www.qualicen.de
www.munichre.com

Requirements-Smells: Automatische Unterstützung bei der Qualitätssicherung von Anforderungsdokumenten

Manuelle Reviews sind effektiv, um die Qualität von Anforderungsdokumenten sicherzustellen. Sie sind aber auch arbeitsintensiv und es dauert lange, bis der Autor Feedback bekommt. Daraus folgt, dass manuelle Reviews oft nur unregelmäßig durchgeführt werden. In diesem Artikel wird beschrieben, wie manuelle Reviews durch automatische Techniken ergänzt werden können. Damit ist es Requirements-Engineers möglich, bereits während des Schreibens oder des Änderns von Dokumenten wertvolles Feedback zu bekommen, ohne auf die Ergebnisse von manuellen Reviews warten zu müssen.

Anforderungen sind das primäre Mittel, um die Ziele von Softwareprojekten zu kommunizieren: Requirements-Engineers erheben Anforderungen und vermitteln diese unter anderem an Entwicklungs- und Testteams. Da diese Kommunikation meist asynchron abläuft und oft, unter anderem aus gesetzlichen Gründen, schriftlich festgehalten werden muss, werden Anforderungen in Anforderungsdokumenten niedergeschrieben. Diese bestehen neben Diagrammen und Tabellen größtenteils aus Text in natürlicher Sprache. Dadurch sind sie einfach zu erstellen und für alle Stakeholder lesbar (**siehe Abbildung 1**).

Qualität von Anforderungsdokumenten ist essenziell

Natürliche Sprache eröffnet allerdings auch das Risiko, sich unpräzise oder missverständlich auszudrücken. Sind Anforderungsdokumente schlecht geschrieben, so hat das Auswirkungen auf das gesamte Projekt:

- Kompliziert formulierte Anforderungen erhöhen den Aufwand beim Lesen.
- Missverständliche Anforderungen führen zu unnötigem Aufwand durch Rückfragen oder sogar zu Fehlinterpretation.
- Kompliziert geschriebene Anforderungen werden auf lange Zeit vernachlässigt und irgendwann gänzlich ignoriert, weil der Einarbeitungsaufwand zu hoch ist.
- Fehlen Teile von Anforderungen, kann das dazu führen, dass ein Produkt entwickelt wird, das für Nutzer nicht zu gebrauchen ist.

Manuelle Reviews sind gut, aber aufwändig

Um eine hohe Qualität von Anforderungsdokumenten sicherzustellen, haben sich manuelle Reviews als effektives Mittel zur Identifikation von Qualitätsproblemen bewährt. Hierbei werden Anforderungsdokumente von anderen Personen des Projektteams manuell inspiziert und Qualitätsdefekte angemerkt. Im Anschluss wird das annotierte Anforderungsdokument an den ursprünglichen Autor zur Nachbesserung übergeben. Obwohl diese Methode sehr effektiv ist, hat sie leider auch Nachteile:

- Sowohl die Überprüfung und Anmerkung durch Reviewer, als auch die Koordination und Kommunikation der Review-Ergebnisse sind sehr arbeitsintensiv.
- Manuelle Reviews haben lange Feedback-Zeiten: Erst nachdem die Reviewer das Dokument gesichtet und bewertet haben, erhält der Autor Rückmeldung auf das Dokument und kann die Anmerkungen der Reviewer einarbeiten.
- Viele Review-Kommentare hängen nicht mit inhaltlichen Problemen zusammen, sondern beziehen sich auf ungenaue Formulierungen, Abweichungen von der Unternehmens-Guideline oder ähnlichen, einfach zu entdeckenden Mängeln. Dies kostet unnötig Zeit und geht zu Lasten der Konzentration auf schwierige, inhaltliche Probleme.

Die Folge des hohen Review-Aufwands ist, dass Anforderungsdokumente oft nur unregelmäßig inspiziert werden. Dieser Effekt verstärkt sich insbesondere unter dem üblichen Projekt-Zeitdruck. Dennoch stellen manuelle Reviews heutzutage die mit Abstand wichtigste Qualitätssicherungsmethode dar.

Automatische Qualitätssicherung von Anforderungen

Die genannten Probleme bei Reviews werfen die Frage auf, ob Gutachter durch automatisierte Prüfungen unterstützt werden können und die Reviews so effizienter durchgeführt werden können. Ein Beispiel für eine sehr einfache Art einer solchen automatischen Unterstützung sind Rechtschreib- und Grammatikprüfungen, wie sie in Textverarbeitungsprogrammen heutzutage üblich sind. Sie helfen, Fehler schon während der Texterstellung zu vermeiden. (Die Kombination von schnellen automatischen Erkennungstechniken mit manuellen Reviews wurde in der Wissenschaft unter dem Begriff *Phased Inspections* eingeführt – **siehe Abbildung 2**.) Dabei behandelt der Autor im ersten Schritt alle Fehler oder

