

Effiziente Erfassung und Pflege von Traceability-Modellen zur Entwicklung technischer Systeme

vorgelegt von
Asmus Figge
Dipl.-Ing.
aus Kiel

von der Fakultät V – Verkehrs- und Maschinensysteme
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr.-Ing. Roland Jochem
1. Gutachter: Prof. Dr.-Ing. Rainer Stark
2. Gutachter: Prof. Dr.-Ing. habil. Ralph Stelzer

Tag der wissenschaftlichen Aussprache: 4. Februar 2014

Berlin 2014

D 83

GELEITWORT DER HERAUSGEBER

Die Schriftenreihe „Berichte aus dem Produktionstechnischen Zentrum Berlin“ wird von den Professoren der im Produktionstechnischen Zentrum Berlin dauerhaft angelegten Fach- und Forschungsgebiete der TU Berlin gemeinsam herausgegeben. Zweck der Schriftenreihe ist es, die auf den Gebieten der Produktentstehung, Produktionstechnik und Informationstechnik erarbeiteten Forschungsergebnisse einer breiten Fachöffentlichkeit zugänglich zu machen. In der Schriftenreihe erscheinen in erster Linie die an den Fachgebieten des Instituts für Werkzeugmaschinen und Fabrikbetrieb (IWF) der TU Berlin und am Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik entstandenen Dissertationen. Daneben werden aber auch andere Forschungsberichte, die in den thematischen Rahmen passen und vom allgemeinen Interesse sind, in die Schriftenreihe aufgenommen. Die Herausgeber wünschen sich ein reges Interesse an der Schriftenreihe und würden sich freuen, wenn hieraus fruchtbare Dialoge mit Praktikern und Forschern entstünden.

Die vorliegende Dissertationsarbeit von Herrn Dr. Figge beschäftigt sich mit der Erforschung und der Evaluierung der nächsten Generation von Methoden und Werkzeugen des „*Traceability Engineering*“ (*Effiziente Erfassung und Pflege von Traceability-Modellen zur Entwicklung technischer Systeme*), einer grundlegenden Arbeitsdisziplin des Systems Engineering und somit Wegbereiter der modernen disziplinenübergreifenden Produktentwicklung. In der vorliegenden Arbeit werden drei neuartige Methoden zur Erfassung und Pflege von Traceability-Informationen entwickelt und evaluiert. Ziel der Arbeit ist es, das Traceability Engineering im Sinne der expliziten Aufnahme von Verlinkungen zwischen digitalen Modellen als moderne und durchgängig nutzbare Entwicklungsmethode zu etablieren, so dass dem derzeit misslichen Umstand, dass nur die Branchen konsistent Traceability Dokumentationsmethoden nutzen, die hierzu gesetzlich verpflichtet sind, gekonnt entgegen getreten werden kann.

Der Verfasser der vorliegenden Dissertationschrift hat eindeutig bewiesen, dass er mit wissenschaftlich fundierten Vorgehensweisen bestens vertraut ist und diese gewinnbringend zur Erforschung von neuen Konzepten und Werkzeugen der Traceability-basierten Entwicklung von technischen Systemen einsetzen kann. Insbesondere hervorzuheben ist die gewählte wissenschaftliche Vorgehensweise, welche sich ausgehend von dem Erkennen der Probleme in der Praxis, über daraus abgeleitete konkrete Anforderungen an verbesserte Modellierungsumgebungen, dem Stand der Technik sowie der sich anschließenden Hypothesenbildung, der gezielten Entwicklung von neuen Methoden für die durchgängige Verknüpfung, der prototypischen Implementierung dieser Methoden in Softwaresysteme bis hin zur Evaluierung im Rahmen von mehreren wissenschaftlichen Nutzerstudien erstreckt.

ABSTRACT

In the past, the development process for technical products primarily focused on the detailed development of mainly mechanical working principles. Today, research and development challenges have shifted towards the orchestration of different technical disciplines. In this complex context it is often impossible for developers to fully apprehend all dependencies of the systems under consideration.

Traceability is a method that allows for the explicit documentation of these dependencies, for instance in order to assess the impact of changes across disciplines. However, traceability is usually only applied when required by law, as it requires considerable effort for recording and maintenance. Hence, this thesis addresses three research questions aimed at supporting the use of traceability.

To assist in the efficient modeling of traceability-models, the method EcoTracing is designed. To reduce the modeling effort, EcoTracing utilizes the hierarchical structure of the models describing the systems under development. Its performance was evaluated in a user study, showing that arithmetically averaged savings of about 60 % are possible.

Traceability-models have to be evaluated and updated continuously to maintain their quality. In this thesis, TraceEvaluation is proposed as a new approach to support this quality management. It applies crowdsourcing methods to enable developers to easily document mistakes in traceability-models that they identify during their daily business. In doing so, the feedback of many users can be aggregated rapidly and analyzed centrally. The effectiveness of TraceEvaluation was verified in a user study in which all planted mistakes were precisely identified.

In addition to the recognition of mistakes it is important to identify dependencies missing from the traceability-model. To address this issue, TraceLegacy is presented in this thesis. It makes use of development knowledge of legacy projects to identify dependencies in current projects. The viability of TraceLegacy was proved in an evaluation with an example in which several dependencies could be detected.

The contribution of this thesis is the development and evaluation of three innovative methods for the recording and maintenance of traceability-models. It significantly contributes to the state of traceability-research, especially as the effectiveness of the methods was proved using realistic examples of technical systems as well as in user studies.

ZUSAMMENFASSUNG

Lag früher der Fokus bei der Entwicklung technischer Produkte auf der Detaillierung meist mechanischer Wirkprinzipien, finden sich die Herausforderungen heute eher im Bereich der Orchestrierung unterschiedlicher Entwicklungsdisziplinen. In diesem komplexen Kontext ist es Entwicklern vielfach unmöglich, alle Abhängigkeiten der zu entwickelnden Systeme vollständig zu erfassen.

Traceability ist eine Methode, um diese Abhängigkeiten explizit zu dokumentieren und so bspw. die Auswirkungen von Änderungen über Disziplinengrenzen hinweg abschätzbar zu machen. Jedoch beschränkt sich die Anwendung der Traceability hauptsächlich auf Bereiche, in denen sie explizit durch den Gesetzgeber gefordert wird. Gründe hierfür sind u. a. der große Aufwand für die Erfassung und Pflege der Traceability. Der Bedarf nach einer Reduktion dieses Aufwands stellt den Ausgangspunkt für die drei in dieser Arbeit behandelten Forschungsfragen dar.

Dazu wird zunächst die effiziente Modellierung von Traceability-Modellen betrachtet und die Methode EcoTracing konzipiert. EcoTracing nutzt die hierarchische Struktur systembeschreibender Modelle, um Aufwände bei der Modellierung zu reduzieren. Bei der Evaluation der Methode im Rahmen einer Studie konnte im arithmetischen Mittel eine Aufwandsersparnis von ca. 60 % erreicht werden.

Einmal erstellt, müssen die Traceability-Modelle kontinuierlich gepflegt werden, um ihre Qualität dauerhaft zu erhalten. Als neuen Ansatz, dies zu unterstützen, wird die auf Crowdsourcing basierende Methode TraceEvaluation entwickelt. Sie ermöglicht es Entwicklern, Fehler in Traceability-Modellen, die sie während der Bearbeitung ihrer Aufgaben im Tagesgeschäft automatisch entdecken, einfach zu dokumentieren. Durch dieses Verfahren gelingt es, das Feedback vieler Nutzer schnell zu aggregieren und zentral auszuwerten. In einer Studie konnten mit TraceEvaluation alle Fehler präzise identifiziert und damit die Wirksamkeit der Methode nachgewiesen werden. Um neben falschen auch fehlende Tracelinks identifizieren zu können, wird zusätzlich die Methode TraceLegacy in dieser Arbeit beschrieben. Sie nutzt das Entwicklungswissen vergangener Projekte, um Abhängigkeiten in aktuellen Projekten zu identifizieren. Im Rahmen der Evaluation konnten so zahlreiche fehlende Tracelinks erkannt und damit die Funktionsfähigkeit der Methode nachgewiesen werden.

Der Forschungsbeitrag dieser Arbeit liegt in der Entwicklung und Evaluation dreier neuartiger Methoden zur Erfassung und Pflege von Traceability-Modellen. Die Methoden stellen einen signifikanten Beitrag zum Stand der Wissenschaft im Bereich der Traceability-Forschung dar, zumal ihre jeweilige Wirksamkeit durch Nutzerstudien sowie anhand vergleichender Beispiele belegt werden konnte.

DANKSAGUNG

Die Erstellung dieser Dissertation war mit einem eingangs nicht für möglich gehaltenen Aufwand verbunden, dessen Bewältigung mir ohne die Unterstützung zahlreicher Personen in meinem privaten und beruflichen Umfeld nicht möglich gewesen wäre.

Zunächst möchte ich meinem Doktorvater Prof. Dr.-Ing. Rainer Stark für die Betreuung dieser Arbeit danken. Seine berechtigten Fragen und geäußerte Kritik haben das vorliegende Ergebnis wesentlich beeinflusst. Prof. Dr.-Ing. habil. Ralph Stelzer danke ich für die Begutachtung der Arbeit und Prof. Dr.-Ing. Roland Jochem für die Übernahme des Promotionsausschuss-Vorsitzes.

Grischa Beier, der seine Dissertation zeitgleich erstellt hat, danke ich für fünf Jahre außerordentliche Zusammenarbeit. Die täglichen Diskussionen über Traceability-Grundlagen, den Stand der Technik, entwickelte Methoden, deren Evaluation und der gemeinsame Konsum hunderter Liter Kaffee haben diese Arbeit erst möglich gemacht. Michel Bornath und Robert Müller, die als studentische Mitarbeiter die prototypische Implementierung meiner Methoden durchgeführt haben, danke ich für ihre hervorragende Arbeit, ihr stetiges Mitdenken und die Tatsache, dass sie meine maschinenbäuerlichen Fragen ertragen haben. Johann Habakuk Israel und Elisabeth Brandenburg danke ich für die Unterstützung bei der Konzeption der durchgeführten Studien und Simon Königs für die stundenlangen Diskussionen über die Grundlagen der Traceability. Konrad Wernicke und Tobias Lange danke ich für die Korrektur meiner Dissertation und entschuldige mich nachträglich für deren Umfang.

Allen Kollegen im Geschäftsfeld Virtuelle Produktentstehung des Fraunhofer IPK und im Fachgebiet Industrielle Informationstechnik der TU Berlin danke ich für eine ausgesprochen angenehme Arbeitsatmosphäre, den freundschaftlichen Umgang im Arbeitsalltag und zahlreiche Gespräche insbesondere abseits der Virtuellen Produktentstehung.

Ein besonderer Dank gilt meinen Eltern Hanna und Norbert. Ohne ihre große Unterstützung in jeder Hinsicht wären weder Abitur noch Studium möglich gewesen, was wahrscheinlich bei Begutachtung, spätestens jedoch bei Einreichung der Arbeit aufgefallen wäre. Auch Opa Siegfried, Jonas, Kadl, meinen Schwiegereltern Hanne und Werner sowie Dirk, Esther, Jan und Christin Schönzart danke ich für ihre Unterstützung.

Mein größter Dank gilt jedoch meiner kleinen Familie. Lena, vielen Dank, dass du immer für mich da warst und unsere Wochenendbeschäftigung der letzten anderthalb Jahre ertragen hast. Ohne dich wäre diese Arbeit nicht möglich gewesen. Lotta, vielen Dank, dass du die stressige letzte Phase der Promotion entschleunigt und durch jedes Lächeln so viel schöner gemacht hast.

INHALTSVERZEICHNIS

INHALTSVERZEICHNIS	IX
ABKÜRZUNGSVERZEICHNIS.....	XV
ABBILDUNGSVERZEICHNIS.....	XVII
TABELLENVERZEICHNIS.....	XXIII
1 EINLEITUNG UND ZIELSETZUNG	1
1.1 Ausgangssituation und Zielsetzung der Arbeit	1
1.2 Forschungsfragen	3
1.3 Vorgehen und Aufbau der Arbeit.....	4
2 ENTWICKLUNG MECHATRONISCHER SYSTEME	7
2.1 Vorgehensmodelle zur Entwicklung mechatronischer Systeme	7
2.1.1 VDI 2206 – Entwicklungsmethodik für mechatronische Systeme	8
2.1.2 Systems Engineering	10
2.1.3 ISO 26262 – Road vehicles – Functional Safety	13
2.1.4 Industrielle Vorgehensweise am Beispiel der Automobilindustrie	15
2.1.5 V-Modell des Fraunhofer IPK	18
2.1.6 Vergleich der Vorgehensmodelle	21
2.2 Artefakte zur Entwicklung mechatronischer Systeme	23
2.2.1 Anforderungsartefakte	23
2.2.2 Funktionsartefakte	24
2.2.3 Verhaltensartefakte.....	26
2.2.4 Produktstrukturartefakte	28
2.2.5 Abhängigkeiten zwischen Artefakten	28
2.3 Zusammenfassung und Diskussion	30
3 DURCHGÄNGIGE NACHVERFOLGBARKEIT	31
3.1 Bedarf nach durchgängiger Nachverfolgbarkeit	32
3.2 Ursprung und Verbreitung durchgängiger Nachverfolgbarkeit	35
3.3 Grundbegriffe durchgängiger Nachverfolgbarkeit	36
3.4 Phasen der durchgängigen Nachverfolgbarkeit	42

3.4.1	Erfassung von Tracelinks	42
3.4.2	Verwendung von Tracelinks	44
3.4.3	Pflege von Tracelink-Modellen.....	44
3.4.4	Planung durchgängiger Nachverfolgbarkeit	45
3.5	Einordnung der durchgängigen Nachverfolgbarkeit in den strategischen Unternehmenskontext	48
3.6	Software zur Erfassung, Pflege und Verwendung von Tracelinks.....	50
3.6.1	Anforderungsmanagement-Software.....	50
3.6.2	PLM-Systeme	54
3.6.3	Analyse-Software	57
3.6.4	Integrationssoftware	61
3.7	Herausforderungen der durchgängigen Nachverfolgbarkeit	68
3.7.1	Allgemeine Darstellung der Herausforderungen.....	69
3.7.2	Darstellung der Herausforderungen am Beispiel der Entwicklung des mechatronischen Außenspiegels.....	71
3.8	Zusammenfassung und Diskussion	74
4	FORSCHUNGSFRAGEN UND HYPOTHESEN	77
4.1	Methodische Unterstützung zur effizienten Modellierung von Tracelinks.....	78
4.2	Methodische Unterstützung zur verteilten Qualitätssicherung	79
4.3	Verwendung des Wissens vergangener Projekte zur Qualitätssicherung.....	80
5	METHODEN ZUR EFFIZIENTEN MODELLIERUNG VON TRACELINKS.....	81
5.1	Stand der Technik und Forschung	82
5.2	Grundlagen zur Konzeption und Bewertung der Methoden.....	84
5.2.1	Strukturanalyse der Artefakte der Entwicklung mechatronischer Systeme	85
5.2.2	Granularität im Kontext der Traceability	100
5.2.3	Ermittlung des Aufwands bei der manuellen Erfassung von Tracelinks	105
5.2.4	Zusammenfassung und Schlussfolgerungen	107
5.3	Evaluation der Hypothese 1* im Kontext der Systementwicklung	108
5.3.1	Aufbau und Durchführung der Studie.....	108

5.3.2	Auswertung und Diskussion der Ergebnisse	113
5.4	EcoTracing – Methode zur effizienten Modellierung von Tracelinks	118
5.4.1	Funktionsprinzip der Methode EcoTracing	118
5.4.2	Algorithmus der Methode EcoTracing.....	119
5.4.3	Prototypische Implementierung der Methode EcoTracing.....	123
5.4.4	Ermittlung des Aufwands bei der Erfassung von Tracelinks mit EcoTracing	126
5.4.5	Evaluation der Methode EcoTracing.....	128
5.5	Zusammenfassung und Diskussion	137
6	METHODEN ZUR PFLEGE VON TRACELINK-MODELLEN	141
6.1	Stand der Technik und Forschung.....	142
6.1.1	Regelbasierte Methoden	143
6.1.2	Linguistische und IR-Methoden.....	145
6.1.3	Zusammenfassung und Diskussion.....	147
6.2	Grundlagen zur Konzeption und Bewertung der Methoden	149
6.2.1	Crowdsourcing.....	149
6.2.2	Trefferquote und Präzision	152
6.3	TraceEvaluation – Methode zur verteilten Bewertung von Tracelinks	154
6.3.1	Funktionsprinzip der Methode TraceEvaluation	154
6.3.2	Prototypische Implementierung der Methode TraceEvaluation	157
6.3.3	Evaluation der Methode TraceEvaluation	160
6.3.4	Zusammenfassung und Diskussion.....	165
6.4	TraceLegacy – Methode zur Identifikation fehlender Tracelinks	166
6.4.1	Funktionsprinzip der Methode TraceLegacy	167
6.4.2	Algorithmus der Methode TraceLegacy	168
6.4.3	Prototypische Implementierung der Methode TraceLegacy	171
6.4.4	Evaluation der Methode TraceLegacy	173
6.4.5	Zusammenfassung und Diskussion.....	179
6.5	Zusammenfassung und Diskussion	181
7	ANWENDUNG DER METHODEN ZUR ENTWICKLUNG EINES TECHNISCHEN SYSTEMS	183
7.1	Verortung der Methoden im V-Modell.....	184

7.2	Demonstration der Methoden am Beispiel der Entwicklung eines mechatronischen Außenspiegels	185
7.2.1	Anforderungen	187
7.2.2	Anforderungskaskade	187
7.2.3	Vorentwurf der Funktionen und Eigenschaften	189
7.2.4	Erfassung von Tracelinks zwischen Anforderungen und Funktionen..	190
7.2.5	Vorentwurf des Systemmodells	193
7.2.6	Erfassung von Tracelinks zwischen Funktionen und Verhaltensmodell	194
7.2.7	Integrierte Absicherung.....	195
7.2.8	Entwicklung mechanischer Komponenten	196
7.2.9	Erfassung von Tracelinks zwischen Produktstruktur und Verhaltensmodell sowie Anforderungen	196
7.2.10	Durchführung einer Auswirkungsanalyse aufgrund einer geänderten Anforderung	198
7.2.11	Durchführung einer Qualitätskontrolle des Tracelink-Modells	199
7.3	Zusammenfassung	201
8	ZUSAMMENFASSUNG UND AUSBLICK	203
8.1	Zusammenfassung der Ergebnisse dieser Arbeit	204
8.1.1	EcoTracing	204
8.1.2	TraceEvaluation.....	205
8.1.3	TraceLegacy.....	206
8.1.4	Fazit	207
8.2	Einsatzmöglichkeiten der Methoden und deren Auswirkungen auf die IT-Architektur	208
8.3	Ausblick	210
	LITERATURVERZEICHNIS	213
	GLOSSAR	223
	ANHANG	231
A.	Material zur Studie „Evaluation der Hypothese 1* im Kontext der Systementwicklung“	232

B.	Material zur Studie „Evaluation der Methoden EcoTracing und TraceEvaluation“	244
C.	Material zur Evaluation der Methode TraceLegacy.....	254

ABKÜRZUNGSVERZEICHNIS

ASIL	Automotive Safety Integrity Level
BMBF	Bundesministerium für Bildung und Forschung
CAD	Computer Aided Design
COTS	Commercial off-the-shelf: Standardsoftware
DIN	Deutsches Institut für Normung
DMM	Domain Mapping Matrix
DMU	Digital Mock-Up
DRM	Design Research Methodology
DSM	Design Structure Matrix
E/E	Elektrik/Elektronik
FEM	Finite Elemente Methode
FMEA	Fehlermöglichkeits- und Einfluss-Analyse (engl.: Failure Mode and Effects Analysis)
GUI	Graphical User Interface: grafische Benutzeroberfläche
HiL	Hardware in the Loop
ID	Identifikationsnummer
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering
IR	Information Retrieval: Informationsrückgewinnung
ISO	International Organization for Standardization
IT	Informationstechnik
LED	Leuchtdiode
MBE	Modellbasierte Entwicklung
MDM	Multiple-Domain Matrix
MKS	Mehrkörpersimulation
OSLC	Open Services for Lifecycle Collaboration
PDM	Produktdatenmanagement
PSE	Produktstruktur-Editor
QFD	Quality-Function-Deployment

ReqIF	Requirements Interchange Format
ROI	Return on Investment
SE	Systems Engineering
SiL	Software in the Loop
SysML	Systems Modeling Language
UID	Unique ID: Eindeutige Identifikationsnummer
VDI	Verein Deutscher Ingenieure
XML	Extensible Markup Language: erweiterbare Auszeichnungssprache

ABBILDUNGSVERZEICHNIS

Abbildung 1:	Vorgehen zur Bearbeitung der Forschungsfragen	4
Abbildung 2:	Aufbau der Arbeit (Kapitelnummern in Kreisen)	5
Abbildung 3:	V-Modell als Makrozyklus [VDI-Richtlinie 2206]	9
Abbildung 4:	Systems Engineering und die darin beinhalteten Methoden und Vorgehensmodelle [NASA STI 2007, S. 4]	11
Abbildung 5:	Der Systems Engineering Prozess nach [Department of Defense 2001, S. 31]	12
Abbildung 6:	V-Modell nach ISO 26262 [ISO 26262-2]	14
Abbildung 7:	Matrixorganisation in der Automobilindustrie [Königs et al. 2012, S. 925]	16
Abbildung 8:	Übergreifende Vorgehensweise in der Automobilindustrie [Weber 2009, S. 12]	17
Abbildung 9:	V-Modell des Fraunhofer IPK	19
Abbildung 10:	Vergleich der Vorgehensmodelle	21
Abbildung 11:	Simulationsumgebungen auf Basis von Modelica [Otter 2011, S. 6] ...	27
Abbildung 12:	Darstellung der LEDs des Toter-Winkel-Assistenten im Geometriemodell des Außenspiegels (vgl. [Amin et al. 2012])	29
Abbildung 13:	Schematische Darstellung des Objekts „Tracelink“	38
Abbildung 14:	Quantitative Traceability am Beispiel des mechatronischen Außenspiegels	40
Abbildung 15:	Darstellungsform von Tracelinks in DOORS im Textmodus	51
Abbildung 16:	Theoretisches Traceability-Schema in Teamcenter [Siemens PLM Software 2011]	56
Abbildung 17:	Die METUS-Methode für das Systemdesign [ID-Systems GmbH 2013]	58
Abbildung 18:	Matizen als Kernelement der Software LOOME0 [Teseon GmbH 2012]	59
Abbildung 19:	Tracelinks in ToolNet (in Anlehnung an [Rosenauer 2008, S. 100])	62
Abbildung 20:	Darstellung der Suchergebnisse von ConWeaver (strukturiert nach Ergebnislisten) [Abbildung mit freundlicher Genehmigung zur Publikation von der ConWeaver GmbH (www.conweaver.de) bereitgestellt]	63
Abbildung 21:	Anlegen eines Tracelinks im ModelTracer	65
Abbildung 22:	Datenmodell des ModelTracers	66
Abbildung 23:	Ariadne's Eye zur Visualisierung der Artefakte und Tracelinks	67

Abbildung 24: Methode "Distributed Modeling of Component DSM" zur Verringerung des Aufwands bei der Modellierung von Tracelinks (in Anlehnung an [Tilstra et al. 2009, S. 245])	83
Abbildung 25: Beispiel einer Inklusionshierarchie [Grobstein 1973, S. 32]	87
Abbildung 26: Kategorisierung der Tierwelt mit Hilfe einer subsumtiven Inklusionshierarchie	87
Abbildung 27: Ableitung von Tracelinks durch Ausnutzung der Transitivität	88
Abbildung 28: Schematische Darstellung der Zusammensetzung von Eigenschaften (visualisiert durch farbige Quadrate) in kompositionellen (links) und subsumtiven Inklusionshierarchien (rechts)	90
Abbildung 29: Strukturierung des Anforderungsartefakts mit Hilfe der Konkretisierung	92
Abbildung 30: Strukturierung des Anforderungsartefakts mit Hilfe der Partitionierung	92
Abbildung 31: Strukturierung des Anforderungsartefakts mit Hilfe der Projektion	93
Abbildung 32: Strukturierung des Funktionsartefakts mit Hilfe einer Hierarchie	94
Abbildung 33: Dreidimensionale Blockdarstellung einer Funktionsstruktur [Pahl et al. 2007, S. 45]	95
Abbildung 34: Strukturierung des Funktionsartefakts mit Hilfe einer Funktionsstruktur	95
Abbildung 35: Strukturierung eines Verhaltensartefakts.....	96
Abbildung 36: Ausschnitt einer beispielhaften Strukturierung eines Außenspiegels a) unter Berücksichtigung von Montagegesichtspunkten und b) entsprechend der Funktionen.....	98
Abbildung 37: Schematischer Aufbau einer Produktstruktur.....	98
Abbildung 38: Aufbau von Artefakten im Systems Engineering und Einordnung der Begriffe "Granularität", "Detaillierungsgrad" und "hierarchische Ebene"	100
Abbildung 39: Granularitätsebenen bei der Modellierung von Tracelinks.....	101
Abbildung 40: Auszug aus dem Anforderungs- und Funktionsartefakt des mechatronischen Außenspiegels	102
Abbildung 41: Folgen der Reduktion der Granularität auf die Auswertung von Tracelinks: a) durch eine niedrige Granularität werden viele Tracelinks zwischen Elementen vererbt, bei denen keine Abhängigkeit besteht. b) durch die Wahl einer höheren Granularität werden nur tatsächlich Abhängige Elemente verknüpft.	104
Abbildung 42: Vorgehensweise zur Erfassung von Abhängigkeiten am Beispiel einer DSM (in Anlehnung an [Maurer 2007, S. 98])	105

Abbildung 43: Abstraktes Beispiel zur Ermittlung des Aufwands.....	106
Abbildung 44: Bei der Auswertung wird überprüft, ob auf höchster hierarchischer Ebene ein Tracelink vorhanden ist (dargestellt durch die graue gestrichelte Linie), wenn deren Kinderelemente einen Tracelink aufweisen (dargestellt durch die rote durchgängige Linie). Ist dem nicht so, wird ein Widerspruch registriert.....	109
Abbildung 45: Beispieldarstellung einer in der Studie verwendeten Abhängigkeitsmatrix	110
Abbildung 46: Schematische Darstellung des Kaffeevollautomaten (links: Übersicht, rechts: Darstellung der Baugruppe Auffangeinheit mit beschrifteten Einzelteilen)	111
Abbildung 47: Schematische Darstellung des Fahrrads (links: Übersicht, rechts: Darstellung der Antriebssystems mit beschrifteten Einzelteilen)	111
Abbildung 48: Korrelation zwischen der Selbsteinschätzung und der Anzahl der Widersprüche	115
Abbildung 49: Funktionsprinzip der Methode EcoTracing.....	119
Abbildung 50: Untersuchung der Artefakte auf Abhängigkeiten zwischen den Elementen	119
Abbildung 51: Markierung von Elementkombinationen bei nicht vorhandener Abhängigkeit	120
Abbildung 52: Modellierung von Tracelinks und Markierung der Kinder-Elemente für eine spätere Untersuchung	121
Abbildung 53: Algorithmus der EcoTracing Methode (zunächst Erfassung der Tracelinks auf hoher hierarchischer Ebene und folgend schrittweise Erhöhung der Granularität).....	122
Abbildung 54: Vergleich der Online- mit der Offline-Erfassung von Tracelinks: kontinuierliche, jedoch unvollständige Online-Analyse gegenüber einer vollständigen Offline-Analyse zu bestimmten Zeitpunkten im Entwicklungsprozess (hier dargestellt: zum Ende der jeweiligen Phase)	123
Abbildung 55: Graphical User Interface (GUI) des EcoTracers	125
Abbildung 56: Notwendige Parameter zur Bestimmung des Aufwands unter Verwendung von EcoTracing.....	127
Abbildung 57: Beispiel zur Erläuterung der Ermittlung der Anzahl der Entscheidungen unter Verwendung von EcoTracing (Elementkombinationen, bei denen eine Entscheidung getroffen werden muss, sind schraffiert dargestellt)	128
Abbildung 58: Übersicht der Artefakte des Fahrradbeispiels.....	129

Abbildung 59: Häufigkeit in der unterschiedliche Anzahlen von Probanden übereinstimmen und somit einen bzw. keinen Tracelink modelliert haben	136
Abbildung 60: Ersparte Entscheidungen durch EcoTracing im Vergleich zur manuellen Methode (in %)	136
Abbildung 61: Regeln zur Ableitung von Abhängigkeiten nach [Maurer 2007, S. 95].....	144
Abbildung 62: Verwendung eines Glossars, um wichtige Begriffe in textuellen Beschreibungen identifizieren zu können [Cerbah und Euzenat 2001, S. 33].....	146
Abbildung 63: Typisches Crowdsourcing System [Geiger et al. 2011, S. 2]	150
Abbildung 64: Taxonomie zur Beschreibung der Eigenschaften von Crowdsourcing-Systemen [Geiger et al. 2011, S. 6]	151
Abbildung 65: Ablauf der Methode TraceEvaluation	155
Abbildung 66: TraceEvaluation als Crowdsourcing-System in Anlehnung an [Geiger et al. 2011, S. 2]	157
Abbildung 67: Melden eines falschen Tracelinks in Ariadne's Eye.....	158
Abbildung 68: Auswertungsmodul zur Analyse der gemeldeten falschen Tracelinks	159
Abbildung 69: Algorithmus zur Zusammenstellung angezweifelter Tracelinks	160
Abbildung 70: Funktionsprinzip der Methode TraceLegacy.....	168
Abbildung 71: Schematische Darstellung der Verwaltung der Ähnlichkeitswerte in Form von Matrizen	170
Abbildung 72: Algorithmus der Abhängigkeitsanalyse mit TraceLegacy	171
Abbildung 73: GUI des Prototyps der Methode TraceLegacy	172
Abbildung 74: Verwaltung der beiden Projekte im ModelTracer (Artefakte des Vorgängerprojekts „Fahrrad“ wurden mit dem Präfix „_“ versehen).	175
Abbildung 75: Ergebnisliste der prototypischen Implementierung der Methode TraceLegacy anhand der Beispiele „Fahrrad“ und „Pedelec“	176
Abbildung 76: Verortung der Methoden im V-Modell des Fraunhofer IPK.....	184
Abbildung 77: Darstellung der durchgeführten Prozessschritte während der Entwicklung des mechatronischen Außenspiegels.....	186
Abbildung 78: Darstellung eines Ausschnitts der Anforderungshierarchie (links) sowie Eigenschaften der Anforderung „Maximum Angle“ (rechts) in CATIA V6 (vgl. [Amin et al. 2012])	188
Abbildung 79: Funktionsstruktur des mechatronischen Außenspiegels [Amin et al. 2012]	189

Abbildung 80: Anlegen einer Implementierungsbeziehung zwischen einer Anforderung und einer Funktion (vgl. [Amin et al. 2012]).....	191
Abbildung 81: Anwendung der Methode EcoTracing zur Abhängigkeitsanalyse zwischen Anforderungen und Funktionen des Außenspiegels	192
Abbildung 82: Modelica-Modell zur Simulation des Verhaltens der Spiegelglasbeheizung (vgl. [Amin et al. 2012])	194
Abbildung 83: Impact Analyse in Ariadne's Eye & Anwendung der Methode TraceEvaluation	199
Abbildung 84: Auswertungsmodul der Methode TraceEvaluation.....	199
Abbildung 85: Ergebnis der Analyse mittels TraceLegacy	200
Abbildung 86: Übersicht der erstellten Artefakte, modellierter Tracelinks (rote Pfeile) sowie der erreichten Ersparnis durch EcoTracing (wenn angewendet)	201
Abbildung 87: Einordnung der Methoden EcoTracing, TraceEvaluation und TraceLegacy in einem beispielhaften vierstufigen IT-Architekturkonzept (in Anlehnung an [Eigner und Stelzer 2009, S. 43])	209

