Herausforderungen für halbautomatische Traceability-Erkennung

Jörg Leuser
Konzernforschung & MBC Entwicklung, Daimler AG
Postfach 2360, 89013 Ulm, Deutschland
joerg.leuser@daimler.com

Zusammenfassung

Die Erkennung und Pflege von Traceability-Links mittels halbautomatischer Methoden wird in der Praxis durch mehrere Faktoren behindert: Die Größe der Spezifikationen und andere Spezifikationssprachen als Englisch. Diese Veröffentlichung beschreibt acht Herausforderungen für den Einsatz halbautomatischer Methoden in der Automobilindustrie. Es werden erste Ergebnisse einer Fallstudie sowie Optimierungen, die die Arbeit mit realen Spezifikationen erleichtern, beschrieben: Das Filtern von Kandidaten-Links in Abhängigkeit der strukturellen Eigenschaften der verlinkten Objekte sowie der Einsatz einer Kompositazerlegung in der Vorverarbeitungsphase der Algorithmen. Insgesamt zeigen die Optimierungen leichte Verbesserungen, sind jedoch nicht ausreichend, halbautomatische Methoden in der Praxis direkt einsetzbar zu machen.

Schlagwörter: Traceability, tf/idf, LSI, Kompositazerlegung, Natürlichsprachige Spezifikation

1 Einleitung

Es ist wichtig, eine durchgängige Verfolgbarkeit (Traceability) von Anforderungen über den Entwicklungszyklus sicherzustellen. Verfolgbarkeit kann durch explizite oder implizite Verknüpfungen (Links) innerhalb und zwischen Entwicklungsartefakten sichergestellt werden. Explizite Verfolgbarkeit ist dabei zu bevorzugen, da sie automatisierte Analysen erlaubt. Ein Aspekt, weshalb die Automobilindustrie Traceability herstellen und pflegen muss, sind Standards wie CMMI sowie der zukünftige Standard ISO 26262. Die Standards fordern zwar Traceability, beschreiben jedoch nicht genau, wie detailliert die Traceability ausgeführt werden muss [15].

Während die RE-Community Traceability als wichtiges Instrument zur Beherrschung der Komplexität ansieht, ist diese Sichtweise in der Praxis noch nicht weit verbreitet: Die Entwickler, die Spezifikationen erstellen, sehen in Traceability mehr eine Zusatzlast als eine nützliche Methode. Das gilt insbesondere für Verlinkungen innerhalb von Spezifikationen. Verlinkungen zwischen Anforderungen und Testfällen sind hingegen besser akzeptiert. Ein Problem ist jedoch, dass das Verlinken – wie es in der Praxis geschieht – aufwändig und umständlich ist. Insbesondere ist es ein größtenteils manueller Prozess. Deshalb sind viele Entwickler der Ansicht, dass das Erstellen und Pflegen von Traceability für sie mehr Aufwand bedeutet als dass es nutzt. Dies ist der Hauptgrund dafür, dass explizite Traceability meist nur erstellt und gepflegt wird, wenn es den Entwicklern vorgeschrieben wird.

Ein Ansatz, zu mehr Traceability zu gelangen ist es, den Prozess der Erstellung und Pflege zu beschleunigen bzw. zu vereinfachen. Eine vollständige Automatisierung der Traceability-Erkennung und Pflege wäre zwar optimal, ist für natürlichsprachige Dokumente allerdings utopisch. Jedoch zeigt Forschung im Bereich des Information Retrieval (IR) zum halbautomatischen Erkennen von Links bereits vielversprechende Ergebnisse für kleine Spezifikationen. Hauptsächlich werden dabei IR Methoden, die auf dem Vektorraum-Modell (VSM) [1, 10, 21], dem Probabilistic Network Model [10] oder dem Latent Semantic Indexing (LSI) [12, 21, 27, 28, 37] basieren, eingesetzt. Kritik an solch halbautomatischen Vorgehen kommt von Asuncion [2] der fehlende semantische Informationen in den erkannten Links bemängelt. Deshalb können aus seiner Sicht nicht alle Vorteile von Traceability ausgenutzt werden. Zwar mögen die IR Methoden nicht in der Lage sein, die Semantik der Links zu liefern, doch ist es möglich, diese aus Meta-Daten wie z.B. Dokumenten-Strukturen abzuleiten.

Vor einer praktischen Einsatzfähigkeit solcher halbautomatischen Methoden in der Industrie sind aber noch einige Herausforderungen zu lösen. Im folgenden Abschnitt werden die Eigenschaften der Automobil-Domäne beschrieben, die Einfluss auf solch halbautomatische Verfahren haben.

2 Die Automobil-Domäne

Soweit es das Traceability-Problem betrifft beschäftigt sich die Automobil-Domäne mit eingebetteten Systemen. Im Allgemeinen beinhaltet das Traceability-Problem Verlinkungen innerhalb und zwischen Lastenheften und von dort transitiv Verlinkungen zur Architektur, dem Design und letztendlich zu Tests. In dieser Veröffentlichung fokussieren wir auf die Links innerhalb von Anforderungs-Spezifikationen. Zwar ist das Traceability-Problem für diese Domäne bisher noch nicht gelöst [33], doch gibt es bereits einige Bereiche, in denen Traceability akzeptiert ist: Verlinkungen von normativen Einschränkungen [32] – auch als mitgeltende Unterlagen bekannt – aus den Spezifikationen sowie Verlinkungen zwischen Anforderungen und zugehörigen Testfällen.

In der Automobilindustrie sind Lastenhefte größtenteils text- und nicht modellbasiert. Diagramme in den Spezifikationen dienen hauptsächlich der Illustration, weniger der Spezifikation. Regelsysteme (z.B. der Motorfunktion) bilden eine Ausnahme da sie teilweise modellbasiert spezifiziert werden. Ein Punkt, der zusätzlich zu beachten ist, ist dass Spezifikationen – zumindest bei Daimler – auf deutsch geschrieben werden. Diese Spezifikationen werden nur bei Bedarf ins Englische übersetzt.

Eine typische Spezifikation eines modernen Autos aus der Luxusklasse besteht aus einer Fahrzeug-, ungefähr 100 System- und mehreren hundert Komponenten-Spezifikationen. BMW gliedert die Systemebene dabei weiter auf in Subsysteme und Kundenfunktionen [20, S. 12]. Die einzelnen Spezifikationen sind dabei zwischen wenigen Seiten (für einfache Komponenten) und 2000 Seiten (für komplexe Steuergeräte) lang. Zwar bestehen diese Spezifikationsdokumente nicht komplett aus Anforderungen, doch ein Großteil des Volumens stellen Anforderungen dar. Zusätzlich verweisen die Spezifikationen auf weitere Dokumente, sogenannten normativen Einschränkungen [32]. Dies können Gesetzte, (Firmen-)Normen oder Standards sein die zusätzliche Anforderungen enthalten. Aktuell summieren sich diese Dokumente auf weitere 4000 Seiten. Je nach Größe und Komplexität werden Spezifikationen von einzelnen Entwicklern oder Teams erstellt. Typischerweise werden spezielle Abschnitte von Bereichs-Experten spezifiziert und in die Spezifikation mit aufgenommen.