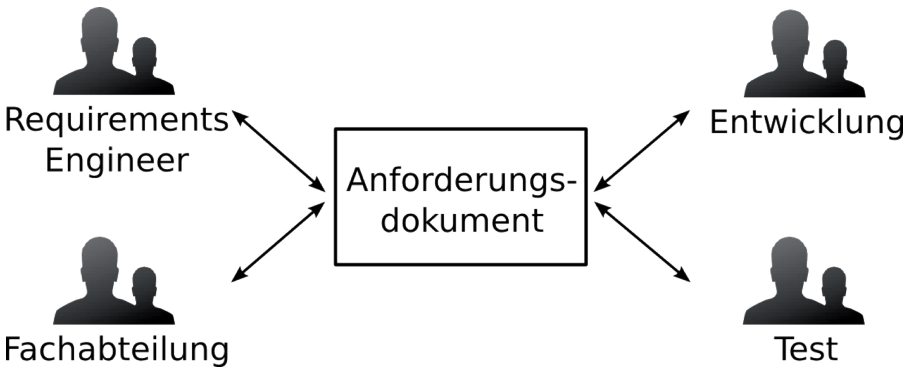


Abb. 1: Anforderungsdokumente sind das zentrale Artefakt in der Softwareentwicklung.

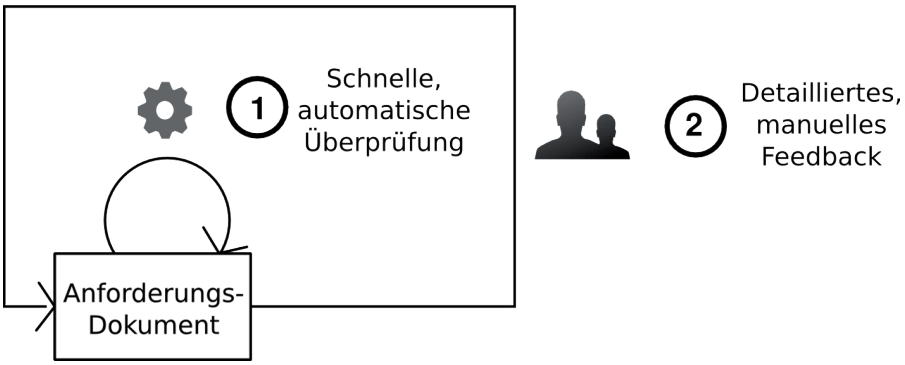


Abb. 2: Phased Inspection von Anforderungsdokumenten.

Hinweise, die ein entsprechendes Werkzeug automatisch detektiert. Erst anschließend geht das Dokument in das langwierige, aber effektive manuelle Review.
Um zu klären, welche Anforderungsprobleme automatisch entdeckt werden können, ist zunächst folgende Unterscheidung hilfreich:

- *Strukturelle Probleme* beziehen sich auf die Dokumentstruktur der Anforderungen. Ein Beispiel für ein Problem dieser Kategorie ist die Abweichung von vorgegebenen Dokumentenvorlagen für die Anforderungsdokumentation.
- *Lexikalische Probleme* beziehen sich auf einzelne Begriffe in den Anforderungen. Der Einsatz von vagen Begriffen wie „schnell" oder Weichmachern wie „wenn möglich" fallen in diese Problemkategorie.
- *Grammatikalische Probleme* beziehen sich auf die Satzstruktur der Anforderungen. Ein Beispiel ist die Nutzung von Passivsätzen, da sie den Akteur verschleiern können.
- *Semantische Probleme* beziehen sich auf den Inhalt der Anforderungen. Inkorrekte oder unvollständige Anforderungen sind Beispiele für diese Problemklasse.

Vor allem semantische Probleme lassen sich nur schwer eindeutig automatisch aufdecken. Häufig finden sich jedoch Indizien, die auf solche Probleme hindeuten. Hat beispielsweise ein Ablauf in einem Use-Case nur einen Schritt, so ist dies ein Indiz auf einen möglicherweise unvollständigen Ablauf. In Anlehnung an Code-Smells (Stellen im Programmcode, die auf Qualitätsmängel und notwendige Überarbeitungen hindeuten, beispielsweise lange Methoden) werden solche Hinweise auf Mängel in Anforderungen *Requirements-Smells* genannt.