TABELLENVERZEICHNIS

Tabelle 1:	Traceability-Eigenschaften von DOORS (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)	52
Tabelle 2:	Traceability-Eigenschaften von Reqtify (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)	54
Tabelle 3:	Traceability-Eigenschaften der V6-Suite (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)	55
Tabelle 4:	Traceability-Eigenschaften von Teamcenter (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)	57
Tabelle 5:	Traceability-Eigenschaften von METUS (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)	59
Tabelle 6:	Traceability-Eigenschaften von LOOME0 (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)	60
Tabelle 7:	Traceability-Eigenschaften von ToolNet (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)	62
Tabelle 8:	Traceability-Eigenschaften von ConWeaver (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)	64
Tabelle 9:	Traceability-Eigenschaften des ModelTracers (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)	68
Tabelle 10:	Traceability-Eigenschaften der betrachteten Software-Tools (D – DOORS, R – Reqtify, V – V6-Suite, T – Teamcenter, M – Metus, L – Loomo. N – ToolNet, C – ConWeaver, MT – ModelTracer).....	76
Tabelle 11:	Zuordnung von verwendeten Artefakten zur Art der Hierarchie	99
Tabelle 12:	Eigenschaften der Studie zur Überprüfung der Hypothese 1*	113
Tabelle 13:	Ergebnis der Studie zur Überprüfung der Hypothese 1* (Erläuterung, was als Widerspruch gezählt wird, in Abbildung 44) ...	114
Tabelle 14:	Kumulierte Widersprüche des Fahrrad-Beispiels.....	116
Tabelle 15:	Kumulierte Widersprüche des Kaffeevollautomaten-Beispiels	117
Tabelle 16:	Eigenschaften der Studie zur Überprüfung der Hypothese 1	131

Tabelle 17:	Darstellung der durch Proband 1 modellierten Tracelinks für die Artefaktkombination (Anforderungen «» Funktionen) 131
Tabelle 18:	Studienergebnisse für die Anzahl der Entscheidungen und die Ersparnis an Entscheidungen durch Anwendung von EcoTracing pro Proband zwischen Anforderungen (A), Funktionen (F) und Produktelementen (P) 132
Tabelle 19:	Aggregierte Anzahl modellierter Tracelinks zwischen Anforderungen und Funktionen (fette Zahlen markieren eine Zelle in der ein Tracelink durch den Autor modelliert wurde)..... 134
Tabelle 20:	Kategorisierung der Methoden des Stands der Technik 147
Tabelle 21:	Bewertung der Größen Präzision (Precision) und Trefferquote (Recall) [Hayes et al. 2006, S. 11] 153
Tabelle 22:	Elementepaare, zwischen denen im Vorfeld der Studie falsche Tracelinks modelliert wurden 161
Tabelle 23:	Eigenschaften der Studie zur Überprüfung der Hypothese 2 162
Tabelle 24:	Meldung von Tracelinks pro Proband 163
Tabelle 25:	Übersicht der Elementepaare, die von den Probanden als falsch gemeldet wurden 164
Tabelle 26:	Charakterisierung des Vorgängerprojekts Fahrrad 173
Tabelle 27:	Charakterisierung des aktuellen Projekts Pedelec 174
Tabelle 28:	Evaluationsergebnis ohne Stammformrückführung (Auszug) 176
Tabelle 29:	Evaluationsergebnis mit Stammformrückführung (Auszug) 178
Tabelle 30:	Eigenschaften der initialen Anforderungsspezifikation 187
Tabelle 31:	Eigenschaften der Anforderungsspezifikation des mechatronischen Außenspiegels 188
Tabelle 32:	Eigenschaften der Funktionsstruktur des mechatronischen Außenspiegels 190
Tabelle 33:	Eigenschaften des Verhaltensmodells des mechatronischen Außenspiegels 194
Tabelle 34:	Eigenschaften der Produktstruktur des mechatronischen Außenspiegels 196

1 EINLEITUNG UND ZIELSETZUNG

Die im Titel erwähnte *Traceability* ist ein aus der Informatik stammender Ansatz, dessen Anwendung im Kontext der Systementwicklung eine Antwort auf die Herausforderungen heterogener Entwicklungslandschaften bietet. Die vorliegende Arbeit widmet sich diesem Thema aus Sicht der *Entwicklung von Methoden zur effizienten Modellierung und Pflege von Tracelinks*, die für eine stärkere industrielle Verbreitung der durchgängigen Nachverfolgbarkeit unabdingbar ist. In Kapitel 1.1 wird zunächst die Ausgangssituation und Zielsetzung der Forschungen auf diesem Gebiet erläutert, bevor anschließend in Kapitel 1.2 die Formulierung der Forschungsfragen erfolgt. Die für die Beantwortung dieser Fragen gewählte Vorgehensweise sowie die Verortung der Ergebnisse in der Struktur der vorliegenden Arbeit wird abschließend in Kapitel 1.3 erläutert.

1.1 AUSGANGSSITUATION UND ZIELSETZUNG DER ARBEIT

Die Entwicklung von Produkten hat sich in den vergangenen Jahrzehnten stark verändert. Lag früher der Fokus auf der Detaillierung meist mechanischer Wirkprinzipien, finden sich die Herausforderungen heute im Bereich der Orchestrierung der unterschiedlichen Entwicklungsdisziplinen, um ein optimales mechatronisches Produkt zu entwickeln [VDI-Richtlinie 2206, S. 4].

Zahlreiche Vorgehensmodelle befassen sich mit dieser Herausforderung, indem das Produkt bspw. zunächst unter funktions- oder systemorientierten Gesichtspunkten konzipiert wird, um einen gemeinsamen Ausgangspunkt für die weitere Entwicklung

zu etablieren. In der Praxis haben diese jedoch einen geringen Verbreitungsgrad, da die hohen aufzubringenden Aufwände einem, in der industriellen Anwendung als zweifelhaft angesehenen, Nutzen gegenüberstehen. Vor diesem Hintergrund ist es den Entwicklern komplexer Systeme unmöglich, Abhängigkeiten der Subsysteme vollständig zu erfassen und die Auswirkungen von Änderungen in ihrem Entwicklungsbereich umfassend abzuschätzen.

Die Etablierung durchgängiger Nachverfolgbarkeit (engl.: Traceability) ist in diesem Zusammenhang eine Methode, die Antworten auf diese Herausforderungen bereitstellt. Dabei werden die Abhängigkeiten zwischen den unterschiedlichen Modellen, die während der Produktentwicklung entstehen, mit Hilfe sog. Tracelinks explizit dokumentiert, um bspw. die Umsetzung der vom Kunden definierten Anforderungen verfolgen zu können. Jedoch beschränkt sich die Anwendung aktuell hauptsächlich auf Bereiche, in denen sie durch den Gesetzgeber explizit gefordert wird. Die Potenziale, die Tracelinks über den reinen Dokumentationszweck hinaus zu nutzen, werden bisher in der Praxis noch nicht erkannt. Ferner gibt es eine Reihe von technologischen, methodischen und sozialen Herausforderungen, die neben dem Aufwand für die Erfassung der Abhängigkeiten sowie der Pflege der modellierten Tracelinks eine sehr große Hürde für die industrielle Nutzung dieser Methode darstellen.

Diesen Aufwand gilt es zu reduzieren, um eine stärkere Verbreitung der Traceability zu erreichen. Dabei wird es hinsichtlich der (teil-)automatisierten Unterstützung in absehbarer Zeit wohl keinen Goldstandard geben, der für alle denkbaren Situationen während der Systementwicklung eine optimale Hilfe bei der Modellierung und Pflege von Tracelinks darstellt. Dies liegt u. a. an den unterschiedlichen Ausprägungen der in der Entwicklung verwendeten Modelle sowie deren informellen Notation. Somit ist momentan eine Kombination unterschiedlicher unterstützender Methoden anzustreben. Aizenbud-Reshef et al. sprechen in diesem Zusammenhang von einem Puzzle [Aizenbud-Reshef et al. 2006, S. 523], bei dem die Methoden einzelne Puzzle-Teile sind. Diese werden situationsgerecht und zielgerichtet während der Entwicklung eingesetzt, um in Kombination eine möglichst umfassende Unterstützung zur Reduzierung des Aufwands bei der Erstellung und Pflege von Tracelink-Modellen zu bilden.

Vor diesem Hintergrund besteht die Zielsetzung der vorliegenden Arbeit darin, Methoden für die Vervollständigung dieses Puzzles bereitzustellen. Konkret handelt es sich um die drei Methoden *EcoTracing*, *TraceEvaluation* sowie *TraceLegacy*, die der effizienten Modellierung von Tracelinks und der Pflege von Tracelink-Modellen dienen. Alle drei Methoden bauen auf dem aktuellen Stand der Technik auf und gehen jeweils in Teilaspekten über diesen hinaus. Dabei liegt der Forschungsbeitrag, den diese Arbeit leistet, neben der grundlegenden konzeptionellen Entwicklung der Me-

thoden, in deren Evaluation, die anhand dreier Studien sowie einem vergleichenden Beispiel durchgeführt wurde.

1.2 FORSCHUNGSFRAGEN

Die Forschungsfragen, die in diesem Zusammenhang untersucht werden, sind hinsichtlich ihrer Ausrichtung den Bereichen Modellierung von Tracelinks sowie Pflege von Tracelink-Modellen zuzuordnen. Der erste Bereich adressiert dabei methodische Vorgehensweisen für die Reduktion des Aufwands zur Erfassung von Tracelinks:

Kann methodische Unterstützung helfen, den Aufwand bei der Modellierung von Tracelinks zu reduzieren? Wie muss diese ausgeprägt sein?

Die Forschungsfrage adressiert den Aspekt, dass bislang zu wenige Methoden zur Erfassung von Tracelinks existieren und eher unstrukturierte Vorgehensweisen, bei denen leicht Abhängigkeiten übersehen werden können, verbreitet sind. Darüber hinaus besteht die Frage, ob eine Methode zur strukturierten Modellierung von Tracelinks ebenfalls die dafür notwendigen Aufwände reduzieren kann.

Der zweite zu erforschende Bereich betrifft die Pflege von Tracelink-Modellen, die aufgrund sich dynamisch ändernder Modelle zwingend notwendig ist. Es wurden dabei zwei unterschiedliche Forschungsrichtungen verfolgt. Zum einen, ob die Nutzer der Tracelinks in die Qualitätssicherung des Tracelink-Modells mit einbezogen werden können:

Können Methoden des Crowdsourcings zur Steigerung der Qualität des Tracelink-Modells genutzt werden?

Das Crowdsourcing (deutsch: Schwarmauslagerung) ist eine relativ neuer Ansatz, der vor allen Dingen bei einer Vielzahl von Diensten im Internet zu finden ist [Estelles-Arolas und Gonzalez-Ladron-de-Guevara 2012], bislang noch keine Berücksichtigung im Kontext der durchgängigen Nachverfolgbarkeit findet, sich aber potenziell für die Steigerung der Qualität des Tracelink-Modells eignen könnte.

Die zweite Forschungsfrage im Bereich der Pflege von Tracelink-Modellen befasst sich hingegen mit einer speziellen Art der Wiederverwendung von Traceability-Informationen:

Kann das Wissen, welches mithilfe von Tracelinks in vergangenen Projekten dokumentiert wurde, in aktuellen Projekten zur Steigerung der Qualität des Tracelink-Modells genutzt werden?

In diesem Kontext wurde erforscht, ob Wissen über Abhängigkeiten zwischen Elementen, welches in vergangenen Projekten aufwändig mit Hilfe von Tracelinks dokumentiert wurde, wiederverwendet und somit zur Generierung von Vorschlägen für fehlende Tracelinks in aktuellen Projekten herangezogen werden kann.

Grundsätzlich ist anzumerken, dass die im Rahmen dieser Arbeit vorgestellten Softwareprototypen primär dem Zweck dienen, die entwickelten Methoden zu evaluieren, weshalb der wissenschaftliche Fokus der Ausführungen jeweils auf der Entwicklung und Bewertung der Methoden und nicht auf der entsprechenden Software-Implementierung liegt.

1.3 VORGEHEN UND AUFBAU DER ARBEIT

Das methodische Vorgehen zur Beantwortung der im vorangehenden Kapitel 1.2 zusammengestellten Forschungsfragen wurde an die Design Research Methodology (DRM) von Blessing und Chakrabarti angelehnt [Blessing und Chakrabarti 2009]. Diese umfasst vier grundlegende Untersuchungen, die je nach Forschungsaufgabe unterschiedlich ausgeprägt und unterschiedlich oft durchlaufen werden können. Im vorliegenden Fall wurden die ersten beiden Phasen (Klärung der Forschung und deskriptive Untersuchung I) zusammengefasst und anschließend die präskriptive Untersuchung sowie die deskriptive Untersuchung II für jede der drei identifizierten Forschungsfragen durchgeführt (Abbildung 1).

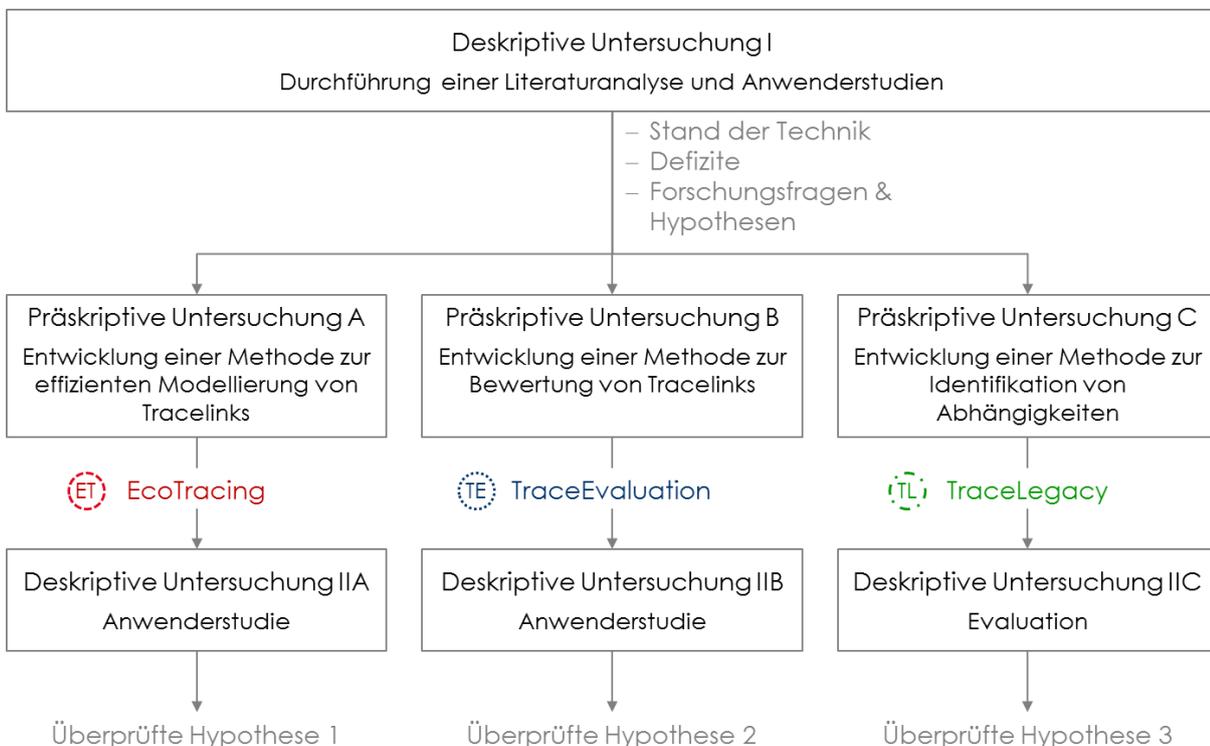


Abbildung 1: Vorgehen zur Bearbeitung der Forschungsfragen

Die erste deskriptive Untersuchung umfasste eine Literaturanalyse und die Durchführung zweier Anwender-Studien, um den Stand der Technik im Bereich der Vorgehensmodelle und der durchgängigen Nachverfolgbarkeit zu erfassen und auf dieser Basis Defizite der aktuellen Methoden zu identifizieren. Aus diesen wurden im Folgenden drei Forschungsfragen und zugehörige Hypothesen entwickelt, welche die Forschungsrichtungen für die präskriptiven Untersuchungen A-C vorgegeben haben. In diesen Untersuchungen wurden Methoden durch den Autor dieser Arbeit entwickelt und prototypisch implementiert. Konkret handelt es sich um Methoden zur effizienten Modellierung von Tracelinks, zur Bewertung von Tracelinks sowie zur Verwendung von Abhängigkeits-Informationen aus vergangenen Projekten, die in den anschließenden deskriptiven Untersuchungen II mit Hilfe von Studien und Evaluationen überprüft wurden.

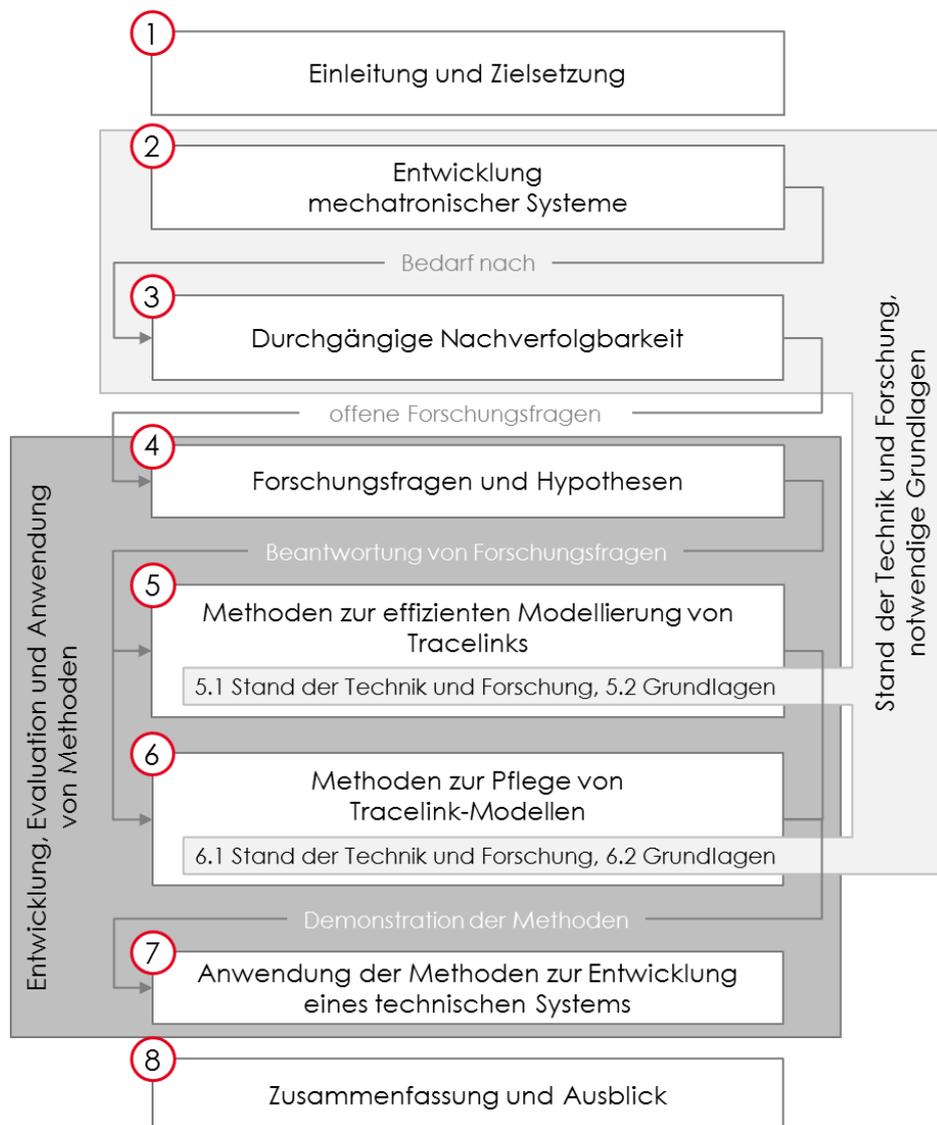


Abbildung 2: Aufbau der Arbeit (Kapitelnummern in Kreisen)

Auch der Aufbau der Arbeit orientiert sich folglich an der DRM und ist in Abbildung 2 inkl. einer Unterteilung der Kapitel nach Stand der Technik (hellgrau hinterlegt) und Methodenentwicklung¹ (dunkelgrau hinterlegt) festgehalten. Die Ergebnisse der Analyse des Standes der Technik (deskriptive Studie I) und damit auch eine Einführung in Grundbegriffe durchgängiger Nachverfolgbarkeit sind in den Kapiteln 2 und 3 dokumentiert. Die detaillierte Beschreibung der Methode zur effizienten Modellierung von Tracelinks (EcoTracing) ist in Kapitel 5 zu finden. Die Methoden zur Bewertung von Tracelinks (TraceEvaluation) und Verwendung von Wissen über Abhängigkeiten in vergangenen Projekten (TraceLegacy), die thematisch der Pflege von Traceability-Informationen zugerechnet werden, sind in Kapitel 6 dokumentiert.

Um die Anwendung und Funktionsweisen der entwickelten Methoden in einem realistischen Umfeld anschaulich darzustellen, wird zusätzlich auf die Ergebnisse der Lehrveranstaltung „Virtual Engineering in Industry“ des Wintersemesters 2011/12 zurückgegriffen. Im Rahmen dieser Veranstaltung wurde in Gruppenarbeit ein mechatronischer Außenspiegel durch die Studenten Aroon Amin, Mohamad Said al-Dahabi, Fabian Friedrich und Justus Thiele entwickelt [Amin et al. 2012]. Dieses Beispiel wird im Rahmen dieser Arbeit analysiert und die Ergebnisse dieser Analyse, zugehörig zu den einzelnen behandelten Aspekten, in den unterschiedlichen Kapiteln dargestellt. So wird die Vorgehensweise der Studenten in Kapitel 2.1 im Kontext anderer Vorgehensmodelle erläutert und diskutiert. Zusätzlich wird der Einsatz der im Rahmen dieser Arbeit entwickelten Methoden anhand des Außenspiegel-Beispiels in Kapitel 7 beschrieben, um deren Anwendung bei der Entwicklung eines mechatronischen Produkts zu verdeutlichen.

¹ Bei der Entwicklung und Evaluation der Methoden (EcoTracing, TraceEvaluation und TraceLegacy) sowie der Demonstration ihrer Anwendung handelt es sich um die eigentliche Forschungsleistung des Autors dieser Arbeit.

2 ENTWICKLUNG MECHATRONISCHER SYSTEME

Vor dem Hintergrund durchgängiger Nachverfolgbarkeit ist eine genauere Untersuchung des Vorgehens bei der Entwicklung mechatronischer Systeme notwendig. Dieses wird in Kapitel 2.1 beschrieben, unterschiedliche Vorgehensweisen verglichen und Unterschiede sowie Gemeinsamkeiten herausgestellt. Eine dieser Gemeinsamkeiten ist, dass während der Entwicklung verschiedene Artefakte² zur Beschreibung des Entwicklungsgegenstands ausgearbeitet werden. Diese Artefakte werden in Kapitel 2.2 vorgestellt und hinsichtlich ihrer Abhängigkeiten zueinander analysiert. Abschließend werden die wichtigsten Aspekte der Vorgehensmodelle und Artefakte in Kapitel 2.3 zusammengefasst, die Abhängigkeiten zwischen diesen noch einmal herausgestellt und somit die Überleitung zur durchgängigen Nachverfolgbarkeit, die im Detail in Kapitel 3 vorgestellt wird, geschaffen.

2.1 VORGEHENSMODELLE ZUR ENTWICKLUNG MECHATRONISCHER SYSTEME

Zur Entwicklung mechatronischer Systeme wurden von unterschiedlichen Institutionen Vorgehensmodelle entwickelt, die helfen sollen, disziplinspezifische Vorgehensweisen zusammenzuführen und die Entwicklung zu koordinieren. So hat der Verein Deutscher Ingenieure (VDI) ergänzend zu seinen Richtlinien 2221 und 2422 die Richtlinie 2206 zu deren Integration herausgegeben (siehe Kapitel 2.1.1). Aus dem Bereich der Luft-

² Als Artefakt werden zusammenfassend jegliche Art von produktbeschreibenden Informationen (Spezifikationen, Funktionsstrukturen, Produktstrukturen etc.) bezeichnet [Cleland-Huang et al. 2003, S. 797].

und Raumfahrt heraus hat sich das Systems Engineering entwickelt und gilt heute als wichtiges Vorgehensmodell für die disziplinübergreifende und vor allen Dingen systemübergreifende Entwicklung mechatronischer Systeme (siehe Kapitel 2.1.2). Die von der Internationalen Organisation für Normung (ISO) herausgebrachte Norm ISO 26262 adressiert hingegen die Automobilbranche und in diesem Kontext insbesondere die Entwicklung sicherheitsrelevanter E/E-Systeme (siehe Kapitel 2.1.3). Da diese Vorgehensmodelle in der Industrie nur punktuell und selten ohne Änderungen eingesetzt werden, wird in Kapitel 2.1.4 zusätzlich exemplarisch auf industrielle Vorgehensweisen eingegangen. Darüber hinaus wird das am Fraunhofer IPK entwickelte V-Modell in Kapitel 2.1.5 beschrieben. Dieses Vorgehensmodell ist insbesondere vor dem Hintergrund relevant, da es von einer Studentengruppe zur Entwicklung eines mechatronischen Außenspiegels verwendet wurde, welcher im weiteren Verlauf der Arbeit als durchgängiges Beispiel fungiert. Abschließend werden die unterschiedlichen Vorgehensmodelle in Kapitel 2.1.6 verglichen und Unterschiede diskutiert.

Das Ziel dieses Kapitels ist es, darzustellen wie unterschiedliche Entwicklungsdisziplinen für die Entwicklung mechatronischer Systeme zusammenwirken müssen und welche unterschiedlichen Schritte dafür notwendig sind. Diese Grundlagen helfen im weiteren Verlauf der Arbeit bei der Identifikation der unterschiedlichen Artefakte und im Folgenden bei der Ableitung des Bedarfs nach einer durchgängigen Nachverfolgbarkeit bei der Entwicklung mechatronischer Systeme.

2.1.1 VDI 2206 – ENTWICKLUNGSMETHODIK FÜR MECHATRONISCHE SYSTEME

Die VDI 2206 ist eine vom Verein Deutscher Ingenieure herausgegebene Richtlinie in der eine Entwicklungsmethodik für mechatronische Systeme beschrieben wird [VDI-Richtlinie 2206]. Sie zielt auf die Bereitstellung von Methoden und Werkzeugen für die frühen Phasen des Entwickelns mit dem Schwerpunkt des Systementwurfs. Dabei liegt besonderes Augenmerk auf Kommunikation und Kooperation zwischen den unterschiedlichen, an der Entwicklung beteiligten Disziplinen. Sie ergänzt damit die Richtlinien VDI 2221 und VDI 2422 und stellt einen übergreifenden Integrationsansatz für deren domänenspezifische Entwicklungsvorgehensweisen dar.

Die Richtlinie umfasst drei grundsätzliche Elemente, von denen im Folgenden jedoch vor allen Dingen das V-Modell als Makrozyklus sowie die Prozessbausteine für wiederkehrende Arbeitsschritte, die die Prozessschritte im V-Modell darstellen, näher erläutert werden. Der Problemlösungszyklus auf Mikroebene, der als Hilfestellungen für Produktentwickler dienen soll, ist vor dem Hintergrund der vorliegenden Arbeit weniger relevant und wird deshalb nicht beschrieben.

Das V-Modell als Makrozyklus beschreibt eine Vorgehensempfehlung für die Entwicklung mechatronischer Systeme. Es besteht aus den Tätigkeiten Systementwurf, domänenspezifischer³ Entwurf, Modellbildung und -analyse, Systemintegration und Eigenschaftsabsicherung, um von den definierten Anforderungen zum fertigen Produkt zu gelangen (siehe Abbildung 3). Dabei kann das V-Modell mehrfach durchlaufen und so bspw. zunächst Labormuster entwickelt werden, um sich dann sukzessive zum finalen Produkt zu iterieren.

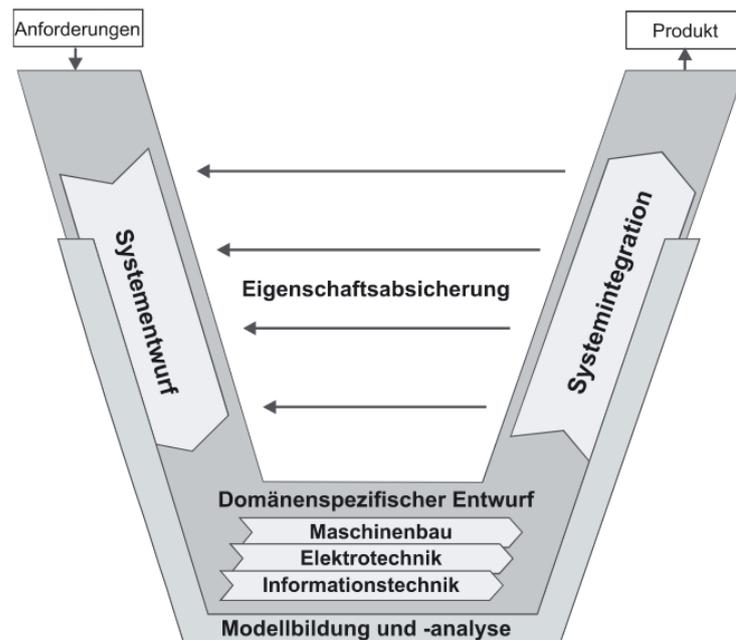


Abbildung 3: V-Modell als Makrozyklus [VDI-Richtlinie 2206]

In jedem Fall beginnt der Zyklus mit der Definition von Anforderungen, die die Aufgabenstellung und damit die Basis für den Systementwurf darstellen. Dieser Entwurf beinhaltet zunächst die Entwicklung einer Funktionsstruktur, die aus den Teilfunktionen des Produkts aufgebaut wird. Deren Ziel ist es, das Verhalten zu beschreiben, Inkonsistenzen zu identifizieren und Wirkprinzipien bzw. Lösungselemente für die einzelnen Funktionen zu finden. Diese so definierten Wirkstrukturen werden weiter zu prinzipiellen Lösungsvarianten detailliert, die bewertet werden. Ergebnis des Systementwurfs ist ein Lösungskonzept, welches die grundsätzlichen Funktionsprinzipien und Zusammenhänge des Systems disziplinübergreifend definiert.

Dieses Lösungskonzept, in dem die Funktionen auf die Disziplinen partitioniert werden, bildet die Grundlage für den folgenden disziplinspezifischen Entwurf. Dieser soll laut VDI nach etablierten Vorgehensweisen ablaufen (z. B. VDI 2221) und wird im Rahmen der Richtlinie nicht weiter spezifiziert. Das Ergebnis dieses detaillierten Designs sind mehrere Lösungsvarianten, die im Rahmen der anschließenden Systemintegration zu

³ Im Rahmen dieser Arbeit wird der Begriff „disziplinspezifisch“ verwendet.

mehreren Gesamtsystemvarianten zusammengeführt und auf mögliche Inkompatibilitäten untersucht werden. Die Integration wird dabei durch die Eigenschaftsabsicherung begleitet, in der die integrierten Gesamtsystemvarianten gegenüber dem Systementwurf bzw. der Anforderungsliste verifiziert und validiert werden. Mit Hilfe dieser Bewertungen können die unterschiedlichen Varianten verglichen, nicht umsetzbare ausgeschlossen und eine optimale ausgewählt werden.