Die Spezifikationen der verschiedenen Abstraktionsebenen folgen definierten Dokumentenstrukturen. Komponenten-Spezifikationen sind ähnlich strukturiert wie in [36] beschrieben. Die bekannten Strukturen helfen dabei, Informationen in neuen Spezifikationen schneller zu finden. Dadurch ist eine implizite Traceability bereits über die Dokumentenstruktur gegeben. Zusätzlich enthalten Spezifikationen einige textuelle Querbezüge. Die Form, die Uusitalo et al. [35] als Personen-Verlinkung bezeichnen, stellt bei Daimler heutzutage allerdings die Hauptform der Traceability dar. Traceability wird dadurch gepflegt, dass die beteiligten Personen den jeweiligen Autor von Anforderungen kennen und mit ihm kommunizieren.

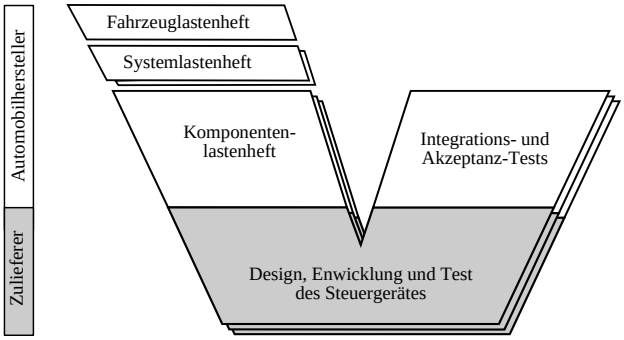

Abbildung 1. Spezifikations-Hierarchie

Die Wiederverwendung von Funktionalität zwischen verschiedenen Fahrzeug-Generationen beträgt 80-90% [33]. Sogar die Spezifikation der Mensch-Maschine-Schnittstelle, die u.A. das visuelle Design von Komponenten festelegt, hat eine Wiederverwendungsrate von bis zu 50%. Ähnlich viel Wiederverwendung gibt es im Unterhaltungs-Elektronik-Bereich [27]. Allerdings gibt es Wiederverwendung nicht nur zwischen verschiedenen Generationen von Fahrzeugen sondern auch zwischen verschiedenen Varianten derselben Generation.

Die meisten Systeme entstehen in einer Kooperation zwischen einem Automobilhersteller (OEM) und einem Zulieferer. Typischerweise wird das Lastenheft vom OEM erstellt, während das System-Design und die Implementation vom Zulieferer übernommen wird. Abbildung 1 zeigt, dass das Komponentenlastenheft die Schnittstelle zwischen OEM und Zulieferer darstellt. Die Entwicklungspartner müssen daher unterschiedliche Arten der Traceability beherrschen. Der OEM konzentriert sich hauptsächlich auf Links innerhalb von Lastenheften bzw. innerhalb der Lastenheft-Hierarchie sowie Traceability zu den Testfällen. Der Zulieferer hingegen verlinkt von Anforderungen zu Design und Code sowie von dort zu den jeweiligen Testfällen.

Da die Entwickler fast ständig unter Zeitdruck stehen sind alle Verfahren, die die Erstellung und Pflege von Traceability beschleunigen, ein Schritt in Richtung Lösung des Traceability-Problems. Die halbautomatischen Verfahren versprechen eine solche Beschleunigung.

3 Halbautomatische Methoden

Die halbautomatischen Methoden zur Erkennung und Pflege von Traceability in natürlichsprachigen Dokumenten basieren auf Methoden des Information Retrievals (IR). Die Methoden sind „nur“ halbautomatisch, da die Algorithmen zwar eine Liste von potentiellen Links generieren können (die Kandidaten-Liste), ein menschlicher Analyst aber immer noch über die Korrektheit der Vorschläge entscheiden muss. Die Schritte zum Erkennen von Links können in mehrere Phasen eingeteilt werden: Vorverarbeitung, Anwendung der Algorithmen, Erstellen der Kandidaten-Liste sowie Kontrolle durch den Benutzer. Bis auf die letzte Phase sind alle Schritte automatisiert durchführbar. Insbesondere sind die aufwändigen Berechnungen in der Vorverabeitungsphase und bei der Anwendung der Algorithmen unabhängig von der Erstellung der Kandidaten-Liste möglich. D.h., die Abfragen die der Analyst stellt (indem er sich Kandidaten-Listen generieren lässt), können vom System sehr schnell beantwortet werden. Die einzelnen Phasen werden in Abschnitt 5 genauer beschrieben. Da die Kandidaten-Liste sehr viele möglichen Links enthält (auch die sehr unwahrscheinlichen), wird dem Benutzer in der Regel nur eine sortierte Liste präsentiert, die nur Kandidaten-Links enthält, die „besser“ als ein gewisser Wert (Cutoff-Point) sind.

Zur Qualitätsbestimmung der Kandidaten-Liste werden zwei weit verbreitete Metriken aus dem IR eingesetzt: *Recall* (Trefferquote) und *Precision* (Genauigkeit). Wir verwenden die Generalisierung auf mehrere Abfragen wie z.B. auch de Lucia et al. [11]:

Definition 1

$$Recall = \frac{\sum_i |korrekte\ Links_i \cap gefundene\ Links_i|}{\sum_i |korrekte\ Links_i|}$$

Definition 2

$$Precsion = \frac{\sum_i |korrekte\ Links_i \cap gefundene\ Links_i|}{\sum_i |gefundene\ Links_i|}$$

Beide Metriken haben einen Wertebereich von 0 - 100% wobei 100% in beiden Fällen das Optimum ist.

Um die Größe von Datensätzen (d.h. einer Menge von Dokumenten, die analysiert werden sollen) zu klassifizieren schlagen Hayes et al. [22] vor, die Anzahl der theoretisch maximal möglichen Links als Basis zu verwenden. Ein fiktiver Datensatz, der 20 abstrakte und 50 detaillierte Anforderungen enthält, liefert 1000 theoretisch mögliche Links. Ein Datensatz gilt laut Klassifizierung als „klein“, wenn er maximal 3000 mögliche Links enthält und als „groß“, wenn er mehr mögliche Links enthält. Deshalb fallen die meisten Spezifikationen der Automobilindustrie – wenn nicht sogar alle – in die Kategorie „groß“.