Beispiele für Requirements-Smells

Für viele praktisch bedeutsame Probleme in Anforderungen existieren Requirements-Smells als Indikatoren. Ein Beispiel für einen strukturellen Smell sind Verweise auf *undefinierte Referenzpunkte*. Dieser Smell ist spezifisch für Anforderungen in Form von Anwendungsfällen mit Szenarienbeschreibung. In solchen Szenarien werden normalerweise Standardabläufe spezifiziert und Alternativabläufe an festgelegten Stellen eingehängt. Ist die angegebene Stelle allerdings nicht vorhanden, ist das ein Hinweis auf ein unvollständiges oder fehlerhaftes Szenario (**siehe Abbildung 3**).
Der *Vage-Phrasen*-Smell kennzeichnet ungenaue Ausdrücke, die von einem Entwickler anders gedeutet werden können, als es die Intention des Anforderers war. Ein Beispiel hierfür ist der Begriff „other": In der Anforderung in **Abbildung 4** ist es beispielsweise unklar, zu welchen Dateien eine Verbindung erzeugt werden soll.

Ein typischer grammatikalischer Smell ist der Einsatz von Passivsätzen. Formulierungen im Passiv sind häufig Schlupflöcher, die eine präzise Nennung der involvierten Akteure verschleiern. Anforderungen im Passiv können daher potenziell unvollständig oder missverständlich sein, wie **Abbildung 5** zeigt. Hier ist unklar, ob es die Aufgabe des Benutzers oder des Systems ist, die „un-hide"-Aktion durchzuführen.
Ein letztes Beispiel, diesmal für einen semantischen Smell, sind Details zu *GUI-Schnittstelle in Anforderungen*. Es gilt als schlechter Stil, das Design der grafischen Benutzungsschnittstelle schon in Szenarien von Anwendungsfällen festzuschreiben. Erkennen lassen sich solche Details über Begriffe wie „Button" oder Phrasen wie „left side of the screen" (**siehe Abbildung 6**).

Automatische Erkennung

Zum Aufspüren von Requirements-Smells kommen verschiedene Techniken in Frage: Manche Smells sind recht einfach zu erkennen, andere benötigen aufwändigere Ansätze. Einige der genutzten Verfahren stellen wir im Folgenden vor:

- *Wortlisten* sind ein gängiges Mittel zum Aufspüren von simplen Smells. Hierfür werden vorab Listen mit Begriffen oder Begriffsmuster definiert, vor deren Benutzung gewarnt werden soll. Beispiele sind die schon erwähnten vagen Worte oder Begriffe, die auf Nutzerschnittstellen-Details hindeuten. Verfahren, die Wortlisten nutzen, gehen davon aus, dass die Anforderungen in einem einfachen Textformat vorliegen,

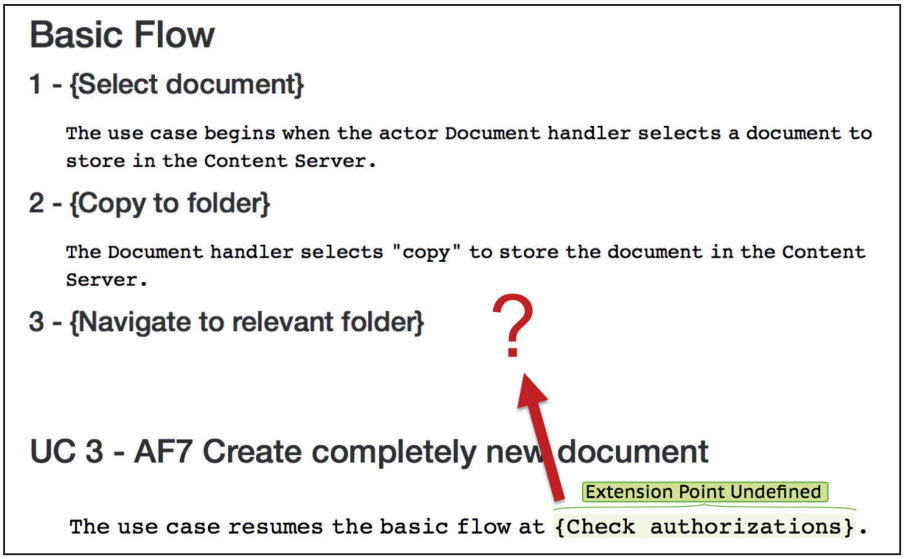


Abb. 3: „Undefinierte Referenzpunkte" Smell: Der Alternativablauf AF7 soll im Schritt {Check authorizations} fortgeführt werden. Dieser ist jedoch im Beispiel nicht definiert.

Weak

The system generates relations to other files, based on attribute values.

Abb. 4: Ein Beispiel für den Smell „Vage Phrasen“.

das leicht eingelesen werden kann. Diese Verfahren sind einfach umsetzbar, allerdings auch nur wenig mächtig und führen häufig zu vielen unerwünschten Warnungen (*False-Positives*).

