haben. Da SIFEI auf Konzepten aus der Softwaretechnik basiert, ist es wahrscheinlich leichter für sie, die umgesetzten Konzepte zu verstehen.

7.2. Hintergrund

7.2.1. Vorherige Studien

Zu und mit SIFEI wurden bereits mehrere Studien durchgeführt. Eine von ihnen diente zur initialen Bewertung von SIFEI vor dem Beginn der Bachelorarbeit. Eine andere wurde durchgeführt um Text-Tutorials mit Video-Tutorials zu vergleichen [KKW16]. In diesen Tutorials wurde SIFEI beschrieben.

7.2.2. Relevanz

Die Usability eines Programms sollte auch von Nutzern bewertet werden, die nichts mit der Entwicklung zu tun hatten. Da die Entwickler eines Programms wissen, wie sie dieses verwenden müssen, können sonst Usability-Probleme übersehen werden.

7.3. Vorbereitung

7.3.1. Vorgehen

Die Probanden sahen zuerst ein Video, in welchem das Konzept von Spreadsheetprüfungen vorgestellt hat. In diesem wurden einige Beispiele für schlechte Spreadsheets gegeben und SIFEI eingeführt. Danach erhielten sie ein Text-Tutorial, das SIFEI und statische Tests vorstellte [KKW16]. Hiernach mussten sie die erste Aufgabe lösen. Diese war kurz und darauf ausgerichtet, das Gelesene anzuwenden. Als Nächstes bekamen die Probanden ein Text-Tutorial, das die Szenarientechnik einführte. Auch danach mussten sie eine kleine Aufgabe bearbeiten, um das Gelesene zu festigen. Nachdem sie beide Aufgaben bearbeitet hatten,

lösten sie die Hauptaufgabe (Die genaue Aufgabenstellung liegt der Bachelorarbeit bei).
Die Probanden führten die Studie an einem klimatisierten Arbeitsplatz in einem Computerpool durch. Während der Studie wurde der Bildschirminhalt der Probanden aufgezeichnet. Außerdem wurde der Bildschirm auf einen anderen Monitor gespiegelt. Hieraus sollte der Fortschritt verfolgt werden, ohne dass sich die Probanden beobachtet fühlten.
Alle Probanden hatten Probleme mit einigen Excel-Funktionen. Hierzu gehörten VLOOKUP und SVERWEIS. Konnten sie diese nicht selbst lösen, konnten sie im Internet nach den Formeln suchen.

7.3.2. Material

Die Probanden bekamen zu Beginn mehrere Dateien. Außerdem bekamen sie ein ausgedrucktes Aufgabenblatt, das sie durch die Studie leitete. Nach jeder gestellten Aufgabe mussten sie einige Fragen zu dieser ausfüllen. Außerdem bekamen sie Text-Tutorials zu den statischen und den dynamischen Tests und ein Video zur Einführung in SIFEI.
Bei diesen Aufgaben mussten sie drei verschiedene Dateien bearbeiten. Diese enthielten bewusst Fehler, die die Probanden finden und, wenn möglich, lösen sollten. Bei der dritten Datei handelte es sich um ein Spreadsheet, dass Telefonrechnungen berechnen konnte. Hierzu konnte ein individueller Verbrauch und ein Tarif angegeben werden.

7.3.3. Aufgaben

Die Probanden erhielten insgesamt drei Aufgaben. Diese waren in eine Reihe von kleineren Aufgaben unterteilt. Bei der ersten Aufgabe sollten sie die Grundlagen von SIFEI kennenlernen. Die Probanden mussten in einem Spreadsheet statische Regeln aktivieren und eine Prüfung starten. SIFEI meldete hierauf drei Befunde. Zwei dieser Befunde sollten die Probanden lösen. Einen Befund bezüglich der Leserichtung und einen bezüglich gleicher Verweise hintereinander (Abbildung 7.1). Um diese zu lösen, mussten zwei Zellen verschoben werden und ein mehrfacher Verweis gelöscht werden.

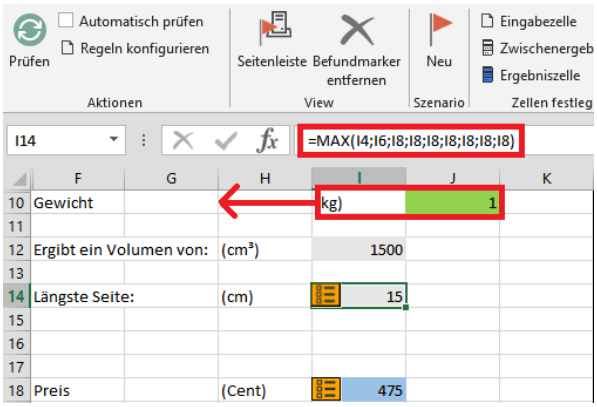


Abbildung 7.1.: Befunde in der ersten Aufgabe

In der zweiten Aufgabe mussten die Probanden ein Szenario erstellen. Damit sollten sie dann das Spreadsheet auf Fehler untersuchen. Dieses Szenario fand einen weiteren Fehler. Auch diesen sollten die Probanden beheben. Dazu musste eine überflüssige Subtraktion in einer Formel entfernt werden. Danach mussten die Probanden das Szenario umbenennen, neue Werte eintragen und eine neue Prüfung starten.
Anschließend mussten die Probanden die letzte und umfangreichste Aufgabe lösen. Hierfür mussten sie das dritte Spreadsheet erweitern. Sie mussten zuerst einen neuen Tarif erstellen und dann für jeden Tarif die Kategorie "netzinterne SMS", hinzufügen.

7.3.4. Hypothesen

Die folgenden Hypothesen sollten getestet werden:

- Die Probanden empfinden die Usability von SIFEI als gut.
- SIFEI stört den Arbeitsfluss nicht.
- Die Probanden empfinden SIFEI als zuverlässig.