Sowohl der Systementwurf, der domänenspezifische Entwurf als auch die Systemintegration werden dabei von der Modellbildung und -analyse begleitet, um bspw. das Systemverhalten abzubilden oder die Feindimensionierung durchzuführen. Auch bei der Eigenschaftsabsicherung spielen Modelle im Rahmen von zunehmend virtuellen Versuchen eine große Rolle. Die Modelle, die hierbei entwickelt werden, bauen laut Richtlinie idealerweise aufeinander auf. In diesem Zusammenhang wird zwar nicht von Traceability, jedoch von Durchgängigkeit gesprochen, die über alle Entwicklungsphasen angestrebt werden sollte.

2.1.2 SYSTEMS ENGINEERING

Die INCOSE (International Council on Systems Engineering) definiert Systems Engineering als eine Vorgehensweise und Hilfsmittel für die Realisierung erfolgreicher Systeme [Haskins 2006, S. 2.01]. Etwas spezifischer ist die Definition des US amerikanischen Verteidigungsministeriums (Department of Defense), welches Systems Engineering als interdisziplinären Managementprozess für Entwicklungen sieht, mit dessen Hilfe Systeme, die Kundenbedürfnisse befriedigen, entwickelt und verifiziert werden können [Department of Defense 2001, S. 3].

Das zu entwickelnde System wird dabei als Zusammenstellung unterschiedlicher, miteinander interagierender Systemelemente definiert, die ein gemeinsames oder mehrere Ziele verfolgen [Department of Defense 2001, S. 3; Haskins 2006, S. 1.5]. Dabei wird der Mehrwert des Systems gegenüber den einzelnen Elementen durch deren Zusammenwirken, und damit deren Abhängigkeiten untereinander, definiert. Unterscheidungsmerkmal im Vergleich zu anderen Vorgehensmodellen ist beim Systems Engineering, dass diese Systemelemente nicht ausschließlich technischer Natur sein müssen, sondern, je nach Festlegung der Systemgrenze, neben Bauteilen oder Software auch Personen, Einrichtungen, Dokumente oder Richtlinien sein können [NASA STI 2007, S. 3].

Zur Entwicklung dieser Systeme stellt das Systems Engineering zahlreiche technische und nicht-technische Vorgehensweisen und Methoden bereit, die neben der eigentlichen Entwicklung auch auf deren Management zielen (siehe Abbildung 4) [NASA STI 2007, S. 4]. Besonderer Fokus liegt dabei auch auf der Reduzierung von Risi-

ken, die mit der Neuentwicklung bzw. Anpassung bestehender komplexer Systeme verbunden sind [Haskins 2006, S. 2.05].

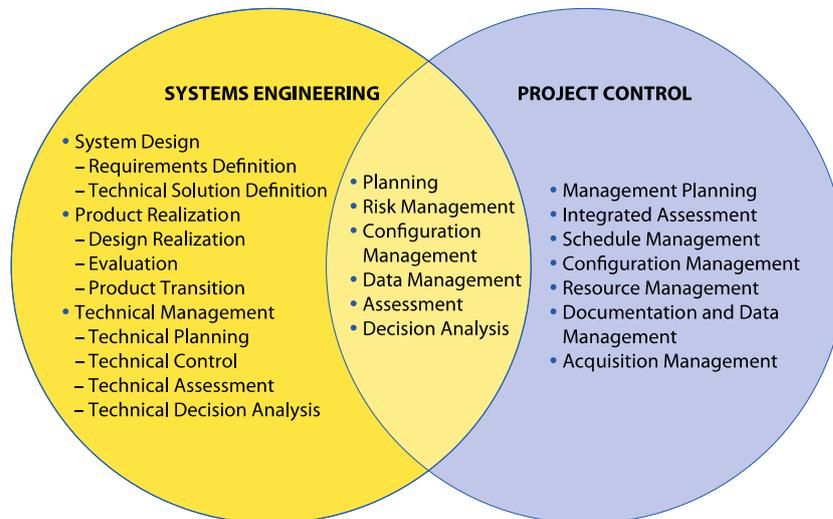


Abbildung 4: Systems Engineering und die darin beinhalteten Methoden und Vorgehensmodelle [NASA STI 2007, S. 4]

Grundsätzlich ist anzumerken, dass zum Thema Systems Engineering unterschiedliche Institutionen Handbücher mit Beschreibungen der Vorgehensweisen und Methoden sowie Hinweisen für deren Anwendung herausgebracht haben, welche sich im Detail leicht unterscheiden (vgl. bspw. [Department of Defense 2001; Haskins 2006; NASA STI 2007]). Diese Unterschiede liegen vor allen Dingen in der Wahl unterschiedlicher Bezeichnungen für inhaltlich gleiche Abläufe und Methoden. Aufgrund der übersichtlichen Darstellung wird im Folgenden der sog. Systems Engineering Prozess, anhand der Ausführungen des US amerikanischen Verteidigungsministeriums beschrieben [Department of Defense 2001] (siehe Abbildung 5).

Als Input für den Systems Engineering Prozess wirken dabei Anforderungen bzw. Wünsche der Kunden sowie anderer Stakeholder. Stakeholder können in diesem Zusammenhang alle Individuen oder Organisationen sein, die ein Interesse an dem zu entwickelnden System aufweisen, somit auch Gesetzgeber, die bspw. Anforderungen bzgl. der Sicherheit des zu entwickelnden Systems haben. In der darauf folgenden Anforderungsanalyse werden alle erfassten Anforderungen überprüft, untereinander abgeglichen und aufbereitet. Das Ergebnis sind funktionale und nicht-funktionale Anforderungen, die präzise und validierbar beschreiben, was das System in welchen Grenzen zu leisten hat.

Im Rahmen der anschließenden funktionalen Analyse dienen die technischen Anforderungen als Grundlage für die Identifikation der Gesamtfunktion(en), über die das System verfügen muss. Diese werden anschließend weiter in die zugehörigen Teilfunktionen heruntergebrochen und den funktionalen und nicht-funktionalen Anforderun-

gen zugeordnet. Zusätzlich werden die Abhängigkeiten dieser Funktionen untereinander identifiziert sowie dokumentiert und somit die funktionale bzw. logische Architektur des Systems entwickelt. Die sog. Anforderungsschleife (Requirements Loop) sorgt dafür, dass das durch die Architekturentwicklung gewonnene Systemverständnis für die erneute Überarbeitung der Anforderungen genutzt wird und somit zu deren Detaillierung und Verbesserung beiträgt. In der anschließenden Synthese wird eine Partitionierung der Funktionen auf die Systemelemente unterschiedlicher Disziplinen, und somit die Definition des Systems im Sinne seiner physikalisch und softwaretechnisch umzusetzenden Elemente, durchgeführt. Das Ergebnis ist die physikalische Architektur des Systems, die als Grundlage der Detailentwicklungen der unterschiedlichen Disziplinen wirkt. Die sog. Designschleife sichert dabei, ähnlich der bereits beschriebenen Anforderungsschleife, die Umsetzbarkeit der entwickelten Architektur ab bzw. sorgt für Anpassungen und Detaillierungen der funktionalen Architektur, falls eine Umsetzung gemäß der ursprünglichen Planung nicht möglich ist. Der Abgleich der Systemarchitektur mit den Anforderungen erfolgt abschließend in der Verifikationsphase (siehe Abbildung 5).

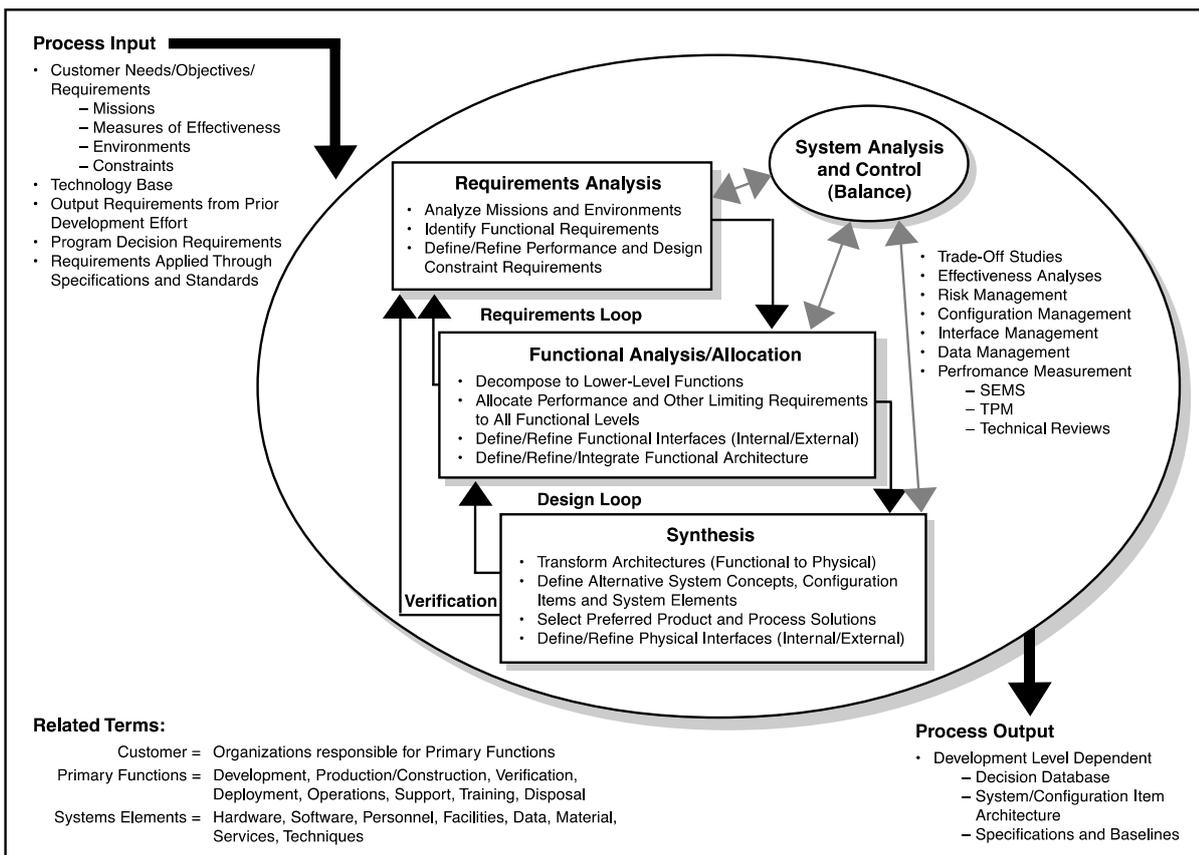


Abbildung 5: Der Systems Engineering Prozess nach [Department of Defense 2001, S. 31]

Zusätzlich zu den beschriebenen Prozessen läuft parallel die Systemanalyse und -kontrolle, in der bspw. Fortschrittskontrollen, die Dokumentation von Entscheidungen oder die Evaluation und Auswahl mehrerer Alternativen während der Systeme-

mentwicklung durchgeführt werden. Darüber hinaus ist es Ziel dieses Prozesses, eine durchgängige Nachverfolgbarkeit zwischen Anforderungen und Systemarchitektur zu etablieren und zu pflegen [Department of Defense 2001, S. 31-33].

2.1.3 ISO 26262 – ROAD VEHICLES – FUNCTIONAL SAFETY

Die ISO 26262 „Road vehicles – Functional Safety“ ist ein internationaler Standard der die funktionale Sicherheit von Elektronik/Elektrik-Systemen (E/E-Systemen) in Straßenfahrzeugen bis 3,5 Tonnen adressiert. Er stellt eine branchenspezifische Ableitung des Standards IEC 61508 dar und besteht aus zehn Teilen.

Der Standard zielt darauf, funktionale Sicherheit für sicherheitsrelevante E/E-Systeme zu gewährleisten. Funktionale Sicherheit ist erreicht, wenn ein System so ausgelegt ist, dass Gefahrensituationen erkannt und ein Unfall verhindert oder zumindest das Ausmaß des Schadens verringert wird [Horstkötter et al. 2010]. Da Einzelmaßnahmen bei der Entwicklung komplexer Systeme zur Erreichung funktionaler Sicherheit nicht ausreichen, stellt der Standard ein umfassendes Rahmenwerk für die Entwicklung von E/E-Systemen bereit, welches durch die Anwendung unterschiedlicher Maßnahmen funktionale Sicherheit gewährleisten soll. Darüber hinaus wird ausdrücklich betont, dass alle sicherheitskritischen Systeme, unabhängig von den entwickelnden Branchen, nach diesem Vorgehensmodell entwickelt werden können.

Die Norm orientiert sich am Aufbau eines V-Modells und ist in separate Teile für die Konzeptentwicklung, die Entwicklungen auf System-, Hardware- und Softwareebene sowie Produktion und Betrieb gegliedert (siehe Abbildung 6).

Die Besonderheit der ISO 26262 ist im Vergleich zu den anderen in dieser Arbeit beschriebenen Normen, dass in jedem der Prozessschritte großer Wert auf die Abschätzung von Risiken sowie Gegenmaßnahmen gelegt wird. Dies spiegelt sich auch in dem großen Stellenwert, der dem Automotive Safety Integrity Level (ASIL)⁴ zugerechnet wird, wider (ISO 26262 Part 9: Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses [ISO 26262-9]).

⁴ Die zu entwickelnden Systeme werden mit dem Automotive Safety Integrity Level (ASIL) im Rahmen einer Gefahren- und Risikoanalyse bewertet und, je nach Einstufung von A (geringes Risiko) bis D (hohes Risiko), Empfehlungen für Maßnahmen ausgegeben.

nächst eine vorläufige grobe Systemarchitektur erstellt, auf deren Basis die technischen Sicherheitsanforderungen als Verfeinerung des Sicherheitskonzepts detailliert und der erfüllenden Hard- und Software zugewiesen werden (ISO 26262 Part 4: Product development at the system level [ISO 26262-4]). Ferner werden Schnittstellen spezifiziert, unterstützende Prozesse festgelegt (vgl. ISO 26262 Part 8: Supporting processes [ISO 26262-8]) und abschließend das Systemdesign finalisiert. Dieses Design stellt die Voraussetzung für die Hard- und Softwareentwicklungen dar, die beide nach ähnlichen Vorgehensmodellen ablaufen (ISO 26262 Part 5: Product development at the hardware level [ISO 26262-5] und Part 6: Product development at the software level [ISO 26262-6]): zunächst werden Anforderungen an die Hard- bzw. Software spezifiziert (inkl. der Sicherheitsanforderungen) und das Hardware Design bzw. die Softwarearchitektur erstellt, welche im Anschluss implementiert werden. Dabei wird durch die Norm gefordert, dass die Umsetzung der Sicherheitsanforderungen in der Implementierung nachverfolgbar dokumentiert wird und somit nachgewiesen werden kann. Im Anschluss an Integration, Tests sowie Verifikationen auf Hardware- und Softwareebene werden die Ergebnisse der domänenspezifischen Entwicklungen auf Systemebene integriert und im Systemverbund gegenüber den Sicherheitsanforderungen verifiziert und validiert.

Da sowohl Produktion, als auch der eigentliche Betrieb der Entwicklungsgegenstände von großer Bedeutung für die funktionale Sicherheit sind, werden diese parallel zur Entwicklung auf Systemebene ebenfalls betrachtet (ISO 26262 Part 7: Production and operation [ISO 26262-7]). Dabei werden Sicherheitsanforderungen an die Produktion und den Betrieb detailliert und bspw. Instruktionen für den Zusammenbau oder die Reparatur der Systeme definiert.

In den restlichen fünf Teilen der Norm werden genutzte Fachwörter (ISO 26262 Part 1: Vocabulary [ISO 26262-1]) erläutert und unterstützende Prozesse und Methoden beschrieben. So definiert ISO 26262 Part 2: Management of functional safety [ISO 26262-2] Sicherheitsmanagement-Rollen und -Verantwortlichkeiten sowie Anforderungen an Organisationen, die nach ISO 26262 entwickeln wollen. In Part 8 Supporting processes [ISO 26262-8] werden Hinweise für die Spezifikation und Verwaltung von Sicherheitsanforderungen gegeben und explizit Nachverfolgbarkeit für diese gefordert. Darüber hinaus werden u. a. Methoden für das Konfigurationsmanagement und das Änderungsmanagement empfohlen.

2.1.4 INDUSTRIELLE VORGEHENSWEISE AM BEISPIEL DER AUTOMOBILINDUSTRIE

Die in den Kapiteln 2.1.1 bis 2.1.3 beschriebenen Vorgehensmodelle zur Entwicklung mechatronischer Systeme sind von ihren Autoren für den direkten Einsatz in der industriellen Anwendung vorgesehen. Dort herrschen jedoch eher Mischformen bzw.

an die jeweilige Situation angepasste Formen der genannten Vorgehensmodelle vor [Königs et al. 2012, S. 925]. Um Unterschiede und Ähnlichkeiten zwischen industrieller und von Richtlinien und Normen empfohlener Anwendung darzustellen, wird im Folgenden beispielhaft die Vorgehensweise in der Automobilindustrie beschrieben⁵.

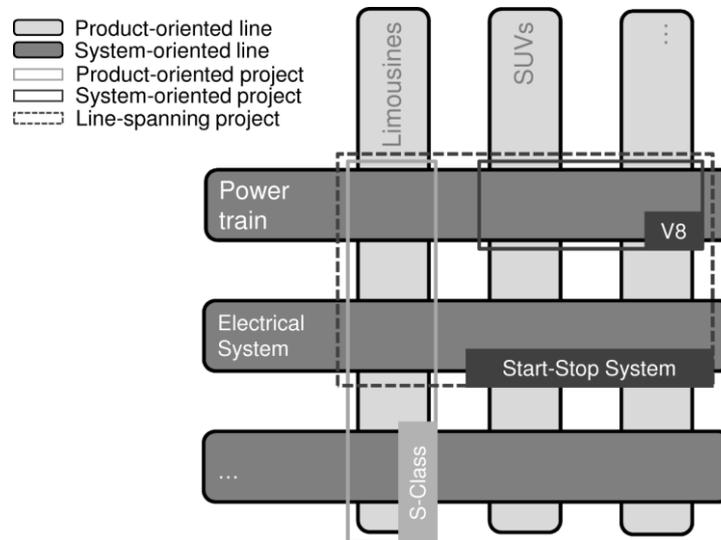


Abbildung 7: Matrixorganisation in der Automobilindustrie [Königs et al. 2012, S. 925]

Der Aufbau der Organisationen ist in der Automobilbranche vielfach als Matrix ausgelegt, die sich an den vorhandenen Produktlinien (vertikale Linien) und den integrierten Systemen (horizontale Linien) orientiert (siehe Abbildung 7).

Während die in den vertikalen Linien strukturierten Organisationseinheiten für die Integration der Systeme in die spezifischen Produktlinien zuständig sind, werden die Systeme in den Organisationseinheiten der horizontalen Linien entwickelt [Königs et al. 2012, S. 925-926]. Laut Weber lassen sich dabei sechs gängige Systeme oder Entwicklungsbereiche unterscheiden [Weber 2009, S. 9]:

- Antriebsstrang: Motor, Getriebe, Differentiale, Kühlungssystem, Auspuff usw.
- Fahrwerk: Achsen, Federung, Lenkung, Räder, Reifen usw.
- Karosserie: Karosseriestruktur, Türen, Motorhaube, Heckklappe usw.
- Exterieur: Stoßstangen, Fenster, Türsystem usw.
- Interieur: Cockpit, Verkleidungen, Teppich, Sitze usw.
- E/E: Sensoren, Aktuatoren, Verkabelung, Steuergeräte usw.

Die Organisation von Entwicklungen erfolgt in Projekten, die sich, in Abhängigkeit der Komplexität des Entwicklungsvorhabens, des Neuheitsgrads (Anpassungs- oder Neu-

⁵ Obwohl die Automobilhersteller unterschiedliche Prozesse nutzen, ist deren grundsätzlicher Ablauf auf der betrachteten Ebene durchaus vergleichbar [Weber 2009, S. 1]. Aus diesem Grund werden nicht spezifische Ausprägungen, sondern ein verallgemeinerter Prozess beschrieben.

entwicklung) und des Umfangs der Entwicklung, unterscheiden [Weber 2009, S. 1]. Die Projekte überspannen dabei je nach Ausrichtung eine oder mehrere horizontale und vertikale Linien der Matrix [Königs et al. 2012, S. 926; Königs 2012, S. 14].

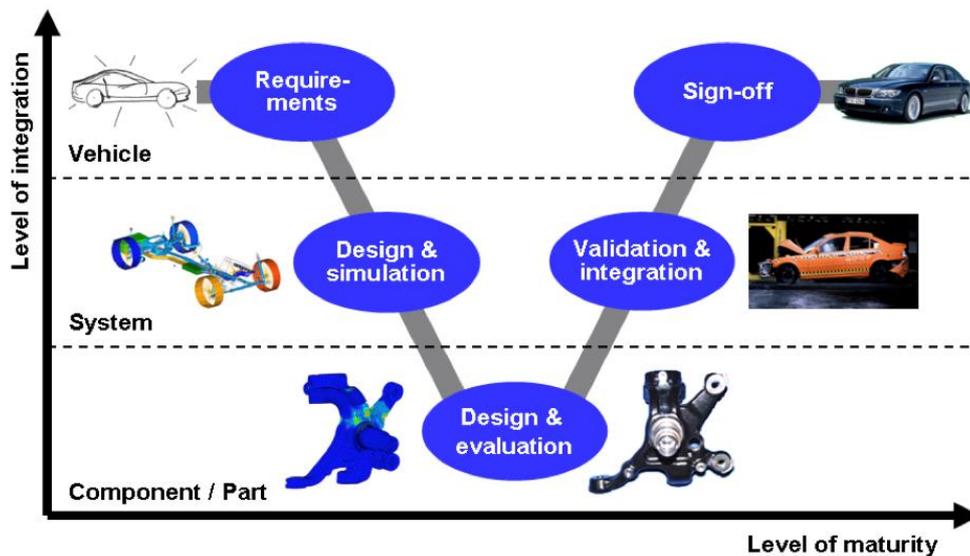


Abbildung 8: Übergreifende Vorgehensweise in der Automobilindustrie [Weber 2009, S. 12]

Das übergeordnete Vorgehen innerhalb der Projekte orientiert sich am V-Modell (siehe Abbildung 8), welches ähnlich ausgeprägt ist, wie in Kapitel 2.1.1 beschrieben. Ausgehend von der Spezifikation des Gesamtfahrzeugs werden die Anforderungen auf die zuvor genannten Systeme und weiter auf die beinhalteten Bauteile heruntergebrochen [Weber 2009, S. 11]. Da die grundsätzliche Architektur des Gesamtfahrzeugs überwiegend bekannt ist, erfolgt jedoch im Unterschied zu V-Modell nach VDI 2206 (siehe Kapitel 2.1.1) keine lösungsneutrale Beschreibung auf funktionaler Ebene, sondern eine direkte Weitergabe der Spezifikationen an die unterschiedlichen System-Entwicklungsbereiche (und damit die unterschiedlichen Entwicklungsdisziplinen). Innerhalb dieser Bereiche haben sich unterschiedliche Vorgehensweisen etabliert, die sich an den Bedarfen der jeweiligen Disziplin orientieren: So arbeiten bauteilorientierte Bereiche auf einem von Beginn an vergleichsweise detaillierten Niveau mit CAD-Modellen, MKS-, FEM- und zahlreichen weiteren spezialisierten IT-Werkzeugen zur Auslegung der Bauteile und Baugruppen und sichern diese mit Hilfe von DMUs (später mit realen Prototypen) ab [Weber 2009, S. 41-52]. Im Bereich der E/E-Entwicklung wird hingegen aufgrund der starken Vernetzung der einzelnen Komponenten und Systeme eine abstraktere Sichtweise benötigt, weshalb laut Weber domänenintern in Anlehnung an das Systems-Engineering-Vorgehensmodell verfahren wird: aus den meist textuellen Anforderungen wird eine Systemarchitektur abgeleitet bzw. eine bestehende angepasst [Weber 2009, S. 53-78]. Diese umfasst eine Funktionsstruktur, welche die für den Fahrer erlebbaren Funktionen beschreibt, weiter detailliert und deren Abhängigkeiten abbildet. Darüber hinaus wird die Partitionie-

rung auf Funktionsträger (z. B. Steuergeräte) ebenfalls zur Systemarchitektur (auch technische Systemarchitektur genannt) hinzugerechnet. Erst im Anschluss werden Steuergeräte, Embedded Software und Anwendungssoftware entwickelt. Dies kann zum Teil auf Basis der Funktionsstruktur geschehen, indem diese weiter detailliert und abschließend mit Hilfe von Codegeneratoren (z. B. TargetLink) in Programmcode für Steuergeräte umgewandelt wird [Conrad et al. 2005, S. 6]. Die Absicherung erfolgt hierarchisch von Bauteil bis Gesamtfahrzeug. Zunächst werden Komponenten wie Steuergeräte inkl. Software in sog. Hardware-in-the-Loop (HiL) Tests getestet. Dabei werden Schnittstellen-Parameter (von Sensoren und Aktuatoren) durch eine HiL-Simulationsumgebung emuliert und somit schnelle Tests ermöglicht. Im Folgenden werden die E/E-Subsysteme und Systeme integriert, getestet und gegenüber den Anforderungen abgesichert [Weber 2009, S. 67-69]. Die Integration des Gesamtfahrzeugs erfolgt dann wieder in den vertikalen Linien (siehe Abbildung 7) und dient der Absicherung und der Evaluierung der Gesamtfahrzeugeigenschaften. Dabei kommen u. a. folgende Integrations- und Absicherungsmethoden zum Einsatz [Weber 2009, S. 45-49; 70-71]: Die geometrische Integration und Absicherung dient der Verteilung und Überwachung des Bauraums. Zum Einsatz kommen bspw. DMUs, mit deren Hilfe Kollisionen und Spiel analysiert werden. Die funktionale Integration und Absicherung dient der Überprüfung der funktionalen Eigenschaften des Gesamtfahrzeugs (z. B. passive Sicherheit). Die Systemintegration dient der Integration und Absicherung des vollständigen E/E-Systems auf Fahrzeugebene.

2.1.5 V-MODELL DES FRAUNHOFER IPK

Das im Folgenden beschriebene V-Modell wurde am Fraunhofer IPK entwickelt [Beier et al. 2012, S. 14-17] und orientiert sich an dem V-Modell der VDI 2206 (siehe Kapitel 2.1.1), welches jedoch insbesondere in den frühen Phasen erweitert wurde (siehe Abbildung 9).

Ausgangspunkt des V-Modells ist eine übergeordnete Produktplanung, bei der zunächst das zu entwickelnde Produkt in seinen Grundzügen (vom Management) spezifiziert wird. Darauf folgt die Architekturreflexion, bei der Vorgängerprodukte hinsichtlich ihrer Architektur analysiert und ggf. neue Systeme auf abstrakter Ebene integriert werden. Ergebnis dieser beiden Schritte ist ein initiales Set an Anforderungen, welches als Ausgangspunkt für die eigentliche Entwicklung dient.

Die Anforderungen werden im nächsten Schritt in der Anforderungskaskade heruntergebrochen, bis die zu realisierenden Funktionen sowie deren Eigenschaften identifiziert werden können.

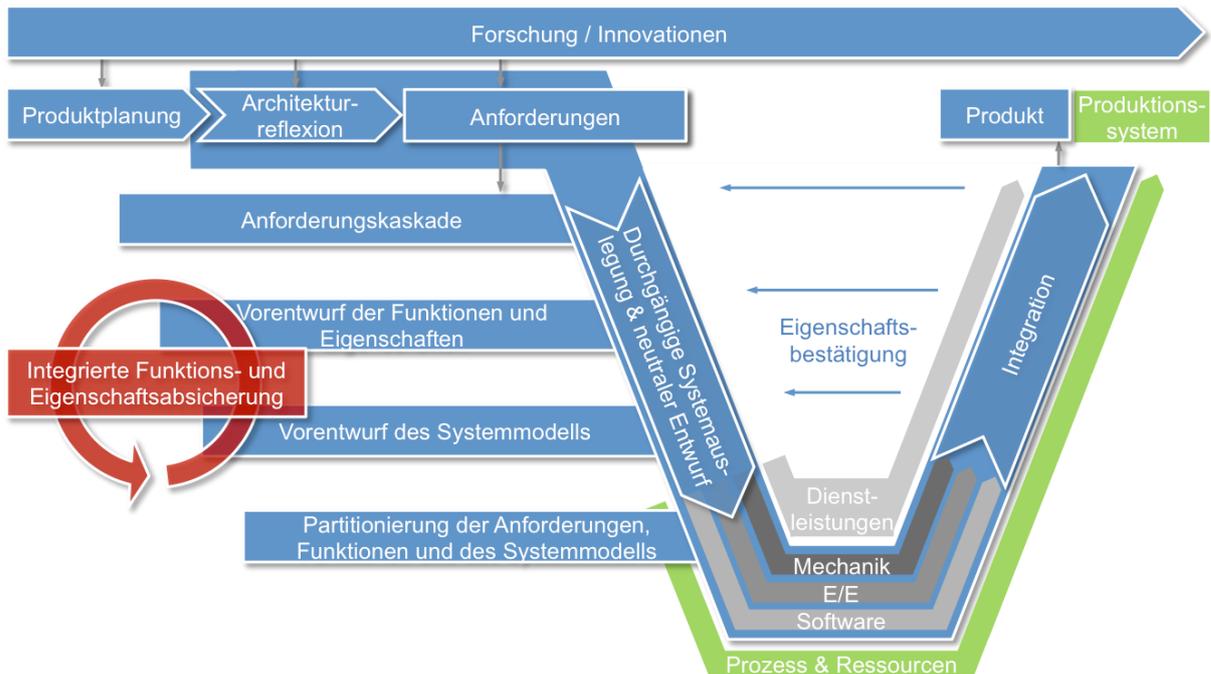


Abbildung 9: V-Modell des Fraunhofer IPK

Die Funktionen werden im folgenden Schritt, dem Vorentwurf der Funktionen und Eigenschaften, modelliert und deren Abhängigkeiten abgebildet. Dieser funktionale Vorentwurf wird anschließend zum Vorentwurf des Systemmodells weiterentwickelt, indem die Funktionen hinsichtlich ihrer Wirk- und Lösungsprinzipien spezifiziert werden und deren logisches Verhalten modelliert wird. Dieses Systemmodell sowie das Funktionsmodell und die Anforderungen, die in den vorangegangenen Schritten entwickelt wurden, unterliegen dabei einer ständigen Iteration, um eine integrierte Funktions- und Eigenschaftsabsicherung zu erreichen. Ziel ist es die Eigenschaften des zu entwickelnden Produkts so weit wie möglich abzusichern bevor die detaillierte Entwicklung, getrennt nach Entwicklungsdisziplinen, erfolgt. Als Abschluss der Phase der durchgängigen Systemauslegung und des neutralen Entwurfs werden die zuvor kaskadierten Anforderungen, detaillierten Funktionen und des Systemmodells partitioniert und somit den Disziplinen Mechanik, Elektronik/Elektrik und Software sowie Dienstleistungsentwicklung und Produktionsplanung zugeordnet.

Die abschließende Integration wird von einer stetigen Überprüfung der Eigenschaften gegenüber dem Systementwurf begleitet. Aufgrund umfassender Simulationen auf dem linken Ast des Vs sollte diese allerdings eher einer Bestätigung denn einer Absicherung (siehe V-Modell nach VDI 2206 in Kapitel 2.1.1) gleich kommen.

Begleitet werden die zuvor genannten Prozessschritte von einem kontinuierlichen Forschungsprozess aus dem insbesondere zu Beginn der Entwicklung Innovationen in den eigentlichen Ablauf des V-Modells eingesteuert werden können.