- *Parsen von Dokumenten:* Vor allem für strukturelle Smells, wie etwa die *undefinierten Referenzpunkte*, benötigt man Informationen über die Struktur des Anforderungsdokuments, beispielsweise über die Gliederungsebenen oder auch die Formatierung. Um an diese Struktur zu gelangen, muss entweder in das entsprechende Autoren-Tool eingegriffen werden (z.B. in Form eines Word-Plug-ins) oder es müssen spezielle Parser herangezogen werden (z.B. Apache POI zum Einlesen von Office-Dokumenten).
- *Natural Language Processing (NLP)* bezeichnet eine Familie von Techniken, die es ermöglichen, natürliche Sprache maschinell zur analysieren. Solche Techniken stellen detaillierte Informationen über die grammatikalische Struktur eines Textes bereit und erlauben dadurch beispielsweise die zuverlässige Erkennung von Passivsätzen. Ein Beispiel für die Möglichkeiten von NLP findet sich in **Kasten 1**.

Mit Hilfe dieser Verfahren lassen sich unserer Erfahrung nach viele nützliche Requirements-Smells zuverlässig erkennen. Auch die technisch komplexen NLP-Techniken sind heute so effizient, dass sie eine Prüfung on-the-fly, also schon während des Schreibens, zulassen.

Erfahrungen aus der Praxis

Um herauszufinden, ob automatische Methoden reif genug sind, um die Qualitätssicherung in industrieller Softwareentwicklung zu unterstützen, haben wir die Requirements-Smell-Erkennung bei mehreren ausgewählten Softwareentwicklungsprojekten von Munich Re eingeführt. Zur Dokumentation von Anforderungen werden insbesondere Use-Cases nach dem Essentials-Modell von Ivar Jacobsen (vgl. [Jac99]) verwendet.

Auswahl von Smells

In unserem Beispiel existierte eine Review-Guideline, die auf knapp 30 Seiten die grundlegenden Maßgaben des Unternehmens für Use-Cases erklärt. Diese beinhaltet Konventionen (wie für die Benennung von Use-Cases und Alternative Flows), Schreibhilfen (wie das Vermeiden von langen Schachtelsätzen) und inhaltliche Hilfestellungen (wie „Transportiert der Use-Case-Name den Wert des Use-Cases?“).
Neben diesen Unternehmensvorgaben bilden Standards einen anderen Blickwinkel auf die Anforderungsqualität. So bietet beispielsweise ISO-29148 (vgl. [ISO11]) eine Liste von so genannten *Requirements Language Criteria*. Diese enthalten sprachliche Empfehlungen, wie etwa auf Komparative und Superlative zu verzichten.
Aus dieser Menge an Richtlinien haben wir eine initiale Menge von Regeln ausgewählt und dafür automatische Erkennungstechniken implementiert (**siehe Kasten 1**). Nachdem wir diese auf die existierenden Use-Case-Dokumente angewendet haben, zeigte sich schnell, dass nicht alle Regeln der ISO-29148 praxistauglich sind: So rät die ISO etwa von negierten Beschreibungen (z.B. „Das System darf nicht ...”) ab, um die Lesbarkeit und Eindeutigkeit zu verbessern. In der Praxis lässt sich allerdings kaum ein Anforderungsdokument rein positiv ausdrücken (schwierig sind insbesondere Einschränkungen des Systems, die so genannten *System Constraints*, positiv auszudrücken), und auch wissenschaftlich ließ sich bisher noch kein Nachteil durch negative Formulierungen feststellen (vgl. [Mun15]). Daher haben wir bei Munich Re darauf verzichtet, negative Beschreibungen in Anforderungsdokumenten anzuzeigen.

We want to change (hide or un-hide) an object.

Pass

Especially the "un-hide" must be done on the "parent" folder.

Abb. 5: Ein Beispiel für den Smell „Passivsätze ohne Akteur“.

Was lässt sich gut erkennen?

Zügig zeigt sich, welche Requirements-Smells präzise erkennbar sind: Insbesondere strukturelle Smells, wie die oben erwähnten toten Verweise auf Referenzpunkte, oder sprachliche Smells, wie unerwünschte Bedingungen und Verzweigungen im Main Flow eines Use-Cases (diese sollten in alternative Flows verschoben werden, um nicht den Main-Flow zu verwässern), finden besonders treffsicher Regelverletzungen. Folgende gute Erfahrungen haben wir ausschnittsweise gemacht:

- Fast alle strukturellen Smells sind akkurat erkennbar: Zu den strukturellen Problemen in Anforderungen gehören unter anderem Abweichungen von Templates, das Fehlen von bestimmten Inhalten und insbesondere fehlerhafte Verweise. Auch diese *undefinierte Referenzpunkte* lassen sich gut finden, allerdings erst nachdem man die Dokumente korrekt einlesen kann.
- Viele lexikalische Smells können genau erkannt werden: Beispiele dafür sind die genannten vagen Phrasen, die gegen Best-Practices im Requirements-Engineering verstoßen. Hier gibt es allerdings meist hohen Anpassungsbedarf bei den Analysetechniken (siehe nächsten Abschnitt).
- Die meisten grammatikalischen Smells werden von NLP-Techniken präzise erkannt: Zu den grammatikalischen Smells gehört die Verwendung von Superlativen und Komparativen, die oft vergleichende Anforderungsdokumente erzeugen (etwa *„Das System muss schneller sein als das Konkurrenzsystem”*). Diese vergleichenden Anforderungsdokumente führen zu Problemen in Entwicklung, Test und insbesondere Abnahme, etwa wenn das Vergleichssystem sich verbessert.
- Semantische Smells sind bislang nur gut aufzuspüren, wenn sie durch ein abgeschlossenes Vokabular definiert werden können, d.h. die Begriffe eindeutig auf den Inhalt schließen lassen: Ein gutes Beispiel dafür sind Benutzungsschnitt-