3.1 Die Herausforderungen

In der Automobil-Domäne gibt es eine Reihe von Herausforderungen für halbautomatische Traceability-Methoden. Diese Herausforderungen sind jedoch nicht auf die Automobil-Domäne beschränkt:

1. **Spezifikationsgröße:** Verglichen mit den beispielsweise in [10] oder [21] beschriebenen Spezifikation sind nahezu alle Spezifikationen der Automobil-Domäne groß. Zwar steigt mit der Größe der Spezifikationen auch der Verarbeitungsaufwand, das eigentliche Problem stellt aber die große Anzahl an Kandidaten-Links dar, die vom Benutzer kontrolliert werden muss.

2. **Spezifikationssprache:** Während die meisten Ergebnisse für englische Spezifikationen berichtet werden, sind Daimler-Spezifikationen größtenteils auf deutsch verfasst. Da mindestens die Vorverarbeitung der Algorithmen sprachspezifisch ist, hat die Spezifikationssprache Einfluss auf die Ergebnisse.

3. **Tiefe Hierarchien:** Neben den bereits beschriebenen drei Abstraktionsebenen Fahrzeug, System und Komponente (siehe Abb. 1) besitzt jede dieser Spezifikationen eine eigene Hierarchie. Die Hierarchien von Spezifikationen auf derselben Ebene sind im Groben jedoch identisch.

4. **Wiederverwendung:** Der Großteil der Spezifikationen sind Weiterentwicklungen vorhandener Spezifikationen, keine von Grund auf neuen Spezifikationen.

5. **Verteilte/Gemeinsame Spezifikationserstellung:** Spezifikationen werden selten von einer einzelnen Person erstellt. Meistens kommen zumindest Fachinhalte von unterschiedlichen Fachbereichen in ein Lastenheft. Spezifikationen werden auch von den Zulieferern weiter verfeinert, wobei zusätzlich Firmengrenzen überschritten werden.

6. **Zeitpunkt der Traceability-Erstellung:** Es ist bisher nicht klar, zu welchem Zeitpunkt die Erstellung der Traceability optimal ist. Während der Erstellung der Spezifikation oder nach Abschluss der Spezifikationsarbeiten? Ggf. ist es auch möglich, komplett auf explizite Links zu verzichten, falls eine On-Demand-Erstellung problemlos funktioniert.

7. **Vorgeschriebene Links:** Ob sichergestellt werden kann, dass alle vorgeschriebenen Links (z.B. durch die zukünftige ISO 26262) durch halbautomatische Methoden sicher gefunden werden, ist nicht geklärt. Außerdem steht nicht fest, ob explizite Links vorhanden sein müssen oder ob eine automatische Erkennung on-demand ausreichend ist.

8. **Geheime Anforderungen:** Teile der Anforderungen eines Automobils sind nur einem begrenzten Personenkreis bekannt/zugreifbar. Dazu zählen u.a. das Fahrzeuglastenheft wie auch Anforderungen sensibler Systeme wie z.B. der Schließ- oder Alarmanlage. Müssen die zugriffsberechtigten Personen also alle Links erzeugen?

Diese Herausforderungen anzugehen muss dabei nicht zwingend bedeuten, die eingesetzten Algorithmen zu verbessern. Viele der Herausforderungen ließen sich durch bessere Anleitungen zur Verwendung der Algorithmen (z.B. [13]) verringern. Aus Sicht des Autors lassen sich die Herausforderungen 1 – 3 am besten durch verbesserte Algorithmen (auch der Vorverarbeitung) angehen, während die restlichen besser durch geeignete Anleitungen angegangen werden.

3.2 Hilfreiche Domänen-Eigenschaften

Während die Domäne eine Menge an Herausforderungen bietet, besitzt sie gleichzeitig jedoch einige positiven Eigenschaften, die die Traceability-Erkennung erleichtern. Beispielsweise sind die Spezifikationen gut strukturiert, was eine Voraussetzung für gute Retrieval-Ergebnisse ist [7]. Zusätzlich kann die Struktur dazu verwendet werden, die Retrieval-Ergebnisse zu verbessern [10]. Besonders hilfreich ist dabei, dass die Struktur der Spezifikationen größtenteils im Voraus bekannt ist und sich selten ändert.

Weiterhin hilfreich ist, dass die Anforderungen typischerweise mit zusätzlichen Meta-Daten angereichert sind. Diese Meta-Daten erscheinen besonders dazu geeignet zu sein, die Precision zu erhöhen indem False Positives (fälschlicherweise als Link erkannte Zusammenhänge) aus der Kandidaten-Liste entfernt werden wenn sie auf Grund der Meta-Daten unwahrscheinlich oder gar unmöglich sind.

4 Anwendungsfälle

Links in Anforderungsdokumenten werden für verschiedene Anwendungsfälle erstellt. Links werden beispielsweise zur Änderungs-Einflussanalyse (Impact Analysis) benötigt. Ebenso dienen sie der Dokumentation von Abhängigkeiten innerhalb eines Systems und helfen somit beim Systemverständnis. Links zwischen Anforderungen und Testfällen können als notwendige Bedingung zur Testabdeckung dienen (wenn auch nicht als hinreichende). Für die Anwendung der halbautomatischen Methoden wurden drei Hauptanwendungsälle für die Industrie identifiziert:

- **Initiale Erstellung von Traces**: In bisher nicht verlinkten Artefakten eine initiale Verlinkung herstellen, die dann ggf. manuell weiter gepflegt wird.

- **Erstellung von Links on-demand**: Dynamische Erkennung von Links in Artefakten zum Bedarfszeitpunkt. Falls dieser Anwendungsfall gut funktioniert, ist es nicht nötig, Links „auf Verdacht“ zu erstellen und pflegen sondern nur bei Bedarf den Aufwand zur Erstellung zu investieren.

- **Qualitätscheck für Links**. Links können über Zeit degenerieren. Zwar können Tools wie Doors sicherstellen, dass keine „toten“ Links entstehen, sie können aber nicht sicherstellen, dass die Links noch inhaltlich korrekt sind.

4.1 Zusätzlicher Nutzen

Während die Entdeckung und Pflege von Links das Hauptziel der Methoden ist, lassen sich weitere Anwendungsfälle ohne größeren Aufwand ebenfalls unterstützen.

Auf Grund des hohen Wiederverwendungsanteils entsteht immer wieder der Bedarf, unterschiedliche Spezifikationen zu vergleichen. Beispielsweise wird eine Prüfung nötig, ob alle Inhalte der „alten“ Spezifikation auch in die „neue“ Spezifikation integriert wurden. Typischerweise werden die Anforderungen bei der Wiederverwendung zumindest leicht verändert, sodass exakte Treffer unwahrscheinlich sind. Mittels der IR-Methoden ist solch ein Vergleich jedoch relativ einfach zu implementieren.