Das beschriebene V-Modell stellte die Grundlage für die Entwicklung des eingangs erwähnten mechatronischen Außenspiegels, der in dieser Arbeit als durchgängiges Beispiel verwendet wird, dar. Er wurde im Rahmen der Lehrveranstaltung „Virtual Engineering in Industry“ mit der Tool-Landschaft V6 von Dassault Systèmes (siehe auch Kapitel 3.6.2.1) entwickelt [Amin et al. 2012; Dassault Systèmes 2013]. Die Projektarbeit, welche im Wintersemester 2011/12 durchgeführt wurde, gliederte sich aufgrund vorgegebener Design Reviews in vier Phasen, von denen die ersten drei die Entwicklung des Außenspiegels und die letzte die Dokumentation des Projekts behandelten [Amin et al. 2012, S. 7]. In Phase 1 wurden zunächst vorgegebene Anforderungen aufgenommen und auf dieser Basis unterschiedliche Konzepte entwickelt. Die grobe Evaluation dieser Konzepte hinsichtlich ihrer Kosten, Gewichte und grundsätzlichen Machbarkeit führte zur Auswahl eines weiterzuverfolgenden Konzepts. Die bereits aufgenommenen Anforderungen wurden ergänzt, strukturiert und in ENOVIA dokumentiert. Hinsichtlich des V-Modells handelt es sich hierbei um die Anforderungskaskade. Ebenfalls in Phase 1 wurden anschließend alle notwendigen Funktionen des mechatronischen Außenspiegels identifiziert und daraus eine Funktionsstruktur in CATIA aufgebaut. Es handelt sich um den Vorentwurf der Funktionen, bei dem sog. „Building Blocks“ genutzt werden, um die Funktionen zu modellieren. In Projektphase 2 schloss sich die Wahl der Wirkprinzipien und die Modellierung des Verhaltensmodells mit Hilfe des in CATIA integrierten DYMOLA Editors an. In Bezug auf das V-Modell handelt es sich dabei um die Entwicklung des Systemmodells, bei dem mit Hilfe der Sprache Modelica das Verhalten des Systems modelliert wird. Ebenfalls parallel bzw. leicht zeitversetzt begann die Detailkonstruktion der Bauteile und Baugruppen in CATIA. Dieser Prozessschritt entspricht der mechanischen Entwicklung im V-Modell. Aspekte wie die Entwicklung der Steuergeräte (E/E-Entwicklung im V-Modell) oder der zugehörigen Software (Softwareentwicklung im V-Modell) wurden aufgrund zeitlicher und kapazitiver Restriktionen nicht behandelt. In Phase 3 der Projektarbeit wurden abschließend Anpassungen an den System- und geometrischen Modellen durchgeführt, die aufgrund entsprechenden Feedbacks in den Design Reviews notwendig wurden. Zusätzlich wurde die Erfüllung der Anforderungen mit Hilfe der zuvor erstellten Modelle nachgewiesen. Weitere Schritte wie bspw. Prototypenbau oder die Absicherung der Produktion mit Methoden der Digitalen Fabrik waren nicht im Umfang der Lehrveranstaltung vorgesehen und wurden daher nicht weiter betrachtet.

2.1.6 VERGLEICH DER VORGEHENSMODELLE

Die in den Kapiteln 2.1.1 bis 2.1.5 beschriebenen Vorgehensmodelle haben einen grundsätzlich ähnlichen Ablauf, der sich in die Phasen „Planung“, „Entwurf“, „Detailentwicklung“, „Integration“ und „Nutzung“ unterteilen lässt. Dabei werden jedoch nicht alle Phasen von allen Vorgehensmodellen abgedeckt (siehe Abbildung 10).

Ähnlichkeiten bestehen zudem auch bei der detaillierteren Betrachtung, bei der sich innerhalb der Phasen ähnliche, wiederkehrende Prozessschritte offenbaren. So findet sich die Abfolge „Anforderungsanalyse“ – „Funktionsdefinition“ – „Synthese“, wenngleich unter unterschiedlichen Bezeichnungen und in unterschiedlichen Umfängen, in der VDI 2206 [VDI-Richtlinie 2206], dem Systems Engineering [Department of Defense 2001], der ISO 26262 [ISO 26262-4; ISO 26262-3] sowie im Fraunhofer IPK Vorgehensmodell [Beier et al. 2012]. Lediglich in der industriellen Anwendung wird anders vorgegangen, da eine lösungsneutrale Beschreibung des zu entwickelnden Systems im Tagesgeschäft als zu aufwändig und nicht zielführend wird.

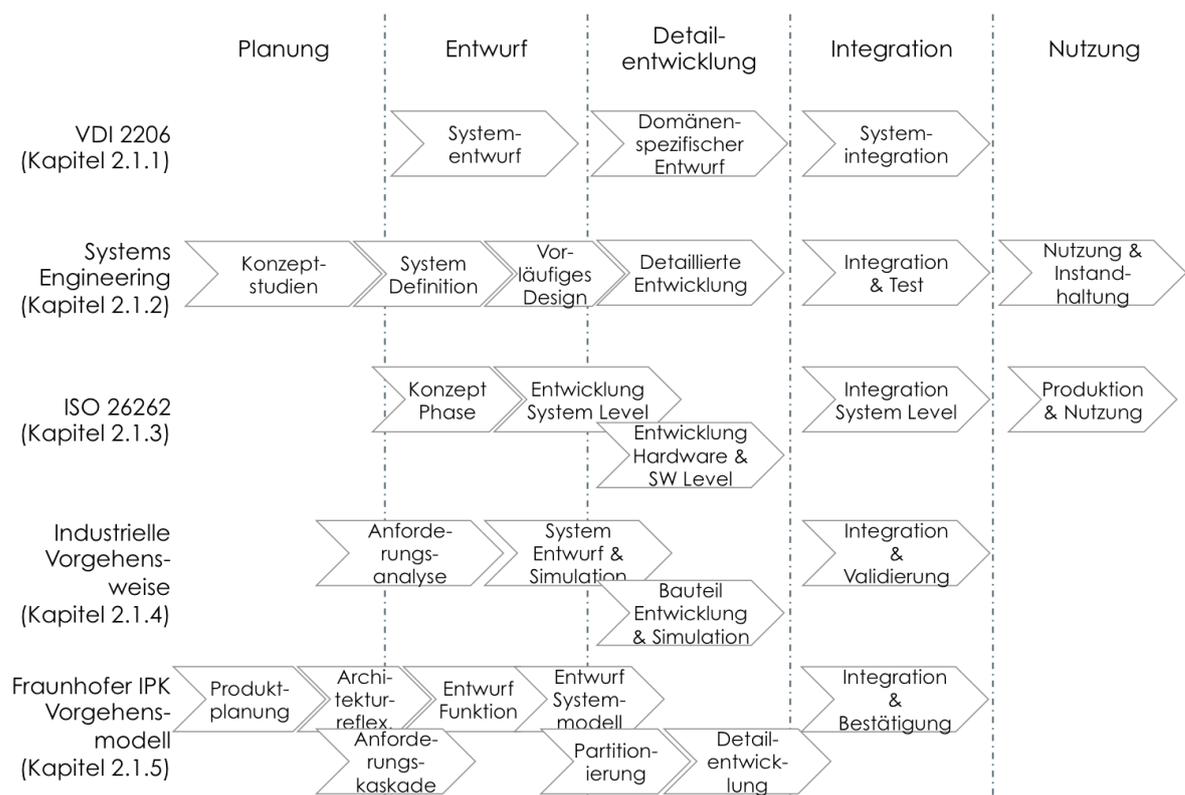


Abbildung 10: Vergleich der Vorgehensmodelle

Die Vorgehensmodelle sind sich somit dahin gehend ähnlich, dass basierend auf Anforderungen eine Art Systembeschreibung entwickelt wird bzw. bereits vorhanden ist. Diese kann sich im Detail unterscheiden, jedoch überwiegt der Ansatz, die Funktionen des Systems zu definieren und diese dann schrittweise in Richtung Umsetzung zu konkretisieren. Im Anschluss folgt bei allen Vorgehensmodellen die Detailentwicklung,

die getrennt nach Entwicklungsdisziplinen durchgeführt wird. Abschließend ist eine Integration erforderlich, bei der das Zusammenspiel der Systeme gegenüber den Anforderungen getestet wird.

Neben diesen Ähnlichkeiten gibt es jedoch auch Unterschiede, sowohl auf abstrakter Betrachtungsebene (so adressiert bspw. das Systems Engineering und das Fraunhofer IPK Vorgehensmodell als einzige der betrachteten Vorgehensmodelle die Phase der Planung) als auch im Detail. Diese Unterschiede ergeben sich u. a. aus der Herkunft und den Zielen der Vorgehensmodelle. So geht das Systems Engineering von einer umfassenden Gesamtsystembetrachtung aus. Unterscheidungsmerkmal im Vergleich zu anderen Vorgehensmodellen ist dabei, dass Systemelemente nicht ausschließlich technischer Natur sein müssen sondern, je nach Festlegung der Systemgrenze, neben Bauteilen oder Software auch Personen, Einrichtungen, Dokumente oder Richtlinien sein können [NASA STI 2007, S. 3].

Die VDI 2206 wurde hingegen entwickelt, um bereits bestehende domänenspezifische Vorgehensmodelle (VDI 2221 & VDI 2422) zu integrieren und damit die Entwicklung mechatronischer Systeme zu ermöglichen. Daher wird ein großer Wert auf den lösungsneutralen Systementwurf gelegt und die domänenspezifischen Entwicklungsvorgehensweisen nicht näher betrachtet. Auch Tätigkeiten wie Projektmanagement, die integraler Bestandteil des Systems Engineerings sind, werden nicht berücksichtigt.

Die ISO 26262 richtet sich zunächst an die Entwicklung sicherheitskritischer E/E-Systeme, jedoch wird ausdrücklich darauf hingewiesen, dass sie auch für die Entwicklung sicherheitskritischer Funktionen anderer Entwicklungsdisziplinen anwendbar ist. Vor diesem Hintergrund ist nicht verwunderlich, dass der größte Unterschied zu den anderen Vorgehensmodellen der Fokus auf die Abschätzung von Risiken sowie der Entwicklung von Gegenmaßnahmen ist.

In der industriellen Vorgehensweise, die anhand des Beispiels der Automobilindustrie beschrieben wurde, unterscheidet sich die Vorgehensweise von den anderen Vorgehensmodellen. Es wird auf die lösungsneutrale Beschreibung der Systeme verzichtet, da die überwiegende Anzahl der Entwicklungen evolutionäre Weiterentwicklungen und keine Neukonstruktionen sind. Das bedeutet, dass der grundsätzliche Aufbau des Produkts bekannt ist und anhand diesem Entwicklungsbereiche unterschieden werden, die das Fahrzeug auf den Ebenen Gesamtfahrzeug, System und Bauteil entwickeln. Innerhalb dieser Bereiche haben sich unterschiedliche Vorgehensweisen durchgesetzt, die sich an den Bedarfen der Disziplinen orientieren.

Unabhängig vom genutzten Vorgehensmodell werden in jedem Fall verschiedene Modelle und Dokumente genutzt, um den Systementwurf und die Detailentwicklung durchzuführen und zu dokumentieren. Diese sog. Artefakte stellen damit eine wichti-

ge Beschreibungsform während der Systementwicklung dar. Die Beschreibung ausgewählter Artefakte erfolgt im folgenden Kapitel 2.2.

2.2 ARTEFAKTE ZUR ENTWICKLUNG MECHATRONISCHER SYSTEME

Dokumente, Modelle, Code und weitere digitale Beschreibungsformen, die während des Entwicklungsprozesses erstellt werden, können zusammenfassend als Artefakte bezeichnet werden [Cleland-Huang et al. 2003, S. 797]. Im Fokus dieser Arbeit stehen dabei strukturierte Artefakte, bei denen sich einzelne Elemente identifizieren lassen. Das können bspw. einzelne Anforderungen in der objektorientierten Software DOORS oder Bauteile einer Baugruppe in einem CAD-System sein. Genauso lassen sich die folgenden Erläuterungen jedoch bspw. auch auf strukturierte, in Word verfasste Anforderungsspezifikationen, bei denen Überschriften genutzt werden, um die Anforderungen eindeutig identifizierbar zu machen, anwenden.

In den folgenden Kapiteln wird auf die Artefakte „Anforderungsartefakt“ (Kapitel 2.2.1), „Funktionsartefakt“ (Kapitel 2.2.2), „Verhaltensartefakt“ (Kapitel 2.2.3) und „Produktstrukturartefakt“ (Kapitel 2.2.4) eingegangen. Das Ziel des Kapitels ist es, die wichtigsten Artefakte zur Entwicklung technischer Systeme kurz vorzustellen und zu vermitteln, dass Abhängigkeiten zwischen diesen Artefakten bestehen, die während der Entwicklung berücksichtigt werden müssen (Kapitel 2.2.5).

2.2.1 ANFORDERUNGSARTEFAKTE

Die IEEE definiert den Begriff der Anforderung als [IEEE 610.12-1990, S. 62]:

1. Eine Bedingung oder Eigenschaft, die ein System oder eine Person benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen.
2. Eine Bedingung oder Eigenschaft, die ein System oder eine Systemkomponente aufweisen muss, um einen Vertrag zu erfüllen oder einem Standard, einer Spezifikation oder einem anderen formell auferlegten Dokument zu genügen.
3. Eine dokumentierte Repräsentation einer Bedingung oder Eigenschaft wie in 1. oder 2. definiert.

Zielsetzung der Formulierung von Anforderungen ist somit die lösungsneutrale⁶ Spezifikation eines Systems im Hinblick auf seine später auszuführenden Funktionen und zu

⁶ In der Realität sind die Anforderungen häufig lösungsspezifisch formuliert, da es sich bei den meisten Entwicklungen um Anpassungen eines Vorgängersystems handelt und somit auf dem vorhergehenden Wissensstand aufgebaut wird [Almefelt et al. 2006, S. 121; Wrasse 2011, S. 22-23].

erfüllenden Eigenschaften [Weilkiens 2006, S. 38-44; Pahl et al. 2007, S. 215]. Sie spielen damit eine zentrale Rolle in der Produktentwicklung, wobei sich ihre genaue Ausprägung in Abhängigkeit des Charakters des Entwicklungsprojekts unterscheiden kann: während Anforderungen in kleinen unabhängigen Projekten ausschließlich zur bereits erwähnten Spezifikation verwendet werden, dienen sie in großen Entwicklungsprojekten zusätzlich der Aufgabenverteilung [Almefelt et al. 2006, S. 119].

Für die eindeutige Dokumentation von Anforderungen sind je nach Literaturquelle unterschiedliche Informationen notwendig. Übereinstimmend sind jedoch mindestens eine eindeutige Identifikationsnummer sowie ein beschreibender Text erforderlich [Weilkiens 2006, S. 40; NASA STI 2007, S. 48; Pahl et al. 2007, S. 216]. Darüber hinaus sind die Angabe der oder des Verantwortlichen, Datum der Aufnahme der Anforderung und der letzten Änderung, Quelle der Anforderung sowie erläuternde Informationen möglich [Pahl et al. 2007, S. 217].

Anforderungen lassen sich dabei in Abhängigkeit ihrer Ausprägung in verschiedene Arten unterscheiden. Während bspw. funktionale Anforderungen die Funktionen, die ein System oder eine Komponente ausführen können soll, beschreiben, dienen Leistungsanforderungen (nicht-funktionale Anforderungen) der Spezifikation, wie gut diese ausgeführt werden sollen [NASA STI 2007, S. 41]. Zusätzlich gibt es je nach konsultierter Literaturquelle die Unterscheidung zahlreicher weiterer Kategorien: Schnittstellen-Anforderungen, Design-Anforderungen, operationelle Anforderungen, Anforderungen an Ressourcen, Verifikationsanforderungen usw. [Department of Defense 2001, S. 35-36; Hood und Wiebel 2005, S. 102; NASA STI 2007, S. 41-45; Wrasse 2011, S. 27].

Anforderungen bilden somit eine abstrakte Sicht auf das zu entwickelnde System, in der beschrieben wird, was entwickelt werden soll. Sie definieren damit die Ausprägung der im Folgenden entwickelten Artefakte (siehe Kapitel 2.2.2 bis 2.2.4) und stellen den Ausgangspunkt der weiteren Entwicklungsaktivitäten dar. Sie dienen zudem der Kommunikation unter Stakeholdern, der Validierung des Systems sowie der Steuerung der Entwicklung [Yeh und Zave 1980].

2.2.2 FUNKTIONSARTEFAKTE

Mit Hilfe von Funktionsartefakten lassen sich bei der Entwicklung mechatronischer Systeme die Funktionen des zu entwickelnden Systems beschreiben. Es gibt in diesem Zusammenhang sehr viele unterschiedliche Vorgehensweisen und Definitionen, sowohl in der Forschung, als auch in der industriellen Anwendung, die aus unterschiedlichen Anwendungsszenarien motiviert und nicht immer miteinander kompatibel sind

[Vermaas 2009, S. 113]. Eine gute Übersicht über unterschiedliche Ansätze findet sich in [Erden et al. 2008].

Im Kontext dieser Arbeit wird die Definition der Funktion nach Ponn und Lindemann verwendet [Ponn und Lindemann 2008, S. 56], die in ähnlicher Form auch bei Pahl et al. und der VDI-Richtlinie 2206 zum Einsatz kommt [VDI-Richtlinie 2206, S. 32-33; Pahl et al. 2007, S. 243]:

Funktionen stellen im Kontext der Systementwicklung „eine am Zweck orientierte, lösungsneutrale, als Operation beschriebene Beziehung zwischen Eingangs- und Ausgangsgrößen eines Systems“ dar [Ponn und Lindemann 2008, S. 56].

Trotz ihrer großen Verbreitung (insbesondere im deutschsprachigen Raum) ist auch diese Definition in der Wissenschaft nicht frei von Kritik. So weisen van Beek und Tomiyama auf das sog. „ontology problem“ hin, welches in einigen Fällen auch auf die vorliegende Definition zutrifft [van Beek und Tomiyama 2008, S. 4]. Unter diesem Problem verstehen sie einen zu eng (oder auch zu weit) gesteckten Rahmen an unterschiedlichen Funktionen, die mit einer Methode beschrieben werden können. Funktionen, die weder über Energie-, Material- noch Signalflüsse verfügen, können somit, mit der von Ponn und Lindemann definierten Funktion, nicht beschrieben werden. Als Beispiel für eine solche Funktion führen van Beek und Tomiyama die Funktion des Bügels bei einem Kopfhörer an, die sich mit der Wandlung der genannten Flüsse nicht beschreiben lässt [van Beek und Tomiyama 2008, S. 4]. Bei der Wahl der Funktionsdefinition und -ontologie ist es somit wichtig, dass sie auf den Anwendungsfall abgestimmt sind, denn auf eine für alle Anwendungsfälle gültige Definition wurde sich zumindest bis heute noch nicht geeinigt.

Obwohl Schwierigkeiten bei der Definition der Funktion bestehen wird das grundsätzliche Verfahren, die Funktionen bei der Entwicklung mechatronischer Systeme zu beschreiben, insbesondere in der Wissenschaft als wichtige Methode angesehen⁷. Insbesondere die Tatsache, dass es den Begriff der Funktion in allen Disziplinen der mechatronischen Systementwicklung (wenn auch in unterschiedlichem Umfang) gibt und das Konzept der Funktionsmodellierung somit unabhängig von der Entwicklungsdisziplin ist, wird als einer der größten Vorteile angesehen [van Beek und

⁷ In der industriellen Anwendung gibt es zwar einige positive Beispiele (siehe z.B. [Balluchi et al. 2002]), die allgemeine Verbreitung der disziplinübergreifenden Funktionsmodellierung ist jedoch gering. Zum einen ist dies auf den Aufwand und den Schwierigkeitsgrad zurückzuführen, der mit der Erstellung der Funktionsartefakte verbunden ist [Kroll 2012, S. 168]. Darüber hinaus stellen die fehlende eindeutige Definition der Funktion sowie vorhandene „individuelle“ Definitionen ein echtes Einstiegshindernis dar [van Beek und Tomiyama 2008, S. 5].

Tomiyaama 2008, S. 4]. Funktionsartefakte können somit von Experten unterschiedlicher Disziplinen gemeinsam entwickelt und als gemeinsame Sprache bei der Spezifikation des zu entwickelnden Systems und dessen gewünschten Verhalten genutzt werden [Erden et al. 2008, S. 147; van Beek und Tomiyama 2008, S. 4; Vermaas 2009, S. 113].

Funktionen werden aus den funktionalen Anforderungen des Anforderungsartefakts (siehe Kapitel 2.2.1) abgeleitet und in Artefakten dokumentiert [Ponn und Lindemann 2008, S. 56-57]. Die Definition der Haupt- und Hilfsfunktionen (unterschiedlicher Abstraktionsstufen), die ein System zur Verfügung stellen soll, ist damit ein Zwischenschritt zwischen der Aufgabenstellung und der Entwicklung der Lösung. Dabei soll die explizit lösungsneutrale Formulierung der Funktionen⁸ dabei helfen, unvoreingenommen an die Entwicklungsaufgabe heranzugehen und somit die Findung neuartiger Lösungen zu unterstützen [Leemhuis 2005, S. 51; Pahl et al. 2007, S. 243; Ponn und Lindemann 2008, S. 56]. Zusätzlich hilft die Lösungsneutralität auch dabei, das System ohne Bindung an bestimmte Entwicklungsdomänen zu beschreiben und trägt somit zur Verständigung zwischen den Entwicklern unterschiedlicher Domänen bei [Erden et al. 2008, S. 147].

2.2.3 VERHALTENSARTEFAKTE

Im Anschluss an die Modellierung der Funktionen mechatronischer Systeme erfolgt die Festlegung konkreter Wirkprinzipien für deren Umsetzung und damit eine erste grobe Auswahl beteiligter Systemelemente. Da nun die (dynamischen) Eigenschaften und die Interaktion dieser Elemente untereinander das gesamte Systemverhalten beeinflussen [Vajna 2009, S. 29], ist es hilfreich dieses mit Hilfe von Verhaltensmodellen zu simulieren. Verhaltensmodelle sind abstrakte mathematische Modelle, mit deren Hilfe bereits früh im Entwicklungsprozess belastbare Abschätzungen über Eigenschaften des zu entwickelnden Systems getroffen werden können [Janschek 2010, S. 50]. Dabei liegt eine große Herausforderung in der Heterogenität der beteiligten Entwicklungsdomänen begründet [Janschek 2010, S. 11]. So müssen sehr unterschiedliche Systemelemente im Verhaltensmodell integriert und in Kombination miteinander simuliert werden (z. B. Mechanik, Fluidik, digitale Informationsverarbeitung usw.).

⁸ Die Lösungsneutralität bei der Beschreibung der Funktionen ist nicht unumstritten [Umeda und Tomiyama 1995, S. 4; Chakrabarti und Bligh 2001, S. 500; Kroll 2012, S. 166]. Umeda und Tomiyama argumentieren bspw., dass es häufig schwierig ist, die Funktion eines Systems unabhängig von dessen Verhalten (und damit dessen Realisierung) zu betrachten [van Beek und Tomiyama 2008, S. 4].

Eine Sprache, die von vielen frei verfügbaren und kommerziellen Werkzeugen genutzt wird (z. B. OpenModelica der Universität Linköping, Dymola von Dassault Systèmes, MOSILAB von Fraunhofer FIRST, SimulationX der Gesellschaft für ingenieurtechnische Informationsverarbeitung [Modelica Association 2013]), ist Modelica [Modelica Association 2012]. Für sie gibt es zahlreiche freie und kostenpflichtige Bibliotheken mit Objekten unterschiedlicher Disziplinen, die kombiniert und simuliert werden können. Laut Janschek hat Modelica einige wichtige Eigenschaften: sie verfolgt eine modellspezifische, anstatt einer mathematisch-orientierten Beschreibung, ist anwendungsneutral, unterstützt gemischt kontinuierlich-diskrete Systeme und berücksichtigt physikalische Einheiten [Janschek 2010, S. 132].

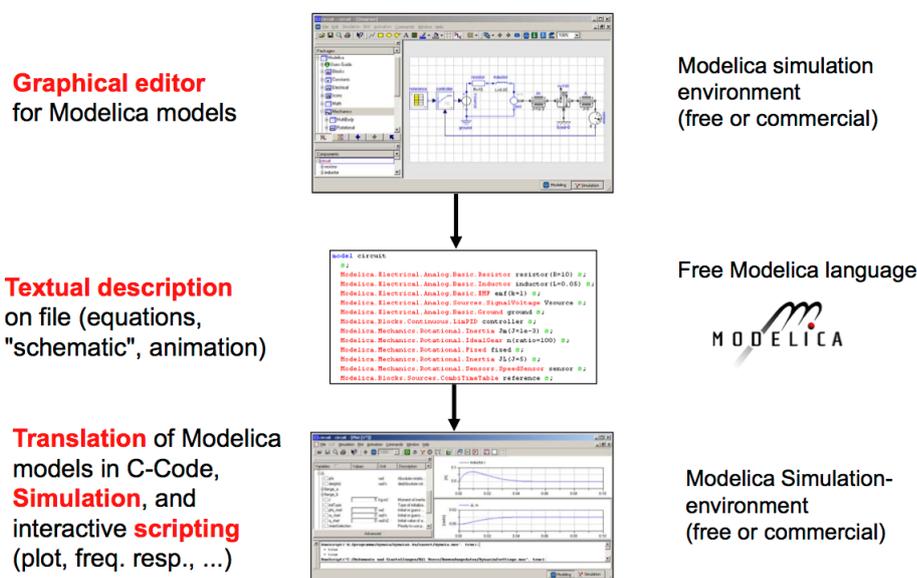


Abbildung 11: Simulationsumgebungen auf Basis von Modelica [Otter 2011, S. 6]

Die Modellierung des Verhaltensmodells und der Simulationdurchlauf verlaufen meistens wie in Abbildung 11 dargestellt ab: die relevanten Objekte (z. B. ein Widerstand) werden in dem grafischen Editor aus Bibliotheken geladen und an ihren „Ports“ mittels Signal- und Energie-Flüssen miteinander verknüpft [Modelica Association 2012]. Jedes Objekt ist dabei textuell mit Modelica beschrieben. Nach Modellierung des Verhaltensmodells werden diese textuellen Beschreibungen in C-Code transformiert und die Simulation durchgeführt. Das Simulationsergebnis wird in Form von Variablen und deren Verlauf über die Zeit ebenfalls in der Simulationsumgebung ausgegeben.

Mit der Hilfe von Verhaltensartefakten lässt sich somit das Verhalten des zu entwickelnden Systems simulieren. Es werden die zu realisierenden Funktionen mit Hilfe von Modellen der gewählten Systemelemente und Wirkprinzipien umgesetzt und deren Zusammenwirken evaluiert.

2.2.4 PRODUKTSTRUKTURARTEFAKTE

Laut Dubbel et al. bildet „die Produktstruktur [...] den Aufbau eines Produktes in Form einer hierarchischen Struktur ab.“ Diese Ansicht teilen zahlreiche andere Autoren, welche die Produktstruktur bzw. die synonym zu gebrauchende Erzeugnisgliederung als hierarchische Zerlegung eines Produkts in seine Haupt- und Teilkomponenten definieren (z. B. [Schönsleben 2007, S. 21; VDI 2219, S. 10]). Im Kontext des Systems Engineerings wird diese Definition im Kern beibehalten und nur insofern erweitert, dass nicht nur Produkte, Baugruppen und Einzelteile, sondern darüber hinaus auch Systeme, Sub-Systeme, Software- und Informationselemente strukturiert werden. Die Struktur wird Product Breakdown Structure [NASA STI 2007, S. 52] oder Work Breakdown Structure [Department of Defense 2001, S. 85] genannt.

Die Produkt- oder Systemstruktur dient somit der Verwaltung von Systemen, Baugruppen sowie Bauteilen und Software- und Informationselementen, deren Leistung und Funktionsweise in den Anforderungen spezifiziert, Funktionen in Funktionsartefakten beschrieben und Verhalten in Verhaltensartefakten überprüft wurde.

2.2.5 ABHÄNGIGKEITEN ZWISCHEN ARTEFAKTEN

Die in den Kapiteln 2.2.1 bis 2.2.4 vorgestellten Artefakte stellen einige (wenngleich wichtige) Beispiele möglicher Artefakte bei der Systementwicklung dar. Insbesondere in der Detailentwicklung gibt es in den unterschiedlichen Entwicklungsdisziplinen zahlreiche auf besondere Aufgabenstellungen spezialisierte Werkzeuge, in denen weitere Artefakte erzeugt werden [Tudorache 2006, S. 2]. Allen Artefakten ist gemein, dass sie nicht inhaltlich isoliert existieren, sondern Redundanzen bzw. inhaltliche Abhängigkeiten zu anderen Artefakten aufweisen [Egyed 2000, S. 28; VDI-Richtlinie 2206, S. 47; Tudorache 2006, S. 2]. So kann bspw. das Trägheitsmoment einer Komponente einmal als Parameter in einem Verhaltens-Modell und einmal über die Abmessungen und Materialauswahl in einem CAD-Programm beschrieben werden. Andererseits erfüllen bspw. Funktionen des Funktionsartefakts Anforderungen, die in einer Anforderungsspezifikation dokumentiert sind. Diese Funktionen werden wiederum von Lösungselementen realisiert, das Verhalten in Verhaltensmodellen simuliert und deren Gestalt in Form von CAD-Modellen in der Produktstruktur abgelegt wird.

Auch bei der Entwicklung des mechatronischen Außenspiegels lassen sich zahlreiche Redundanzen und Abhängigkeiten zwischen den vier modellierten Artefakten (Anforderungsspezifikation, Funktionsstruktur, Verhaltensmodell und Produktstruktur) identifizieren. So ist bspw. in den Anforderungen die maximale Dauer der Beheizung für die Außenspiegel angegeben (= 360 Sekunden). Diese Dauer findet sich ebenfalls im Verhaltensmodell als modellierter „Timer“ wieder, der die Schaltung der Spiegelhei-

zung steuert. Diese Redundanz, die dadurch entsteht, dass der Parameter sowohl in der Anforderungsspezifikation, als auch im Verhaltensmodell dokumentiert ist bzw. verwendet wird, kann insbesondere bei einseitiger Änderungen zu Inkonsistenzen zwischen den Artefakten und damit zur Nicht-Erfüllung von Anforderungen führen.

Gleiches gilt auch für Vorgaben für maximalen Abmessungen sowie die maximale Masse, welche in den Anforderungen beschrieben sind. Zwar wird in diesem Fall der Parameter nicht unverändert in die geometrischen Bauteile im CAD-Modell übernommen, jedoch müssen die Abmessungen bzw. die Masse bei der Validierung gegenüber diesen abgesichert werden. Es bestehen somit Abhängigkeiten zwischen den Anforderungen und der Produktstruktur.

Auch wenn keine Parameter involviert sind, können Abhängigkeiten zwischen den unterschiedlichen Elementen der Artefakte bestehen. So ist in den Anforderungen des Außenspiegels spezifiziert, dass ein Toter-Winkel-Assistent in den Außenspiegel integriert werden soll. Dies hat zur Folge, dass die Studenten eine gleichnamige Funktion im Funktionsmodell modelliert haben, die im Verhaltensmodell anschließend näher hinterlegt wurde. Auch Abhängigkeiten zum Geometrie-Modell bestehen, da für den Assistenten LEDs im Spiegel montiert und Kabel verlegt werden müssen. In Abbildung 12 ist beispielhaft die ausgewählte LED-Baugruppe und ihre Positionierung im CAD-Modell des Außenspiegels zu erkennen.

