Automatisierte Identifikation und Charakterisierung von Anforderungsabhängigkeiten – Literaturstudie zum Vergleich von Lösungsansätzen

Automated Identification and Characterization of Requirement Dependencies - Literature Study to Compare Solution Approaches

Iris Gräßler[1], Daniel Preuß[1]*, Christian Oleff[1]

[1] Heinz Nixdorf Institute, Chair of Product Creation, Paderborn University

** Korrespondierender Autor:*
Daniel Preuß
Universität Paderborn
Heinz Nixdorf Institut
Lehrstuhl für Produktentstehung
Fürstenallee 11
33102 Paderborn
Telefon: 05251/606259
Mail: daniel.preuss@hni.upb.de

Abstract

Many development projects of complex, interdisciplinary products fail due to an inefficient handling of requirement changes. One reason for this is that requirement dependencies and the resulting change propagation are not sufficiently recognized and considered by existing approaches. Based on a literature study and two industry workshops, approaches of automated dependency analysis of requirements for interdisciplinary development are evaluated. The result is that ontology-based approaches are particularly suitable for the automated dependency analysis of requirements due to the use of expert knowledge and the possibility of automation. On this basis, subsequent research activities can further develop the handling of requirements changes in interdisciplinary projects in a targeted manner.

Keywords

Requirements Engineering, Requirements Change Management, Dependency Detection, Interdisciplinary Product Development, Literature Study

 | https://doi.org/10.35199/dfx2020.21

1. Motivation

Eine wesentliche Herausforderung bei der Entwicklung komplexer, interdisziplinärer Systeme ist der effiziente Umgang mit Anforderungsänderungen. Untersuchungen zeigen, dass Anforderungsänderungen unvermeidbar sind, in allen Phasen eines Entwicklungsprojekts auftreten und über die Hälfte aller Anforderungen an das System einer hohen Änderungswahrscheinlichkeit unterliegen [1]. Die Handhabung und Implementierung dieser Änderungen führt zu einem zusätzlichen Kosten- und Zeitaufwand, welche auch zu einem Fehlschlag des Entwicklungsvorhabens führen können [2]. Anforderungsänderungen sind im Ursprung auf das dynamische Projektumfeld sowie den fortschreitenden Erkenntnisgewinn in der Entwicklung zurückzuführen. Darüber hinaus sind Anforderungen abhängig voneinander, weshalb die Änderung einer Anforderung dazu führen kann, dass weitere Anforderungsänderungen ausgelöst werden. Dadurch werden die Auswirkungen einer initialen Änderung im Sinne einer Änderungspropagation vervielfacht.

Der effektive Umgang mit Anforderungsänderungen erfordert die Einbeziehung der kollektiven Auswirkungen im gesamten Anforderungsset, welche sich nur auf Grundlage der Anforderungsabhängigkeiten abschätzen lassen [3]. Dies verdeutlich ein Beispiel aus der Entwicklung eines LED-Frontscheinwerfers für BMW [4]: Die Anforderung an die maximale Temperatur des LED-Substrats ist ein zentraler Einflussfaktor auf die Leistungsfähigkeit des Scheinwerfers und wurde ursprünglich auf 120°C festgelegt. Dies wurde im Laufe der Entwicklung auf 90°C reduziert. Zur Reduktion der Temperatur mussten mehr LEDs verbaut werden, deren Energiebedarf jeweils geringer ist. Dies wiederum erforderte die Anpassung der Anforderungen an den Bauraum, sodass aus der ursprünglichen Anforderungsänderung Auswirkungen auf Anforderungen an Energiebedarf, Anzahl der LEDs und Geometrie resultierten.

2. Forschungsproblem und Forschungsziel

Der Umgang mit Anforderungsänderungen ist in drei **Handlungsfelder** einzuordnen: das Risikomanagement im Vorfeld einer Änderung, die Entscheidungsfindung bei einem vorliegenden Änderungsbedarf (Change Request) und die darauffolgende Implementierung dieser Änderung in das Anforderungsset [5; 6]. Allen Handlungsfeldern ist gemein, dass Wissen über Anforderungsabhängigkeiten erforderlich ist. Sowohl im Risikomanagement als auch bei der Entscheidungsfindung können nur anhand der Abhängigkeiten die ganzheitlichen Auswirkungen einer Änderung abgeschätzt werden [7]. Hinsichtlich der Implementierung einer Änderung sind die Abhängigkeiten insbesondere für eine konsistente Anpassung der Anforderungen ausschlaggebend, da Inkonsistenzen zu unzureichender Anforderungserfüllung oder späten und ressourcenintensiven Änderungen führen [5]. **Anforderungsabhängigkeiten** werden nach der Art oder dem Grad der Abhängigkeit charakterisiert. Arten von Anforderungsabhängigkeiten werden in Referenzmodellen abgebildet DAHLSTEDT [8] unterteilt die Abhängigkeitsarten in verschiedene Sub-Klassen. So ist „Requires“ z. B. eine Abhängigkeitsart der Sub-Klasse „Condition“. Diese Abhängigkeitsart beschreibt, dass eine Anforderung nur realisiert werden kann, wenn eine andere Anforderung realisiert wird. Anforderungsabhängigkeiten sind bidirektional oder, wie in dem Beispiel, unidirektional. In der Literatur existieren weitere Referenzmodelle wie von POHL [9], KARLSSON [10] und CARLSHAMRE [11]. Sie unterscheiden sich in den Arten von Anforderungsabhängigkeiten und deren Unterteilung in Sub-Klassen.