Ein zusätzlicher Anwendungsfall ist die Qualitäts-Überprüfung von Anforderungsdokumenten (im Gegensatz zum Anwendungsfall „Qualitätscheck für *Links*“). Gerade in großen Spezifikationen besteht die Gefahr der Re-Spezifikation. Re-Spezifikation meint dabei, dass eine Anforderung mehrfach in einer Spezifikation auftaucht – unter Umständen sogar sich widersprechend. Da es sich um textuelle Spezifikationen handelt ist es sehr unwahrscheinlich, dass die Duplikate exakt gleich formuliert sind. Wie Natt och Dag et al. [31] berichten, sind IR-Methoden in der Lage solche Re-Spezifikationen aufzudecken.

5 Der Lösungsansatz

Das Vorgehen ist darauf ausgelegt, reale Projekte bei der Erstellung und Pflege von Traceability zu unterstützen. Daher ist es notwendig, auf aktuelle, reale Daten zugreifen zu können – in unserem Fall bedeutet das den Zugriff auf Anforderungsdokumente in DOORS. Aus diesem Grund ist das TraceTool – das Forschungs-Tool welches die IR Algorithmen implementiert – in der Lage, Daten direkt aus DOORS zu laden. Da das interne Datenmodell der Struktur des Requirements Interchange Format (RIF) [23] ähnelt, sollten auch andere Anforderungsmanagement-Tools bei Bedarf leicht einbindbar sein. Es ist erklärtes Ziel, die Spezifikationen nicht speziell für die Traceability-Erkennung zu optimieren sondern zu untersuchen, ob reale Spezifikationen direkt sinnvoll verarbeitet werden können.

Für die Erkennung von Traceability wurden zwei vielgenutzte Algorithmen implementiert: tf/idf-gewichtetes VSM [4] und tf/idf-gewichtetes LSI [4]. Beide Algorithmen bauen eine Term/Dokument-Matrix auf. LSI verringert die Dimension der Matrix nach einer Singulärwertzerlegung um das Synonymproblem in den Griff zu bekommen. Basierend auf dieser Term/Dokument-Matrix berechnen die Algorithmen Ähnlichkeiten zwischen Dokumenten (jedes Dokument entspricht einer Spalte in der Matrix und wird als Vektor interpretiert). In unserem Fall entsprechen Dokumente jeweils einer einzelnen Anforderung, bzw. einem einzelnen Strukturelement; d.h. ein Dokument entspricht einem Doors-Objekt.

Als Ähnlichkeitsmaß zwischen zwei Dokumenten wird der Kosinus des Winkels zwischen beiden Dokumentenvektoren verwendet. Mittels Ähnlichkeitswerten generieren die Algorithmen eine Liste möglicher Links, der sogenannten Kandidaten-Liste, die durch einen menschlichen Analysten auf Korrektheit kontrolliert werden muss. Um dem Analysten die Entscheidung zu erleichtern wird der Ähnlichkeitswert auf das Intervall [0,100] projiziert mit 100 als bestmöglichen Wert. Dieser Wert wird im Folgenden als *Trust Level* bezeichnet.

Eine Übersicht über den Bearbeitungs-Ablauf ist in Abbildung 2 gegeben. Zwar ist der Vorverarbeitungs-Teil des TraceTools sowohl für englische wie auch deutsche Spezifikationen ausgelegt, doch liegt der Fokus auf deutschsprachigen Texten. Stoppwörter wie z.B. Artikel und Konjunktionen, welche keinen Mehrwert liefern, werden entfernt. Für deutschsprachige Texte wird dabei auf die Stoppwortliste von snowball.tartarus.org zurückgegriffen, die um Schreibeweisen der neuen deutschen Rechtschreibung erweitert wurde. Für tf/idf werden die Wörter zusätzlich mit Caumanns' Stemming-

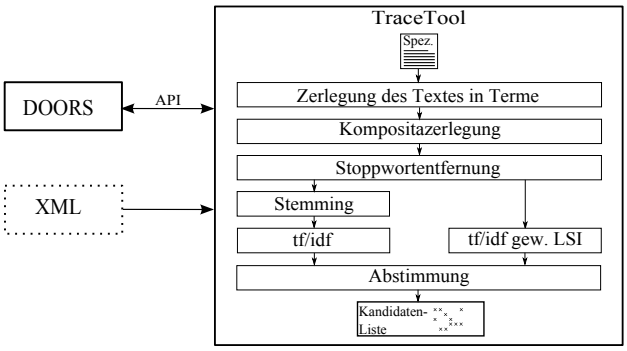

Abbildung 2. Programmstruktur des Tools

Algorithmus auf ihren Wortstamm reduziert. Dieser Stemming-Algorithmus wird beispielsweise auch im Suchmaschinenprojekt Apache Lucene eingesetzt. Auf Grund der Eigenschaften des LSI-Algorithmus ist Stemming für LSI nicht notwendig. Zusätzlich wurde für deutsche Texte eine wörterbuchbasierte Kompositazerlegung implementiert, da sich gezeigt hat [25], dass Spezifikationen der Automobilindustrie viele Komposita enthalten. Nachdem Braschler und Ripplinger [6] im „klassischen" IR zeigen konnten, dass Kompositazerlegung für deutsche Texte im Allgemeinen positive Auswirklungen auf Precision und Recall hat, scheint dies ein sinnvoller Zusatzschritt im Vorverabeitungsablauf zu sein.

Um die Ergebnisse der beiden IR-Algorithmen zu konsolidieren wird ein von Dekhtyar et al. [13] vorgeschlagenes Abstimmungs-Schema eingesetzt. Das vorgeschlagene Abstimmungs-Schema „Zustimmung" zeigt gute Ergebnisse bei der Reduktion von false positives; dies wird dadurch erreicht, dass nur Links in die Kandidaten-Liste mit aufgenommen werden, die von allen beteiligten Algorithmen als Link angesehen werden. Allerdings wird nicht erwartet, dass mit nur zwei Algorithmen der von Dekhtyar et al. beschriebene Effekt in vollem Umfang eintritt.

In der beschriebenen Fallstudie wurde eine Spezifikation eines Außenlichtsystems eines PKWs untersucht. Tabelle 1 beschreibt die Eigenschaften des Datensatzes. Dieser Datensatz enthält alle in Abschnitt 3.1 beschriebenen Herausforderungen, die aus Sicht des Autors durch verbesserte Algorithmen angegangen werden können. Weiterhin ist er größer als die meisten bisher in Veröffentlichungen beschriebenen Datensätze für halbautomatische Methoden. Der Datensatz „AB", der in [37] beschrieben ist, hat mit mehreren hundert Anforderungen auf zwei Abstraktionsebenen eine vergleichbare Größe. Allerdings werden im beschriebenen Experiment nicht alle Teile des Datensatzes verwendet. Zum Vergleich: Konrad und Gall [24] berichten, dass das manuelle Herstellen von Traceabi-

Tabelle 1. Datensatz-Eigenschaften

Außenlicht-Systemlastenheft	
Eigenschaft	Wert
Größe	ca. 500 Seiten
Größe (für Automobilindustrie)	mittel
Größe (Trace Komplexität [22])	groß
Spezifikationssprache	deutsch
Von Entwicklern gesetzte Traces	ca. 400
Anzahl textueller Anforderungen	2000+
Anzahl Strukturelemente	1000+
ø # Wörter pro Anforderung	20,2
ø # Wörter pro Strukturelement	3,1

lity von ungefähr 4000 Benutzer-Anforderungen durch den Entwicklungsprozess sehr aufwändig ist.