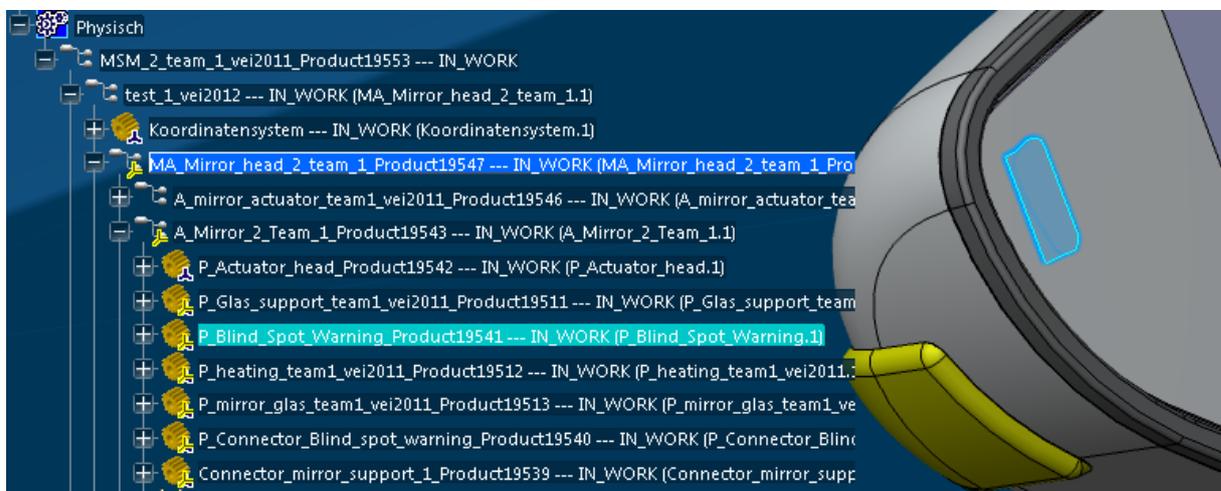


Abbildung 12: Darstellung der LEDs des Toter-Winkel-Assistenten im Geometriemodell des Außenspiegels (vgl. [Amin et al. 2012])

Zusammenfassend kann festgestellt werden, dass inhaltliche Abhängigkeiten unterschiedlicher Ausprägung zwischen den Artefakten bei der Entwicklung technischer Systeme bestehen. Diese zu kennen und explizit zu dokumentieren ist essentiell, da es sonst zu Inkonsistenzen zwischen den Artefakten und damit zu Fehlern während der Entwicklung kommen kann. Dies gilt insbesondere für die Entwicklung komplexer Sys-

teme, da die Anzahl und Komplexität der Artefakte sowie die Anzahl der beteiligten Entwickler hoch und somit das Potenzial für Inkonsistenzen besonders groß ist.

2.3 ZUSAMMENFASSUNG UND DISKUSSION

In Kapitel 2.1 wurden unterschiedliche Vorgehensmodelle zur Entwicklung mechatronischer Systeme vorgestellt und miteinander verglichen. Diese haben zum Teil einen ähnlichen Ablauf, wenngleich sie nicht alle denselben Zeitraum im Lebenszyklus eines Systems abdecken. Auch bei der detaillierteren Betrachtung fallen Ähnlichkeiten auf, die sich insbesondere anhand gleicher Prozessschritte ausprägen.

Neben diesen Ähnlichkeiten gibt es jedoch auch Unterschiede. Diese ergeben sich u. a. aus der Herkunft und den Zielen der Vorgehensmodelle. So geht das Systems Engineering bspw. von einer umfassenden Gesamtsystembetrachtung unter Berücksichtigung nicht-technischer Systeme aus. Die VDI 2206 wurde hingegen entwickelt, um bereits bestehende domänenspezifische Vorgehensmodelle zu integrieren. Daher wird großer Wert auf den lösungsneutralen Systementwurf gelegt und die domänenspezifischen Entwicklungsvorgehensweisen nicht näher betrachtet. Die ISO 26262 wurde für die Entwicklung sicherheitskritischer E/E-Funktionen spezifiziert, weshalb im Unterschied zu den anderen Vorgehensmodellen der Fokus auf der Abschätzung von Risiken sowie der Entwicklung von Gegenmaßnahmen liegt. Die industrielle Vorgehensweise, die in dieser Arbeit am Beispiel der Automobilindustrie beschrieben wurde, unterscheidet sich am stärksten von den sonstigen Vorgehensmodellen. Es wird auf die lösungsneutrale Beschreibung der Systeme verzichtet, da die überwiegende Anzahl der Entwicklungen evolutionäre Weiterentwicklungen sind, womit der grundsätzliche Aufbau des Produkts unverändert bleibt.

Unabhängig vom genutzten Vorgehensmodell werden verschiedene Modelle und Dokumente genutzt, um den Systementwurf und die Detailentwicklung durchzuführen und zu dokumentieren. Zusammenfassend können diese digitalen Ergebnisse als Artefakte bezeichnet werden. Neben der kurzen Vorstellung der Artefakte war die Erkenntnis des vorliegenden Kapitels, dass die Artefakte nicht inhaltlich isoliert existieren, sondern Redundanzen und Abhängigkeiten zueinander aufweisen, ein essentielles Ergebnis dieses Kapitels. Auf Probleme, die sich daraus ergeben und Lösungsansätze, die diese adressieren, wird im folgenden Kapitel 3 eingegangen.

3 DURCHGÄNGIGE NACHVERFOLGBARKEIT

Ein Ansatz, um die in Kapitel 2 beschriebenen Redundanzen und Abhängigkeiten der Artefakte der Systementwicklung zu dokumentieren und somit allen an der Entwicklung beteiligten Entwicklern zur Verfügung zu stellen, ist die durchgängige Nachverfolgbarkeit bzw. Traceability⁹. Dabei werden die inhaltlichen Abhängigkeiten zwischen den systembeschreibenden Artefakten und deren Elementen explizit mit Hilfe von sog. Tracelinks abgebildet. Das Ergebnis ist eine durchgängige Nachverfolgbarkeit, mit deren Hilfe die Artefakte untereinander konsistent gehalten werden können.

Die folgenden Kapitel geben einen Überblick über die Grundlagen, die Vorteile und Nachteile der durchgängigen Nachverfolgbarkeit. Zunächst wird in Kapitel 3.1 näher auf die Motivation, durchgängige Nachverfolgbarkeit zu praktizieren, eingegangen und anschließend in Kapitel 3.2 ihre geschichtliche Entwicklung betrachtet. Eine Einordnung in den Unternehmenskontext erfolgt in Kapitel 3.5. Notwendige Grundbegriffe werden in Kapitel 3.3 zusammengefasst, der grundlegende Ablauf zur Erstellung und Verwendung von Traceability in Kapitel 3.4 dargestellt und die Herausforderungen, die mit dieser Methode verbunden sind, in Kapitel 3.7 diskutiert. Kapitelübergreifend werden dabei Traceability-Kriterien vorgestellt, mit deren Hilfe sich Software-Werkzeuge charakterisieren lassen. Diese Kriterien wurden durch den Autor in Zu-

⁹ Die beiden Begriffe „(durchgängige) Nachverfolgbarkeit“ und „Traceability“ werden im Rahmen dieser Arbeit synonym verwendet.

sammenarbeit mit Simon Königs, Grisca Beier und Rainer Stark während der Erarbeitung der Veröffentlichung [Königs et al. 2012] erarbeitet und werden im Text „**fett**“ gekennzeichnet und die zugehörigen Ausprägungen bzw. Eigenschaften „*kursiv*“ dargestellt.

In Kapitel 3.6 werden die Traceability-Kriterien anschließend genutzt, um die Eigenschaften der dort vorgestellten aktuell verfügbaren kommerziellen Werkzeuge zur Verwaltung, Modellierung und Verwendung von Traceability zu beschreiben. Besondere Berücksichtigung findet dabei der ModelTracer (siehe Kapitel 3.6.4.3), welcher im Rahmen des BMBF-geförderten Projekts ISYPROM in Kooperation der TU Berlin, dem Fraunhofer IPK und der InMediasP GmbH entwickelt wurde, da er als Basis für die prototypische Implementierung und anschließende Validierung der im Rahmen dieser Arbeit entwickelten Methoden dient.

Abschließend werden in Kapitel 3.8 vorhandene Methoden, deren Umsetzung in Software-Tools und Herausforderungen diskutiert und die Bedarfe für Ansätze in den Bereichen der Erfassung von Tracelinks und Pflege von Tracelink-Modellen abgeleitet.

Das Ziel dieses Kapitels ist es, einen Überblick über das Thema durchgängige Nachverfolgbarkeit zu geben. Nach dem Lesen sollten die Grundbegriffe, Methoden sowie deren Umsetzungen bekannt und Vor- und Nachteile dieses Ansatzes verstanden sein. Aus den Nachteilen sowie deren Eingrenzung auf die unterschiedlichen Phasen der durchgängigen Nachverfolgbarkeit werden im folgenden Kapitel 4 die Forschungsfragen abgeleitet und auf dieser Basis die Hypothesen formuliert.

3.1 BEDARF NACH DURCHGÄNGIGER NACHVERFOLGBARKEIT

Eine häufig getroffene Aussage im Zusammenhang mit modernen mechatronischen Systemen ist, dass deren hohe Komplexität eine große Herausforderung für die Entwicklung darstellt (vgl. bspw. [Tretow et al. 2008, S. 36]). Um die Ursachen dafür zu analysieren, ist es zunächst notwendig, die Begriffe „System“ und „Komplexität“ näher zu betrachten.

Allgemein definiert besteht ein System aus einer Anzahl von Elementen bzw. Teilsystemen, die miteinander in Beziehung stehen [Ehrlenspiel 2007, S. 20]. Es erfüllt zumeist eine oder mehrere Funktionen [Haskins 2006, S. 1.5] und lässt sich durch eine Systemgrenze von der Umwelt abgrenzen.

Die Komplexität ist eine Eigenschaft dieser Systeme, die sich objektiv aus der Anzahl der das System ausmachenden Systemelemente und deren Abhängigkeiten ermitteln lässt [VDI-Richtlinie 2206, S. 4; Weilkiens 2006, S. 10; Ehrlenspiel 2007, S. 36]. Je nach Definition werden zusätzlich noch die Anzahl der möglichen Systemzustände

(und damit die Dynamik des Systems) in die Ermittlung der Komplexität miteinbezogen (vgl. bspw. [Probst und Ulrich 1988, S. 58]).

Bei der Betrachtung heutiger Systeme fällt auf, dass eine Verschiebung von rein mechanischen Systemen hin zu mechatronischen Systemen stattgefunden hat, sodass Elektronik und Software mittlerweile einen immer größeren Stellenwert für die Realisierung von Funktionen einnehmen [Conrad et al. 2005, S. 3; Dannenberg und Burgard 2007, S. 4]. Das bedeutet, dass die unterschiedlichen Systemelemente häufig nicht mehr nur einer Disziplin zuzuordnen sind, sondern vielmehr allen drei Disziplinen Mechanik, Elektronik/Elektrik und Software. Darüber hinaus erfolgt die Realisierung der Funktionen durch ein enges Zusammenspiel der Wirkprinzipien der Disziplinen, was zu einer großen Anzahl von Abhängigkeiten zwischen den Disziplinen führt [Conrad et al. 2005, S. 3-4]. Zusätzlich sind bspw. im Automobilbau immer mehr Teilsysteme an der Realisierung mehrerer Funktionen beteiligt¹⁰, was weitere Abhängigkeiten zwischen den Teilsystemen und -funktionen zur Folge hat [Conrad et al. 2005, S. 3-4; Dannenberg und Burgard 2007, S. 12]. Es kommt somit zu einer Erhöhung der Anzahl der Abhängigkeiten zwischen den Elementen in einem System. Hinzu kommt, dass sich auch die Anzahl dieser Elemente stetig erhöht und somit ebenfalls zu einer Steigerung der Komplexität beiträgt. So beinhaltet ein modernes Fahrzeug heute, im Vergleich zu früher, zahlreiche Systeme, die Sicherheits- oder Komfortfunktionen erfüllen: die Anzahl verfügbarer Sonderausstattungen hat sich bspw. in einem 7er BMW innerhalb von zwanzig Jahren von 14 im Jahr 1986 auf 92 im Jahr 2006 gesteigert [Dannenberg und Burgard 2007, S. 15]. Ebenfalls anschaulich illustriert wird diese Steigerung der Anzahl der Elemente dadurch, dass 1974 wenige DIN A4 Seiten ausreichten, um die Anforderungen an das Instrumentenpanel eines Fahrzeugs zu erfassen. Heutzutage ähnelt der Umfang einer solchen Spezifikation eher dem eines Buches [Almfelt et al. 2006, S. 121].

Zu der Komplexität des Systems kommt die Komplexität der Entwicklung des Systems hinzu. Wie in Kapitel 2.2 dargestellt, werden für die Entwicklung eines mechatronischen Systems unterschiedliche Artefakte durch zahlreiche Entwickler der beteiligten Fachabteilungen in spezialisierten Werkzeugen erstellt, die zwar separat aber trotzdem inhaltlich aufeinander aufbauend entwickelt werden, wodurch Redundanzen und Abhängigkeiten zwischen ihnen bestehen können [Egyed 2000, S. 28; VDI-Richtlinie 2206, S. 47; Tudorache 2006, S. 2]. Das Wissen über diese Abhängigkeiten liegt jedoch meistens nur implizit vor und ist nicht allen Entwicklern bewusst [Aizenbud-Reshef et al. 2006, S. 515; Tudorache 2006, S. 3]. Insbesondere bei Systemen, die

¹⁰ So werden bspw. beim PRE-Safe-System von Mercedes-Benz Crash-Sensoren, ESP, Sitzverstellung, Sicherheitsgurt und Schiebedach zusätzlich genutzt, um im Verbund Sicherheitsfunktionen zu realisieren [Dannenberg und Burgard 2007, S. 12].

verteilt entwickelt werden, kann es deshalb aufgrund fehlender Kommunikation bei Änderungen zu Schwierigkeiten kommen, was wiederum Inkonsistenzen zwischen den Artefakten zur Folge haben kann [Pinheiro 2003, S. 91; Aizenbud-Reshef et al. 2006, S. 515; Tudorache 2006, S. 3]. Dies wurde auch im Rahmen einer Studie, in der 1401 Ingenieure aus unterschiedlichen Branchen befragt wurden, bestätigt: bei Änderungen der Artefakte bemängeln 51 % der Ingenieure, nicht rechtzeitig informiert zu werden [Müller et al. 2013, S. 24].

Dies stellt vor dem Hintergrund, dass Änderungen während der Entwicklung sehr häufig sind, ein Problem dar. So ändern sich üblicherweise mehr als 50 % der Systemanforderungen, bevor ein System zum Einsatz kommt [Kotonya und Sommerville 1998]. Dies wurde auch im Bereich des Systems Engineering (vgl. Kapitel 2.1.2) erkannt, und das Management bzw. die Abschätzung der Auswirkungen von Änderungen als wichtige Aufgabe des Systems Engineers definiert [Haskins 2006, S. 8.05].

Ein Ansatz, diese Abhängigkeiten zwischen den unterschiedlichen Artefakten der Systementwicklung zu verwalten, ist es, sie mit Hilfe von Methoden der Traceability explizit zu dokumentieren. Die so erfassten Tracelinks können genutzt werden, um sog. Auswirkungsanalysen¹¹ durchzuführen, mit welchen auch bei komplexen Systemen von Änderungen betroffene Elemente identifiziert werden können [Gotel und Finkelstein 1994, S. 99; Spanoudakis et al. 2004, S. 105; Almfelt et al. 2006, S. 128; Eigner und Stelzer 2009, S. 79; Mäder et al. 2009c, S. 7]. Auf ähnliche Weise helfen sie, das Systemverständnis und die Transparenz der Entwicklung zu steigern, indem sie Abhängigkeiten für alle Entwickler einsehbar dokumentieren [Sutinen et al. 2004, S. 5; Maurer 2007, S. 37]. Darüber hinaus ermöglichen es die Tracelinks, die Entstehung der unterschiedlichen Artefakte nachzuvollziehen (z. B. lassen sich die Begründung oder der Urheber für Anforderungen mit ihrer Hilfe festhalten) [Ramesh und Edwards 1993, S. 257; Gotel und Finkelstein 1994, S. 99; Dömges und Pohl 1998, S. 56; Ramesh und Jarke 2001, S. 21; Egyed und Grünbacher 2002, S. 168-169; Pinheiro 2003, S. 101; Spanoudakis et al. 2004, S. 105; Almfelt et al. 2006, S. 122], deren Konsistenz untereinander zu gewährleisten [Ramesh und Edwards 1993, S. 256; Sutinen et al. 2000, S. 12; Tudorache 2006, S. 52] sowie bei der Absicherung der Systeme, die Anforderungen ihren erfüllenden Elementen zuzuordnen [Ramesh und Edwards 1993, S. 256; Gotel und Finkelstein 1996, S. 167; Dömges und Pohl 1998, S. 56; Ramesh und Jarke 2001, S. 16-17; Egyed und Grünbacher 2002, S. 168-169; Spanoudakis et al. 2004, S. 105; Mäder et al. 2009c, S. 7; ISO 26262-3, S. 16]. Traceability kann demnach eine erhebliche Unterstützung bei der Entwicklung komplexer Systeme leisten.

¹¹ engl.: Impact Analysis

3.2 URSPRUNG UND VERBREITUNG DURCHGÄNGIGER NACHVERFOLGBARKEIT

Die Ursprünge dieser Methode liegen im Requirements Engineering der Softwareentwicklung, wo bereits 1978 das erste Werkzeug zur Verwaltung von Tracelinks in einem Zeitschriftenaufsatz veröffentlicht wurde [Pierce 1978]. Seitdem wurde dieser, sehr spezifische und nur auf Anforderungen bezogene, Ansatz erweitert und kann zur Nachverfolgung aller, während der modellbasierten Entwicklung von Software entstehender, Artefakte genutzt werden. In diesem Zusammenhang ergeben sich, neben reinen Dokumentationszwecken, weitere Vorteile bei der Validierung und Verifikation der Software gegenüber den Anforderungen oder der Softwarewartung. Darüber hinaus können bei Modelltransformationen Abhängigkeiten automatisch dokumentiert und somit die manuellen Aufwände minimiert werden [Aizenbud-Reshef et al. 2006, S. 515; Winkler und Pilgrim 2010, S. 531].

Aus diesen Gründen wird im Bereich der Softwareentwicklung Traceability als Kennwert für Systemqualität und Prozessreife angesehen und insbesondere für sicherheitskritische Softwarefunktionen durch zahlreiche Standards vorgeschrieben [Aizenbud-Reshef et al. 2006, S. 515]. So gilt bspw. die MIL-STD-498 [Military Standard MIL-STD-498] bei militärischen Auftragsentwicklungen für das Department of Defense der USA und die DIN EN 9115 [DIN EN 9115] bei der Entwicklung von Luftfahrtssystemen. Die Normen ISO 12207 [ISO 12207:2008], ISO 15504 [ISO/IEC 15504-5] und die IEEE Std. 1219 Software Maintenance [IEEE 1219] adressieren allgemein die Softwareentwicklung bzw. die Wartung von Software. Dabei fordern alle genannten Standards ähnliche Formen der Traceability. So soll die Nachverfolgbarkeit zwischen den Kunden-, System- und Softwareanforderungen, der Systemarchitektur sowie dem Softwareentwurf und teilweise sogar dem detaillierten Softwaredesign etabliert und gepflegt werden. Selbst die ISO 9000 [DIN EN ISO 9000], welche beschreibt, welche Anforderungen ein Unternehmen bzgl. des Qualitätsmanagements erfüllen muss, befasst sich mit Traceability. Die geforderte Form ist allerdings deutlich weniger spezifisch als bei den zuvor genannten Standards. Es wird lediglich definiert, dass Dokumentation im Allgemeinen der Rückverfolgbarkeit dient und es ermöglicht werden soll, u. a. den Werdegang und die Verwendung eines Produkts zu verfolgen.

Aufgrund der weit verbreiteten Anwendung gibt es im Bereich der Softwareentwicklung bereits zahlreiche positive Erfahrungen im Umgang mit Traceability. Diese äußern sich bspw. in der Verringerung der Anzahl später Änderungen, da ein Vergessen von Anforderungen vermieden wird, was einen direkten Einfluss auf die Entwicklungskosten hat. Die Vermeidung des Vergessens von Anforderungen ist auch bei sicherheitskritischen Anforderungen von großer Bedeutung, da bei diesen eine Nichterfüllung schwerwiegende Folgen haben kann. Darüber hinaus werden Traceability-Informationen verwendet, um bspw. ein verbessertes Projektmanagement zu realisie-

ren. Dabei werden die Status der Teilentwicklungen über Tracelinks aggregiert und somit der übergreifende Projektfortschritt ermittelt. Zudem werden die Tracelinks genutzt, um bei Änderungen deren Auswirkungen auf andere Entwicklungsbereiche, die Kosten der Änderung abzuschätzen und über die Notwendigkeit und den Umfang der Änderung zu entscheiden [Ramesh et al. 1997, S. 411; Dömges und Pohl 1998, S. 54].

Bei der Entwicklung mechatronischer Systeme findet durchgängige Nachverfolgbarkeit aktuell noch keine flächendeckende Anwendung, obwohl sie gerade bei interdisziplinären Änderungsprozessen großes Potenzial birgt. Vielmehr wird sie sehr punktuell bspw. bei der Entwicklung von Embedded Systems eingesetzt, da Traceability hier bei sicherheitskritischen Funktionen gesetzlich durch die ISO 26262 vorgeschrieben ist (vgl. [ISO 26262-8, S. 8]). Daneben kommt Traceability, im Sinne einer Dokumentation von Abhängigkeiten, zur Beantwortung sehr spezifischer Fragestellungen zum Einsatz, um bspw. eine Modularisierung von Systemen zu planen [Maurer 2007].

3.3 GRUNDBEGRIFFE DURCHGÄNGIGER NACHVERFOLGBARKEIT

Vor dem Hintergrund, dass Traceability Anwendungsfelder in unterschiedlichen Disziplinen hat und in diesen jeweils weiterentwickelt wurde, ist es nicht verwunderlich, dass sich keine übergreifend einheitliche Definition für Traceability herausgebildet hat [Winkler und Pilgrim 2010, S. 530]. Vielmehr gibt es zahlreiche Definitionen, die Aspekte der jeweiligen Forschungsfelder aufgreifen und integrieren.

Eine allgemein gehaltene Definition kann dem *Standard Glossary of Software Engineering Terminology* des Institute of Electrical and Electronics Engineers (IEEE) entnommen werden. Diese besagt, dass Traceability dem Ausmaß entspricht, in dem eine Beziehung zwischen mindestens zwei Produkten des Entwicklungsprozesses etabliert werden kann. Dies gilt insbesondere, wenn diese eine Vorgänger-Nachfolger- oder Eltern-Kind-Beziehung zueinander haben [IEEE 610.12-1990, S. 78]. Edwards und Howell definieren Traceability ebenfalls allgemein als Technik, mit deren Hilfe Beziehungen zwischen Anforderungen, dem Design und der finalen Implementierung des Systems zur Verfügung gestellt werden können [Edwards und Howell 1992, S. 3.7].

Die Definitionen für Traceability aus dem Bereich der Modellbasierten Entwicklung (MBE) (von Software) sind ähnlich gefasst, beziehen sich jedoch spezifisch auf die Gegebenheiten der MBE, bei der ein besonderer Fokus auf Modelltransformationen liegt [Winkler und Pilgrim 2010, S. 534]. So bezeichnen Paige et al. Traceability als Fähigkeit, eindeutig identifizierbare Objekte chronologisch auf eine Art und Weise zu verknüpfen, die Sinn macht. Dabei handelt es sich bei den Objekten vorrangig um

Modelle und Modellelemente, die mit Hilfe verketteter manueller oder automatisierter Operationen ineinander überführt werden [Paige et al. 2008, S. 49]. Aizenbud-Reshef et al. haben ein ähnliches Verständnis und bezeichnen jegliche Beziehung zwischen Artefakten der Software-Entwicklung als Traceability¹². Ausdrücklich schließen die Autoren in dieser Definition Verknüpfungen ein, die durch Transformationen generiert, die basierend auf existierenden Informationen berechnet und die basierend auf Änderungshistorien ermittelt wurden [Aizenbud-Reshef et al. 2006, S. 516].

Im Bereich des (Software) Requirements Engineerings wird Traceability hingegen deutlich spezifischer aus Sicht der Anforderung und der zugehörigen Anforderungsspezifikation definiert. Requirements Traceability bezeichnet in diesem Zusammenhang die Fähigkeit, den Lebenszyklus einer Anforderung zu beschreiben und nachzuvollziehen [Gotel und Finkelstein 1994, S. 97]. Werden dabei auch andere Artefakte als Anforderungsspezifikationen mit einbezogen, wird von Extra-Requirements Traceability gesprochen [Pinheiro 2003, S. 95]. Dabei wird weiter in Forward-Traceability und Backwards-Traceability bzw. Pre- und Post-Requirements-Specification-Traceability unterschieden, wodurch zum Ausdruck gebracht wird, dass die Nachverfolgbarkeit einmal von der Anforderung über beliebige Zwischenprodukte zu den Komponenten des zu entwickelnden Systems und einmal von der Anforderung zu ihren Ursprüngen realisiert wird [Gotel und Finkelstein 1994, S. 97; Pinheiro 2003, S. 93]. Eine Anforderung wird somit als nachverfolgbar bezeichnet, wenn die Quelle, der Grund für die Existenz, die Beziehungen zu weiteren Anforderungen und Artefakten sowie der Status der Anforderung bekannt sind [Sutinen et al. 2000, S. 5].

Obwohl Traceability im Rahmen des Systems Engineerings als wichtig anerkannt wird [Haskins 2006, S. 4.04], hat sich bisher noch keine an die spezifischen Vorgehensweisen des Systems Engineerings angepasste Definition von Traceability herausgebildet. Aus diesem Grund wird in dieser Arbeit die folgende Definition verwendet, die an die vorangegangenen angelehnt wurde:

Traceability ist eine Methode zur Unterstützung der Entwicklung mechatronischer Systeme, bei welcher Abhängigkeiten zwischen den Elementen produktbeschreibender Artefakte explizit dokumentiert werden. Das Ziel der Methode, der Zustand einer durchgängigen Nachverfolgbarkeit zwischen den produktbeschreibenden Artefakten, wird ebenfalls mit dem Begriff Traceability beschrieben. (1)

Bei den erwähnten Tracelinks handelt es sich um Software-Objekte, die je nach Umsetzung bspw. die IDs der voneinander abhängigen Elemente oder Parameter bein-

¹² Im Gegensatz zu den zuvor genannten Definitionen wird hier nicht zwischen Traceability und Tracelink unterschieden.

- Describes-Tracelinks (beschreiben) drücken aus, dass ein Element den Inhalt des anderen näher beschreibt.
- Depends-on-Tracelinks (von etwas abhängen) beschreiben einen Sachverhalt, bei dem ein Element von dem Vorhandensein eines anderen abhängt.
- Validates-Tracelinks (validieren) verbinden zwei Elemente, wenn das eine zur Validierung des anderen vorgesehen ist.

In Bezugnahme auf die in Kapitel 2.1.5 beschriebene Entwicklung eines mechatronischen Außenspiegels werden somit die Funktion „Außenspiegel beheizen“ und die Anforderung „Automatisches Entfernen von Eis vom Außenspiegel innerhalb von $t = 360 \text{ s}$ “ mit Hilfe eines Satisfy-Tracelinks verknüpft, der ausdrückt, dass die Anforderung durch die Funktion erfüllt wird. Dieselbe Anforderung ist zusätzlich über einen Describes-Tracelink mit einem detaillierten Dokument, in dem Verfahren zur Berechnung des Aufheizverhaltens beschrieben sind, verknüpft. Die Unterscheidung der Tracelink-Typen hat bspw. bei einer Auswirkungsanalyse, in der die Folgen der Verringerung der geforderten Zeit in der Anforderung auf $t = 300 \text{ s}$ bestimmt werden sollen, den Vorteil, dass der Analyseumfang deutlich eingeschränkt werden kann. Da beschreibende Dokumente von der Änderung nicht betroffen sind, können diese über die Einschränkung der untersuchten Typen von Tracelinks bei der Analyse ausgeklammert werden. Diese Einschränkung wäre im Fall von untypisierten Tracelinks nicht möglich, da die Anforderung sowohl mit der Funktion als auch mit dem beschreibenden Dokument mit einem Tracelink verknüpft wäre, der lediglich eine grundsätzliche Abhängigkeit ausdrückt, ohne deren Art zu spezifizieren.

Dieses Beispiel zeigt, dass insbesondere bei der Bearbeitung komplexer Artefakte mit vielen Elementen und Abhängigkeiten eine Unterscheidung unterschiedlicher Arten von Tracelinks notwendig ist. Aus Sicht des Autors dieser Arbeit sollte daher die Typisierung zur Steigerung des Nutzens durchgängiger Nachverfolgbarkeit praktiziert werden. Dazu muss jedoch zunächst festgelegt werden, wie die Tracelinks verwendet werden und welche Arten von Tracelinks dementsprechend notwendig sind. Um die Erfassung der Tracelinks übersichtlich zu gestalten, sollten darüber hinaus nur die Tracelink-Arten bei der Erfassung angeboten werden, die für die Kombination der jeweiligen Artefakte zuvor als sinnvoll festgelegt wurden.

Bei der Umsetzung der in Kapitel 5.4 beschriebenen Methode zur effizienten Erfassung von Tracelinks wurde jedoch auf die Typisierung verzichtet. Hintergrund dieser Vereinfachung ist, dass nicht die Verwendung der Tracelinks (für die die Typisierung notwendig ist) im Fokus der Forschung lag, sondern deren effiziente Modellierung. Für die Entwicklung und Evaluation der Methode war eine Unterscheidung unterschiedlicher Arten von Tracelinks nicht notwendig. Allerdings ist eine Erweiterung der Methode um diese Aspekte problemlos möglich und sollte zukünftig angestrebt werden.

Darüber hinaus kann Tracelinks eine Richtung (gerichtet von Element 1 nach Element 2; gerichtet von Element 2 nach Element 1 oder ungerichtet zwischen Element 1 und Element 2) zugewiesen werden, die eine Reihenfolge in Kausalität oder Zeit ausdrückt. Diese Richtung dient jedoch nur als Hinweis für den Anwender, da es für die Auswertung notwendig ist, in beide Richtungen entlang der Links navigieren zu können [Ramesh und Edwards 1993, S. 257; Winkler und Pilgrim 2010, S. 532]. Somit kann trotz vorgegebener Richtung, bspw. von Anforderungen zu den sie erfüllenden Bauteilen, und anders herum navigiert werden.

Weiter besteht die Möglichkeit, den Tracelinks eine Art Gewichtung zuzuordnen, mit deren Hilfe bspw. das Ausmaß des Einflusses der Änderung eines Ausgangselements auf die davon abhängigen Elemente beschrieben werden kann. In diesem Fall spricht man bei der **Art der Traceability** von *quantitativer Traceability*, während die *qualitative Traceability* lediglich die Identifikation betroffener Elemente erlaubt [Sutinen et al. 2002, S. 1]. Ein weiteres Beispiel quantitativer Traceability ist die Verknüpfung von Produkt- und Funktionsstruktur mit gewichteten Tracelinks. Mit Hilfe eines Attributs des Tracelinks kann anschließend abgeschätzt werden, in welchem Ausmaß ein Bauteil an der Erfüllung einer Funktion beteiligt ist. Bei Kenntnis der Kosten der Bauteile können somit die Kosten einer Funktion bestimmt und bspw. für die Kalkulation verwendet werden. Eine Umsetzung dieser quantitativen Traceability findet sich in der Software METUS der Firma ID Consult (siehe auch Kapitel 3.6.3.1). Bei dem zuvor bemühten Beispiel des Außenspiegels sind bspw. neben der Heizspirale und dem An/Aus-Schalter zusätzlich der Kabelstrang (teilweise), die Spiegelhalterung (hinsichtlich der Befestigung der Heizspirale) und Rechenressourcen eines Steuergeräts an der Realisierung der Funktion „Außenspiegel beheizen“ beteiligt (siehe Abbildung 14).