Unterstützt wird der Umgang mit Anforderungsänderungen sowie die Modellierung von Abhängigkeiten durch diverse Requirements Engineering **Werkzeuge**. Den größten Marktanteil unter Werkzeugen, die für das Requirements Engineering genutzt werden, hat Microsoft Office. Dabei wird die geringe Einstiegshürde und Alltagstauglichkeit von Excel (Verwaltung) und Word (Dokumentation) genutzt. Es sind keine spezifischen Funktionalitäten

zur Identifikation oder Modellierung von Abhängigkeiten bzw. Requirements Engineering im Allgemeinen vorhanden. Softwareprodukte mit Fokus auf das Requirements Engineering wie z. B. IBM DOORS oder Siemens Polarion Requirements haben im Gegensatz dazu eine Vielzahl an Funktionalitäten hinsichtlich der Modellierung von Abhängigkeiten und der Nachverfolgbarkeit von Anforderungen (Traceability) [12]. Diesen Produkten ist jedoch gemein, dass lediglich manuelles Einpflegen von Abhängigkeiten möglich ist. Eine (teil-) automatisierte Identifikation von Abhängigkeiten wird nicht oder ausschließlich auf der Basis von Inkonsistenzen in der bestehenden Datengrundlage unterstützt. Im Zuge dieses manuellen Prozesses ist die Charakterisierung von Abhängigkeiten vielfach möglich, wird aber auf Aspekte der Verwaltung von Anforderungen fokussiert (z. B. hierarchische Abhängigkeiten oder Abhängigkeiten zu Testfällen) und lässt nur sehr rudimentäre Rückschlüsse auf den Propagationseffekt zu.

Derzeit werden Anforderungsabhängigkeiten gar nicht oder unsystematisch, reaktiv und auf Basis einer lückenhaften Informationsgrundlage in die **Entwicklungsentscheidungen** eingebunden [1; 3]. Das führt insbesondere bei komplexen Systemen mit einer hohen Anzahl an Anforderungen zu Inkonsistenzen im Anforderungsset und Korrekturkosten in der Entwicklung. Lösungsansätze, welche die Identifikation und Charakterisierung von Abhängigkeiten in interdisziplinären Anforderungssets ermöglichen, existieren nur vereinzelt [1; 13; 14] und zeigen unter anderem Schwachstellen hinsichtlich Automatisierungsgrad und Aussagekraft der Ergebnisse zur Änderungspropagation. Der Automatisierungsgrad ist für die Anwendbarkeit im Kontext komplexer Systemen bedeutsam, da durch die hohe Anzahl an Anforderungen der Arbeitsaufwand einer manuellen Beurteilung, z. B. durch paarweisen Vergleich, nicht praxistauglich ist [3]. Für die Abschätzung des Propagationseffekts ist zudem eine Charakterisierung der Abhängigkeiten nach Art und Stärke von Relevanz, weshalb auch dieser Aspekt durch Lösungsansätze abzudecken ist [15]. Da beide Funktionalitäten im Kontext interdisziplinärer Anforderungen unzureichend erforscht sind, ist das Ziel des Forschungsvorhabens die Beantwortung der folgenden Forschungsfrage: „*Wie können frühzeitig Abhängigkeiten in Anforderungssets interdisziplinärer Systeme für die Propagationsanalyse automatisiert identifiziert werden?*“.

3. Methodik

Die Beantwortung dieser Fragestellung ist Teil des vom BMBF geförderten Projekts „Automated Requirement Change Analysis“ (ARCA), das entsprechend Typ 5 der Design Research Methodology nach BLESSING und CHAKRABATI [16] strukturiert ist. Dieser Beitrag ist der Phase Descriptive Study I (DS-I) des übergeordneten Vorhabens zuzuordnen und folgt einem dreischrittigen Vorgehen, um Lösungsansätze für die automatisierte Abhängigkeits-analyse im interdisziplinären Entwicklungskontext zu identifizieren und zu vergleichen. Im ersten Schritt werden auf Basis eines Industrieworkshops und einer Vorab-Literaturanalyse zu bereits publizierten Anforderungen 28 Stakeholder-Anforderungen (im Sinne von [17]) an einen Lösungsansatz zur Identifikation und Charakterisierung von Anforderungs-abhängigkeiten aufgenommen und dazu Use-Cases entwickelt. Nach dem Ansatz von BALZERT [18] folgend werden die Anforderungen als essenziell, bedingt notwendig oder optional priorisiert. Anforderungen sind beispielsweise: *„Der Ansatz soll unternehmensspezifisches Wissen einbeziehen.“* und *„Anforderungsabhängigkeiten sollen hinsichtlich des Propagationsverhaltens unterschieden werden“*. Die Ausarbeitung von Use Cases wird genutzt, um in nachfolgenden Gesprächen mit den Industrieanwendern Korrektheit und Vollständigkeit der erhobenen Anforderungen sicherzustellen. Nachdem die Anforderungsdefinition abgeschlossen ist, erfolgt im zweiten Schritt eine systematische Literaturstudie nach MACHI und MCEVOY [19] zur Identifikation von Lösungsansätzen. Dieses Vorgehen wird gewählt, da es durch Abstraktion der Lösungsansätze in einer Tally-Matrix den inhaltlichen Vergleich von Ansätzen aus unterschiedlichsten Anwendungsfeldern erlaubt. Im