7.4. Durchführung

Vor der Ankunft der Probanden wurde der Computer und die Bildschirmaufzeichnung gestartet. Auch wurde der Ordner geöffnet, in dem alle benötigten Dateien waren. Außerdem wurde die Umfrage und die Aufgabenstellungen ausgedruckt und bereitgelegt.

7.5. Analyse der Ergebnisse

Ziel war herauszufinden, wie die Probanden SIFEI bewertet haben. Sie sollten auf einer Likert-Skala angeben, wie sehr sie gewissen Aussagen zustimmen. Stimmten sie der Aussage überhaupt nicht zu, wird dies in den folgenden Diagrammen mit einer 1 dargestellt. Stimmten sie der Aussage vollkommen zu, mit einer 7.
Wie die Probanden die Usability von SIFEI bewertet haben, wurde durch verschiedene Fragen in der Umfrage untersucht. Wie sich in Abbildung 7.2 erkennen lässt, wurde die Usability von den meisten Probanden als gut bewertet.

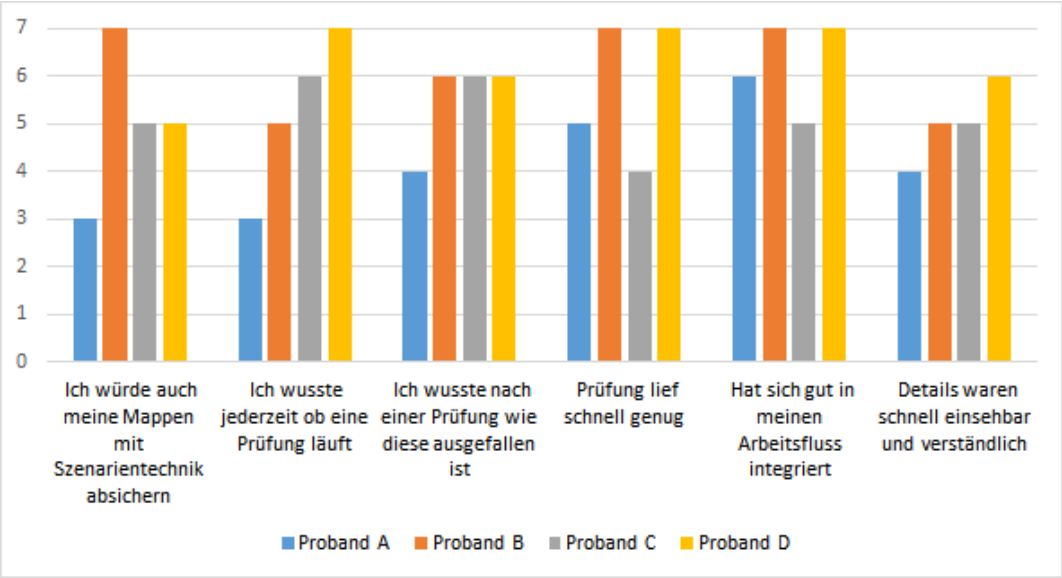


Abbildung 7.2.: Fragen bezüglich der Usability

Die Ergebnisse der Aussage “Ich hatte das Gefühl, dass [SIFEI] zuverlässig und korrekt arbeitet“ lassen sich in Abbildung 7.3 ablesen. Man kann erkennen, dass drei der Probanden die Zuverlässigkeit als ausreichend empfinden. Proband A hat die Zuverlässigkeit jedoch schlecht bewertet. Diese Bewertung wurde abgegeben, da bei ihm Excel abgestürzt ist. Dabei wurden alle Szenarien, die in dem Spreadsheet enthalten waren, gelöscht.
Außerdem wurde in den Freitextkommentaren geschrieben, SIFEI solle “ausfallsicherer“ sein und dass zu oft “Prüfungen [fehlschlagen]“. Daher lässt sich erkennen, dass die Probanden SIFEI vertrauen solange keine größeren Abstürze stattfinden. Wenn diese vorkommen, sinkt das Vertrauen in SIFEI stark.

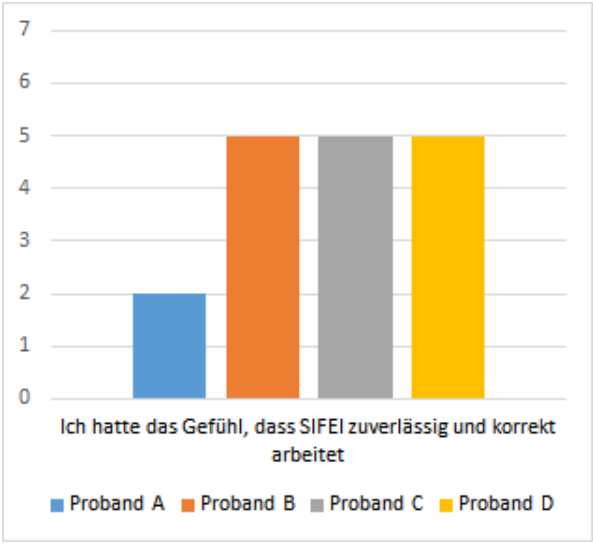


Abbildung 7.3.: Bewertung der Zuverlässigkeit durch Probanden

7.6. Diskussion

7.6.1. Bewertung der Hypothesen

Die Probanden bewerten die Usability von SIFEI relativ gut. Man kann aber erkennen, dass durchaus noch Verbesserungen nötig sind. Bestätigt hat sich die Hypothese, dass SIFEI sich gut in den Arbeitsablauf der Probanden integriert. In Abbildung 7.2 kann man erkennen, dass alle Probanden zufrieden damit waren. Widerlegt wurde die Hypothese, dass die Probanden SIFEI zuverlässig finden.

7.6.2. Gefährdungen der Validität

Die größte Gefährdung stellt die geringe Anzahl an Probanden da. Aussagen lassen sich schlecht verallgemeinern, wenn nur so wenige Daten vorhanden sind. Andererseits ging es bei der Studie vor allem darum, Tendenzen zu erkennen.
Eine weitere Gefährdung bestand aus der Zusammensetzung der Probanden. Es kann davon ausgegangen werden, dass sie die Konzepte die in SIFEI umgesetzt wurden, überdurchschnittlich gut verstehen, da sie alle in die Richtung Informatik studieren. Dies stellt einen Teil der Zielgruppe dar, jedoch werden viele Nutzergruppen nicht abgedeckt. Außerdem kann es sein, dass Informatikstudenten Usability anders empfinden als andere Nutzer.