stellen-Details. Falls ein Use-Case über Richtungsangaben, Combo-Boxen oder bestimmte Farbmarkierungen spricht, beschreibt dieser fast immer das reine Design der Benutzungsschnittstelle, anstatt sich auf die Interaktion zwischen System und Benutzer zu fokussieren. Die Konsequenz dieses Qualitätsdefekts: Bei Änderungen am reinen Design der Benutzungsschnittstelle müssen auch die Use-Cases gewartet werden. Das sind unnötige Aufwände und birgt die Gefahr, inkonsistente Anforderungsdokumente zu erzeugen. Hier zeigte sich bei der praktischen Anwendung, dass die von unserer Requirements-Smell-Erkennung identifizierten Use-Cases tatsächlich genau die waren, die laut Aussage der Autoren viele Details der Benutzungsschnittstelle beinhalteten.

Präzision: Der Teufel steckt im Detail

Bei genauer Betrachtung der automatisch gefundenen Qualitätsdefekte stellte sich allerdings heraus, dass eine plumpe Anwendung von Regeln zu vielen ungewünschten Fehlmeldungen (*False-Positives*) führt. Oftmals muss die Präzision durch Ausnahmen und Filter geschärft werden. Ein Beispiel hierfür ist die passive Formulierung „Die Daten werden gespeichert". Hier ist nicht klar, von wem die Daten gespeichert werden sollen. Ist es eine Anforderung an die zu entwickelnde Software oder werden die Daten von einem anderen System gespeichert?
Wird der Akteur allerdings durch einen Zusatz beschrieben, sind die Zuständigkeiten eindeutig. Die Verwendung von Passiv stellt in diesem Fall also doch kein Problem dar. Beispiel: „Die Daten werden vom System gespeichert." (Hier ist der Akteur explizit genannt.)

Anpassungen an den Projektkontext

Des Weiteren erkannten wir schnell, dass die Smell-Erkennungstechniken nicht *out-of-the-box* angewendet werden können. Manche Smells sind beispielsweise nur in gewissen Teilen von Anforderungsdokumenten relevant: Ein Beispiel hierfür sind wieder Passivsätze. Während der handelnde Akteur in Use-Case-Schritten ein wichtiger Informationsträger ist, sind die Akteure in Vor- und Nachbedingungen eines Use-Case irrelevant, da diese Abschnitte beschreiben, in welchem Zustand das System sein soll, und nicht, wie es in diesen Zustand gekommen ist. Für diese Unterscheidung muss jedoch das Format der Anforderungsdokumente bekannt sein (beispielsweise durch ein Template). Sollte die Smell-Erkennung auf ein anderes Projekt mit einem anderen Template angewendet werden, so müsste die Analyse angepasst werden.
Insbesondere bereitete uns dabei das Einlesen aus MS-Word-Dateien (mittels Apache POI) die größten Kopfschmerzen: Zu unausgereift ist die Bibliothek, sodass wir an manchen Stellen direkt auf MS-Office-Open-XML arbeiten mussten, um die gewünschten Informationen aus den Dateien auszulesen. Zu den Aufwänden für die Anpassung des Einlesens kam gelegentlich noch ein Anpassen der Analysen. Sobald es jedoch einmal eingerichtet war, funktionierten die Analysen reibungslos.

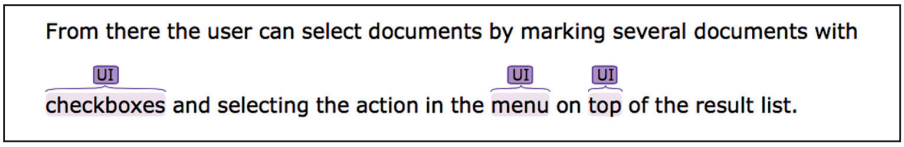
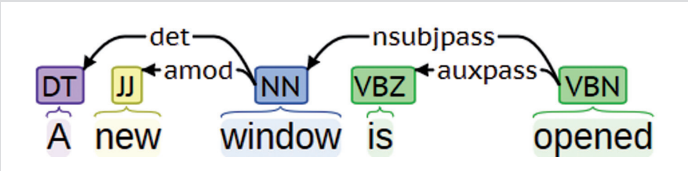


Abb. 6: Ein Beispiel für den Smell „GUI-Schnittstelle in Anforderungen".