letzten Schritt wird die Bewertung und Auswahl geeigneter Ansätze umgesetzt. Dafür wird auf eine Nutzwertanalyse zurückgegriffen. Die Kriterien der Nutzwertanalyse werden auf Basis der Anforderungen erarbeitet und in einem zweiten Workshop mit sechs Industrievertretern diskutiert und gewichtet. Es wird ein paarweiser Vergleich der Kriterien durchgeführt, um die Priorität zu ermitteln. Dies erfolgt ausschließlich durch die Industrievertreter, um die Bedeutung der Kriterien aus Anwender-Perspektive zu berücksichtigen. Anschließend wird unter Berücksichtigung der Anwender-Priorisierung sowie der Anzahl und Priorität der zugrundeliegenden Anforderungen eine qualitative Gewichtung für jedes Kriterium festgelegt. Um unterschiedliche Rollen in der Entwicklung zu berücksichtigen, werden für beide Workshops dieselben Industrievertreter aus allen für die Anwendung relevanten Aufgabenfelder beteiligt: Abteilungsleiter, Entwickler, Anforderungs- und Prozessmanager, Projektleiter und Spezialist.

Für die Literaturstudie wird auf Google Scholar und im Web of Science eine bool'sche Stichwortsuche durchgeführt. Die Stichworte werden direkt aus der Forschungsfrage abgeleitet, z. B. „Dependency Detection" AND „Requirement". Auf Basis der Titel und der Abstracts der Veröffentlichungen werden relevante Veröffentlichungen selektiert. Die Ergebnisse der Literaturstudie werden in einer Tally-Matrix [19] dokumentiert (siehe Tabelle 1). In einer Tally-Matrix werden Konzepte und Argumente sowie die Argumentationsstruktur von Autoren gesammelt und hinsichtlich verschiedener Kriterien wie der Qualität der Daten bewertet. Durch die Stichwortsuche wird unter anderem eine Veröffentlichung von MOTGER ET AL. im Kontext des EU-Projekts „OpenReq" identifiziert [20]. In diesem Projekt wird ein Ansatz zur automatisierten Abhängigkeitsanalyse mit Fokus auf die Software-Entwicklung entwickelt. Auf Basis der Projektergebnisse und Publikationen werden weitere relevante Ansätze zur Abhängigkeitsanalyse identifiziert und in die Tally-Matrix aufgenommen [21–29]. Die Inhalte der Tally-Matrix werden genutzt, um die Lösungsansätze hinsichtlich der verschiedenen Kriterien (siehe Abschnitt 4) zu bewerten.

4. Bewertungskriterien

In diesem Abschnitt werden die zehn Kriterien zur Bewertung der Lösungsansätze erläutert. Die Kriterien werden inhaltlich anhand der **vier Kategorien** Input/Datengrundlage, Output/Ergebnisse, Anwendung und Aufwand gruppiert. Die Bewertungskriterien sind überschneidungsfrei und unabhängig von Aspekten der Nutzerfreundlichkeit.

Der Lösungsansatz soll **Expertenwissen (K1.1)** bei der Anwendung berücksichtigen. Dies erlaubt eine Anwendbarkeit in frühen Entwicklungsphasen ohne umfassende Datenbasis und stellt eine Möglichkeit dar, das Erfahrungswissen der Anwender einzubeziehen. Dieses implizite Wissen der Experten ist häufig die wichtigste Entscheidungsgrundlage in Unternehmen. Da es neben Erfahrungswerten auch auf Fachwissen beruht, ermöglicht es zudem die Identifkation von Abhängigkeiten, welche einer rein restrospektiven Bewertungsgrundlage unzugänglich sind. Um bereits **früh in der Entwicklung (K1.2)** das Änderungsrisiko von Anforderungen beurteilen zu können, muss die eingeschränkte Datengrundlage, die zu diesem Zeitpunkt vorliegt (insbesondere natürlichsprachige Anforderungstexte), ausgewertet werden können.

Für den Output bzw. die Ergebnisse werden drei Kriterien unterschieden. Erstes Kriterium ist die **Differenzierung der Abhängigkeiten (K2.1)** nach Intensität und Art. Dies ist wichtig zur Beurteilung des Propagationsverhalten, da eine binäre Beurteilung, ob eine Abhängigkeit vorherrscht oder nicht, nur eingeschränkte Aussagekraft hat und zu starken Verzerrungen führen können. Hierarchische oder kausale Abhängigkeiten (z. B. naturwissenschaftliche Gesetzmäßigkeiten) führen beispielsweise zwangsläufig zu einer Propagation der Änderung, wenn auch mit unterschiedlichen Auswirkungen. Weniger ausgeprägte Abhängigkeitsarten (z. B. „konfliktär") hingegen lassen nur bedingt Rückschlüsse auf das Propagationsverhalten zu und erfordern eine kontextspezifische Analyse. Auch wenn eine Abhängigkeit vorliegt, muss

diese nicht zwangsläufig zu einer Änderungspropagation führen. Ein weiteres Kriterium ist die **Lernfähigkeit (K2.2)** des Ansatzes. Dieses Kriterium beschreibt, inwiefern die Ergebnisgüte des Ansatzes mit wiederholter Anwendung zunimmt. Ein Ansatz gilt als **erprobt (K2.3)**, wenn dieser für die Anwendung in interdisziplinären Entwicklungsprojekten etabliert ist, bzw. regelmäßig Anwendung findet.