8. Erneute Bewertung von SIFEI

SIFEI wurde nach der Implementierungsphase erneut bewertet. So sollte beurteilt werden, ob Änderungen zu Verbesserungen geführt haben. Um eine möglichst objektive Bewertung durchführen zu können, wurden wieder die selben Aspekte, wie bei der initialen Bewertung, mit gleichen Verfahren und Parametern geprüft. So wurden auch bei der abschließenden Bewertung die Usability mittels der Usability-Patterns, die Wartbarkeit mit Teamscale, die Zuverlässigkeit mit Lasttests und die Korrektheit mit Unit-Tests und einem Systemtest überprüft. Die Usability wurde zusätzlich mit der in Kapitel 7 beschriebenen Nutzerstudie bewertet.

8.1. Vergleich der Resultate

8.1.1. Zuverlässigkeit

Nach den vorgenommenen Änderungen ist nur eine Prüfung gleichzeitig möglich. Hierdurch konnten die Kommunikationsprobleme zwischen SIFEI und SIFCore verringert werden. Dies führt außerdem zu einer höheren Ausfallsicherheit.
Die Kommunikation zwischen SIFEI und SIFCore wurde nicht nennenswert verbessert. Daher kommt es noch immer zu Ausfällen, die auf die Kommunikation zurückzuführen sind.

8.1.2. Usability

Die Usability konnte verbessert werden. Es wurden zwei neue Usability-Patterns in SIFEI umgesetzt. Die Rückmeldung an den Nutzer konnte erheblich verbessert werden. Nun erfolgt eine Rückmeldung bei laufenden Prüfungen und das Ende einer Prüfung wird angezeigt. Ist die Prüfung gescheitert, wird dies noch stärker hervorgehoben. Außerdem lässt sich durch die Nutzerstudie erkennen, dass sich Details von Befunden schnell erkennen lassen.
Betrachtet man die Nutzerstudie, konnte die Usability automatischer Prüfungen verbessert werden. Diese erfolgen nun bei Änderungen an Zellen oder bei Speichervorgängen.

8.1.3. Wartbarkeit

Die durch Teamscale gemessene Wartbarkeit hat sich im Laufe der Bachelorarbeit verbessert. Die Anzahl an Verstößen wurde von 571 auf 420 vermindert. Dies entspricht einer Reduktion von über 20% (Abbildung 8.1) . Dies liegt vor allem daran, dass Kommentare hinzugefügt wurden, die Nomenklatur angepasst wurde und Methoden verkürzt wurden.
Da aber immer noch ein Großteil der Befunde durch fehlende Kommentare verursacht wird, sollte die Dokumentation des Codes in Zukunft noch weiter verbessert werden.

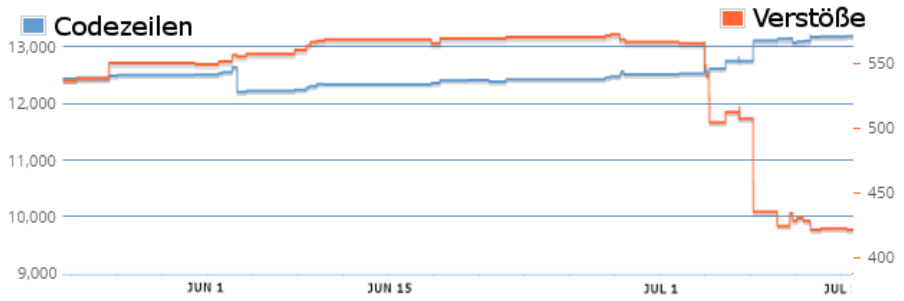


Abbildung 8.1.: Entwicklung der Wartbarkeit gemessen von Teamscale

8.1.4. Korrektheit

Nach den vorgenommenen Änderungen an SIFEI werden noch immer alle Unit-Tests in SIFCore erfolgreich beendet. Der Regressionstest hat also keine Fehler gefunden.

9. Zusammenfassung und Ausblick

9.1. Zusammenfassung

Das an der Universität Stuttgart entwickelte Werkzeug SIFEI hilft Fehler in Spreadsheets zu beheben und zu vermeiden. Um einen sinnvollen Einsatz in der Praxis zu ermöglichen, sollte SIFEI erweitert und verbessert werden.
Als erstes wurden Bewertungsverfahren gesucht, um SIFEI zu untersuchen. Betrachtet werden mussten die Aspekte Wartbarkeit, Zuverlässigkeit, Korrektheit und Usability. Nachdem geeignete Verfahren gefunden wurden, wurde SIFEI mit diesen bewertet. Da diese Verfahren sehr viele Schwächen aufgedeckt haben, mussten diese weiter priorisiert werden. Auch hierfür musste ein geeignetes Verfahren gefunden werden. Die Schwächen wurden gruppiert, um so entscheiden zu können, welche Schwächen verbessert werden.