Die ersten Untersuchungen des Datensatzes [25] zeigten eine sehr große Zahl an Kandidaten-Links in der Ergebnis-Liste.

Der Datensatz enthält zwar Anforderungen verschiedener Abstraktionsebenen (Systemebene und Komponentenebene), jedoch sind die Anforderungen in nur einem Dokument gesammelt. Deshalb ist eine Unterscheidung der verschiedenen Ebenen nur über die Dokumentenstruktur möglich. Dies ist den Algorithmen momentan allerdings nicht möglich. Eine Analyse der Ergebnisse ergab aber eine Möglichkeit, False Positives zu entfernen: Links zwischen einer Überschrift und einem direkten Kind können ignoriert werden, da die Struktur im Dokument ja genau diese Zuordnung liefert. Außerdem zeigt sich, dass Links zwischen Geschwister-Anforderungen (Anforderungen die zur selben Überschrift gehören) entfernt werden können, da Abschnitte eine überschaubare Anzahl von Anforderungen enthalten. Bereits durch ihre Gruppierung unter der gleichen Überschrift ist die Zusammengehörigkeit der Anforderungen dokumentiert. Um diese zwei Arten von False Positives aus der Kandidaten-Liste zu entfernen werden Filter vorgeschlagen, die dann zum Einsatz kommen, wenn Links innerhalb einer Dokumentenstruktur gesucht werden. Es ist dem Autor bewusst, dass es für die Erkennung leichter wäre, den Datensatz in zwei Gruppen aufzuteilen die jeweils nur eine Abstraktionsebene enthalten. Das erklärte Ziel war aber, mit realen, unveränderten Spezifikationen zu arbeiten.

6 Ergebnisse

Wie in [25] beschrieben enthält der verwendete Datensatz neben hauptsächlich deutschem Text auch englischsprachige Anteile in Form von Domänen-Begriffen. Littman et al. [26] berichten, dass mittels LSI auch

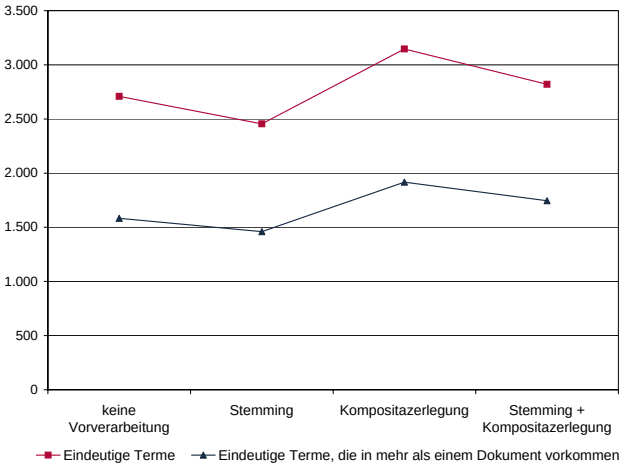

Abbildung 3. Anzahl Terme abhängig von gewählter Vorverarbeitung

mehrsprachige Datensätze sinnvoll durchsucht werden können. Diese Zweisprachigkeit ist für den Stemmer nicht optimal, er ist jedoch trotzdem in der Lage, die Anzahl der Terme zu verringern. Abbildung 3 zeigt, wie sich die einzelnen Vorverarbeitungsschritte auf die Termzahl im Datensatz auswirken. Die Termzahl bestimmt direkt eine Dimension der Term/Dokument-Matrix und beeinflusst damit auch die Berechnungszeit.

Der Datensatz enthält zwar bereits von Entwicklern erzeugte Links, diese sind allerdings nicht vollständig. Insbesondere sind die Links größtenteils auf Kapitel-Ebene (Strukturelementen-Ebene) angesiedelt und nicht auf Ebene der einzelnen Anforderungen. Mangels vollständigem Referenz-Traceability-Satz wurden die vorhandenen Verlinkungen als Referenz-Satz herangezogen.

Während man typischerweise darauf schaut, dass Precision (Abb. 5) und Recall (Abb. 6) in sinnvollen Bereichen liegen, ist es bei großen Dokumenten weiterhin notwendig, die Gesamtzahl der Links im Auge zu behalten. Abbildung 4 zeigt, wie schnell die absolute Zahl an potentiellen Links in der Kandidaten-Liste steigt wenn der Cutoff-Point nur leicht herabgesetzt wird. Angenommen, der Analyst benötigt nur fünf Sekunden um zu entscheiden ob ein Kandidaten-Link korrekt ist, so benötigt er bei 10.000 Links immerhin fast 14 Stunden, die Kandidaten-Liste zu analysieren.

Für alle beschriebenen Ergebnisse fand eine Stoppwort-Elimination statt. Der Algorithmus LSI verringerte die Größe der Matrix jeweils auf 5%, da dies die bestmöglichen Werte für den Recall ergab.