Die **Transferfähigkeit (K3.1)** beschreibt, ob der Ansatz auch auf andere Anwendungsfälle übertragbar ist. Die Ansätze werden hauptsächlich in der Software-Entwicklung angewandt. Der Aufwand zur Übertragbarkeit wird nicht betrachtet, sondern ausschließlich die Machbarkeit. Der Lösungsansatz ist **nachvollziehbar (K3.2)**, wenn es für den Anwender transparent ist, wie die Ergebnisse gebildet wurden und diese auf kausalen Zusammenhängen beruhen. Dies ist wichtig für die Akzeptanz und Weiterverwendung der Ergebnisse. Die **Robustheit (K3.3)** beschreibt, inwiefern die Ergebnisse sich ändern, wenn bestimmte Randbedingungen nicht eingehalten wurden. Dies bezieht sich vor allem auf das Format der Input-Daten. Viele der Ansätze wenden Techniken zur natürlichen Sprachverarbeitung an. Ein Ansatz ist z.B. robust, wenn er keine kontrollierte Sprache oder die Einhaltung eines Template verlangt, um eine hohe Ergebnisgüte zu erzeugen.

Ein **hoher Automatisierungsgrad (K4.1)** ist notwendig, um umfangreiche Anforderungssets mit dem Lösungsansatz bearbeiten zu können, ohne das ein hoher Anwendungsaufwand für den Nutzer resultiert. Dies ist besonders für komplexe interdisziplinäre Entwicklungsprojekte relevant. Das Kriterium **Datenvorbereitung (K4.2)** beschreibt wie hoch der Aufwand ist, um den Ansatz anwenden zu können.

5. Ergebnisse der Literaturstudie

Die verschiedenen Lösungsansätze zur Erkennung von Anforderungsabhängigkeiten werden hinsichtlich der Differenzierung der Anforderungen (K2.1) geordnet. Es wird zwischen Ansätzen unterschieden, die eine differenzierte, semi-differenzierte und singuläre Betrachtung vornehmen. Ansätze zur differenzierten Betrachtung bestimmen die Art der Abhängigkeit zwischen Anforderungen. Ansätze zur semi-differenzierten Betrachtung bestimmen lediglich den Grad der Abhängigkeit zwischen Anforderungen. Ansätze zur singulären Betrachtung bestimmen, ob eine bestimmte Abhängigkeitsart vorliegt oder nicht.

5.1. Ansätze zur differenzierten Betrachtung von Anforderungsabhängigkeiten

Die Ansätze zur differenzierten Betrachtung von Abhängigkeitsarten werden, der Art des Lösungsansatzes folgend, zwischen ontologiebasierten Ansätzen, Ansätze des Maschinellen Lernens und manuellen Ansätzen unterschieden. In **ontologiebasierten Ansätzen** wird Expertenwissen in einer Wissensbasis, der Ontologie, formalisiert (K1.1). Hierbei werden Entitäten mit bestimmten Abhängigkeitsarten vernetzt. Der Zeitpunkt, an dem die Daten verfügbar sind, ist abhängig vom Ansatz (K1.2). Einige Ansätze beginnen bereits mit der Aufnahme von Anforderungen [30; 31; 26], andere setzen erst an, wenn die Architektur des Systems bekannt ist [20; 32; 33]. Eine Lernfähigkeit ist teilweise gegeben, da die Ontologie angepasst werden kann (K2.2). Der ontologiebasierte Ansatz aus dem EU-Projekt OpenReq wurde ausführlich anhand von drei Industrie-Fallbeispielen validiert [20]. Es existieren noch keine kommerziellen Lösungen, die eine automatisierte Abhängigkeitsanalyse ermöglichen, weshalb automatisierte Ansätze höchstens eine Teil-Erprobtheit besitzen (K2.3). Unterschiede bei den Ansätzen bestehen in der modellierten Domäne. Ontologien können für den bestimmten Anwendungsfall angepasst werden (K3.1). Die Nachvollziehbarkeit der Ergebnisse ist teilweise gegeben, da es auf kausalen Zusammenhängen zwischen den einzelnen Entitäten beruht (K3.2). Die Berechnung der semantischen Ähnlichkeit von Anforderungen zu Entitäten in der Ontologie basiert jedoch auf probabilistischen Methoden. Einige Ansätze fordern eine bestimmte Struktur zur Formulierung der Anforderungen (K3.3).

Es werden Techniken der natürlichen Sprachverarbeitung angewandt, um Anforderungen den einzelnen Entitäten automatisiert zuzuweisen (K4.1). Anhand der Zuordnung der Entitäten zu den Anforderungen werden Anforderungsabhängigkeiten direkt abgeleitet. Die Erstellung der Ontologie für den jeweiligen Anwendungsfall ist zeitaufwendig (K4.2).

In Ansätzen, die **Techniken des Maschinellen Lernens** verwenden [34; 27; 28], werden Klassifikatoren mit dem Lerntyp (schwaches) überwachtes Lernen trainiert. Input ist die textuelle Beschreibung von Anforderungen.Beim überwachten Lernen wird der Output vom Experten für das Training bewertet (K1.1). Nachteilig ist, dass eine große Menge an Daten notwendig ist, um die Klassifikatoren zu trainieren. Die frühzeitige Bereitstellung solcher Daten kann zudem nur gewährleistet werden, wenn das Entwicklungsprojekt ähnlich zu vergangenen Projekten ist und der Klassifikator mit historischen Daten trainiert werden kann (K1.2). Eine größere Datengrundlage führt zu einer höheren Ergebnisgüte (K2.2). Entwicklungsprojekte besitzen einen hohen Neuheitsgrad, wodurch die Übertragbarkeit eines Klassifikators, der für ein anderes Entwicklungsprojekt trainiert wurde, auf ein weiteres Projekt jedoch nicht möglich ist (K3.1) oder zu einer schlechten Ergebnisgüte führen kann. Ansätze des Maschinellen Lernens basieren auf probabilistischen Methoden und sind daher in der Regel nicht nachvollziehbar (K3.2). Anforderungen können als textuelle Beschreibungen ohne spezielle Restriktionen verarbeitet werden (K3.3). Der Output der Ansätze ist die Klassifikation der Art der Abhängigkeit zwischen den Anforderungen und wird automatisiert berechnet (K4.1). Der Aufwand zur Anwendung eines Klassifikators, der bereits trainiert wurde, ist gering (K4.2).