Dann wurden Änderungen vorgenommen, die versuchten, die Schwächen zu beheben. Zum Beispiel wurde versucht, die Wartbarkeit von SIFEI zu verbessern. Diese Änderungen wurden vorgenommen, um zukünftige Erweiterungen und Verbesserung zu vereinfachen. Vor allem die Anzahl der Kommentare wurde erhöht. Im Bereich der graphischen Benutzeroberfläche wurden viele kleine Änderungen vorgenommen. Mit diesen sollte die Usability verbessert werden. So wurde zum Beispiel die Interaktion mit den Befund-Icons verbessert, das Zeichnen dieser beschleunigt, mehr Rückmeldungen bei Prüfungen eingebaut und das Bearbeiten von Szenarios vereinfacht.
Durch die abschließende Bewertung konnte beurteilt werden, welche Änderungen zu Verbesserungen geführt haben. Größtenteils wurden die gleichen Bewertungsverfahren verwendet, wie bei der initialen Bewertung von SIFEI. Die Usability wurde aber noch zusätzlich durch eine Benutzerstudie bewertet. Sie ergab, dass sich die Usability von SIFEI verbessert hat. Trotzdem kann diese noch weiter erhöht werden, da noch immer Funktionen existieren, die nicht intuitiv sind.
Die Zuverlässigkeit wurde indirekt durch Verbesserungen an der Usability erhöht. Da nur eine Prüfung gleichzeitig möglich ist, kommt es zu weniger Kommunikationsproblemen zwischen SIFEI und SIFCore.
Auch die Wartbarkeit von SIFEI hat sich deutlich verbessert. Da in manchen Teilen des Quellcodes kaum Kommentare vorhanden waren, wurden die wichtigsten Kommentare ergänzt. Dies sollte die nötige Einarbeitungszeit erheblich vermindern.

Somit konnten alle geänderten Aspekte verbessert werden. Hierdurch wurde SIFEI einem produktiven Einsatz in der Praxis weiter angenähert.

9.2. Ausblick

In zukünftigen Arbeiten sollte die Zusammenarbeit zwischen SIFEI und SIFCore weiter verbessert werden . Hierdurch könnte eine höhere Ausfallsicherheit erzielt werden. Diese ist wichtig, damit SIFEI in der Praxis zuverlässig verwendet werden kann. Im Moment kommunizieren SIFEI und SIFCore, sowohl über einen Socket, als auch über XML-Dateien. Diese Kommunikation sollte vereinheitlicht werden, da so der Umfang des Programms vermindert und die Kommunikation einfacher modifiziert werden kann.
Um die Benutzererfahrung noch weiter zu verbessern, sollten Prüfungen auch im Hintergrund ausgeführt werden. So wird der Nutzer weniger in seinem Arbeitsfluss gestört. Außerdem würde eine einfachere Szenario-Erstellung die Nutzung vereinfachen.
Bei all diesen Erweiterungen und Verbesserungen, sollte aber auch auf die Qualität des Quellcodes geachtet werden. Vor allem die Lesbarkeit und die Kommentierung sollten weiter verbessert werden, da immer noch ein Teil des Quellcodes keine oder wenig Kommentare enthält.

Nach dieser Bachelorarbeit scheinen die in SIFEI gebotenen Funktionen zur Prüfung von Spreadsheets ausreichend. Der Nutzer kann Prüfungen innerhalb von Excel konfigurieren, verwalten und starten; er kann die Ergebnisse betrachten und bekommt Hilfe um Mängel zu beheben. Daher sollte in zukünftigen Version die vorhandenen Funktionen weiter verbessert und vereinfacht werden.

Literaturverzeichnis

[BA05] Berander, Patrik ; Andrews, Anneliese: Requirements prioritization. (2005), S. 69–94 (Zitiert auf Seite 27)

[Bec14] Beck, Sebastian: Spreadsheet-Fehlermuster. (2014) (Zitiert auf Seite 14)

[BFB99] Beck, Kent ; Fowler, Martin ; Beck, Grandma: Bad smells in code. In: *Refactoring: Improving the design of existing code* (1999), S. 75–88 (Zitiert auf den Seiten 23 und 34)

[But00] Butler, Raymond J.: Is this spreadsheet a tax evader? How HM Customs and Excise test spreadsheet applications. In: *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on* IEEE, 2000, S. 6–pp (Zitiert auf Seite 11)

[cqs] cqse: *Teamscale.* https://www.cqse.eu/de/produkte/teamscale/ueberblick/ (Zitiert auf Seite 18)

[Dou13] Doust, Ehssan: Visualisierung von Fehlern in Spreadsheets. (2013) (Zitiert auf Seite 14)

[eus] *European Spreadsheet Risks Interest Group.* http://www.eusprig.org/ (Zitiert auf Seite 11)

[Hal77] Halstead, Maurice H.: *Elements of software science.* Bd. 7. Elsevier New York, 1977 (Zitiert auf Seite 18)

[IEE04] IEEE Draft Standard for Software Engineering - Software Life Cycle Processes - Maintenance. In: *IEEE Std P14764, Nov 2004* (2004) (Zitiert auf Seite 17)

[JN94] *Kapitel* The cognitive walkthrough method. A practioner's guide. In: Jakob Nielsen, Robert L. Mack (.: *Usability Inspection Methods.* John Wiley & Sons, 1994, S. 105–140 (Zitiert auf Seite 16)

[KKW16] Käfer, Verena ; Kulesz, Daniel ; Wagner, Stefan: What is the best way for developers to learn new software tools? An empirical comparison between a text and a video tutorial / PeerJ Preprints. 2016. – Forschungsbericht (Zitiert auf Seite 40)

[Lau02] Lauesen, Soren: *Software requirements: styles and techniques.* Pearson Education, 2002 (Zitiert auf Seite 28)

[Lem13] Lemcke, Manuel: Dynamische Prüfung von Spreadsheets. (2013) (Zitiert auf Seite 14)

[LL12] Ludewig, Jochen ; Lichter, Horst: *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken.* dpunkt. verlag, 2012 (Zitiert auf Seite 21)

[LW00] Leffingwell, Dean ; Widrig, Don: *Managing software requirements: a unified approach.* Addison-Wesley Professional, 2000 (Zitiert auf Seite 27)

[McC76] McCabe, Thomas J.: A complexity measure. In: *Software Engineering, IEEE Transactions on* (1976), Nr. 4, S. 308–320 (Zitiert auf Seite 17)

[OHA91] Oman, P ; Hagemeister, J ; Ash, D: A Definition and Taxonomy for Software Maintainability, report SETL Report 91-08-TR. In: *University of Idaho* (1991) (Zitiert auf Seite 17)