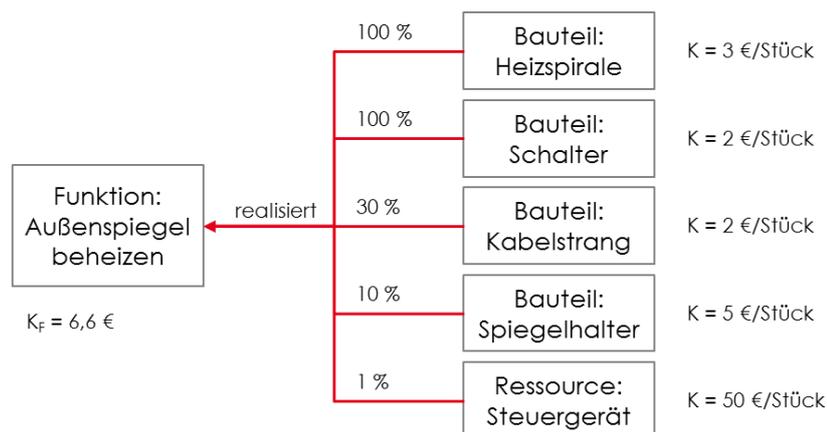


Abbildung 14: Quantitative Traceability am Beispiel des mechatronischen Außenspiegels

Mit Hilfe der gewichteten Tracelinks lassen sich nun die Funktionskosten zu 6,6 € berechnen und ggf. in Kombination mit der Durchführung eines Quality-Function-Deployment (QFD), bei dem die Wichtigkeit der einzelnen Funktionen in Abhängig-

keit der Kundenwünsche bestimmt wird, festgelegt werden, ob die Realisierung unter den gegebenen Umständen lohnenswert ist.

Wie bei der Typisierung von Tracelinks wird auch die Unterscheidung in horizontale und vertikale Traceability in der Forschung ohne Einigung auf ein gemeinsames Verständnis diskutiert. Pfleeger und Bohner sowie Bianchi et al. bezeichnen Traceability dann als vertikal, wenn sie durch Tracelinks zwischen voneinander abhängigen Elementen desselben Artefakts hergestellt wird. Besteht die Abhängigkeit der Elemente hingegen über mehrere unterschiedliche Artefakte hinweg, sprechen sie von einer horizontalen Traceability [Pfleeger und Bohner 1990, S. 323; Bianchi et al. 2000, S. 150]. Nejmech et al. beschreiben die gleichen Kategorien, vertauschen jedoch deren Bedeutungen. Nach ihrer Definition liegt somit vertikale Traceability dann vor, wenn Elemente unterschiedlicher Artefakte mit Tracelinks in Beziehung gesetzt werden und horizontale Traceability, wenn Elemente desselben Artefakts verknüpft werden [Nejmech et al. 1989]. Auch Winkler und Pilgrim unterscheiden zwischen horizontaler und vertikaler Traceability. Im Gegensatz zu den vorangegangenen Definitionen beziehen sie diese jedoch nicht auf Artefakte, sondern auf den zeitlichen Ablauf einer Entwicklung: während Tracelinks zwischen Artefakten derselben Projektphase Teil einer horizontalen Traceability sind, gehören die zwischen Artefakten unterschiedlicher Projektphasen zur vertikalen [Winkler und Pilgrim 2010, S. 533].

Horizontale und vertikale Traceability haben somit grundlegend unterschiedliche Bedeutungen. Die erste – chronologische – bezieht sich auf den zeitlichen Ablauf der Prozessphasen, in denen die Artefakte erstellt werden. Die zweite – artefaktbezogene – behandelt die Frage, ob die Tracelinks innerhalb eines oder zwischen zwei unterschiedlichen Artefakten modelliert werden. Bei letzterer Bedeutung gibt es sogar zwei invertierte Meinungen bzgl. der Ausrichtung von horizontal und vertikal, was sich aus der visuellen Darstellung, wie Pfleeger und Bohner, Bianchi et al. und Nejmech et al. einen Prozess dokumentieren (von links nach rechts bzw. von oben nach unten), ergibt. Aus diesem Grund werden diese Bedeutungen für die Charakterisierung einer Traceability im Rahmen dieser Arbeit getrennt betrachtet und mit **Prozessphasen-Kontext** und **Artefakt-Kontext** bezeichnet [Königs et al. 2012, S. 930].

Der **Prozessphasen-Kontext** beschreibt den Zustand, ob ein Tracelink (2)
Elemente zweier Artefakte der gleichen Phase (*phasenintern*) oder
unterschiedlicher Prozessphasen (*phasenübergreifend*) verbindet.

Somit sind bspw. Tracelinks zwischen Anforderungen einer oder mehrerer Spezifikationen phasenintern, während solche von Anforderungen zu den sie erfüllenden Funktionen der Funktionsstruktur oder Bauteilen der Produktstruktur phasenübergreifend sind.

Der **Artefakt-Kontext** bezieht sich hingegen auf die Artefakt-Zugehörigkeit der in Abhängigkeit zueinander stehenden Elemente. Sind beide in einem Artefakt enthalten, handelt es sich um eine *artefaktinterne*, bei unterschiedlichen Artefakten um eine *artefaktübergreifende* Traceability. (3)

Tracelinks zwischen Anforderungen einer Spezifikation sind somit artefaktintern, während Tracelinks zwischen Anforderungen und den sie erfüllenden Funktionen der Funktionsstruktur artefaktübergreifend sind.

3.4 PHASEN DER DURCHGÄNGIGEN NACHVERFOLGBARKEIT

Durchgängige Nachverfolgbarkeit lässt sich in vier Phasen unterteilen: Planung von Traceability, Erfassung von Tracelinks, Verwendung von Tracelinks und Pflege von Tracelink-Modellen [Winkler und Pilgrim 2010, S. 539]. Im Folgenden werden diese Phasen vorgestellt.

3.4.1 ERFASSUNG VON TRACELINKS

Die Erfassung von Tracelinks beinhaltet die Identifikation von Abhängigkeiten zwischen Elementen unterschiedlicher Artefakte und die Modellierung der entsprechenden Tracelinks. Dabei kann zwischen der Online-Erfassung, bei der die Links (häufig automatisch) als Nebenprodukt der Entwicklung dokumentiert werden, und der Offline-Erfassung, bei der die Links entweder manuell oder automatisch im Anschluss an die Erstellung der Artefakte ermittelt werden, unterschieden werden [Winkler und Pilgrim 2010, S. 539].

Es handelt sich um einen wichtigen und zugleich sehr anspruchsvollen Arbeitsschritt, da alle Methoden, die während der Verwendungsphase angewendet werden (vgl. Kapitel 3.4.2), auf dem Ergebnis dieser Phase aufbauen [Maurer 2007, S. 18 & 37; Biedermaier et al. 2010, S. 309].

Grundsätzlich gilt, dass bei der Tracelink-Erfassung ein möglichst hoher Grad der Automatisierung erreicht werden sollte [Ramesh und Edwards 1993, S. 259; Pinheiro 2003, S. 110]. Die Techniken, die dabei helfen sollen, den Aufwand zu reduzieren, sind heutzutage jedoch hauptsächlich im Bereich der Forschung und weniger in der industriellen Anwendung zu finden (vgl. bspw. [Ramesh und Jarke 2001, S. 39]). Die Ausnahme von dieser Regel stellen Entwicklungsdisziplinen dar, in denen Artefakte automatisch aus anderen generiert und dabei Tracelinks abgeleitet werden können. So lassen sich Tracelinks bspw. in der modellbasierten Softwareentwicklung während Modelltransformationen (bspw. bei der Transformation von UML in Quellcode) ver-

gleichsweise einfach als Nebenprodukt dokumentieren. Alternativ müssen die Inhalte der Artefakte für eine automatisierte Erfassung von Tracelinks analysiert werden, um Abhängigkeiten zwischen deren Elementen zu identifizieren. Dies wird jedoch in der industriellen Anwendung durch die fehlende Formalisierung der Artefakte erschwert [Winkler und Pilgrim 2010, S. 535]. So werden bspw. Anforderungen häufig in Form von Fließtexten beschrieben. In diesen Texten müssen zunächst die Kerninhalte identifiziert werden, um sie dann letztlich mit Elementen anderer Artefakte zu vergleichen. Dies stellt eine weitere Herausforderung dar. Da im Bereich der Entwicklung technischer Systeme sehr verschiedene Artefakte zur Anwendung kommen (siehe Kapitel 2.2), ist ein einfacher Vergleich aufgrund der Verwendung unterschiedlichen Vokabulars nicht immer zielführend. Die Abhängigkeit zwischen der Funktion „Eis entfernen“ des Außenspiegels und dem Bauteil „Heizfolie“ könnte somit durch so einen vergleichenden Algorithmus nicht erkannt werden. Dies führt dazu, dass die manuelle Erfassung der Tracelinks insbesondere im industriellen Umfeld noch sehr weit verbreitet ist und massiv zu dem schlechten Aufwand-Nutzen-Verhältnis der durchgängigen Nachverfolgbarkeit beiträgt. Bei der manuellen Erfassung sollten Tracelinks möglichst von den Entwicklern und nicht von dedizierten Qualitätsteams erfasst werden, da nur sie die Inhalte der Artefakte wirklich verstehen und somit über die Existenz von Abhängigkeiten entscheiden können [Pinheiro 2003, S. 110; Arkley und Riddle 2005, S. 386]. Zusätzlich sollten die Tracelinks nach Möglichkeit direkt bei der Feststellung einer Abhängigkeit und damit bei der Erstellung der Artefakte dokumentiert werden, um Fehler zu vermeiden [Pinheiro 2003, S. 104; Arkley und Riddle 2005, S. 386]. Ist eine direkte Erfassung nicht möglich, sollte zumindest eine möglichst strukturierte Vorgehensweise angewendet werden, um Fehler, wie das Vergessen von Tracelinks, zu vermeiden [Maurer 2007, S. 18].

Methoden, die der Automatisierung, Unterstützung oder Verringerung des Aufwandes bei der Erfassung dienen, lassen sich unter dem Oberbegriff **Modellierungsunterstützung** zusammenfassen. Diese Unterstützung kann z. B. mit Hilfe von *Template*-Ansätzen erfolgen, bei denen die Artefakte zunächst generisch ausgelegt und explizit für die spätere Anpassung vorbereitet werden. Auch die Verwendung von Assistenten (bzw. *Wizards*) ist möglich, die Nutzer durch den Erfassungsprozess führen und den Aufwand u. U. durch Verringerung der zu betrachtenden Elementkombinationen verringert (siehe bspw. Kapitel 5.4). Ansätze, die sich mit der (teil-) automatisierten Identifikation von Abhängigkeiten und der anschließenden Modellierung von Tracelinks befassen, lassen sich in solche, die mit Hilfe von *Regeln*, und solche, die mit Hilfe von *Informationsrückgewinnung* (engl.: Information Retrieval) vorgehen, einteilen [Winkler und Pilgrim 2010, S. 550].

Unabhängig von der Art der Modellierungsunterstützung können immer nur sog. Kandidaten-Links ausgegeben werden, also Vorschläge für Tracelinks, die mit einer

gewissen Wahrscheinlichkeit (die wiederum von der Art der Unterstützung abhängt) zwischen zwei Elementen bestehen. Dabei gibt es keine Garantie, dass die vorgeschlagenen Tracelinks vollständig oder korrekt sind [Egyed et al. 2009, S. 243; Winkler und Pilgrim 2010, S. 551], weshalb letztlich die Entscheidung über die Existenz eines Tracelinks immer beim Entwickler liegen muss [Pinheiro 2003, S. 111; Hayes et al. 2006, S. 10]. Insbesondere der Aspekt der fehlenden Vollständigkeit führt dazu, dass Methoden, die auf Regeln bzw. Informationsrückgewinnung basieren, für die automatisierte Erfassung von Tracelinks ohne menschliche Beurteilung nicht geeignet sind. Deshalb werden sie im Rahmen dieser Arbeit ausschließlich zur Pflege von Tracelink-Modellen (vgl. Kapitel 3.4.3 und 6.1) verwendet, da sie auf „Fehler“ in Form von fehlenden und falschen Tracelinks im Modell hinweisen.

3.4.2 VERWENDUNG VON TRACELINKS

Die Verwendung von Tracelinks stellt den eigentlichen Nutzen der Traceability dar. Dabei gibt es je nach Art des Entwicklungsprojekts unterschiedliche **Zwecke**, für die sich Tracelinks verwenden lassen und nach denen sich die unterschiedlichen Ansätze kategorisieren lassen. So können sie zur *Verifikation* verwendet werden, indem sie eine Unterstützung bei dem Vergleich von ist-Werten eines technischen Systems mit den soll-Werten einer Spezifikation darstellen. Ferner können auf Basis der Tracelinks *Analysen* durchgeführt werden, bei denen bspw. die Auswirkungen einer Änderung bestimmt oder aber Zusammenhänge nachvollzogen werden, um ein besseres Systemverständnis zu erlangen. Darüber hinaus können Tracelinks im weiteren Sinne für die *Synthese* eines Systems verwendet werden, indem bspw. einem Steuergerät Funktionen mit Hilfe von Tracelinks zugewiesen werden. Ein weiterer möglicher Zweck der Traceability ist die *Dokumentation* durchgängiger Nachverfolgbarkeit, u. a. wenn dies durch Standards vorgeschrieben sein sollte (vgl. Kapitel 3.2). Eine ebenfalls praktizierte Projektmanagement-Anwendung im Zusammenhang mit Tracelinks ist ihre Nutzung zur *Fortschrittskontrolle*, bei der bspw. Informationen aggregiert und zu Fortschrittsberichten zusammengefasst werden können [Ramesh und Edwards 1993, S. 258; Lago et al. 2009; Beier et al. 2013, S. 25-26]. Zusätzlich können Tracelinks auch zur *Synchronisation* zweier Artefakte genutzt werden.

3.4.3 PFLEGE VON TRACELINK-MODELLEN

Für die Verwendung der Tracelinks im Sinne von Kapitel 3.4.2 ist eine hohe Qualität des Tracelink-Modells notwendig, da dessen Fehler das Ergebnis der Analysen deutlich verfälschen können [Maurer 2007, S. 94; Biedermann et al. 2009, S. 119; Mäder et al. 2009c, S. 7]. Dabei gibt es zwei mögliche Arten von Fehlern: fälschlicherweise modellierte Tracelinks zwischen Elementen, zwischen denen keine Abhängigkeit existiert

(falsch positive Tracelinks) und fehlende Tracelinks zwischen Elementen, die eine Abhängigkeit zueinander aufweisen (falsch negative Tracelinks) [Hayes et al. 2006, S. 7; Biedermann et al. 2009, S. 119].

Diese Fehler im Tracelink-Modell können auf unterschiedliche Arten entstehen. Zum einen können während der aufwändigen, manuellen Erfassung der Tracelinks Fehler gemacht und so falsch positive und falsch negative Tracelinks dokumentiert werden [Egyed et al. 2009, S. 256; Winkler und Pilgrim 2010, S. 539]. Zum anderen führen auch die Weiterentwicklungen und Änderungen der Artefakte selbst [Sugden und Strens 1996, S. 458; Kotonya und Sommerville 1998] dazu, dass Abhängigkeiten zwischen Elementen hinzukommen oder wegfallen [Cleland-Huang et al. 2003, S. 796; Egyed 2003, S. 116; Egyed et al. 2009, S. 242]. Aus diesem Grund ist es notwendig, Tracelink-Modelle kontinuierlich zu pflegen, um einer stetigen Abnahme der Qualität vorzubeugen [Sutinen et al. 2000, S. 12; Egyed und Grünbacher 2002, S. 163; Mäder et al. 2008, S. 1].

Bei der Pflege werden bestehende Tracelinks auf ihre Korrektheit und Elementepaare, die nicht mit Tracelinks verknüpft sind, auf Abhängigkeiten überprüft. Dabei fällt insbesondere ersteres (die Identifikation von falsch positiven Tracelinks) Anwendern in der manuellen Prüfung leichter, da nur existierende Tracelinks überprüft werden müssen und nicht sämtliche Kombinationen von Elementen, die über keine Tracelinks verfügen [Hayes et al. 2006, S. 7]. Wie die Erfassung von Tracelinks ist somit auch deren Pflege sehr aufwändig [Egyed et al. 2009, S. 242], weshalb eine Softwareunterstützung notwendig ist. Dabei werden, wie in Kapitel 3.4.1 beschrieben, Methoden, die auf Regeln bzw. Informationsrückgewinnung basieren, für die Pflege von Tracelink-Modellen genutzt.

Einen Überblick über den Stand der Technik in diesem Bereich bietet Kapitel 6.1, weshalb an dieser Stelle nicht genauer darauf eingegangen wird.

3.4.4 PLANUNG DURCHGÄNGIGER NACHVERFOLGBARKEIT

Die Planung durchgängiger Nachverfolgbarkeit befasst sich mit Aspekten, die vor Beginn der eigentlichen Entwicklung mechatronischer Systeme durchgeführt werden. Zunächst muss für das jeweilige Projekt entschieden werden, wie umfangreich Tracelinks modelliert und damit zwischen welchen Artefakten Tracelinks erfasst werden sollen. Einerseits wird empfohlen, dass insbesondere bei komplexen mechatronischen Systemen ein umfassender Ansatz gewählt werden sollte, der die Artefakte aller beteiligten Disziplinen berücksichtigt [Ramesh et al. 1997, S. 401]. Andererseits führt dies zu erheblichen Mehraufwänden, die dem Nutzen und den Zielen im Rahmen des Projekts gegenüber gestellt werden müssen. Eine kritische Auseinandersetzung

mit dieser Fragestellung unter der Berücksichtigung von Projektplan, Budget, Entwicklungsstandards, Gesetzen und geplanten Anwendungen sollte somit im Rahmen der Planung der Traceability erfolgen [Dömges und Pohl 1998, S. 54]. Im Anschluss an die Festlegung, welche Artefakte einbezogen werden sollen, muss der Traceability-Ansatz näher detailliert und die zu verwendenden Werkzeuge ausgewählt werden. Letztere lassen sich anhand folgender Kriterien näher charakterisieren und kategorisieren.

Akquisition der Artefakte: In Bezug auf die Akquisition der Artefakte wird zwischen zwei unterschiedlichen Philosophien unterschieden: *integrative* Software-Werkzeuge, die Artefakte aus bestehenden Autorenwerkzeugen importieren bzw. synchronisieren und *nicht-integrative* Werkzeuge, bei denen die Tracelinks nur zwischen werkzeugeigenen Artefakten modelliert werden können.

Duplikation der Artefakte: Bei den akquirierten Artefakten kann es sich um *Originaldaten* handeln, so dass die durchgängige Nachverfolgbarkeit zwischen denselben Datenobjekten hergestellt wird, die auch in den Autorensystemen erstellt und verwendet werden. Alternativ kann eine *Kopie* (eines Teils) der Daten der Autorensysteme im Traceability-Werkzeug gehalten und die Tracelinks auf dieser Basis modelliert werden.

Manipulierbarkeit der Artefakte: Dieses Kriterium beschreibt, ob die gewählten Artefakte mit Hilfe des jeweiligen Traceability-Werkzeugs verändert werden können oder nicht. Einige Werkzeuge erlauben bewusst keine Änderungen der original Daten, um Entwickler dazu zu bringen, die Autorenwerkzeuge zu diesem Zweck zu verwenden.

Unterstützte Arten von Datenstrukturen: Die verwendeten Werkzeuge können weiter nach den Arten von Datenstrukturen, die sie unterstützen, charakterisiert werden. Mögliche Strukturen sind *Listen* oder *Hierarchien*, in denen Entwicklungsdaten häufig vorliegen. Weiter sind jedoch auch *poly-hierarchische* (bei denen Kinderelemente mehr als ein Elternelement besitzen können) oder *netzartige* Strukturen möglich.

Speicherort der Tracelinks: In Abhängigkeit des gewählten Traceability-Werkzeugs kann der Speicherort der Tracelinks zur Charakterisierung herangezogen werden. Tracelinks können entweder *verteilt* in mindestens einem der zu verknüpfenden Artefakte oder separat in einer *zentralen* Datenbank bzw. einem Modell gespeichert werden [Winkler und Pilgrim 2010, S. 537]. In letzterem Fall bezeichnet van Gorp et al. das Modell, in dem diese Tracelinks gespeichert werden, als Traceability-Modell [van Gorp et al. 2006, S. 2]. Betrachtet man dieses im Zusammenhang mit den zugehörigen Artefakten, spricht man von einem integrierten Modell. Da insbesondere die letztgenannte Definition jedoch keine Schlüsse auf den Zusammenhang mit Traceability zulässt, werden im Rahmen dieser Arbeit anschaulichere Namen für die beiden

Modelle verwendet. Ersteres wird Tracelink- und letzteres Traceability-Modell¹³ genannt.

Traceability-Schema: Einige Traceability-Werkzeuge sehen vor, dass im Rahmen der Traceability Planung ein Metamodell spezifiziert wird, welches Tracelinktypen für bestimmte Element- bzw. Artefaktkombinationen vorschreibt [Winkler und Pilgrim 2010, S. 539]. Die Restriktion mit Hilfe dieses Traceability-Schemas reduziert einerseits die Flexibilität bei der Modellierung der Tracelinks, da nicht mehr jedes beliebige Elementepaar verknüpft werden kann, ermöglicht aber andererseits umfassendere Software Unterstützung bei der Erfassung und Interpretation der Tracelinks.

Granularität: Ebenfalls während der Planung muss die notwendige Granularität der zu etablierenden Traceability festgelegt werden. Diese beschreibt die Ebene, auf der die Tracelinks zwischen den unterschiedlichen Artefakten modelliert werden. Diese Ebenen reichen von sehr grober Traceability, die mit Hilfe von Tracelinks zwischen zwei *Artefakten* etabliert wird, über die Verknüpfung von *Elementen* bis zur feinsten Stufe der Traceability, bei der bis auf *Parameterebene* nachverfolgt werden kann. Bei der Planung der Granularität ist zu beachten, dass es sich dabei um einen wichtigen Faktor bei der Balance zwischen ökonomischer und detaillierter, umfassender Traceability handelt. Daher sollte die Granularität immer nur so detailliert wie notwendig gewählt werden (siehe dazu auch Kapitel 5.2.2).

Handhabung von Änderungen: Bei der Änderung von Elementen gibt es unterschiedliche Ansätze, wie die Traceability-Werkzeuge reagieren. Zum einen ist es möglich, dass betroffene verknüpfte Elemente von der Änderung *benachrichtigt* werden. Andererseits können im Fall der *Änderungsübertragung* direkt Veränderungen an den betroffenen Elementen als Reaktion auf die Änderung ausgelöst werden.

Darstellung der Tracelinks: Je nach Traceability-Werkzeug stehen unterschiedliche Arten der Darstellung der Tracelinks zur Verfügung. So können die Elemente der Artefakte bspw. auf die Zeilen und Spalten einer *Matrix* verteilt werden. Eine Abhängigkeit wird mit Hilfe eines Eintrags in der gemeinsamen Zelle der betroffenen Elemente dokumentiert. Ebenfalls möglich ist die Darstellung als *Graph*, bei der Elemente als Knoten und Tracelinks als Kanten dargestellt werden. Weit verbreitet ist die vergleichsweise einfache Visualisierung mit Hilfe von *Referenzen*, die Abhängigkeiten zu anderen Elementen beschreiben.

¹³ Das Traceability-Modell wird so genannt, da es alle zur Etablierung einer Traceability benötigten Modelle beinhaltet.

3.5 EINORDNUNG DER DURCHGÄNGIGEN NACHVERFOLGBARKEIT IN DEN STRATEGISCHEN UNTERNEHMENSKONTEXT

Im Kontext eines Unternehmens bedeutet die Etablierung der durchgängigen Nachverfolgbarkeit zahlreiche Veränderungen, die es einzuordnen gilt. Diese Einordnung wird in diesem Kapitel in Anlehnung an [Winter 2003] sowie an das am Produktionstechnischen Zentrum Berlin (PTZ) vorliegende Verständnis des Unternehmenskontextes beschrieben. Dabei sind die vier unterschiedlichen Sichten *Strategie*, *Prozess*, *Methode* und *Technologie* zur Beschreibung des Unternehmens zu betrachten.

Auf Strategieebene wird dabei auf abstraktem Niveau festgelegt, was die Ziele des Unternehmens sind [Winter 2003, S. 93]. So kann bspw. definiert werden, welches Produkt für welches Marktsegment zukünftig entwickelt und produziert werden soll. Aus Sicht der IT-Strategie wäre in diesem Zusammenhang die Entscheidung für die Einführung eines PDM-Systems oder der Wechsel eines CAD-Systems denkbar. Es wird somit vorgegeben, *WAS* gemacht werden soll [Winter 2003, S. 93].

In diesem Kontext sind in Bezug auf die durchgängige Nachverfolgbarkeit insbesondere drei abstrakte Unternehmensziele zu nennen, die verfolgt werden können:

- *Normkonformität*: Aus strategischer Sicht der Unternehmensführung ist es aus Haftungsgründen essentiell, dass auf Normkonformität bei der Entwicklung mechatronischer Systeme geachtet wird. In den Bereichen der funktionalen Sicherheit wird bereits heute die Nachverfolgbarkeit der Umsetzung sicherheitsrelevanter Anforderungen verlangt (siehe auch Kapitel 2.1.3).
- *Entwicklung des richtigen Produkts*: Um das richtige Produkt zu entwickeln ist die Berücksichtigung aller Kundenanforderungen notwendig. Diese kann mittels durchgängiger Nachverfolgbarkeit nachgewiesen werden und darüber hinaus auf Basis der Tracelinks bspw. Vollständigkeitsanalysen hinsichtlich der Umsetzung von Anforderungen durchgeführt werden.
- *Reduzierung später Änderungen*: Späte Änderungen sind häufig teuer [Eigner und Stelzer 2009, S. 16], da sie eine Überarbeitung bereits entwickelter und evtl. abgesicherter Modelle, Systeme usw. notwendig machen. Mit Hilfe durchgängiger Nachverfolgbarkeit können Inkonsistenzen zwischen Entwicklungsartefakten sofort bei Entstehung identifiziert und behoben werden. Ein spätes Erkennen nicht anforderungsgerechten Systemverhaltens und damit späte Änderungen werden somit vermieden.

Die nächste zu betrachtende Sicht ist die Prozessebene, in der die Prozesse zur Umsetzung, der auf Strategieebene vorgegebenen Ziele, beschrieben werden. Es wird somit vorgegeben, *WIE* etwas realisiert werden soll. Dabei werden drei unterschiedliche Arten von Prozessen unterschieden: Leistungsprozesse (auch Geschäftsprozesse

se), in denen Leistungen erbracht werden, Unterstützungsprozesse, welche die Leistungsprozesse unterstützen, sowie Führungsprozesse, mit deren Hilfe die ersten beiden Arten von Prozessen koordiniert werden [Winter 2003, S. 94]. Auf dieser Ebene werden für die Umsetzung der zuvor genannten strategischen Ziele, und damit der Einführung durchgängiger Nachverfolgbarkeit, die Etablierung neuer und Änderungen bestehender, unterstützender Prozesse notwendig. Neu sind bspw. der Planungsprozess durchgängiger Nachverfolgbarkeit, dessen Inhalte in Kapitel 3.4.4 beschrieben sind, die Erfassung im Fall einer Offline-Erfassung (siehe Kapitel 3.4.1), mit dessen Hilfe die Tracelinks nach Fertigstellung der Artefakte parallel zum eigentlichen Entwicklungsprozess modelliert werden, sowie der Pflegeprozess, mit dem die Qualität des Tracelink-Modells gesichert wird (siehe Kapitel 3.4.3). Anpassungen sind vorrangig bei solchen Leistungs- und Unterstützungsprozessen möglich, die durch die Verwendung von Tracelinks unterstützt werden können. So können bspw. im Änderungsmanagement die modellierten Tracelinks verwendet werden, um Entwicklern die Möglichkeit zu geben, selber Auswirkungsanalysen durchzuführen und somit abschätzen zu können, welche Folgen eine geplante Änderung potenziell hat [Fei et al. 2011]. Eine langwierige Abschätzung durch mehrere Funktionsverantwortliche ist im ersten Schritt nicht notwendig, sondern muss erst bei der Indikation einer erheblichen Auswirkung auf detailliertem Niveau erfolgen.

Auch auf methodischer Ebene ergeben sich durch die Einführung durchgängiger Nachverfolgbarkeit in einem Unternehmen sowohl Anpassungen an existierenden, als auch die Einführung neuer Methoden. So können bereits etablierte Methoden, wie bspw. die Fehlermöglichkeits- und -Einflussanalyse (FMEA) [Beier et al. 2011, S. 464], welche zur Vermeidung von Fehlern verwendet wird, oder das House of Quality (HoQ) [Beier et al. 2013, S. 25], mit dem Kundenwünsche den Eigenschaften eines Systems gegenübergestellt werden, durch die Verwendung von Tracelinks teilautomatisiert durchgeführt werden. Für die neu eingeführten Erfassungs- und Pflegeprozesse sind neue Methoden bereitzustellen, welche die Entwickler bei der Erfassung von Abhängigkeiten unterstützen. Einen Überblick über diese neu einzuführenden Methoden bieten die Kapitel 3.4.1, 3.4.3, 5.1 sowie 6.1.

Um diese Methoden optimal anwenden zu können, sind auf technologischer Ebene Anpassungen an den IT-Systemen notwendig. Diese bestehen einerseits aus der Umsetzung der Methoden mit Hilfe von Wizards, Algorithmen usw. und andererseits aus der Bereitstellung von Schnittstellen, um die Elemente der in unterschiedlichen Werkzeugen verwalteten Artefakte mit Tracelinks verknüpfen zu können. Die tatsächliche Realisierung ist dabei von der Planung durchgängiger Nachverfolgbarkeit abhängig (siehe Kapitel 3.4.4, insbesondere das Kriterium „Akquisition der Artefakte“).

Die Einführung durchgängiger Nachverfolgbarkeit hat somit weitreichende Änderungen auf allen Ebenen des Unternehmens zur Folge. Sind diese jedoch durchgeführt und, insbesondere im Bereich der Erfassung von Tracelinks, Methoden zur Unterstützung der Entwickler etabliert, lässt sich umfangreicher Nutzen aus den modellierten Tracelinks gewinnen. Neben den eingangs erwähnten strategischen Zielen ergeben sich auch Vorteile für Entwickler, die sich im Tagesgeschäft in Form von höherer Effizienz oder Entscheidungssicherheit ausprägen.

3.6 SOFTWARE ZUR ERFASSUNG, PFLEGE UND VERWENDUNG VON TRACELINKS

Im Umfeld des Systems Engineerings gibt es zahlreiche Software-Werkzeuge, die Methoden zur Erfassung, Pflege und Verwendung von Tracelinks informationstechnisch unterstützen. Dabei unterscheiden sich die Ausprägungen der unterschiedlichen Ansätze deutlich. Sie lassen sich, anhand ihrer Herkunft und ihres Zwecks, in vier Kategorien einteilen: Anforderungsmanagement-Software, PLM-Software, Software zur Analyse von Systemstrukturen sowie spezialisierte Traceability-Werkzeuge, in denen Traceability zu Integrationszwecken genutzt wird.

In den folgenden Kapiteln werden ausgewählte Werkzeuge, anhand der zuvor genannten Kategorien strukturiert, kurz vorgestellt und jeweils zum Abschluss mit Hilfe der Kriterien aus den Kapiteln 3.3 und 3.4 in Tabellenform charakterisiert.

3.6.1 ANFORDERUNGSMANAGEMENT-SOFTWARE

Anforderungsmanagement-Software bietet in erster Linie Funktionalitäten zur Erfassung und Verwaltung von Anforderungen. Um in diesem Zusammenhang eine Nachverfolgbarkeit zwischen Anforderungsspezifikationen unterschiedlicher Detaillierungstiefe herstellen zu können, werden vielfach auch Funktionalitäten zur Modellierung von Tracelinks und Durchführung von einfachen Änderungsauswirkungsanalysen bereitgestellt. Im Folgenden werden exemplarisch die beiden Werkzeuge DOORS (siehe Kapitel 3.6.1.1) und Reqify (siehe Kapitel 3.6.1.2) vorgestellt.