Die Vorverarbeitungsschritte Stemming und Kom-

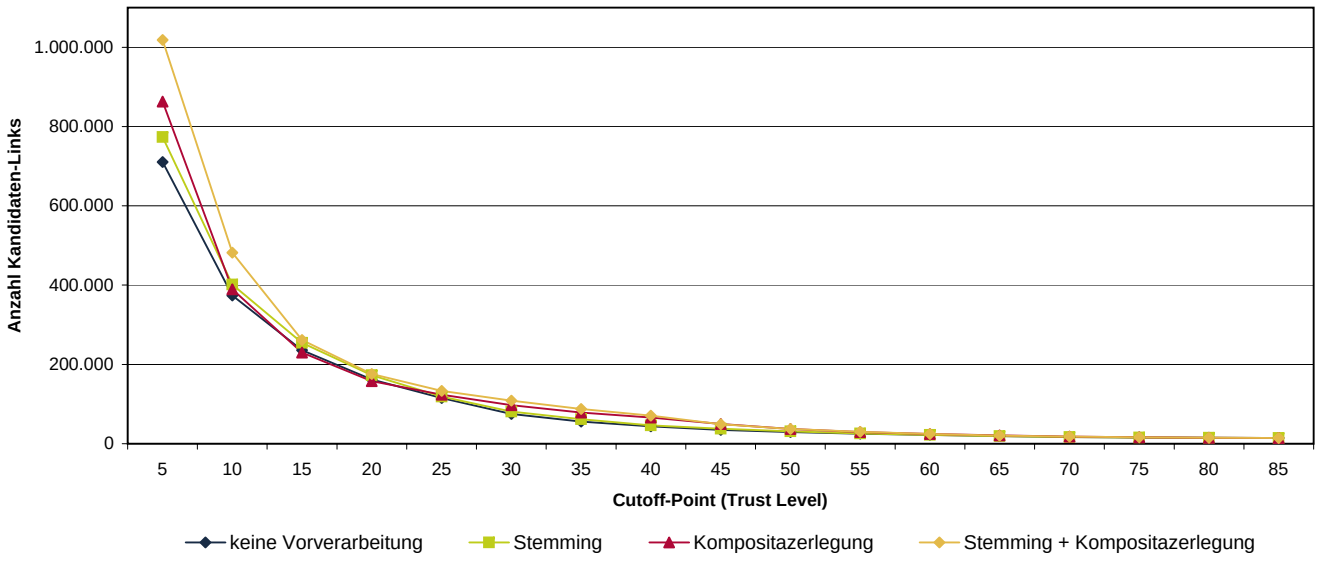

Abbildung 4. Kandidaten-Linkanzahl in Abhängigkeit des Cutoff-Points

positazerlegung haben beide positive Auswirkungen auf den Recall (Abb. 6), jedoch kaum Einfluss auf die Precision (Abb. 5). Allerdings wirkt sich die Kompositazerlegung bei einem Cutoff-Point im Intervall 20 bis 35 negativ auf den Recall aus. Da die Precision über den gesamten Bereich in absolut inakzeptablen Bereichen liegt, soll dieser Tatsache erst einmal keine Bedeutung zugewiesen werden. Da Stemming die Anzahl der Terme reduziert, hilft es zusätzlich zu den positiven Auswirkungen auf das Ergebnis auch den Berechnungsaufwand zu verringern. Wegen des verbesserten Recalls sollte Stemming auf jeden Fall eingesetzt werden.

Die Filterung ist in den Abbildungen nicht extra aufgeführt, da sie in allen Fällen nur geringfügigen Einfluss auf Recall (negativ) und Precision (positiv) hatte. Die Anzahl der Links in der Kandidaten-Liste wurde jedoch um 1-5% verringert.

Für Werte des Cutoff-Points im Bereich 85 bis 100 war der Recall 0, d.h. es wurden keine Links aus dem Referenzsatz mehr gefunden. Deshalb zeigen Abb. 6 und Abb. 5 diesen Wertebereich nicht. Trotzdem wurden noch Kandidaten-Links im fünf- bis vierstelligen Bereich gefunden (siehe Abb. 4). Die Untersuchung des Datensatzes ergab, dass im Datensatz oft Dokumente mit identischem Text vorhanden sind. Das Vorhandensein solcher Dokumente ist der Dokumentenstruktur geschuldet, die für die am System beteiligten Komponenten bestimmte Strukturelemente und Anforderungen vorgibt. Beispielsweise gibt es für jede Komponente einen Abschnitt, der beschreibt, welche Fehler von der Komponente im Fehlerspeicher abgelegt werden. Legt eine Komponente keine Fehler ab, so wird dies durch einen vorgegebenen Text spezifiziert.

Zwar ist der Recall bei sehr niedrigem Cutoff-Point akzeptabel, da jedoch die Precision und die Gesamtzahl an Kandidaten-Links in diesem Bereich so schlecht sind, ist das Ergebnis nicht praxistauglich. Ein Grund kann die Art der Links im Referenz-Satz sein: Da die Links, wie beschrieben, oft auf Struktur-Ebene angesiedelt sind stehen vergleichsweise wenig Daten pro Element zur Verfügung: Im Schnitt nur 3,1 Wörter pro Dokument, d.h. Struktur-Element (siehe Tab. 1). Weiterhin gab es einige Links, die von Dokumenten, die für die Algorithmus nicht zugänglich waren (z.B. Grafiken bzw. eingebettete Objekte), auf Strukturelemente zeigten.

7 Verwandte Arbeiten

Obwohl modellbasiertes Spezifizieren in der Automobilindustrie noch nicht im großen Maßstab eingesetzt wird, geht der Trend in Richtung modellbasierter Spezifikation. In späteren Phasen des Entwicklungsprozesses sind Modelle und Codegenerierung gängig. Verfahren von De Lucia et. al [12] und Cleleand-Huang et al. [10] zeigen, dass das Finden von Links zwischen Anforderungen und Modellen halbautomatisch vonstatten gehen kann.

Ratanotayanon et al. [34] fokussieren auf Traceability in späteren Phasen des Entwicklungszykluses. Mit ZELDA unterstützen sie die Verlinkung von Feature-Beschreibungen mit textbasierten Artefakten (in ih-

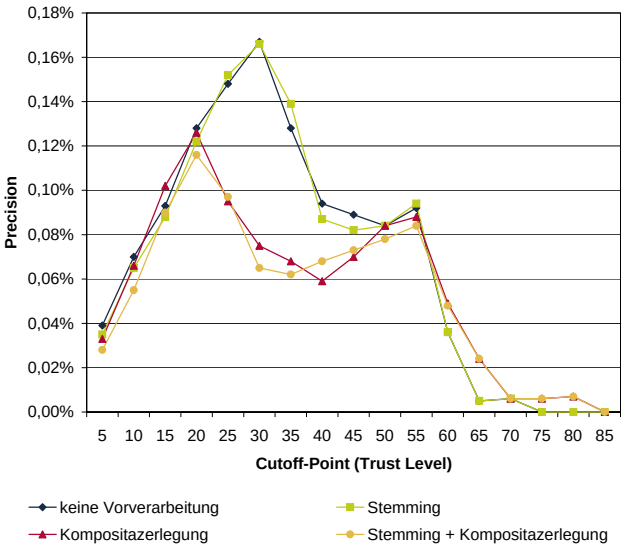

Abbildung 5. Precision in Abhängigkeit des Cutoff-Points

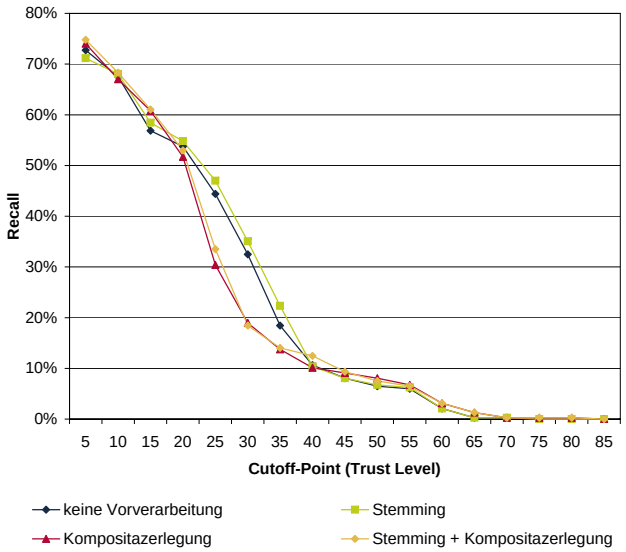

Abbildung 6. Recall in Abhängigkeit des Cutoff-Points

rem Fall Quellcode) und aktualisieren diese Links auch bei Änderungen im Code automatisiert. Auch Marcus und Maletic [28] zeigen, dass Links in Quellcode-Dokumente hinein halbautomatisch gefunden werden können. Egyed et al. [16] empfehlen Links zwischen Anforderungen und Code nur in grober Granularität vorzuhalten. Erst bei Bedarf sollten die vorhandenen Links verfeinert werden.