[Pan98] Panko, Raymond R.: What we know about spreadsheet errors. In: *Journal of Organizational and End User Computing (JOEUC)* 10 (1998), Nr. 2, S. 15–21 (Zitiert auf Seite 11)

[Pan08] Panko, Raymond R.: Spreadsheet errors: What we know. what we think we can do. (2008) (Zitiert auf Seite 11)

[RGK90] Radatz, Jane ; Geraci, Anne ; Katki, Freny: IEEE standard glossary of software engineering terminology. In: *IEEE Std* 610121990 (1990), Nr. 121990, S. 3 (Zitiert auf den Seiten 19 und 20)

[Röd12] Röder, Holger: *Usability Patterns: eine Technik zur Spezifikation funktionaler Usability-Merkmale.* Cuvillier, 2012 (Zitiert auf den Seiten 15 und 16)

[Saa08] Saaty, Thomas L.: Decision making with the analytic hierarchy process. In: *International journal of services sciences* 1 (2008), Nr. 1, S. 83–98 (Zitiert auf Seite 27)

[Sav07] Savoia, Alberto: *Crap official blogpost.* http://www.artima.com/weblogs/viewpost.jsp?thread=210575. Version: 07 2007 (Zitiert auf Seite 18)

[Sch14] Scheurich, Jonas: *Benutzerschnittstelle fuer einen Spreadsheet-Pruefstand*, Universitaet Stuttgart, Bachelor Arbeit, 2014 (Zitiert auf den Seiten 13 und 14)

[SS97] Sommerville, Ian ; Sawyer, Pete: *Requirements engineering: a good practice guide.* John Wiley & Sons, Inc., 1997 (Zitiert auf Seite 27)

[Tot14] Toth, Fabian: Live-Prüfung von Spreadsheets während der Bearbeitung. (2014) (Zitiert auf Seite 14)

[Wel01] Welker, Kurt D.: The software maintainability index revisited. In: *CrossTalk* 14 (2001), S. 18–21 (Zitiert auf Seite 17)

[WS80] Wind, Yoram ; Saaty, Thomas L.: Marketing applications of the analytic hierarchy process. In: *Management science* 26 (1980), Nr. 7, S. 641–658 (Zitiert auf Seite 27)

[Zit12] Zitzelsberger, Sebastian: *Fehlererkennung in Spreadsheets*, Universitaet Stuttgart, Bachelorthesis, 2012. http://elib.uni-stuttgart.de/handle/11682/2821 (Zitiert auf den Seiten 13 und 14)

Alle URLs wurden zuletzt am 10. Oktober 2016 geprüft.

A. Schwächen von SIFEI

A.1. Im Issue Tracker vorher eingetragene Schwächen

Speichern

Diese Schwächen sind nach Kategorien gruppiert

- Es wird eine Ausnahme geworfen, falls das Spreadsheet im csv Format gespeichert ist.
- Die Datei muss vor der ersten Prüfung gespeichert sein.

Multithreading

- SIFEI ist nicht multithreadingsicher.
- Wenn SIFEI gestartet wird, soll überprüft werden, ob SIFCore schon ausgeführt wird. Falls dies nicht der Fall ist, wird die Ausführung von SIFCore gestartet.

Regelanwendung

- Die Prüfung auf die Leserichtung von Formeln soll nur innerhalb eines Worksheets angewendet werden können.

Undo/Redo

- Durch das Verwenden von SIFEI wird die globale Undo/Redo Funktion von Excel so gestört, dass sie nicht mehr zuverlässig verwendet werden kann. Vor allem das Zeichnen der Befundsicons verhindert eine zuverlässige Anwendung dieser Funktion.

Szenarios

- Sind in einem Szenario die Ergebniswerte Gleitkommazahlen, stürzt unter Umständen SIFEI und / oder SIFCore ab.
- Wird die Erstellung eines Szenarios abgebrochen, werden nur teilweise die eingetragenen Werte verworfen.
- Es kann passieren, dass Befund-Icons verschwinden, falls das Szenariofenster geöffnet wird.
- Soll ein leeres Szenario erstellt werden, kommt erst zu spät eine Fehlermeldung, dass dies nicht möglich ist.
- Es wird keine Warnung angezeigt, wenn einzelne Zellen eines Szenarios gelöscht werden sollen.
- Das Bearbeiten von Szenarios sollte nicht standardmäßig eingeschaltet sein.
- Szenarien funktionieren nur auf dem ersten Worksheet des Spreadsheets.
- Bei der Definition eines neuen Szenarios sollten sich alle Zellen visuell verändern, damit deutlich wird, dass man sich im Szenarieneditor befindet. Nur solche Zellen die auch als Eingabe-, Zwischenergebnis oder Ergebniszellen markiert wurden, sollten bearbeitet werden können und sollten dies visuell verdeutlichen.

Seitenleiste

- Markierte Einträge in der Seitenleiste bleiben markiert, auch wenn danach nur eine Teilmenge dieser Zellen ausgewählt / markiert wird.
- Wird ein Eintrag der Seitenleiste ausgewählt und dann auf eine andere Zelle geklickt, kann durch den erneuten Klick auf den Eintrag der Seitenleiste der Fokus nicht auf diesen zurückgebracht werden.

Befunde

- Befunde welche mit “später“ markiert wurden, tauchen bei einer erneuten Prüfung wieder in der originalen Liste auf und verschwinden aus der “später“-Liste.

Windows Presentation Framework / Windows Forms

- Die Klasse PolicyConfiguration verwendet im Moment noch Windows Forms anstatt des Windows Presentation Frameworks

A.2. Neu gefundene Schwächen

Diese Schwächen sind nach den Verfahren bzw. Werkzeugen gruppiert mit denen sie gefunden wurden.

A.2.1. Visual Studie Code Analysis

Globalisierung

Von der Visual Studio Code Analysis wurden einige Benutzerschnittstellentexte gefunden die noch nicht globalisiert sind. Auch wurden viele falsch positive Globalisierungsfehler gefunden, da viele Methoden mit festen Stringparametern aufgerufen werden.