3.6.1.1 RATIONAL DOORS

Rational DOORS (Dynamic Object Oriented Requirements System) ist eine Anforderungsmanagement-Software von IBM. Sie basiert auf einer Client-Server-Architektur und ist für die gleichzeitige Verwendung durch mehrere Nutzer ausgelegt [Plette 2009, S. 3; Abma 2009, S. 57]. Die graphische Benutzungsschnittstelle orientiert sich an Textverarbeitungsprogrammen, so dass keine Datenbankankenntnisse für die Anwendung notwendig sind [IBM Corporation 2008; Plette 2009, S. 6]. Durch die Konfigurati-

on von Sichten können die Informationsumfänge an die Bedarfe unterschiedlicher Rollen angepasst werden [Plette 2009, S. 10].

Die Strukturierung der Datenbank erfolgt in sog. Projekten, in denen wiederum sogenannte Formal Modules gespeichert werden [Plette 2009, S. 4]. Diese Module entsprechen unterschiedlichen Spezifikationen innerhalb eines Projekts. Sie sind hierarchisch strukturiert und einzelne Informationen wie Anforderungen, Überschriften, Bilder usw. werden innerhalb dieser Module als Objekte gespeichert. Ihnen werden eindeutige Identifikationsnummern (UID) und optional eine beliebig definierbare Anzahl an Attributen zugewiesen [IBM Corporation 2008; Abma 2009, S. 60]. Dadurch lassen sie sich eindeutig identifizieren, einzeln versionieren [Plette 2009, S. 16] und umfassend beschreiben. Abhängigkeiten zwischen diesen Objekten werden mit gerichteten Tracelinks ausgedrückt [IBM Corporation 2008]. Sie werden im Link Module gespeichert und können beliebige Attribute zugewiesen bekommen. Die Darstellung der Tracelinks erfolgt mithilfe von Pfeildarstellungen für ein- und ausgehende Tracelinks im Textmodus (siehe Abbildung 15), blauen Quadraten im Matrixmodus und Linien im Grafikmodus.

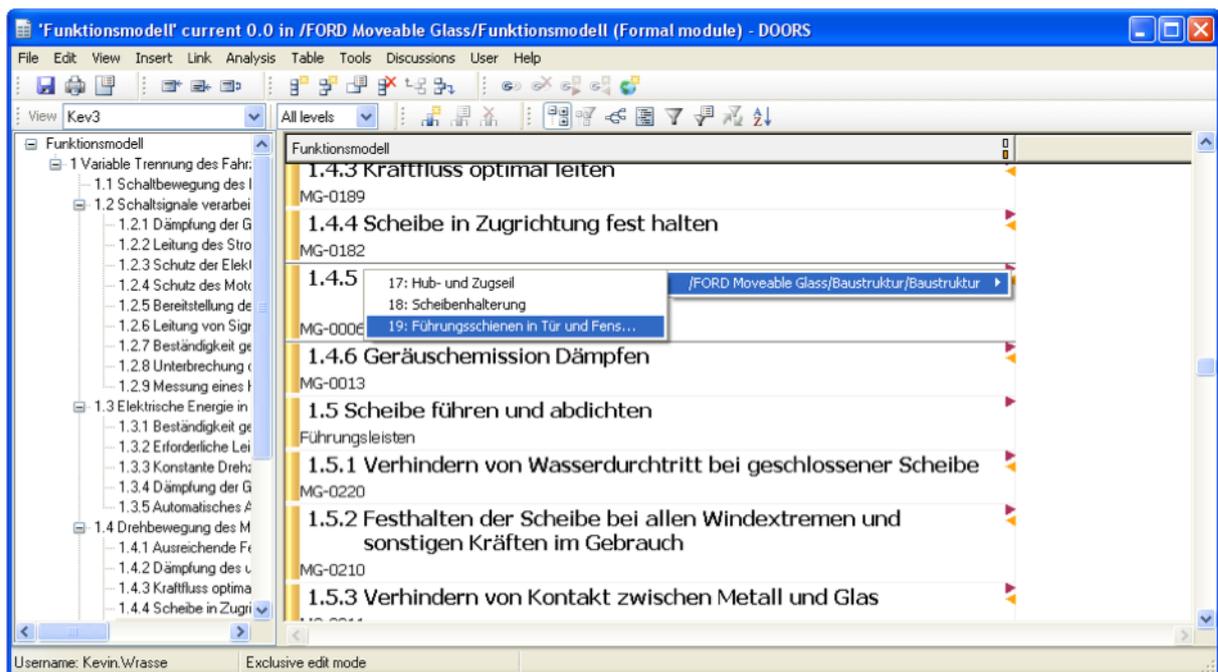


Abbildung 15: Darstellungsform von Tracelinks in DOORS im Textmodus

Das Anlegen von Tracelinks kann manuell oder semi-automatisch durchgeführt werden [Abma 2009, S. 61-62]:

- Manuell per Drag & Drop im Textmodus [IBM Corporation 2008] oder durch Auswählen der entsprechenden Zelle im Matrixmodus.

- Semi-Manuell, indem ein Attribut definiert wird, in dem die UUIDs der zu verlinkenden Objekte eingetragen werden. Die Funktion „Link by attribute“ durchsucht dieses Attribut nach den UUIDs und erstellt die entsprechenden Links im Link Module.

Die erfassten Tracelinks können genutzt werden, um bspw. Auswirkungsanalysen durchzuführen [IBM Corporation 2008]. Zu diesem Zweck werden Filter definiert, die alle ein- und ausgehenden Tracelinks zusammenfassen aus Sicht des jeweiligen Objekts. Darüber hinaus werden bei der Änderung eines Objekts grundsätzlich alle seine Links auf den Status „Suspect“ gesetzt, um Nutzer nach dem Push-Prinzip über die Änderungen zu informieren [IBM Corporation 2008; Abma 2009, S. 67].

In der folgenden Tabelle 1 sind die Eigenschaften von DOORS bzgl. der Etablierung einer durchgängigen Nachverfolgbarkeit zusammengefasst.

Akquisition der Artefakte	nicht-integrativ
Manipulierbarkeit der Artefakte	ja
Duplikation der Artefakte	original Daten
Traceability Schema	nein
Granularität	Elementebene
Arten von Datenstrukturen	Listen, Hierarchien
Modellierungsunterstützung	Regeln
Art der Traceability	qualitativ
Handhabung von Änderungen	Benachrichtigung
Zweck	Analyse, Dokumentation
Darstellung der Tracelinks	Matrix, Graph, Referenz
Prozessphasen-Kontext	phasenintern
Artefakt-Kontext	artefaktintern, artefaktübergreifend
Semantik	nicht typisiert
Speicherort der Tracelinks	zentral

Tabelle 1: Traceability-Eigenschaften von DOORS (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)

3.6.1.2 REQTIFY

Reqtify ist eine Software von Dassault Systèmes, mit deren Hilfe Nachverfolgbarkeit von Anforderungen bis zum Design gewährleistet und Tracelinks visualisiert und analysiert werden können. Zu diesem Zweck werden Daten wie bspw. Spezifikationen oder Modelle aus unterschiedlichen Quellen akquiriert und über unterschiedliche Schnittstellen in Reqtify zur Verfügung gestellt [Geensoft 2010]. Unterstützte Werkzeuge sind bspw. Rational DOORS, RequisitePro, Borland CaliberRM, 3SL Cradle,

Microsoft Word, Microsoft Excel, Adobe PDF, ARTiSAN Studio, Enterprise Architect und Simulink [Geensoff 2010].

Reqtify selbst verfügt über eine Projektdatei bzw. -datenbank, in der alle importierten Dateien referenziert und projektspezifische Daten gespeichert werden. Der grundlegende Unterschied von Reqtify im Vergleich zu den anderen vorgestellten Werkzeugen ist, dass die Tracelinks nicht in Reqtify erstellt oder gespeichert werden [Pohl und Scheel 2004, S. 9]. Diese werden, als Referenzen in den originalen Dokumenten, mit Hilfe sog. Tagger gesetzt und von Reqtify lediglich ausgelesen und interpretiert.

Um die importierten Dokumente zueinander in Kontext zu setzen, werden die Abhängigkeiten der Dokumente zu Beginn eines Projekts im Konfigurationseditor modelliert und somit ein Traceability-Schema aufgebaut. Dieses beschreibt bspw., dass die funktionalen Anforderungen einer Spezifikation durch ein Simulink-Modell erfüllt werden sollen [Pohl und Scheel 2004, S. 11].

Für die Analyse stehen drei unterschiedliche Modi zur Verfügung. Im Coverage-Modus wird basierend auf dem im Konfigurationseditor definierten Traceability-Schema überprüft, ob die zugehörigen Elemente der Dokumente bereits über Tracelinks verfügen und somit laut Definition von Reqtify adressiert sind. Der Auswirkungsanalyse-Modus ermöglicht ausgehend von einem ausgewählten Element die Abschätzung von Auswirkungen durch Darstellung der verknüpften Elemente. Der grafische Modus steigert das allgemeine Verständnis über die Abhängigkeiten durch die gleichzeitige Darstellung ausgewählter Elemente und deren Tracelinks als Linien [Pohl und Scheel 2004, S. 13-16].

Darüber hinaus verfügt Reqtify über eine Funktion für die Erstellung von frei definierbaren Reports, die für die Zertifizierung eines entwickelten Systems (siehe Kapitel 3.2) notwendig sind. So lassen sich bspw. Traceability-Matrizen, als Übersicht miteinander in Beziehung stehender Anforderungen, oder Coverage Reports, denen die Erfüllung der Anforderungen entnommen werden kann, erzeugen [Pohl und Scheel 2004, S. 22-23].

In der folgenden Tabelle 2 sind die Eigenschaften von Reqtify bzgl. der Etablierung einer durchgängigen Nachverfolgbarkeit zusammengefasst.

Akquisition der Artefakte	integrativ
Manipulierbarkeit der Artefakte	nein
Duplikation der Artefakte	original Daten
Traceability Schema	ja
Granularität	Elementebene
Arten von Datenstrukturen	Listen, Hierarchien

Modellierungsunterstützung	keine
Art der Traceability	qualitativ
Handhabung von Änderungen	Benachrichtigung
Zweck	Analysen, Dokumentation, Fortschrittskontrolle
Darstellung der Tracelinks	Matrix, Graph, Referenz
Prozessphasen-Kontext	phasenintern, phasenübergreifend
Artefakt-Kontext	artefaktübergreifend
Semantik	typisiert
Speicherort der Tracelinks	verteilt

Tabelle 2: Traceability-Eigenschaften von Reqify (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)

3.6.2 PLM-SYSTEME

PLM-Systeme stellen eine Weiterentwicklung von PDM-Systemen um das Lifecycle- und das Anforderungsmanagement dar [Eigner und Stelzer 2009, S. 38]. Sie dienen damit nicht nur der Verwaltung reiner Produktdaten sondern darüber hinaus anderer Artefakte wie bspw. Anforderungsspezifikationen. Im Folgenden werden die PLM Systeme V6 von Dassault Systèmes (siehe Kapitel 3.6.2.1) sowie Teamcenter von Siemens PLM Software (siehe Kapitel 3.6.2.2) näher beschrieben und hinsichtlich ihrer Funktionalitäten zur Herstellung einer durchgängigen Nachverfolgbarkeit analysiert.

3.6.2.1 V6-SUITE (R2011x)

Bei der V6-Suite von Dassault Systèmes handelt es sich um eine Systemlandschaft, die auf der RFLP-Methodik basiert. Diese Methodik wurde laut Bornat und Msadek in Anlehnung an Systems Engineering Vorgehensmodelle (siehe Kapitel 2.1.2) bzw. die VDI 2206 (siehe Kapitel 2.1.1) entwickelt [Bornat und Msadek 2011] und erhielt ihren Namen durch die Zusammensetzung der Anfangsbuchstaben der vier relevantesten Prozessschritte (bzw. Artefakte), die während der Entwicklung durchlaufen werden: R (Requirements), F (Functional), L (Logical), P (Physical Design). Dabei werden die Anforderungshierarchien in ENOVIA verwaltet und können darüber hinaus in CATIA visualisiert werden. Die Modellierung von Funktionsstrukturen, Verhaltensmodellen sowie die Detailentwicklung (zumindest im Fall der mechanischen Konstruktion) erfolgen in CATIA [Dassault Systèmes 2013].

Diese Artefakte bauen aufeinander auf, und Abhängigkeiten zwischen ihnen können mithilfe von typisierbaren Tracelinks explizit dokumentiert werden. So können bspw. Anforderungen und Funktionen mit „Implement Relations“ miteinander verknüpf

werden, um auszudrücken durch welche Funktion eine Anforderung realisiert wird [Pfeifer et al. 2012]. Die Erstellung der Tracelinks erfolgt per Drag-and-Drop.

Eine Zusammenstellung dieser artefaktübergreifenden Tracelinks kann als sog. Traceability-Report ausgegeben werden. Dabei werden für wählbare Artefakt-Paarungen (im vorangegangenen Beispiel wären dies Anforderungs- und Funktionsartefakt) alle modellierten Tracelinks in Listenform ausgegeben.

In der folgenden Tabelle 3 sind die Eigenschaften der V6-Suite bzgl. der Etablierung einer durchgängigen Nachverfolgbarkeit zusammengefasst.

Akquisition der Artefakte	integrativ
Manipulierbarkeit der Artefakte	ja
Duplikation der Artefakte	original Daten
Traceability Schema	ja
Granularität	Elementebene
Arten von Datenstrukturen	Listen, Hierarchien, netzartige Strukturen
Modellierungsunterstützung	keine
Art der Traceability	qualitativ
Handhabung von Änderungen	Benachrichtigung
Zweck	Verifikation, Analyse, Synthese, Dokumentation, Fortschrittskontrolle, Synchronisation
Darstellung der Tracelinks	Referenz
Prozessphasen-Kontext	phasenintern, phasenübergreifend
Artefakt-Kontext	artefaktintern, artefaktübergreifend
Semantik	Typisiert und nicht-typisiert
Speicherort der Tracelinks	zentral

Tabelle 3: Traceability-Eigenschaften der V6-Suite (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)

3.6.2.2 TEAMCENTER

Teamcenter von Siemens PLM Software verfügt ebenfalls über Funktionen zur Verknüpfung von Objekten, die im PLM -System verwaltet werden. Dabei ist insbesondere das Modul „Systems Engineering and Requirements Management“ zu erwähnen, mit dem Tracelinks zwischen Anforderungen, funktionalen Strukturen, Verhaltensmodellen und Produktstrukturen modelliert werden können (siehe Abbildung 16). Darüber hinaus ist es möglich auch andere Artefakte, die in externer Software modelliert werden, nachzuverfolgen. Über Schnittstellen können diese Artefakte auf Elementebene in Teamcenter integriert und mit Tracelinks verknüpft werden.

Zur Erstellung eines gerichteten Tracelinks wird zunächst das Ausgangselement ausgewählt und über eine Menüauswahl das Endelement identifiziert. Die Tracelinks können typisiert bis auf Parameterlevel angelegt und zentral in Teamcenter gespeichert werden [Königs 2012, S. 46-47].

Mit Hilfe der Traceability-View können die Tracelinks elementbasiert ausgewertet werden. Dabei wird ein Element (z. B. eine Anforderung) ausgewählt und für diese im sog. Complying-object-pane alle verknüpften Elemente ausgegeben. Alternativ ist es möglich, eine Traceability-Matrix für die Visualisierung der Abhängigkeiten zwischen Anforderungen zu verwenden.

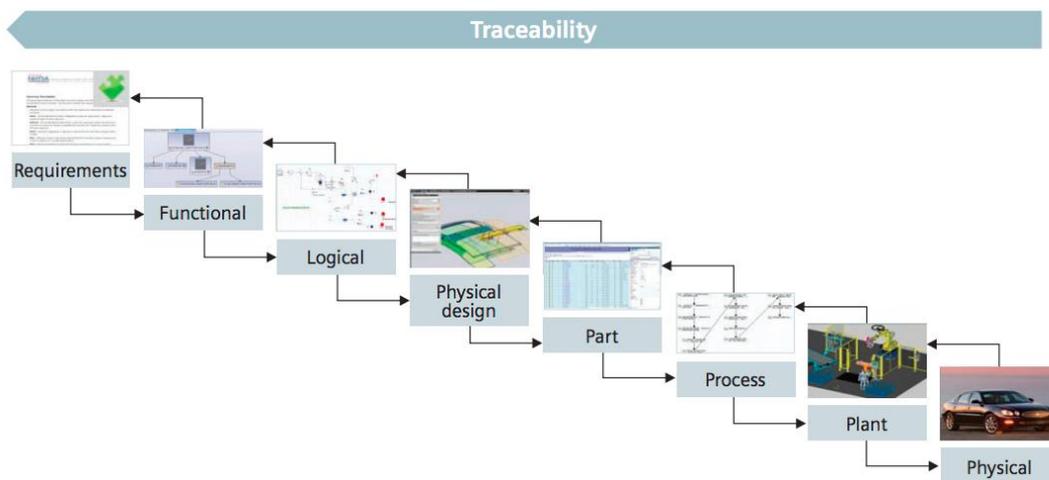


Abbildung 16: Theoretisches Traceability-Schema in Teamcenter [Siemens PLM Software 2011]

Ähnlich der RFLP-Methodik bietet Siemens PLM Software zudem den „Mechatronics Concept Designer“ als integrierten Ansatz zur Systementwicklung an. Dieser setzt auf Teamcenter und der CAD Software NX auf und bietet Funktionalitäten zur Verwaltung und Verknüpfung von textuellen Anforderungen, Funktionsstrukturen, physikbasierten Verhaltensmodellen und 3D-Geometrien [Siemens PLM Software 2010].

In der folgenden Tabelle 4 sind die Eigenschaften von Teamcenter bzgl. der Etablierung einer durchgängigen Nachverfolgbarkeit zusammengefasst.

Akquisition der Artefakte	integrativ
Manipulierbarkeit der Artefakte	ja
Duplikation der Artefakte	original Daten
Traceability Schema	nein
Granularität	Parameterebene
Arten von Datenstrukturen	Listen, Hierarchien, netzartige Strukturen
Modellierungsunterstützung	keine
Art der Traceability	qualitativ

Handhabung von Änderungen	Benachrichtigung, Änderungs-Übertragung
Zweck	Verifikation, Analysen, Synthese, Dokumentation, Fortschrittskontrolle, Synchronisation
Darstellung der Tracelinks	Referenz
Prozessphasen-Kontext	phasenintern, phasenübergreifend
Artefakt-Kontext	artefaktintern, artefaktübergreifend
Semantik	nicht-typisiert
Speicherort der Tracelinks	zentral

Tabelle 4: Traceability-Eigenschaften von Teamcenter (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)

3.6.3 ANALYSE-SOFTWARE

Die im Folgenden vorgestellte Analyse-Software basiert auf der Definition und Analyse von Tracelinks. Sowohl in METUS (siehe Kapitel 3.6.3.1) als auch in LOOME0 (siehe Kapitel 3.6.3.2) werden die Abhängigkeiten innerhalb von Systemen und zwischen deren beschreibenden Artefakten zu dem Zweck abgebildet, unterschiedliche Analysen durchführen zu können. Auf diese Analysen wird folgend näher eingegangen.

3.6.3.1 METUS

METUS ist eine Analyse-Software, die von ID-Consult gemeinsam mit der Daimler AG entwickelt wurde. Sie unterstützt alle Schritte einer zugehörigen Entwicklungsmethodik (System Design Method), deren Ziel die Unterstützung der Entwicklung modularer Produktarchitekturen ist (siehe Abbildung 17)¹⁴ [ID-Systems GmbH 2013].

Erster Schritt dieser Methode ist die Identifikation relevanter Anforderungen und deren Erfassung in METUS. Zu diesem Zweck können auch bestehende Anforderungslisten über XML oder aus Microsoft Excel importiert werden. Anschließend werden notwendige Gesamtfunktionen aus den Anforderungen abgeleitet und in ihre Teilfunktionen heruntergebrochen. Durch die Verknüpfung der Funktionen mit den Anforderungen per Drag-and-Drop wird die Traceability zwischen den Artefakten hergestellt. Den heruntergebrochenen Funktionen werden im nächsten Schritt Bauteile zugeordnet, welche wiederum zu Baugruppen oder Modulen gruppiert werden können. Wie bei den Anforderungen ist es auch bei den Bauteilen möglich, auf bestehende Stücklisten in SAP oder Excel zuzugreifen. Zusätzlich ermöglicht die Integration mit Teamcenter (Siemens PLM) den direkten Zugriff auf die dort verwalteten Produktdaten [Tretow et al. 2008].

¹⁴ Im Folgenden wird nur auf die unter den Gesichtspunkten der Traceability relevanten Schritte dieser Methodik eingegangen.

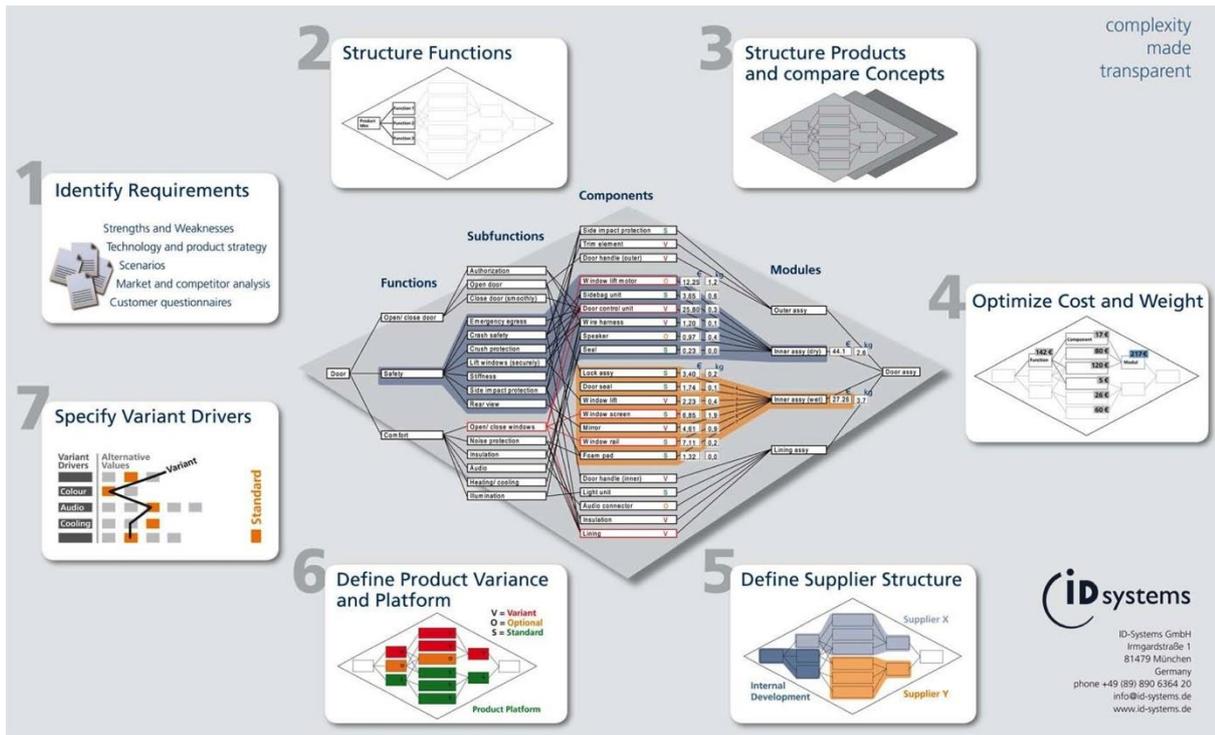


Abbildung 17: Die METUS-Methode für das Systemdesign [ID-Systems GmbH 2013]

Allen in METUS verwalteten Elementen können beliebige Attribute zugewiesen werden. So können bspw. die Herstellungs- oder Bezugskosten der unterschiedlichen Bauteile als Attribute verwaltet werden. Über die Gewichtung der Tracelinks zwischen diesen Bauteilen und deren Funktionen kann dann erfasst werden, wie die Bauteile zur Realisierung der Funktionen beitragen¹⁵. Ziel dieser prozentualen Gewichtung ist es, die Kosten einzelner Funktionen zu ermitteln und somit teure Einzelfunktionen bzw. evtl. Optimierungspotenziale identifizieren zu können.

Zusätzlich sieht die Methodik die Optimierung der Produkte hinsichtlich ihrer Modularisierung vor. In diesem Zusammenhang kommen unterschiedliche Analyseverfahren zum Einsatz. U. a. können die Tracelinks zwischen Funktionen und Bauteilen genutzt werden, um die Modularisierung so zu wählen, dass Funktionen immer nur von einem Modul erfüllt werden [Tretow et al. 2008].

In der folgenden Tabelle 5 sind die Eigenschaften von METUS bzgl. der Etablierung einer durchgängigen Nachverfolgbarkeit zusammengefasst.

Akquisition der Artefakte	nicht-integrativ
Manipulierbarkeit der Artefakte	Nein
Duplikation der Artefakte	kopierte Daten

¹⁵ Die Traceability, die durch diese gewichteten Tracelinks etabliert wird ist somit quantitativ (vgl. Kapitel 3.3).

Traceability Schema	ja
Granularität	Elementebene
Arten von Datenstrukturen	Listen, Hierarchien
Modellierungsunterstützung	keine
Art der Traceability	quantitativ
Handhabung von Änderungen	keine
Zweck	Analysen
Darstellung der Tracelinks	Matrix, Graph
Prozessphasen-Kontext	phasen-übergreifend
Artefakt-Kontext	artefaktübergreifend
Semantik	nicht-typisiert
Speicherort der Tracelinks	zentral

Tabelle 5: Traceability-Eigenschaften von METUS (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)

3.6.3.2 LOOME0

Bei LOOME0 der Teseon GmbH handelt es sich um ein Software-Werkzeug zur Analyse von Systemstrukturen. Zu diesem Zweck bilden unterschiedliche Matrizen die Kernelemente dieser Software (siehe Abbildung 18).

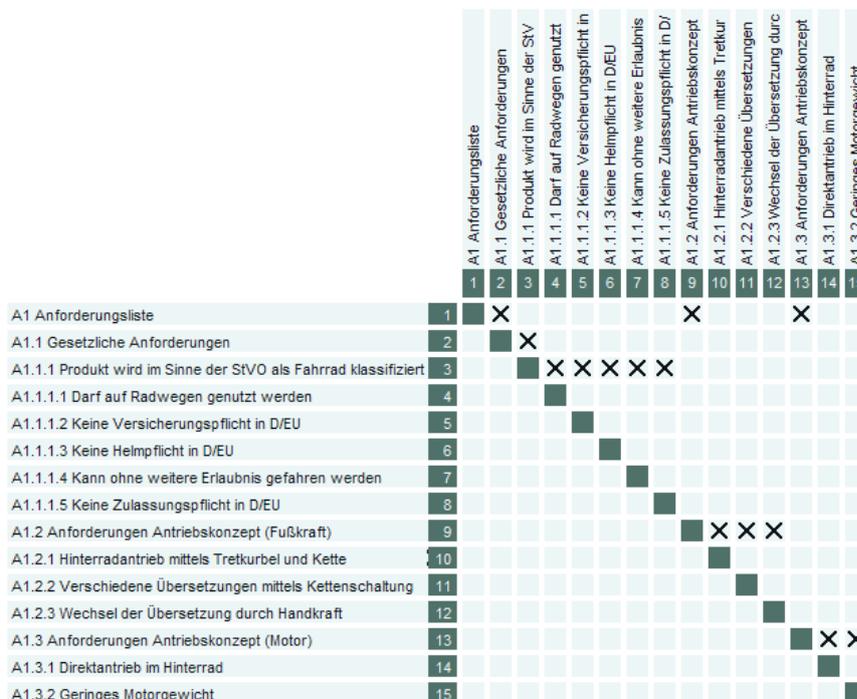


Abbildung 18: Matrizen als Kernelement der Software LOOME0 [Teseon GmbH 2012]

In diesen Matrizen werden die Abhängigkeiten in einem oder zwischen mehreren Artefakten quantitativ oder qualitativ dokumentiert. Zusätzlich können diese Zusam-

menhänge auch mit Graphen dargestellt werden. Diese sind selbstordnend realisiert und erleichtern so bspw. das Erkennen von zusammengehörigen Bauteilen, ähnlichen Varianten oder sich gegenseitig beeinflussenden Organisationseinheiten [Teseon GmbH 2012].

Auf dieser Datenbasis bietet LOOME0, unter Verwendung graphentheoretischer Ansätze, unterschiedliche Analysemöglichkeiten zur Optimierung von Systemstrukturen. So können bspw. aus direkten Abhängigkeiten in den DMMs indirekte Abhängigkeiten in den DSMs abgeleitet oder, durch die Neusortierung der Zeilen und Spalten der Matrizen, stark vernetzte Bereiche in Systemen identifiziert werden, die sich potenziell zur Modularisierung eignen. Die Definition dieser sog. "Filter" erfolgt dabei über die Kombination vorgefertigter Analysebausteine [Teseon GmbH, 2012].

Um auf existierenden Daten aufbauen bzw. die Analyseergebnisse weiterverwenden zu können, stehen ein Import für Excel- und Text-Dateien sowie Exports in die Formate *.png, *.txt und *.svg zur Verfügung.

In der folgenden Tabelle 6 sind die Eigenschaften von LOOME0 bzgl. der Etablierung einer durchgängigen Nachverfolgbarkeit zusammengefasst.

Akquisition der Artefakte	nicht-integrativ
Manipulierbarkeit der Artefakte	nein
Duplikation der Artefakte	kopierte Daten
Traceability Schema	nein
Granularität	Elementebene
Arten von Datenstrukturen	Listen, Hierarchien
Modellierungsunterstützung	keine
Art der Traceability	qualitativ, quantitativ
Handhabung von Änderungen	Benachrichtigung
Zweck	Analysen, Dokumentation
Darstellung der Tracelinks	Matrix, Graph
Prozessphasen-Kontext	phasenintern, phasenübergreifend
Artefakt-Kontext	artefaktintern, artefaktübergreifend
Semantik	nicht-typisiert
Speicherort der Tracelinks	zentral

Tabelle 6: Traceability-Eigenschaften von LOOME0 (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)

3.6.4 INTEGRATIONSSOFTWARE

Integrationssoftware verfolgt das Ziel, Artefakte unterschiedlicher Autorenwerkzeuge zu integrieren, um so die Abhängigkeiten explizit abbilden zu können. Dies wird bei ToolNet (siehe Kapitel 3.6.4.1) und dem ModellTracer (siehe Kapitel 3.6.4.3) umgesetzt. Auch bei ConWeaver (siehe Kapitel 3.6.4.2) handelt es sich um eine Integrationssoftware, die Inhalte aus unterschiedlichen Datenquellen akquiriert. Allerdings ist hier die Zielstellung nicht die explizite Modellierung von Tracelinks, sondern die übergreifende semantische Suche. Da die dafür verwendeten Mechanismen aus Sicht der durchgängigen Nachverfolgbarkeit jedoch sehr interessant sind, wird auch diese Software im Folgenden kurz vorgestellt und charakterisiert.

3.6.4.1 TOOLNET

Bei ToolNet handelt es sich um eine Integrationssoftware, die in einem Verbundprojekt von DaimlerChrysler und EADS entwickelt wurde [Altheide et al. 2002, S. 53]. Das Konzept von ToolNet sieht vor, im Entwicklungsprozess verwendete Autorenwerkzeuge über einen Backbone zu koppeln und somit eine Integration von Funktionen und Daten zu realisieren [van Gorp et al. 2006; Rosenauer 2008, S. 97]. Dabei agiert ToolNet als verknüpfendes Element für die Software-Werkzeuge, die mit Hilfe von spezifischen Adaptern integriert werden. Mit Hilfe dieser Adapter ist es möglich, Anfragen über den Backbone an andere Applikationen zu senden und Teile von deren bzw. ToolNet-spezifische Services zu verwenden.