Da die IR-basierten Methoden auf Ähnlichkeiten von Texten basieren, sind sie nicht in der Lage, Links zwischen Artefakten zu finden, die unterschiedliches Vokabular verwenden. Deshalb schlagen McMillan et al. [30] vor, Links zwischen solchen Artefakten unter Zuhilfenahme von Strukturinformationen indirekt zu finden. In ihrem Vorgehen werden Links zwischen Dokumentations-Artefakten dadurch gefunden, dass diese einen Link zum selben Abschnitt im Quellcode haben. Insofern werden Dokument-Dokument Links auch dann gefunden, wenn eine Verbindung Dokument → Quellcode → Dokument besteht.

Winkler [37] beschreibt Optimierungen der Algorithmen um bei sich ändernden Artefakten nicht bei jeder neuen Suche alle Entscheidungen (durch den menschlichen Analysten) noch einmal treffen zu müssen. Dieses Vorgehen scheint geeignet zu sein, Herausforderung 6 anzugehen.

Für die Domäne e-Science schlagen Asuncion und Tylor [3] vor, Links zwischen Artefakten durch Beobachtung des Benutzers zu erkennen: Werden mehrere Artefakte sequentiell bzw. gemeinsam bearbeitet, so kann davon ausgegangen werden, dass diese miteinander in Beziehung stehen. Dieser Ansatz ist interessant und dürfte zumindest auch auf modellbasierte Entwicklungen übertragbar sein. Zusätzlich empfehlen sie, regelbasiert nach möglichen Links zu suchen wobei die Regeln auf Kontextinformationen basieren.

Marcus et al. [29] schlagen vor, Eigenschaften von Links zu visualisieren, da Zusatzinformationen in Link-Matrizen nur schwer darstellbar sind. Statt die Eigenschaften von Links an sich zu visualisieren wenden Duan, Cleland-Huang [14] und Habrat [8] Visualisierungen an um dem Analysten die Entscheidung über die Korrektheit von Kandidaten-Links zu erleichtern.

Cleland-Huang et al. [9] berichten, dass selbst wenn Traceability gepflegt wird, diese selten verwendet wird. Sie schlagen vor, modellbasiert eine Traceability-Strategie für jedes Projekt zu dokumentieren und darauf basierend dem Benutzer vorbereitete Abfragen an die Traceability zur Verfügung zu stellen. Damit soll es für den Benutzer vereinfacht werden, Nutzen aus den vorhandenen Links zu ziehen.

Verschiedene Forscher geben Hinweise, wie die halbautomatischen Methoden am effektivsten und effizientesten einzusetzen sind. Dekhtyar el al. [13] simulierten den Task eines Analysten, Traceability in einer fertigen Spezifikation herzustellen. Sie fanden heraus, dass zwei Faktoren besonders wichtig sind für Effizienz: Ein strukturiertes Vorgehen bei der Suche nach Links sowie das Finden des „richtigen“ Zeitpunktes, an dem man mit der Suche aufhört. Für Links zwischen Anforderungen und Code liefern Egyed et al. [16] Hinweise auf die optimale Granularität.

Obwohl die Rechenzeit, die für die Berechnungen bei LSI – dem komplexeren der beiden Algorithmen – benötigt wird, mit der aktuellen Datensatzgröße noch akzeptabel ist, kann man sich vorstellen, dass bei ca. 130 GB an Daten in Doors eine vollständige Analyse viel Rechenzeit und Speicher benötigen wird. Der teuerste Teil des LSI ist die Zerlegung der Term/Dokument-Matrix mittels Singular Value Decomposition (SVD). Dafür existieren verschiedene Optimierungen, die SVD inkrementell zu berechnen um beispielsweise bei Änderungen nur vergleichsweise „günstige“ Berechnungen durchführen zu müssen [19, 5]. Auch alternative Algorithmen werden vorgeschlagen [18]. Zusätzlich gibt es Ansätze [17], den benötigten Speicherbedarf zu verringern.

8 Weitere Schritte

Nachdem die Ergebnisse es unwahrscheinlich erscheinen lassen, dass solch große unbearbeitete Spezifikationen sich gut mittels aktueller halbautomatischer Methoden bearbeiten lassen, muss auch in Betracht gezogen werden, die Spezifikationen anzupassen.

Zuerst jedoch sollte ein Referenz-Traceability-Satz für die verwendete Spezifikation erstellt werden, der auch detaillierte Links enthält (d.h. auch auf Anforderungsebene). Es wird erwartet, dass die halbautomatischen Methoden besser auf Anforderungsebene funktionieren, da diese typischerweise mehr Text enthalten als Strukturelemente (siehe Tab. 1). Erste Tests mit einem erweiterten Traceability-Satz (knapp 1000 Links, jedoch noch keine vollständige Referenz-Traceability) zeigen eine Erhöhung des Recalls bei einem Cutoff-Point von 5 auf 90,5% sowie eine Precsion von 0,15% und bestätigen damit die Erwartung.

Auf Grund der großen Zahl an Strukturelementen (Verhältnis ca. 1:2 zu Anforderungen) scheint die Integration von Optimierungen angebracht, die von der vorhandenen Strukturierung profitieren.

9 Fazit

Zwar zeigt sich, dass die Ergebnisse mit halbautomatischen Methoden vielversprechend sind – solange sie sich auf vergleichsweise kleine Datensätze beschränken. Für reale, deutschsprachige Spezifikationen in der Größenordnung von 500 Seiten sind die Ergebnisse nicht praxistauglich. Vorverarbeitungsschritte wie Stemming und Kompositazerlegung zeigen zwar positive Auswirkungen auf die Ergebnisse, in der extrem großen Menge der Kandidaten-Links fallen die positiven Auswirkungen jedoch nicht hinreichend groß aus. Auch das Filtern von Kandidaten-Links in Abhängigkeit der strukturellen Abhängigkeit der verlinkten Elemente ist ein Tropfen auf den heißen Stein. Den aktuellen Ergebnissen nach zu urteilen müssen noch einige Verbesserungen eingesetzt werden um zu praxistauglichen Ergebnissen zu gelangen.

Literatur

[1] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia. Identifying the starting impact set of a maintenance request: a case study. *Software Maintenance and Reengineering, 2000. Proceedings of the Fourth European*, pages 227–230, 2000.