Performance

Es wurden besonders häufig Strings auf Leerheit oder NULL überprüft, mit einem Verfahren, dass laut Visual Studio Code Analysis langsamer ist, als das alternativ vorgeschlagene.

Außerdem wurden noch einige kleinere Fehler gefunden die im Folgenden aufgelistet werden:

- Kleinere Mängel bei den Coding Standards von Microsoft.
- Es werden Methoden aufgerufen deren Ergebnisse aber nie verwendet werden.
- Die Benennung der Methoden entspricht nicht den Microsoft Guidelines.
- Häufig werden Bezeichner wie “this“ verwendet, auch wenn sie nicht nötig sind.
- Einige Collection Properties sind nicht schreibgeschützt, wodurch sich Member hinzufügen und löschen lassen.

A.2.2. Usability-Patterns

Globales Undo

Das globale Undo funktioniert bei Excel mit SIFEI nur eingeschränkt, da z.B. das Zeichnen eines Befund-Icons eine Aktion darstellt, die nicht rückgängig gemacht werden kann.

Abbruch

Eine gestartete Prüfung kann vom Benutzer nicht abgebrochen werden. Gerade bei automatisch gestarteten oder bei besonders zeitintensiven Prüfungen, sollte es für den Benutzer möglich sein eine Prüfung abzubrechen.

Globales Redo

Siehe Globales Undo.

Warnungen

In SIFEI sind ein paar Benutzerinteraktionen möglich, bei denen eine Warnung vor der Ausführung nötig wären. Hierbei handelt es sich Beispielsweise um:

- Der Button “Befundmarker entfernen“ entfernt alle vorhandenen Befund-Icons des Spreadsheets. Gerade bei großen Spreadsheets sollte hier noch einmal nachgefragt werden, da eine erneute Prüfung sehr lange brauchen kann.
- Beim Erstellen eines Szenarios ohne Eingabe-, Zwischenergebnis- und Ergebniszellen sollte die Warnung, dass ein leeres Szenario nicht erstellt werden kann, vor der Namensabfrage kommen.

Fortschrittsanzeige bzw. Verarbeitungsanzeige

Beim Prüfen eines Spreadsheets wäre es für den Nutzer vorteilhaft, wenn ersichtlich ist, dass eine Prüfung stattfindet.

Ausführung im Hintergrund

Wird eine Prüfung gestartet, erfolgt dieser im Vordergrund und blockiert Excel und SIFEI. Gerade bei längeren Prüfungen wäre eine Ausführung im Hintergrund notwendig, da sonst nicht am Spreadsheet weitergearbeitet werden kann.

Filter

In der Seitenleiste ist es nicht möglich die Befunde zu filtern. Es erfolgt nur eine Unterteilung in Befunde und Szenarien und bei den Befunden auf die Kategorien “neu“ “später“ und “ignoriert“. Es kann jedoch nicht nach Art von Befunden gefiltert werden.

Assistent

Gerade die Szenarienerstellung ist nicht intuitiv. Ein Assistent könnte die ersten Schritte erheblich erleichtern.

Systemstatus

Dem Nutzer wird nicht zurückgemeldet was das System gerade macht (aktuelle Prüfung, Szenarioerstellung etc.). Ihm sollte aber rückgemeldet werden, in welchem Status SIFEI sich befindet, damit er es korrekt und effizient verwenden kann.

A.2.3. Teamscale Befunde

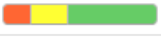
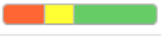
Files	106
Lines of Code	12,424
Source Lines of Code	9,410
File Size Assessment	
Longest Method Length	437
Method Length Assessment	
Maximum Nesting Depth	6
Nesting Depth Assessment	
Last Change Date	Feb 15 2016 00:54
Change Count	120
Number of Findings	536
Findings Density	43.1
Clone Coverage	6.7%
Comment Completeness Assessment	

Abbildung A.1.: Teamscale-Bewertung von SIFEI vor Beginn der Bachelorarbeit

A.2.4. Auswertung der Freitextkommentare in der Studie

In der durch Daniel Kulesz durchgeführten Studie hatten die Teilnehmer die Möglichkeit Freitextkommentare zu verfassen, um Kritik und Änderungswünsche an SIFEI zu äußern. Nachfolgend sind diese thematisch sortiert aufgelistet.

Usability

- Wird ein Befund gefunden taucht der Name der betroffenen Zelle nicht in der Seitenleiste von SIFEI auf.
- Nachdem ein Szenario erstellt wurde, muss erst manuell auf den Prüfung-Starten-Button gedrückt werden, da sonst die Auswertung des Szenarios nicht ersichtlich ist.
- Bei einigen Befundtypen sind die Fehlermeldungen nicht hilfreich.
- Nach einer Prüfung sollte die Seitenleiste automatisch geöffnet werden, damit die Ergebnisse der Prüfung sofort ersichtlich sind.
- Bei einer laufenden Prüfung sollte ein Fortschrittsbalken angezeigt werden, bzw. nach einer erfolgten Prüfung sollte dies dem Benutzer rückgemeldet werden.
- Wenn eine große Anzahl an Befunden vorhanden ist, überfordert diese den Nutzer.
- Sind bei einer Prüfung keine Befunde gefunden worden, sollte dies dem Nutzer mitgeteilt werden. Im Moment erfolgt keinerlei Rückmeldung.

Erleichterte Bedienung

- Es sollte eine Hilfe bzw. Erklärungen zu den einzelnen Regeln geben und warum sie Sinn ergeben.
- Die Szenarioerstellung erscheint nicht intuitiv, da sie die eigentlichen Rechenvorgänge nicht mit einschließt. Dies macht es schwer Fehler zu überprüfen.
- Es sollten genauere Fehlerbeschreibungen der Szenarios vorhanden sein.
- SIFEI sollte allgemein etwas übersichtlicher aufgebaut sein.
- Anstatt nur eine Fehlermeldung auszugeben, wäre es gut, wenn SIFEI direkt Lösungsvorschläge liefern könnte.
- Das Icon um Szenarios zu erstellen, passt nicht zu dem Begriff “Szenario“.
- Die Befunde in der Seitenleiste sollten etwas übersichtlicher dargestellt werden.