Erkenntnisse im Tooling

Ein erster Prototyp wurde so einfach wie möglich umgesetzt: Benutzer legten die Dateien in einem Netzwerk-Verzeichnis ab, innerhalb kurzer Zeit sprang automatisch eine Analyse an und nach einigen Minuten erzeugte die Analyse ein Ergebnis-Dashboard. Dieses Dashboard stellte das komplette Anforderungsdokument als statische HTML-Seite dar und enthielt Annotatio-

Automatische Textanalyseverfahren sind in den letzten Jahren immer mächtiger geworden. Detaillierte Informationen über die Satzstruktur lassen sich damit extrahieren. Für den Satz „A new window is opened" liefert beispielsweise die Library der Computer-Linguisten der Universität Stanford (Stanford NLP, vgl. `http://nlp.stanford.edu`) folgende Ausgabe (Beispielausgabe von Stanford NLP):



Dieser Ausschnitt zeigt einen Teil der Möglichkeiten von NLP auf: Stanford NLP zerlegt den Satz zunächst in einzelne Wörter (*Tokenization*). Dann wird die Wortart jedes Wortes in diesem Satz (*Part-Of-Speech tagging* oder *POS tagging*) ermittelt, entweder über probabilistische Machine-Learning-Ansätze oder linguistische Regeln. In unserem Beispiel stellt Stanford NLP korrekt fest, dass es sich um folgende Kombination handelt (nach dem so genannten „Penn Treebank Schlüssel"):

- Artikel (DT)
- Adjektiv (JJ)
- Nomen (NN)
- Verb, 3. Person Singular (VBZ)
- Verb, Partizip Passiv (VBN)

Aufbauend auf dem POS tagging kann Stanford NLP dann noch Wortzusammenhänge ermitteln (*Dependency Parsing*). Die Pfeile zwischen den Wörtern in der Abbildung zeigen beispielsweise, dass Stanford NLP versteht, dass sich „opened" auf „Window" bezieht. Grammatikalische Smells wie Passivsätze lassen sich mit diesen Informationen einfach detektieren (siehe Smell *Passivsätze ohne Akteur*).
Weiterhin ermöglicht NLP unter anderem das Aufteilen eines Textes in Sätze (*Sentence Splitting*), die Rückführung eines Wortes in seine Grundform (*Lemmatization*), das Aufbrechen des Satzes in seine grammatikalische Struktur (z. B. Haupt- und Nebensätze oder Verbphrasen und Nominalphrasen – das so genannte *Parsing*) und vieles mehr.

Kasten 1: Natural Language Processing (NLP).

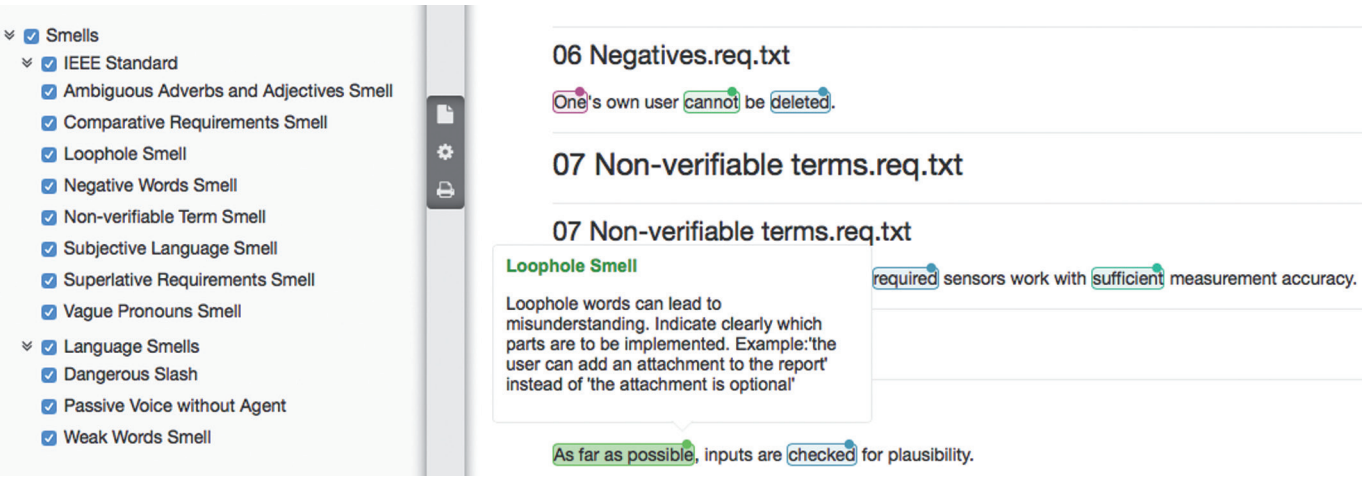


Abb. 7: Ein Ausschnitt aus einem Requirements-Smells-Dashboard in HTML. Einen Teil der Smells können Sie unter www.qualicen.de mit eigenen Texten interaktiv testen.

nen, die auf gefundene Smells hinweisen (siehe Abbildung 7).