In der Literaturrecherche wurde der Fokus auf Ansätze mit hohem Automatisierungsgrad gelegt, die für eine hohe Anzahl an Anforderungen anwendbar sind (s. Abschnitt 2). Aus diesem Grund wird nur ein einzelner, exemplarischer manueller Ansatz betrachtet. Ein Beispiel präsentieren KÖHLER ET AL. Sie prüfen, welche Eigenschaften des Produkts von einem Änderungsantrag abhängig sind [13]. Anschließend wird anhand einer **manuell** erstellten Abhängigkeitsmatrix geprüft, welche Merkmale des Produkts die Eigenschaften beeinflussen. Daraus wird ersichtlich, welche Merkmale durch den Änderungsbedarf betroffen sind.

5.2. Ansätze zur semi-differenzierten Betrachtung von Anforderungsabhängigkeiten

HEIN ET AL. berechnen semantische Ähnlichkeiten zwischen Anforderungen [1]. Semantisch ähnliche Anforderungen werden als abhängig klassifiziert und in einer Matrix repräsentiert. Sie gewichten den Grad der Abhängigkeit zwischen den Anforderungen anhand der Pfadlänge. Das Problem solch einer semi-differenzierten Betrachtung ist, dass die Abhängigkeitsarten nicht bekannt sind. Es ist keine präzise Aussage zur Änderungspropagation möglich. Ursache-Wirkungs-Beziehungen von Anforderungsänderungen werden nicht erkannt. Aus diesem Grund wurde nur ein exemplarischer Ansatz ausgewählt, der typische Charakteristika semi-differenzierter Ansätze vereint und diese Art von Ansätzen nicht weiter betrachtet.

5.3. Ansätze zur singulären Betrachtung von Anforderungsabhängigkeiten

Weiterhin existieren Ansätze, die einzelne Abhängigkeitsarten detektieren und keinen übergreifenden Ansatz anstreben. HAMDAQA hat einen Ansatz zur Erkennung externer Querverweise entwickelt [21]. Es werden automatisiert Textstellen hervorgehoben. OCH DAG ET AL. verwenden lexikalische Analysen in Kombination mit Stemming und der Entfernung von Stopp-Wörtern, um die Ähnlichkeiten von Anforderungen untereinander zu bestimmen [22]. Sie setzen dabei Ähnlichkeit mit Abhängigkeit gleich. MARTINEZ ET AL. verwenden ebenfalls semantische Ähnlichkeiten, um Abhängigkeiten zu erkennen. Hierzu vergleichen sie Anforderungsbeschreibungen mit Use-Cases und Szenarien [35]. PARK ET AL. berechnen die Ähnlichkeit zwischen Anforderungssets, um mögliche Redundanzen und Inkonsistenzen zu identifizieren, und extrahieren die möglichen mehrdeutigen Anforderungen [36]. Die Ähnlichkeitsmessmethode kombiniert ein Schiebefenster- und ein Parsermodell. Es existieren

weitere Ansätze, um Inkonsistenzen in Anforderungssets zu detektieren [23; 24]. DI THOMAZZO ET AL entwickeln einen Ansatz zur automatischen Erstellung einer Requirements Traceability Matrix (RTM) auf der Grundlage von Fuzzy-Logik [37]. Um die Abhängigkeit zwischen funktionalen Anforderungen zu bestimmen, werden die Frequenzvektor- und die Cosinus-Ähnlichkeitsmethode verwendet. ABADI ET AL. vergleichen die Effektivität verschiedener Information Retrieval-Techniken zum Auffinden von Traceability-Links vom Code zur Dokumentation (Anforderungen, Benutzerhandbücher, …) [38]. Sie untersuchen Latent Semantic Indexing, Probabilistic Latent Semantic Indexing, Sufficient Dimensionality Reduction und das Jenson-Shennon Similarity Model.

5.4. Auswertung der Ergebnisse

In der folgenden Abgrenzungsmatrix (s. Tabelle 1) wird die Bewertung der Ansätze nach den erarbeiteten Kriterien visualisiert. Die Ansätze werden entsprechend Kapitel 4 unterteilt. Anhand der Gewichtung der Kriterien und der Bewertung wird eine Summe berechnet (vgl. Nutzwertanalyse). Umso höher die Summe ist, umso besser erfüllt der Ansatz die Kriterien. Die volle Erfüllung des Kriteriums wird mit 1, die teilweise Erfüllung mit 0,5 und die Nicht-Erfüllung mit 0 bewertet.