Featurewünsche

- Es sollte möglich sein, bei “SVERWEIS“, auf nicht sichtbare (weiße) Zahlen zu prüfen.
- Ein integrierter Taschenrechner für die manuelle Prüfung der Szenarioergebnisse wäre hilfreich.
- Es sollte möglich sein, eine Prüfung nur auf dem aktuellen Blatt oder nur im ausgewählten Bereich durchzuführen.

Bugs

- Die Grafik für die Befunde in Zellen sollte verbessert werden.
- Die Prüfung automatisch auszuführen funktioniert noch nicht richtig.
- Das horizontale Scrollen des Fensters für die Bearbeitung von Szenarios ist schwierig.
- Beim Klick auf einen Befund in der Seitenleiste sollte zuverlässig in die dazugehörige Zelle gesprungen werden.
- Änderungen sollten direkt vorgeschlagen werden, wobei diese dann nur noch mit ja/nein bestätigt werden sollen.

Kritik am Backend

- Ein paar der Designrichtlinien sind nicht ganz ausgereift bzw. zeigen nicht unbedingt Fehler auf.
- Es sollte mehr Prüfregeln zur Auswahl geben.

Sonstige Anmerkungen

- Fehler die durch die Szenarien aufgezeigt werden sind manchmal relativ schwer zu finden.
- Erfolgreiche Szenarios sollten bewerten werden können.
- Werte der geprüften Szenarios sollten ersichtlich sein.

A.2.5. Befunde aus Videos der Benutzerstudie

- Die Benutzer sehen sich sehr häufig nicht die Details der einzelnen Befundes an. Diese Details sind zwar einfach zu öffnen, sehen aber sehr unorganisiert aus. Außerdem scheinen sie den Nutzer eher zu überfordern, als dass sie ihm helfen.
- Die Benutzer haben einen Befund korrigiert, merken dies aber nicht, da sie keine Prüfung anstoßen und dadurch weiter nach dem Fehler suchen.
- Es gibt Probanden, die nach beinahe jeder Eingabe, eine manuelle Prüfung anstoßen. Auch hier wäre eine zuverlässige automatische Prüfung hilfreich.
- Die meisten Probanden versuchen Eingaben rückgängig zu machen, was ihnen jedoch nicht gelingt, da SIFEI die globale Undo/Redo Funktion unbrauchbar macht.
- Erstellen die Probanden neue Szenarios haben manche Probleme die Eingabe-, Zwischenergebnis- und Ergebniszellen voneinander abzugrenzen.

A.2.6. Zuverlässigkeits Test

Gefunden wurden folgende reproduzierbare Fehler:

- Das Zeichnen der Befund-Icons dauert sehr lange. Ab 50 Icons dauert das Zeichnen so lange, dass es sehr unangenehm auffällt. Außerdem werden alle Icons aus allen Worksheets für die Dauer des Zeichnens auf dem ersten Worksheet angezeigt. Dies verwirrt den Benutzer vor allem dann, wenn die enthaltenden Zellen leer oder unterschiedlich groß sind.
- Auch beim Scrollen verlangsamen die eingezeichneten Befund-Icons Excel. Hierdurch ist kein flüssiges Scrollen mehr möglich. Hierdurch leidet die Usability sehr, da so Fehler nur schwer korrigiert werden können und das Weiterarbeiten am Spreadsheet sehr erschwert wird.
  Außerdem kann es beim Scrollen dazu kommen, dass die Befund-Icons in den falschen Zellen angezeigt werden, da sie langsamer als das Dokument selbst gescrollt werden.
- Werden Zellen, die Befundsicons enthalten, ausgeschnitten und an einer anderen Stelle wieder eingefügt, so werden teilweise nicht alle Befund-Icons mit verschoben.
- Werden viele Prüfungen hintereinander oder ein großer mit sehr vielen Befunden oder Zellen gestartet, so stürzen sowohl SIFEI als auch SIFCore häufig ab. Der Absturz erfolgt wegen einer XML Parse Exception. Stürzt SIFCore ab, startet SIFEI diese Komponente bei darauffolgenden Prüfungen nicht mehr neu.
  Der kritische Punkt an hintereinander ausgeführten Prüfungen bzw. zu vielen Befunden ist auf Rechnern mit schwächerer CPU und langsamen Festplatten schneller erreicht.

A.2.7. Priorisierung der Schwächen

Die Schwächen wurden folgendermaßen in Gruppen eingeteilt:

Sehr hohe Priorität

In dieser Gruppen befinden sich die Schwächen, die behoben werden müssen, damit SIFEI wirklich produktiv verwendet werden kann.

- Beim Scrollen tauchen "Ghostmarker“ auf. Das heißt beim Scrollen tauchen Icons in leeren Zellen kurz auf.
- Keine Perfomanz beim Scrollen wenn über 50 Befunde vorhanden sind.
- SIFEI zerstört die globale Redo/Undo Funktion, da sie nur eingeschränkt funktioniert.
- Szenarios, die als Output eine Gleitkommazahl haben führen zu einem Absturz von SIFEI/SIFCore.
- Befunde die mit “später“ markiert wurden, tauchen bei einer erneuten Prüfung wieder in der originalen Liste auf. In der “später“-Liste tauchen sie gar nicht auf.
- Eine Ausführung im Hintergrund wird im Moment von SIFEI noch nicht unterstützt. Bei einer Prüfung oder dem Neuzeichnen der Icons, kann der Nutzer keine Eingaben tätigen.
- Wird eine Prüfung beendet, sollte eine Rückmeldung beim Nutzer erfolgen. Wenn also keine Befunde gefunden wurden, sollte dies kommuniziert werden.
- Es kommt zu einem Absturz, wenn das Spreadsheet nicht als “.xls“ oder “.xlsx“ gespeichert ist.
- Datei muss vor der ersten Prüfung gespeichert sein.
- Szenarios funktionieren nur auf dem ersten Tabellenblatt.
- Absturz des Programms mit einer XML Parse Exception.