[2] H. U. Asuncion. Towards practical software traceability. In *ICSE Companion '08: Companion of the 30th international conference on Software engineering*, pages 1023–1026, New York, NY, USA, 2008. ACM.

[3] H. U. Asuncion and R. N. Taylor. Capturing custom link semantics among heterogeneous artifacts and tools. In *Traceability in Emerging Forms of Software Engineering, 2009. TEFSE '09. ICSE Workshop on*, 2009.

[4] R. Baeza-Yates and B. d. A. N. Ribeiro. *Modern information retrieval*. ACM Press books. Pearson Addison-Wesley, reprint edition, 2006.

[5] M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20 – 30, 2006. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[6] M. Braschler and B. Ripplinger. How effective is stemming and decompounding for german text retrieval? *Information Retrieval*, 7(3-4):291–316, September 2004.

[7] J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi, and E. Romanova. Best practices for automated traceability. *Computer*, 40(6):27–35, 2007.

[8] J. Cleland-Huang and R. Habrat. Visual support in automated tracing. *Requirements Engineering Visualization, First International Workshop on*, 0:4, 2007.

[9] J. Cleland-Huang, J. H. Hayes, and J. M. Domel. Model-based traceability. In *Traceability in Emerging Forms of Software Engineering, 2009. TEFSE '09. ICSE Workshop on*, pages 6–10, May 2009.

[10] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou. Utilizing supporting evidence to improve dynamic requirements traceability. In *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pages 135–144, Washington, DC, USA, 2005. IEEE Computer Society.

[11] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Enhancing an artefact management system with traceability recovery features. In *20th IEEE International Conference on Software Maintenance (ICSM'04)*, pages 306–315, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[12] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Trans. Softw. Eng. Methodol.*, 16(4):13, 2007.

[13] A. Dekhtyar, J. H. Hayes, S. Sundaram, A. Holbrook, and O. Dekhtyar. Technique integration for requirements assessment. In *15th IEEE International Requirements Engineering Conferece (RE'07)*, pages 141 –150. IEEE Computer Society, October 2007.

[14] C. Duan and J. Cleland-Huang. Visualization and analysis in automated trace retrieval. *First International Workshop on Requirements Engineering Visualization (REV'06 - RE'06 Workshop)*, page 5, 2006.

[15] A. Egyed, S. Biffl, M. Heindl, and P. Grünbacher. Determining the cost-quality trade-off for automated software traceability. In *20th IEEE/ACM international Conference on Automated software engineering (ASE'05)*, pages 360–363, New York, NY, USA, 2005. ACM.

[16] A. Egyed, P. Grünbacher, M. Heindl, and S. Biffl. Value-based requirements traceability: Lessons learned. *15th IEEE International Requirements Engineering Conference (RE'07)*, 1:115–118, October 2007.

[17] J. Gao and J. Zhang. Sparsification strategies in latent semantic indexing. *Proceedings of the 2003 Text Mining Workshop, San Francisco, CA*, pages 93–103, May 2003.

[18] G. Gorrell and B. Webb. Generalized hebbian algorithm for incremental latent semantic analysis. *Proceedings of Interspeech 2005*, 2005.

[19] J. Götze and P. Rieder. Svd-updating using orthonormal μ-rotations. *The Journal of VLSI Signal Processing*, 14(1):7–17, 1996.

[20] A. Győry. Mapping zwischen Anforderungen und Design in mechatronischen Systemen. Diplomarbeit, Technische Universität München, November 2008.

[21] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Trans. Softw. Eng.*, 32(1):4–19, 2006.

[22] J. H. Hayes, A. Dekhtyar, S. K. Sundaram, and S. Howard. Helping analysts trace requirements: an objective look. *12th IEEE International Requirements Engineering Conference, 2004*, pages 249–259, 2004.

[23] Herstellerinitiative Software (HIS). Requirement interchange format (RIF). Online, May 2007.

[24] S. Konrad and M. Gall. Requirements engineering in the development of large-scale systems. *16th IEEE International Conference on Requirements Engineering (RE'08)*, pages 217–222, 2008.

[25] J. Leuser. Challenges for semi-automatic trace recovery in the automotive domain. *Traceability in Emerging Forms of Software Engineering, 2009. TEFSE '09. ICSE Workshop on*, pages 31–35, May 2009.

[26] M. Littman, S. T. Dumais, and T. K. Landauer. Automatic cross-language information retrieval using latent semantic indexing. *Cross-Language Information Retrieval*, pages 51–62, 1998.

[27] M. Lormans and A. van Deursen. Can LSI help reconstructing requirements traceability in design and test? In *CSMR '06: Proceedings of the Conference on Software Maintenance and Reengineering*, pages 47–56, Washington, DC, USA, 2006. IEEE Computer Society.

[28] A. Marcus and J. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. *Proceedings of the Twenty-Fifth International Conference on Software Engineering*, pages 3–10, 2003.

[29] A. Marcus, X. Xie, and D. Poshyvanyk. When and how to visualize traceability links? In *TEFSE '05: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pages 56–61, New York, NY, USA, 2005. ACM.

[30] C. McMillan, D. Poshyvanyk, and M. Revelle. Combining textual and structural analysis of software artifacts for traceability link recovery. In *Traceability in Emerging Forms of Software Engineering, 2009. TEFSE '09. ICSE Workshop on*, pages 41–48, May 2009.

[31] J. N. och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. A feasibility study of automated natural language requirements analysis in market-driven development. *Requirements Engineering*, 7(1):20–33, 2002.

[32] B. Penzenstadler and J. Leuser. Complying with law for RE in the automotive domain. In *First international workshop on Requirements and Law (RELAW)*, pages 11–15. IEEE, September 2008.

[33] A. Pretschner, M. Broy, I. H. Krüger, and T. Stauner. Software engineering for automotive systems: A roadmap. In *FOSE '07: 2007 Future of Software Engineering*, pages 55–71, Washington, DC, USA, 2007. IEEE Computer Society.

[34] S. Ratanotayanon, S. E. Sim, and D. J. Raycraft. Cross-artifact traceability using lightweight links. In *Traceability in Emerging Forms of Software Engineering, 2009. TEFSE '09. ICSE Workshop on*, pages 57–64, May 2009.

[35] E. Uusitalo, M. Komssi, M. Kauppinen, and A. Davis. Linking requirements and testing in practice. *16th IEEE International Conference on Requirements Engineering (RE'08)*, pages 265–270, 2008.

[36] VDA. *Automotive VDA Standardstruktur Komponentenlastenheft.* Verband der Automobilindustrie (VDA), 1st edition, 2007.

[37] S. Winkler. Trace retrieval for evolving artifacts. In *Traceability in Emerging Forms of Software Engineering, 2009. TEFSE '09. ICSE Workshop on*, pages 49–56, May 2009.