Aus der Abgrenzungsmatrix wird deutlich, dass Ansätze, die eine differenzierte Betrachtung vornehmen, in der Nutzwertanalyse besonders hoch im Vergleich zu den semi-differenzierten und singulären Ansätzen bewertet werden. Dem Alleinstellungsmerkmal, dass nur durch die Unterscheidung der Abhängigkeitsart Ursache-Wirkungs-Beziehungen von Anforderungsänderungen erkannt werden können und somit eine präzisere Propagationsanalyse möglich wird, stehen keine schwerwiegenden Einschränkungen entgegen. Eine Gemeinsamkeit aller betrachteten Ansätze ist, dass noch keine Validierung in einem interdisziplinären Entwicklungsprojekt erfolgt ist.

Die Vorteile der Ansätze des Maschinellen Lernens bestehen in dem hohen Automatisierungsgrad und der Robustheit. Es werden Klassifikatoren anhand von Input-Daten trainiert. Nachteil ist, dass hierbei das Expertenwissen nicht direkt einbezogen wird, was negative Auswirkungen auf die Ergebnisgüte haben kann. Andererseits können diese Ansätze bei Projekten mit unerfahrenen Anwendern durchgeführt werden. In dieser Studie wird dies jedoch negativ bewertet, da Ansätze gesucht werden, die explizit Expertenwissen einbeziehen. Außerdem ist es notwendig, dass für den jeweiligen Anwendungsfall eine ausreichende Datengrundlage vorliegt.

Ontologiebasierte Ansätze haben den Vorteil, dass Expertenwissen in die Anwendung des Lösungsansatzes direkt einbezogen wird. Dadurch wird zum einen eine Anwendbarkeit in frühen Entwicklungsphasen ohne umfassende Datenbasis erlaubt und zum anderen das Erfahrungswissen der Anwender einbezogen. Ein Nachteil dieser Ansätze besteht darin, dass eine Ontologie für den jeweiligen Anwendungsfall erstellt werden muss. Dies ist mit einem hohen Erstellungsaufwand verbunden. Insgesamt zeigt die Auswertung dennoch, dass ontologiebasierte Ansätze am besten für die Abhängigkeitsanalyse geeignet sind. Dabei wird der OpenReq Dependency Detection Ansatz am höchsten bewertet [20]. Bei diesem Ansatz werden Features eines Produkts in einer Ontologie abgebildet und mit Abhängigkeitstypen vernetzt. Der Begriff Feature ist in dem Kontext als Funktionalität einer Software zu verstehen [39]. Anschließend werden die Anforderungen automatisiert den einzelnen Features zugeordnet, sodass die Abhängigkeiten aus der Ontologie abgeleitet werden können. Darüber hinaus könnte auch eine Kombination von Ansätzen, die einzelne Abhängigkeitsarten erkennen, vielversprechend sein. Aufgrund der Heterogenität hinsichtlich Datengrundlage und Anwendungszeitpunkt bedarf dies jedoch einer kontext- und fallspezifischen Untersuchung.

Tabelle 1: Abgrenzungsmatrix

Ansätze	Gruppe	Gewicht	**K1.1** 7,5 % Experten-wissen	**K1.2** 7,5 % frühe Verfüg-barkeit	**K2.1** 17,5 % Differen-zierung	**K2.2** 2,5 % Lern-fähigkeit	**K2.3** 7,5 % Erprobtheit	**K3.1** 10 % Transfer-fähigkeit	**K3.2** 10 % Nachvoll-ziehbarkeit	**K3.3** 7,5 % Robustheit	**K4.1** 17,5 % Automatisie-rungsgrad	**K4.2** 12,5 % Datenvor-bereitung	**Summe mit Gewicht**
[20]	Diff.: Ontologiebasierte Ansätze		●	◐	●	◐	◐	●	◐	◐	●	○	70 %
[33]			●	◐	●	◐	○	●	◐	◐	●	○	66 %
[31]			●	●	●	◐	○	◐	◐	◐	●	○	65 %
[26]			●	●	●	◐	○	◐	●	◐	◐	○	61 %
[30]			●	●	●	◐	○	◐	◐	○	●	○	61 %
[25]			●	○	●	◐	◐	●	●	●	○	○	58 %
[34]	Diff.: ML-Ansätze		○	◐	●	●	○	○	○	●	●	●	61 %
[27]			○	◐	●	●	○	○	○	●	●	◐	55 %
[28]			○	◐	●	●	○	○	○	●	●	◐	55 %
[13]	Diff.: Man.		●	◐	●	○	◐	◐	●	●	○	○	55 %
[1]	S. diff		○	●	◐	○	○	●	○	○	●	●	60 %
[24]	Singulär		○	●	○	○	○	●	○	●	●	●	55 %
[37]			○	◐	○	○	○	◐	◐	●	●	●	55 %
[36]			○	●	○	◐	○	●	○	◐	●	●	53 %
[40]			◐	◐	○	○	○	●	◐	●	●	○	51 %
[41]			◐	●	○	○	○	●	●	●	○	◐	49 %
[35]			○	◐	○	○	○	●	◐	○	●	◐	46 %
[38]			○	○	○	○	○	○	◐	●	●	●	46 %
[23]			●	◐	○	○	○	●	◐	○	●	○	45 %
[22]			○	●	○	◐	○	○	○	●	●	●	45 %
[21]			○	◐	○	○	○	○	○	○	◐	◐	23 %

Legende

● Erfüllt (1) ◐ Teilweise erfüllt (0,5) ○ Nicht erfüllt (0)

ML = Maschinelles Lernen Diff. = Differenziert S. diff. = Semi-differenziert Man. = Manuell