Nach einiger Zeit der Anwendung im Projektalltag war allerdings klar, dass das Öffnen eines weiteren Tools (auch wenn es nur ein HTML-Browser ist) für Requirements-Engineers eine unbequeme Hürde darstellt, die es zu vermeiden galt. Deshalb erweitern wir zurzeit die Tool-Landschaft von Munich Re um ein MS-Office-Plug-in. In einer zusätzlichen Menüleiste (siehe Abbildung 8) können die Smell-Analysen vom Autor direkt gestartet werden. Außerdem werden dem Requirements-Engineer Kommentare dort anzeigt, wo Feedback gewünscht ist: direkt als Kommentare im Anforderungsdokument (siehe Abbildung 9). Dies erfüllt den Wunsch nach schnellem, direktem Feedback.

Grenzen von Requirements-Smells

Zuletzt stellt sich natürlich die Frage: Was kann eine automatische Analyse schaffen, etwa im Vergleich zu manuellen Reviews? Wo kommen auch modernste Verfahren an ihre Grenzen? Unserer Erfahrung nach haben automatische Methoden Grenzen an folgenden Stellen:

- *Semantische Qualitätsdefekte:* Automatisierung kann die Wünsche der Nutzer nicht aus deren Köpfen raten. Dies schließt sowohl Auslassungen und inkorrekte Anforderungen, als auch fehlende Implementierbarkeit ein. Automatische Analysen kennen die Domäne nicht und können ohne weitere Hilfen dazu auch keine Annahmen machen. Beispiel: Die Anforderung fehlt, dass das System auch auf Spanisch verfügbar sein soll.
- *Semantik der Sprache:* Ein weiteres Problem betrifft die Frage: Was wird beschrieben? Zwar kann man mit bestimmten Annahmen erraten, welche groben Inhalte in einem Use-Case in etwa beschrieben werden, aber leider sind NLP-Techniken noch nicht so weit, dass sie ein tieferes Verständnis der Sätze hätten und dadurch zum Beispiel Widersprüche in natürlicher Sprache erkennen könnten. Beispiel: Im ersten Abschnitt steht, dass das System auf Spanisch verfügbar sein soll, im zweiten Absatz werden englische Fehlermeldungen genannt.
- *Bauchgefühl und Ästhetik:* Verständlichkeit ist leider mehr als nur die Länge der Sätze. Wenn sich Menschen schwer tun, genau zu definieren, was ihnen missfällt, wie etwa beim Sprachstil, wird auch eine Automatisierung nur durch eine vorherige Analyse des Problems möglich sein. Beispiel: Irgendwie trifft der Use-Case noch nicht den Kern des Problems.

Feedback, Veränderungen und Auswirkungen

Wir haben die Einführung von Requirements-Smells in Projekten bei Munich Re begleitet und eingefangen, was Requirements-Engineers von der Idee und der Umsetzung halten. Wichtig ist dabei zunächst, bei der Einführung zu betonen und darauf zu achten, dass die Ergebnisse der Smell-Analyse tatsächlich nur für den Autor bestimmt sind. Denn falls die Anzahl der Smells in irgendeiner Form als *Performance Indicator* zur Messung der persönlichen Leistung genutzt wird, würden Nutzer anfangen, das Tool auszutricksen, und die Akzeptanz bei Requirements-Engineers würde umgehend sinken.

Insgesamt stößt der Ansatz auf ausgesprochen positives Feedback bei den professionell anwendenden Testnutzern. Nach ihrer Aussage hilft das Tool:

- Schulungen zu unterstützen, indem Nutzer umgehend auf Verletzungen der grundlegenden Regeln aufmerksam gemacht werden.
- Eine externe Meinung zu bekommen, auf die man nicht selber gekommen wäre.

Abb. 8: Requirements-Smell-Integration als Menüleiste in MS-Word.