Hohe Priorität

In dieser Gruppen befinden sich die Schwächen die behoben werden müssen, damit ein möglicher Nutzer das Programm auch verwenden will.

- Automatische Prüfungen sollten möglich sein.
- Eine genauere Fehlerbeschreibungen der Befunde sollte vorhanden sein.
- Eine vorgenommene Prüfung kann vom Benutzer nicht abgebrochen werden.
- Eine Fortschrittsanzeige fehlt bei einer Prüfung komplett. Es wird von SIFEI nicht zurückgemeldet, wenn eine Prüfung stattfindet.
- Das horizontales Scrollen bei offener Leiste ist schwierig.
- Der Fokus verschwindet von der aktuell ausgewählten Zelle und das Zeichnen von Icons ist sehr langsam.
- Nach dem Ausschneiden von Zellen mit Befunden bleiben die Icons in der alten Zelle enthalten.
- Geistericon: Falls mehr als drei gleiche Befunde untereinander gefunden werden, kann das dritte Icon eventuell nicht mehr gelöscht werden.
- Der Name einer betroffenen Zelle ist nicht immer in der Seitenleiste enthalten.
- Die Leserichtung darf nur innerhalb eines Worksheets betrachtet werden. Nicht über mehrere hinweg.
- SIFEI ist nicht “Multithreadingsicher“.
- SIFEI sollte überprüfen, ob schon eine SIFCore Instanz läuft. Wenn keine läuft, soll eine neue gestartet werden.

Mittlere Priorität

In dieser Gruppe befinden die Schwächen die SIFEI merkbar verbessern würden, die aber keine Voraussetzung für eine flüssige Verwendung sind:

- Eine Hilfe zu den Regeln und deren Sinn wäre hilfreich.
- Es wurden einige Probleme bezüglich der Globalisierung gefunden. Viele davon sind falsch positiv, es gibt aber auch richtig positive.
- In der Seitenleiste ist kein Filter nach Typ der Befunde möglich (Z.b: Leserichtung oder Konstanten in Formeln).

- Befunde in der Seitenleiste bleiben markiert solange eine Untermenge von Zellen markiert ist.
- Wird ein Befund in der Seitenleiste ausgewählt und dann auf eine andere Zelle geklickt, kann durch einen erneuten Klick auf den ersten Befund der Seitenleiste der Fokus nicht zurückgebracht werden.
- Beim Erstellen von Szenarien ist es nicht klar, dass man sich gerade in diesem Modus befindet. Daher kann es passieren, dass Nutzer eine Prüfung starten.
- “Befundmarker entfernen“ löscht ohne nochmaliges Bestätigung alle Befund-Icons. Da eine Prüfung lange Zeit dauern kann, sollte dies verhindert werden.
- Ab einer gewissen Anzahl von Zellen bzw. Befunden (160+) stürzt SIFEI mit einer “XML parse exception“ ab.
- Wird die Erstellung eines Szenarios abgebrochen, werden eingetragenen Werte verworfen. Dies passiert aber nicht für alle Werte, was zu inkonsistenten Verhalten führt.
- Wird das Szenariofenster geöffnet, verschwinden Befund-Icons und tauchen an einer falschen Stelle wieder auf.
- Bei der Definition von Szenarios werden nicht alle Zellen abgedeckt. Nur solche die als Eingabe-, Zwischenwerts- oder Ergebniszelle definiert sind. Wenn möglich sollten alle Zellen abgedeckt werden. Außerdem sollte sichtbar sein, für welche Werte eingetragen werden sollen.
- Keine Warnung wenn Zellen eines Szenarios gelöscht werden sollen.

Niedrige Priorität

In dieser Gruppe befinden sich die restlichen Schwächen, die durchaus verbessert werden sollten, aber die keine großen Einschränkungen für die Verwendung von SIFEI darstellen:

- Details von Befund-Icons sollten mit dem linken Mausbutton geöffnet werden können.
- Ein Assistent bzw. eine Hilfe für die Erstellung von Szenarien fehlt komplett. Dies wäre vor allem eine gute Ergänzung, da dies nicht ganz intuitiv ist.
- Kleinere Mängel bei den C#-Coding-Standards.
- Die Namen von Methoden entspricht nicht den Microsoft Guidelines.
- Es ist redundanter Code enthalten; z.B. die Verwendung von “this.“ wo es nicht nötig ist.
- Vergeleiche, ob string leer oder null ist, sollte perfomanter geschehen.

- Es existieren Methoden, welche aufgerufen werden, deren Ergebnisse nicht verwendet werden.
- Die Seitenleiste sollte automatisch geöffnet werden, wenn eine Prüfung gestartet wird.
- Probanden hatten manchmal Probleme Eingabe-, Zwischenergebniss- und Ergebnisszellen voneinander abzugrenzen.
- Es gibt keine Fehlermeldung, falls leere Szenarios erstellt werden sollen (ohne Eingabe/Zwischenwert/Ergebniszellen). Erst wenn das Szenario gespeichert werden soll kommt eine Fehlermeldung.
- Das Bearbeiten von Szenarios sollte nicht ohne einen zusätzlichen Schritt möglich sein. Erst mittels eines Bearbeiten-Buttons sollte dies möglich sein. Im Moment werden bearbeitete Szenarios gespeichert, indem die Seitenleiste geschlossen wird.
- Es sollten mehr Prüfregeln zur Auswahl stehen. Als Beispiel wurde die Unterstützung beim “SVERWEIS“, auf nicht sichtbare (weiße) Zahlen genannt.
- Es sollten Änderungen an Zellen direkt vorgeschlagen werden. Dann sollte die Änderung nur kurz akzeptiert oder verworfen werden. Außerdem wäre eine ausführliche Hilfe, bei der eine typische Lösung für das Problem gegeben wird, sinnvoll.

Erklärung

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

Ort, Datum, Unterschrift