6. Zusammenfassung und Ausblick

In diesem Beitrag werden auf der Grundlage einer Literaturstudie und zweier Industrie-Workshops mögliche Ansätze für eine automatisierte Abhängigkeitsanalyse von Anforderungen in interdisziplinären Entwicklungsprojekten evaluiert und der Forschungsbedarf identifiziert. Das Ergebnis ist, dass ontologiebasierte Ansätze aufgrund der Informationsdichte und der Möglichkeit zur Automatisierung für die automatisierte Abhängigkeitsanalyse von Anforderungen besonders geeignet sind. Damit kann die Forschungsfrage beantwortet werden, wie Abhängigkeiten in Anforderungssets interdisziplinärer Systeme für die Propagationsanalyse automatisiert identifiziert werden können. Zudem können auf Basis der Abgrenzungsmatrix (siehe Tabelle 1) bedarfsgerechte Kombinationen verschiedener Ansätze abgeleitet werden, sodass nachfolgende Forschungsaktivitäten auf dieser Grundlage den Umgang mit Anforderungsänderungen gezielt weiterentwickeln können.

Literaturverzeichnis

[1] Hein, Phyo Htet; Voris, Nathaniel; Morkos, Beshoy: Predicting requirement change propagation through investigation of physical and functional domains. In: Research in Engineering Design 29, Nr. 2, London: Springer, 2018, S. 309–328.

[2] Pohl, Klaus: Requirements engineering: Fundamentals, principles, and techniques. Berlin: Springer, 2010.

[3] Clarkson, P. John; Simons, Caroline; Eckert, Claudia: Predicting Change Propagation in Complex Design. In: Journal of Mechanical Design 126, Nr. 5, 2014, S. 788–797.

[4] Morkos, Beshoy: Computational representation and reasoning support for requirements change management in complex system design. In: All dissertations, Tiger Prints, 2012.

[5] Wickel, Martina Carolina: Änderungen besser managen. München: Technische Universität München, 2017.

[6] Jayatilleke, Shalinka; Lai, Richard: A systematic review of requirements change management. In: Information and Software Technology 93, 2018, S. 163–185.

[7] Neumann, Marc: Ein modellbasierter Ansatz zur risikoorientierten Entwicklung innovativer Produkte. Shaker Verlag GmbH, 2017.

[8] Dahlstedt, Åsa G.: Requirements Interdependencies – a Research Framework, 2011.

[9] Pohl, Klaus: Process-centered requirements engineering. New York, NY, Taunton, Somerset, England: Wiley; Research Studies Press, 1996.

[10] Karlsson, Joachim; Olsson, Stefan; Ryan, Kevin: Improved practical support for large-scale requirements prioritising. In: Requirements Engineering 2, Nr. 1, 1997, S. 51–60.

[11] P. Carlshamre; K. Sandahl; M. Lindvall; B. Regnell; J. Natt och Dag: An industrial survey of requirements interdependencies in software product release planning. In: Proceedings Fifth IEEE International Symposium on Requirements Engineering, 2001, S. 84–91.

[12] Ebert, Christof: Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten. 6., überarbeitete und erweiterte Auflage. Heidelberg: dpunkt.verlag, 2019.

[13] Köhler, C.; Conrad, J.; Wanke, S.; Weber, C.: A Matrix Representation of the CPM/PDD Approach as a Means for Change Impact Analysis. In: DS 48: Proceedings DESIGN 2008, the 10th International Design Conference, Dubrovnik, Croatia, 2008, S. 167–174.

[14] Mobasher, Bamshad; Cleland-Huang, Jane: Recommender Systems in Requirements Engineering. In: AI Magazine, 32(3), 2011, S. 81-89.

[15] Koh, Y.; Caldwell, M.; Clarkson, John: A method to assess the effects of engineering change propagation. In: Research in Engineering Design 23 (2012), Nr. 4, S. 329–351.

[16] Blessing, Lucienne T.M.; Chakrabarti, Amaresh: DRM, a Design Research Methodology. London: Springer London, 2009.

[17] WALDEN, David D. (Hrsg.); ROEDLER, Garry J. (Hrsg.); FORSBERG, Kevin (Hrsg.); HAMELIN, R. Douglas (Hrsg.); SHORTELL, Thomas M. (Hrsg.): Systems engineering handbook: A guide for system life cycle processes and activities; NJ: Wiley, 2015.

[18] Balzert, Helmut: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. 3. Auflage. Heidelberg: Spektrum Akademischer Verlag, 2009.

[19] Machi, Lawrence A.; McEvoy, Brenda T.: The literature review: Six steps to success. 2. ed. Thousand Oaks, Calif.: Corwin, 2012.

[20] Motger de la Encarnación, Joaquim; Borrull Baraut, Ricard; Palomares Bonache, Cristina; Marco Gómez, Jordi; d'Informació, Universitat Politècnica de Catalunya. Departament d'Enginyeria de Serveis i Sistemes

(Mitarb.); Computació, Universitat Politècnica de Catalunya. Departament de Ciències de la (Mitarb.); Universitat Politècnica de Catalunya. inSSIDE - integrated Software, Service, Information and Data Engineering (Mitarb.) : OpenReq-DD: A requirements dependency detection tool: CEUR-WS.org, 2019

[21] Hamdaqa, Mohammad; Hamou-Lhadj, Abdelwahab: An approach based on citation analysis to support effective handling of regulatory compliance. In: Future Generation Computer Systems 27, Nr. 4, 2011, S. 395–410.

[22] Natt och Dag, Johan; Regnell, Björn; Carlshamre, Pär; Andersson, Michael; Karlsson, Joachim: A Feasibility Study of Automated Natural Language Requirements Analysis in Market-Driven Development. In: Requirements Engineering 7, Nr. 1, 2002, S. 20–33.