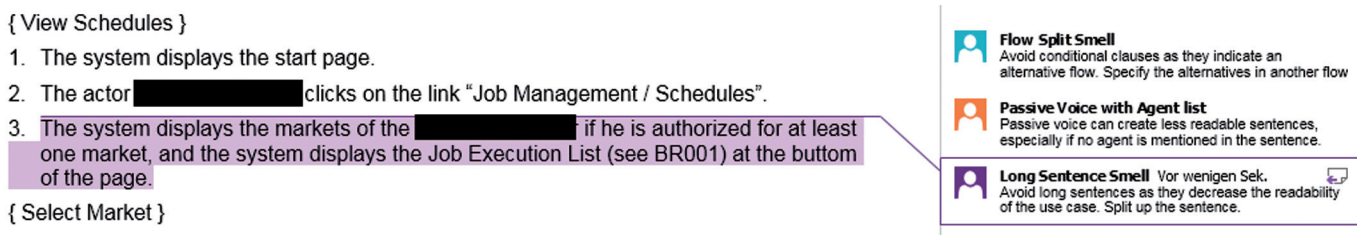


Abb. 9: Automatisch erzeugte Smells als Kommentare in einer MS-Word-Datei.

- Den eigenen Schreibstil zu hinterfragen und zu ändern.

Nach Aussage eines Nutzers hat dieser den Smell sogar mittlerweile im Kopf, bevor er die Worte niedergeschrieben hat.
Was springt dabei finanziell heraus? Wie viel Geld kann das Projekt sparen? Diese sicherlich sinnvollen Fragen sind schwer zu beantworten. Auch wir konnten in diesem Projekt keinen Weg finden, um den monetären Vorteil zu messen. Der Grund liegt allerdings vermutlich tiefer: Noch immer ist es unklar, welche genauen Folgen solche Fehler der reinen Art und Weise der Darstellung von Anforderungsdokumenten nach sich ziehen. Einig ist man sich allerdings, dass sie Auswirkungen haben. Hier ist die Forschung gefragt, zu messen und zu vergleichen.

Resümee

So wie es uns heute antiquiert scheint, ohne automatische Unterstützung in der Entwicklungsumgebung Software zu programmieren, wird automatische Unterstützung im Requirements-Engineering auch bald einen Aufschwung erleben. Die Erfahrungen bei Munich Re haben gezeigt, dass es eine Klasse von Qualitätsdefekten gibt, die wir mit ein wenig Werkzeugunterstützung schnell und einfach erkennen und anschließend ausbauen können.
Wer jedoch von voll-automatischer Qualitätssicherung träumt, wird enttäuscht werden. Die Automatisierung wird auf absehbare Zeit zwar eine Vorstufe, jedoch kein vollwertiger Ersatz zum manuellen Review sein. Als solche könnte sie jedoch, zumindest bei Munich Re, bald nicht mehr wegzudenken sein. Klingt nach Zukunftsmusik? Teile davon haben wir schon in unser tägliches Leben mit aufgenommen, denn wer würde heute noch ein Dokument lesen, das nicht durch eine automatische Rechtschreibprüfung gelaufen ist? ||

Literatur & Links

[Fem14] H. Femmer, D. Méndez Fernández, E. Juergens, M. Klose, I. Zimmer, J. Zimmer, Rapid Requirements Checks with Requirements Smells: Two Case Studies, in: Proc. of the 1st International Workshop on Rapid Continuous Software Engineering, 2014
[Fem15] H. Femmer, J. Mund, D. Mendez Fernandez, It's the Activities, Stupid! A New Perspective on RE Quality, in: Proc. of the 2nd International Workshop on Requirements Engineering and Testing (RET), 2015
[ISO11] ISO, IEC, and IEEE. ISO/IEC/IEEE 29148:2011-Systems and Software Engineering – Life Cycle Processes – Requirements Engineering, Technical report, ISO IEEE IEC, 2011
[Jac99] I. Jacobson, G. Booch, J. Rumbaugh, The Unified Software Development Process, Addison-Wesley Reading, 1999
[Mun15] J. Mund, H. Femmer, M. Daniel, J. Eckhardt, Does Quality of Requirements Specifications matter? Combined Results of Two Empirical Studies, in: Proc. of the 9th International Symposium on Empirical Software Engineering and Measurement (ESEM '15), 2015

Die Autoren

|| Henning Femmer
(henning.femmer@qualicen.de)
ist Doktorand an der TU München mit dem Schwerpunkt Anforderungsqualität. Sowohl in dieser Rolle als auch als Mitgründer der Qualicen GmbH hilft er Firmen, den Begriff Qualität zu verstehen, zu hinterfragen und schließlich – immer einfach und pragmatisch – zu überprüfen.

|| Benedikt Hauptmann
(benedikt.hauptmann@qualicen.de)
ist Mitgründer der Qualicen GmbH, einem Spin-Off der TU München. Als Wissenschaftler an der Universität forscht er gemeinsam mit Firmenpartnern an den Themen Test- und Anforderungsqualität und trägt die entstandenen Forschungsergebnisse in die Praxis.

|| Alexander Widera
(awidera@munichre.com)
ist System Engineer bei der Münchener Rück AG (Munich RE). In seiner Rolle als Produktspezialist verantwortet er die Weiterentwicklung von Software und unterstützt beim Requirements-Engineering. Er ist Mitglied der Requirements Engineering Community.