[23] Xuefeng Zhu; Zhi Jin: Inconsistency measurement of software requirements specifications: an ontology-based approach. In: Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, 2005.

[24] Misra, Janardan: Terminological inconsistency analysis of natural language requirements. In: Information and Software Technology 74. 2016, S. 183–193.

[25] Soomro, Safeeullah; Hafeez, Abdul; Shaikh, Asadullah; Musavi, Syed Hyder Abbas: Ontology Based Requirement Interdependency Representation and Visualization. In: Shaikh, Faisal Karim; Chowdhry, Bhawani Shankar; Zeadally, Sherali; Hussain, Dil Muhammad Akbar; Memon, Aftab Ahmed; Uqaili, Muhammad Aslam (Hrsg.): Communication technologies, information security and sustainable development. Cham: Springer, 2014, S. 259–270.

[26] Verma, Kunal; Kass, Alex: Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents. In: Sheth, Amit; Staab, Steffen; Dean, Mike; Paolucci, Massimo; Maynard, Diana; Finin, Timothy; Thirunarayan, Krishnaprasad (Hrsg.): The Semantic Web - ISWC 2008: 7th International Semantic Web Conference. Berlin: Springer, 2008, S. 751–763.

[27] Atas, Muesluem; Samer, Ralph; Felfernig, Alexander: Automated Identification of Type-Specific Dependencies between Requirements. In: IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2018, S. 688–695.

[28] Samer, Ralph; Stettinger, Martin; Atas, Muslum; Felfernig, Alexander; Ruhe, Guenther; Deshpande, Gouri: New Approaches to the Identification of Dependencies between Requirements. In: IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), 2019, S. 1265–1270.

[29] Tiihonen, Juha; Raatikainen, Mikko; Myllyaho, Lalli; Lüders, Clara Marie; Männistö, Tomi: Coping with Inconsistent Models of Requirements. In: Proceedings of the 21st Configuration Workshop Hamburg, Germany, 2019, S. 1–8.

[30] Schraps, Mathias; Peters, Maximilian: Semantic annotation of a formal grammar by SemanticPatterns. In: Zhao, Liping (Hrsg.): IEEE 4th International Workshop on Requirements Patterns (RePa).NJ: Piscataway, 2014, S. 9–16.

[31] Farfeleder, Stefan; Moser, Thomas; Krall, Andreas; Stålhane, Tor; Omoronyia, Inah; Zojer, Herbert: Ontology-Driven Guidance for Requirements Elicitation. In: Antoniou, Grigoris; Grobelnik, Marko; Simperl, Elena; Parsia, Bijan; Plexousakis, Dimitris; Leenheer, Pieter de; Pan, Jeff (Hrsg.): The semantic web: research and applications: 8th Extended Semantic Web Conference. Berlin: Springer, 2011, S. 212–226.

[32] Borrull Baraut, Ricard: Incorporation of models in automatic requirement dependencies detection. Universitat Politècnica de Catalunya. 2019.

[33] Zichler, Konstantin; Helke, Steffen: Ontologiebasierte Abhängigkeitsanalyse im Projektlastenheft. In: Dencker, Peter; Klenk, Herbert; Keller, Hubert B.; Plödereder, Erhard (Hrsg.): Automotive - Safety & Security 2017: Sicherheit und Zuverlässigkeit für automobile Informationstechnik: Germany. Bonn, 2017.

[34] Deshpande, Gouri; Arora, Chahal; Ruhe, Guenther: Data-Driven Elicitation and Optimization of Dependencies between Requirements. In: Damian, Daniela; Perini, Anna; Lee, Seok-Won (Hrsg.): IEEE 27th International Requirements Engineering Conference: 23-27 September 2019. Piscataway, 2019, S. 416–421.

[35] Martinez, Gonzalo Garcia; Carpio, Alvaro Fernandez Del; Gomez, Luis Nunez: A Model for Detecting Conflicts and Dependencies in Non-Functional Requirements Using Scenarios and Use Cases. In: XLV Latin American Computing Conference (CLEI): IEEE, 2019.

[36] Park, S.; Kim, H.; Ko, Y.; Seo, J.: Implementation of an efficient requirements-analysis supporting system using similarity measure techniques. In: Information and Software Technology 42, Nr. 6, 2000, S. 429–438.

[37] Di Thommazo, Andre; Ribeiro, Thiago; Olivatto, Guilherme; Werneck, Vera; Fabbri, Sandra: An Automatic Approach to Detect Traceability Links Using Fuzzy Logic. In: 27th Brazilian Symposium on Software Engineering: IEEE, 2013, S. 21–30.

[38] Abadi, A.; Nisenson, M.; Simionovici, Y.: A Traceability Technique for Specifications. In: Krikhaar, René (Hrsg.): The 16th IEEE International Conference on Program Comprehension, 2008, S. 103–112.

[39] Oestereich, Bernd: Begriff und Konzept des Features. oose Innovative Informatik GmbH, 2012.

[40] Ninaus, Gerald; Felfernig, Alexander; Stettinger, Martin; Reiterer, Stefan: Intelligent Techniques for Software Requirements Engineering, In: Frontiers in Artificial Intelligence and Applications. Bd. 263, 2014.

[41] Lienhard, A.; Greevy, O.; Nierstrasz, O.: Tracking Objects to Detect Feature Dependencies. In: Wong, Kenny (Hrsg.): 15th IEEE International Conference on Program Comprehension, NJ, 2007, S. 59–68.