Ziel dieser Integration ist es, Funktionalitäten für die Etablierung einer durchgängigen Nachverfolgbarkeit bereitzustellen. Elemente der Artefakte in den verschiedenen angebotenen Werkzeugen können manuell mit sog. ToolNet-Relationen verknüpft werden (siehe Abbildung 19). Zu diesem Zweck werden die beiden eindeutigen Objekt-IDs der zu verknüpfenden Elemente sowie Metadaten in dem ToolNet-Relation-Repository gespeichert. Bei den erwähnten Metadaten handelt es sich bspw. um den Autor, das Erstellungsdatum oder den Tracelink-Typ [Rosenauer 2008, S. 100].

Durch den Ansatz, die Elemente der verschiedenen Software-Werkzeuge nicht zu kopieren sondern zu referenzieren, werden Inkonsistenzen vermieden. Darüber hinaus gibt es die Möglichkeit sog. ChangeEvents für Tracelinks zu definieren. Änderungen eines Elements werden dabei detektiert und das verknüpfte Element informiert. Weiter ist es möglich, direkte Anpassungen der verknüpften Elemente durchzuführen, um die Konsistenz zwischen den verknüpften Elementen wiederherzustellen [Rosenauer 2008, S. 105].

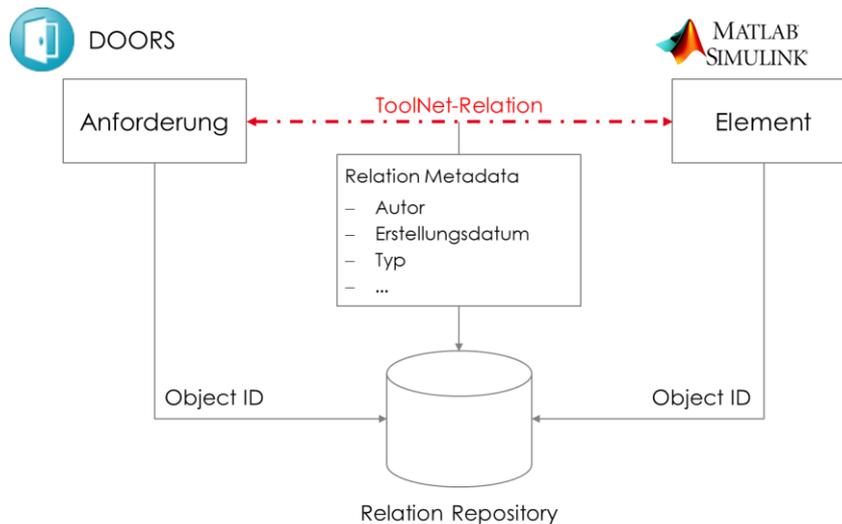


Abbildung 19: Tracelinks in ToolNet (in Anlehnung an [Rosenauer 2008, S. 100])

Zum Verlinken der Elemente muss die eigentliche Arbeitsumgebung nicht verlassen werden. So wird bspw. bei DOORS durch den ToolNet-Adapter das Kontextmenü um einen Eintrag zur Generierung eines Tracelinks erweitert. Bei Selektion des Startelements und Auswahl des Kontextmenüs in DOORS öffnet sich eine ToolNet-Komponente, in der das Zielelement gewählt und der Tracelink bestätigt werden kann [Rosenauer 2008, S. 105-107].

In der folgenden Tabelle 7 sind die Eigenschaften von ToolNet bzgl. der Etablierung einer durchgängigen Nachverfolgbarkeit zusammengefasst.

Akquisition der Artefakte	integrativ
Manipulierbarkeit der Artefakte	nein
Duplikation der Artefakte	original Daten
Traceability Schema	nein
Granularität	Elementebene
Arten von Datenstrukturen	Listen, Hierarchien, netzartige Strukturen
Modellierungsunterstützung	keine
Art der Traceability	qualitativ
Handhabung von Änderungen	Benachrichtigung
Zweck	Analysen, Dokumentation, Synchronisation
Darstellung der Tracelinks	-
Prozessphasen-Kontext	phasenintern, phasenübergreifend
Artefakt-Kontext	artefaktübergreifend
Semantik	typisiert
Speicherort der Tracelinks	zentral

Tabelle 7: Traceability-Eigenschaften von ToolNet (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)

3.6.4.2 CONWEAVER

Bei ConWeaver handelt es sich nicht um eine Software zur Etablierung einer durchgängigen Nachverfolgbarkeit im eigentlichen Sinne, sondern um eine semantische Suchmaschine (siehe Abbildung 20) [Fellner et al. 2008, S. 205]. Aus Sicht der Traceability ist sie trotzdem erwähnenswert, da die Kanten des semantischen Netzes, welches zwischen verschiedenen Unternehmensdaten aufgebaut wird, letztlich Tracelinks zwischen Elementen unterschiedlicher Artefakte sind. Vor diesem Hintergrund sind insbesondere die Algorithmen zur teilautomatisierten Erstellung des semantischen Netzes von Interesse und werden deshalb im Folgenden näher beleuchtet.

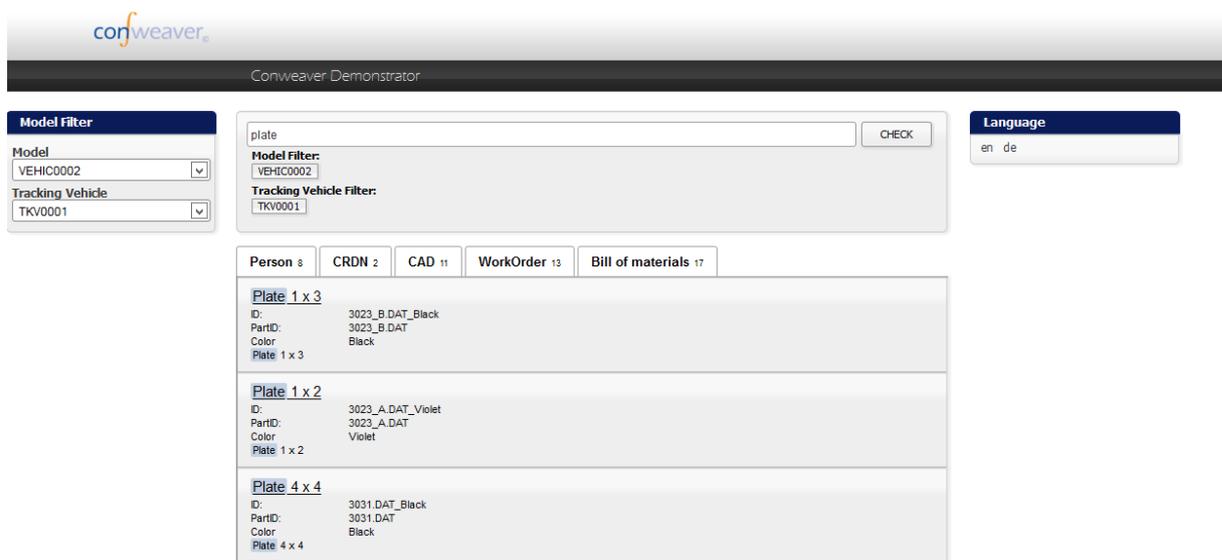


Abbildung 20: Darstellung der Suchergebnisse von ConWeaver (strukturiert nach Ergebnislisten) [Abbildung mit freundlicher Genehmigung zur Publikation von der ConWeaver GmbH (www.conweaver.de) bereitgestellt]

Das Wissensnetz, welches durch ConWeaver aufgebaut wird, besteht aus drei Schichten: Konzept-, Themen- und Wissensnetz. Das Konzeptnetz ist hierarchisch strukturiert und besteht aus sog. Konzepten. Diese Konzepte ermöglichen eine abstrakte Ordnung von Themenbereichen, indem sie untereinander verknüpft werden. So werden bspw. die Konzepte *Krankheit* und *Symptom* mit der Verknüpfung *hatSymptom* verknüpft. Diesen Konzepten werden individuelle Ausprägungen (sog. Individuen) zugeordnet. Das Individuum *Grippe* wird somit dem Konzept *Krankheit* zugewiesen. Zusätzlich gibt es das Themennetz, in dem das Themenvokabular mit sprachlichen und inhaltlichen Zusammenhängen enthalten ist [Dirsch-Weigand et al. 2006, S. 3-4].

Bei der Erstellung des Wissensnetzes muss zunächst das Konzeptnetz manuell erstellt werden. Die individuellen Ausprägungen können hingegen aus existierenden Datenbanken bezogen werden. Es muss lediglich einmalig ein Mapping hergestellt werden, um die Individuen und ihre Attribute (z. B. Preise) korrekt zuordnen zu können. Das

Themenvokabular wird hingegen vollautomatisch aus bestehenden Dokumenten herausgefiltert. Bei dieser Indizierung von Texten kommen Analyseworkflows zum Einsatz, die anwendungsspezifisch konfiguriert werden können. Insgesamt existieren über 200 verschiedene Analyseoperatoren, mit denen nicht nur nach Zeichenfolgen, sondern auch nach der inhaltlichen Bedeutung von Wörtern gesucht werden kann. Diese Analyseoperatoren stammen u. a. aus den Bereichen Textprocessing, Informationsextraktion, Klassifikation und Ähnlichkeit, automatische Übersetzung oder Nutzeranalyse [Dirsch-Weigand et al. 2006, S. 5-10; Fellner et al. 2008, S. 206]. Darin enthalten ist auch die sog. Stammformrückführung, wie sie auch in der Methode TraceLegacy zum Einsatz kommt (siehe Kapitel 6.4).

Damit eignet sich ConWeaver sowohl für strukturierte, als auch für nicht strukturierte Daten, die über eine XML-Schnittstelle aus vielen unterschiedlichen Datenquellen bezogen werden können [Fellner et al. 2008, S. 206].

In der folgenden Tabelle 8 sind die Eigenschaften von ConWeaver bzgl. der Etablierung einer durchgängigen Nachverfolgbarkeit zusammengefasst.

Akquisition der Artefakte	integrativ
Manipulierbarkeit der Artefakte	nein
Duplikation der Artefakte	original Daten
Traceability Schema	trifft nicht zu
Granularität	Elementebene
Arten von Datenstrukturen	Listen, Hierarchien, Netzartige Strukturen
Modellierungsunterstützung	Regelbasiert, Informationsrückgewinnung
Art der Traceability	qualitativ
Handhabung von Änderungen	trifft nicht zu
Zweck	trifft nicht zu
Darstellung der Tracelinks	trifft nicht zu
Prozessphasen-Kontext	phasenintern, phasenübergreifend
Artefakt-Kontext	artefaktintern, artefaktübergreifend
Semantik	typisiert
Speicherort der Tracelinks	zentral

Tabelle 8: Traceability-Eigenschaften von ConWeaver (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)

3.6.4.3 MODELTRACER

Der ModelTracer (vormals ISyFMU) wurde in Kooperation des Fraunhofer IPK, der TU Berlin und der InMediasP GmbH während des Verbundprojekts ISYPROM¹⁶ entwickelt. Dieses prototypische Traceability-Werkzeug dient als Plattform zur Evaluierung verschiedener Methoden, darunter auch EcoTracing (siehe Kapitel 5.4), TraceEvaluation (siehe Kapitel 6.3) und TraceLegacy (siehe Kapitel 6.4). Im Folgenden wird die grundsätzliche Funktionsweise des ModelTracers, die für das Verständnis der später vorgestellten Methoden notwendig ist, vorgestellt. Eine ausführliche Übersicht, insbesondere über Details des Datenmodells und die Implementierung, ist in [Bornath 2011b] zu finden.

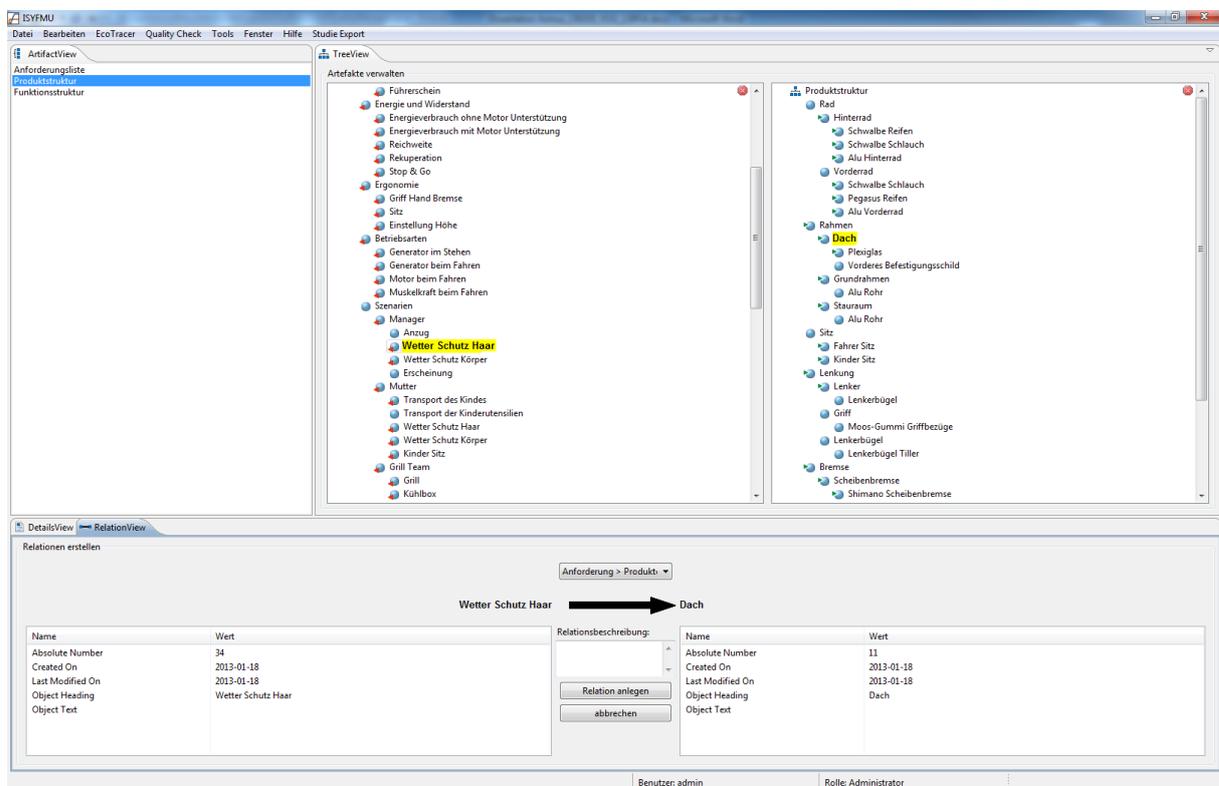


Abbildung 21: Anlegen eines Tracelinks im ModelTracer

Die Artefakte, zwischen denen eine durchgängige Nachverfolgbarkeit hergestellt werden soll, werden aus unterschiedlichen Autorenwerkzeugen geladen und im ModelTracer visualisiert (siehe Abbildung 21, links). Der Import erfolgt dabei über das in der Entwicklung befindliche Mapping-Werkzeug ISYMAP, mit dessen Hilfe beliebige Metamodelle auf das Modell des ModelTracers gemappt werden können. Darüber hin-

¹⁶ Das Projekt ISYPROM – Modellbasierte Prozess- und Systemgestaltung für die Innovationsbeschleunigung – wurde mit Mitteln des Bundesministeriums für Bildung und Forschung im Rahmenkonzept „Forschung für die Produktion von morgen“ (Förderkennzeichen 02PC105x) gefördert und vom Projektträger Forschungszentrum Karlsruhe (PTKA), Bereich Produktion und Fertigungstechnologien (PFT), betreut.

aus existiert eine direkte Schnittstelle, um XML-Dateien im ReqIF-Format importieren zu können. Als Datenstrukturen sind Listen, Hierarchien und Netze verarbeitbar.

Beim Import werden die eindeutigen IDs der verschiedenen Autorenwerkzeuge genutzt, um eine Referenz in der MySQL-Datenbank des ModelTracers anzulegen. Somit bleibt die Verknüpfung zu den originalen Elementen erhalten, weshalb bspw. deren Änderung identifiziert, visualisiert und synchronisiert werden kann.

Das Werkzeug ModelTracer wurde in der Entwicklungsumgebung Eclipse mit der Programmiersprache Java implementiert. Das Datenmodell des ModelTracers ist in Abbildung 22 dargestellt. Ein Produkt besteht aus mehreren Artefakten, die es beschreiben. Diese wiederum werden aus Elementen aufgebaut, die in Hierarchien strukturiert sein können. Die Elemente der unterschiedlichen Artefakte können durch sog. Relation (die Tracelinks entsprechen) miteinander verknüpft werden.

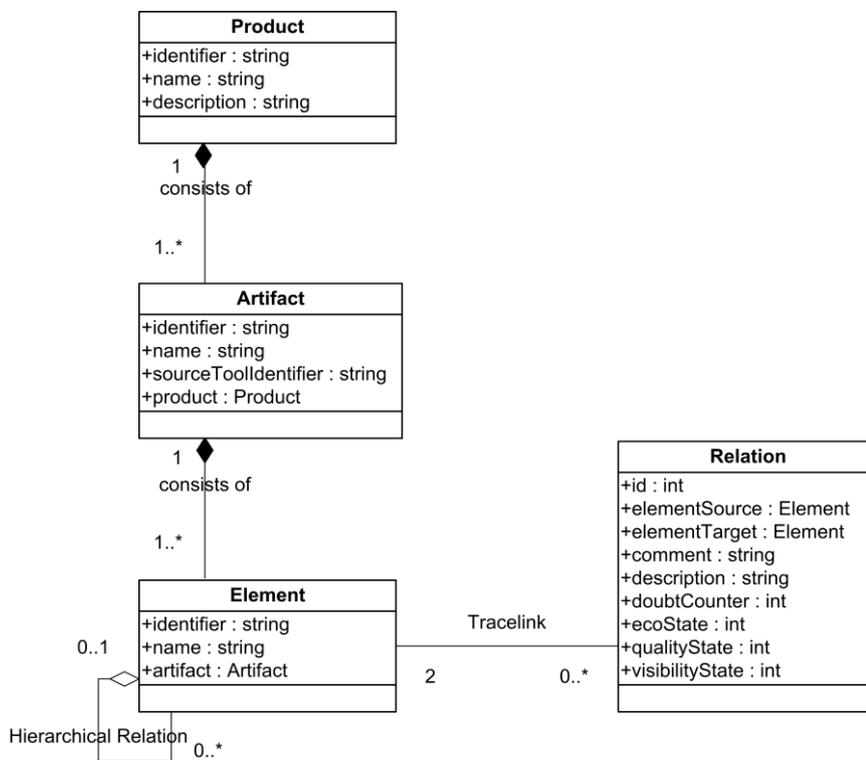


Abbildung 22: Datenmodell des ModelTracers

Das Anlegen eines Tracelinks erfolgt im ModelTracer per Drag-and-Drop des Ausgangselements auf das Endelement. Anschließend werden Details zu den zu verknüpfenden Elementen im unteren Bereich des ModelTracers dargestellt (siehe Abbildung 21).

Hier kann eine nähere Beschreibung des Tracelinks erfolgen und diesem ein Typ zugewiesen werden. Die so erstellten Tracelinks werden, wie die Referenzen der Elemente, in einer MySQL-Datenbank gespeichert. Die realisierbare Granularität liegt dabei auf Elementebene. Parameter können nicht verknüpft werden.

Die Visualisierung der Tracelinks kann mit dem Modul Ariadne's Eye durchgeführt werden. Dazu wird Ariadne's Eye vom ModelTracer aus gestartet und greift auf die in der Datenbank abgelegten Artefakte und deren Tracelinks zu, um eine übersichtliche Visualisierung des Systemzusammenhangs zu ermöglichen (siehe Abbildung 23).

Das Konzept des Visualisierers sieht vor, die Schnittstelle zwischen Entwickler und Traceability-Modell bereitzustellen. Mit seiner Hilfe soll bspw. das Systemverständnis gebildet oder Auswirkungsanalysen durchgeführt werden können. Dabei werden die Artefakte in einem N-Eck angeordnet, um Überschneidungen zwischen Hierarchien und Tracelinks zu vermeiden. Die Tracelinks werden in der Mitte dargestellt und bei Auswahl hervorgehoben. Wird ein sichtbares Element selektiert, werden alle seine Tracelinks farblich hervorgehoben, sodass bestehende Abhängigkeiten auf einen Blick erfassbar sind.

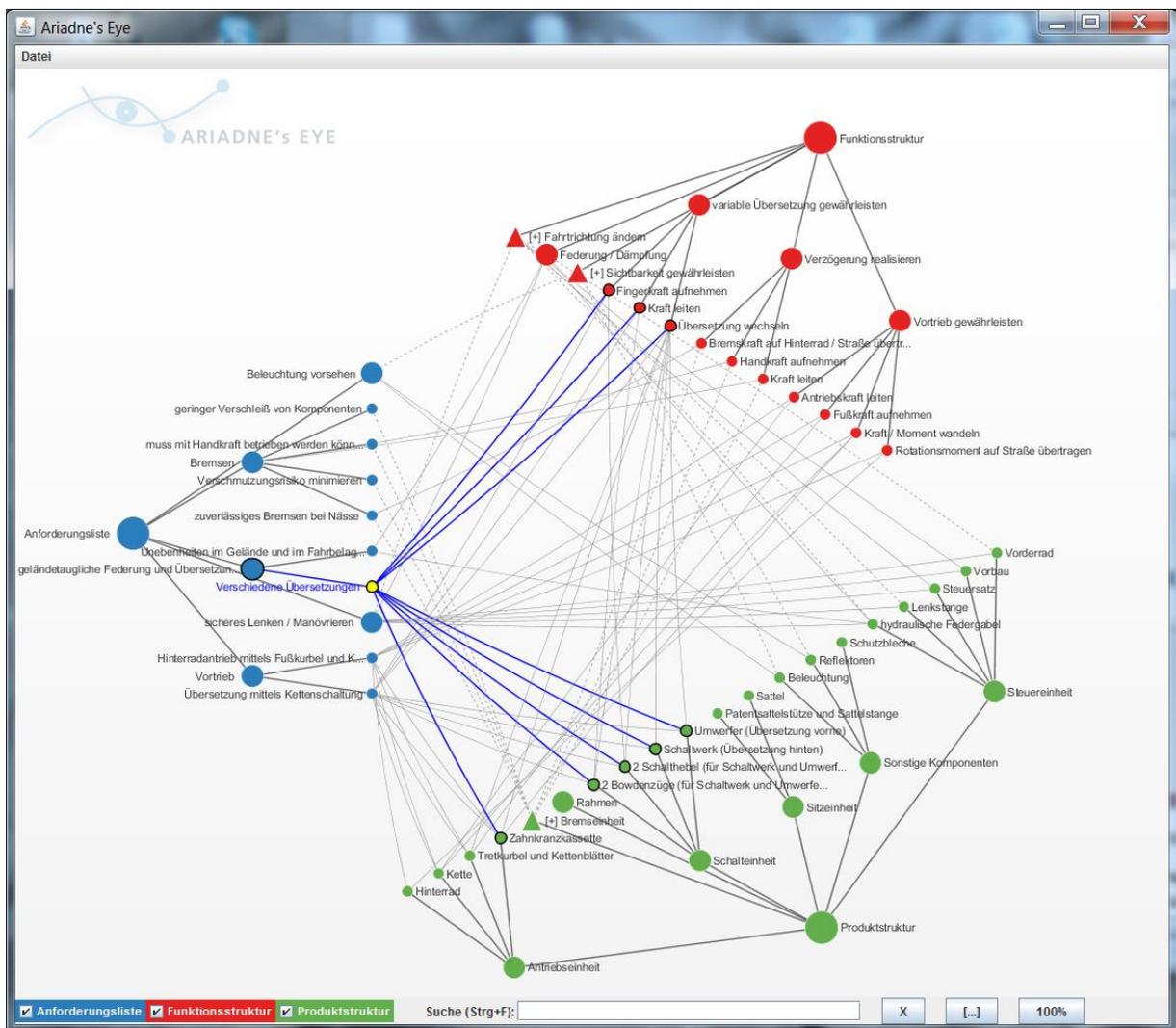


Abbildung 23: Ariadne's Eye zur Visualisierung der Artefakte und Tracelinks

Momentan stellt die begrenzte Anzahl an unterstützten Datenformaten das größte Hindernis für eine verbreitete Anwendung des Werkzeugs ModelTracer dar. Dieses

Defizit soll durch ISyMAP behoben werden, jedoch befindet es sich momentan noch in der Entwicklung. Im Fall der Weiterentwicklung des ModelTracers, über den Status eines Prototypen zur Evaluation von Methoden hinaus, müssten weitere Schnittstellen implementiert oder aber dessen Funktionalitäten bspw. auf ein existierendes PLM-System wie Teamcenter aufgesetzt werden.

In der folgenden Tabelle 9 sind die Eigenschaften des ModelTracers bzgl. der Etablierung einer durchgängigen Nachverfolgbarkeit zusammengefasst.

Akquisition der Artefakte	integrativ
Manipulierbarkeit der Artefakte	nein
Duplikation der Artefakte	original Daten
Traceability Schema	nein
Granularität	Artefaktebene, Elementebene
Arten von Datenstrukturen	Listen, Hierarchien, netzartige Strukturen
Modellierungsunterstützung	Regelbasiert, Informationsrückgewinnung, Assistent
Art der Traceability	qualitativ
Handhabung von Änderungen	Benachrichtigung
Zweck	Analysen, Dokumentation, Fortschrittskontrolle
Darstellung der Tracelinks	Graph
Prozessphasen-Kontext	phasenintern, phasenübergreifend
Artefakt-Kontext	artefaktintern, artefaktübergreifend
Semantik	typisiert und nicht-typisiert
Speicherort der Tracelinks	zentral

Tabelle 9: Traceability-Eigenschaften des ModelTracers (eine Erläuterung der Kategorien sowie Eigenschaften ist in den Kapiteln 3.3 und 3.4 zu finden)

3.7 HERAUSFORDERUNGEN DER DURCHGÄNGIGEN NACHVERFOLGBARKEIT

Die Etablierung einer durchgängigen Nachverfolgbarkeit ist bei der Anwendung zur Entwicklung technischer Systeme mit zahlreichen Herausforderungen verbunden, die im Folgenden aufgezählt und erläutert werden¹⁷. Dazu werden sie zunächst in Kapitel 3.7.1 allgemeingültig beschrieben und kategorisiert. Anschließend werden sie in Kapitel 3.7.2 erneut aufgegriffen und mit Bezug zur Entwicklung des mechatronischen Außenspiegels dargestellt, um die Herausforderungen aus der Perspektive der an der Entwicklung beteiligten Ingenieure aufzuzeigen.

¹⁷ Viele der genannten Herausforderungen sind nicht auf disziplin-übergreifende technische Systeme beschränkt, sondern sind auch im Bereich der durchgängigen Nachverfolgbarkeit von Software zu finden.

3.7.1 ALLGEMEINE DARSTELLUNG DER HERAUSFORDERUNGEN

Die Herausforderungen der Traceability lassen sich grob in technologische, methodische, zwischenmenschliche sowie monetäre klassifizieren¹⁸. Im Bereich der technologischen Herausforderungen ist es insbesondere die *informelle Notation*, in der die Artefakte erstellt werden (vgl. Kapitel 2.2), die eine Vollautomatisierung der Erfassung der Tracelinks so gut wie unmöglich macht [Ramesh und Jarke 2001, S. 39; Egyed 2003, S. 116; Zisman et al. 2003, S. 448; Winkler und Pilgrim 2010, S. 550]. Hinzu kommt, dass die Artefakte in *unterschiedlichen Modellierungssprachen* verfasst [Tudorache 2006, S. 54] und von unterschiedlichen Personen erstellt werden, was dazu führt, dass die Tracelink-Erfassung heute überwiegend manuell durchgeführt werden muss [Ramesh und Jarke 2001, S. 39]. Des Weiteren ist insbesondere bei der Entwicklung mechatronischer Systeme zu berücksichtigen, dass unterschiedliche Disziplinen an der Entwicklung beteiligt sind, die ihre jeweiligen spezialisierten Autorenwerkzeuge verwenden. Die daraus folgende zumeist sehr *heterogene Toollandschaft* ist häufig nicht integriert, so dass die Erfassung der Tracelinks zwischen den Artefakten wiederum erschwert wird [Winkler und Pilgrim 2010, S. 545-546].

Selbst wenn eine Integration erreicht wird, fehlen noch Methoden, die den Entwicklern den Umgang mit der Traceability erleichtern [Ramesh und Jarke 2001, S. 44; Mäder et al. 2009a, S. 1, 2009b, S. 5; Winkler und Pilgrim 2010, S. 556-557]. So sind die Artefakte meist sehr komplex, so dass es ohne methodische Unterstützung schwierig ist, *korrekte Tracelinks zu modellieren* [Ramesh und Jarke 2001, S. 44; Egyed und Grünbacher 2002, S. 163], den *signifikanten Aufwand*¹⁹ *für die manuelle Modellierung zu bewältigen* [Wieringa 1995, S. 16; Hayes et al. 2006, S. 4; Egyed et al. 2009, S. 242] und den *Überblick zu behalten*. Letzteres gilt insbesondere für die Online-Erfassung von Tracelinks. Zwar ist diese Vorgehensweise intuitiv, da Tracelinks modelliert werden sobald Abhängigkeiten während der Erstellung der Artefakte auftreten, allerdings ist die Beurteilung der Vollständigkeit einer solchen Analyse nicht möglich. Darüber hinaus ist das *Bestimmen der richtigen Granularität* für die Modellierung der Links eine bislang unbeantwortete Herausforderung [Mäder et al. 2009b, S. 4]. Da früh im Entwicklungsprozess häufig noch nicht klar ist, zu welchen Zwecken die Tracelinks später genutzt werden, müssen Ingenieure prognostizieren, bis zu welcher Granularität die Links später benötigt werden [Egyed und Grünbacher 2002, S. 163; Egyed et al. 2009, S. 242]. Ist die Granularität zu hoch, steigen die Kosten sowohl für die Ermittlung als auch für die Pflege des Tracelink-Modells [Winkler und Pilgrim 2010, S. 545-546]. Ist sie

¹⁸ Im Folgenden werden die Herausforderungen *kursiv* dargestellt.

¹⁹ Wenn m Anforderungen auf ihre Abhängigkeit zu n Systemelementen überprüft werden sollen, gibt es $m * n$ mögliche Tracelinks, die auf ihre Existenz überprüft werden müssen.

hingegen zu gering, ergeben die Analysen, die auf deren Basis durchgeführt werden sollen, keine hinreichend genaue Ergebnisse. Die Abschätzung ist auch deshalb schwierig, da es viele verschiedene Anwender mit sich teilweise widersprechenden Anforderungen an die Traceability gibt [Gotel und Finkelstein 1994, S. 96; Ramesh et al. 1997, S. 398; Ramesh und Jarke 2001, S. 4]. So liegt es bspw. im Interesse eines Projektmanagers die Erfüllung von Anforderungen zu überprüfen und die Entwicklungszeit für künftige Projekte zu minimieren. Ein Anforderungsmanager dokumentiert mit Hilfe der Traceability die Entwicklung einer Anforderung sowie deren Verifikation und der Entwickler nutzt sie für Auswirkungsanalysen und zur Überprüfung, ob das System vollständig ist [Wieringa 1995, S. 4-5; Aizenbud-Reshef et al. 2006, S. 516]. Es gibt somit keine Standardlösung, die sich auf jedes Entwicklungsprojekt anwenden lässt. Vielmehr muss in jedem Projekt eine angepasste Form der Nachverfolgbarkeit etabliert werden.

Somit fehlen den Anwendern sowohl industriell nutzbare Methoden als auch Werkzeuge, die sie bei der aufwändigen Modellierung von Tracelinks unterstützen. Hinzu kommen zwischenmenschliche Aspekte, die eine weite Verbreitung durchgängiger Nachverfolgbarkeit behindern. Dabei ist vor allen Dingen problematisch, dass *diejenigen, die für die Modellierung der Tracelinks verantwortlich sind, selten auch die Nutzer der Traceability* in späteren Entwicklungsphasen sind [Gotel und Finkelstein 1994, S. 97; Ramesh et al. 1997, S. 398-399; Sutinen et al. 2000, S. 7]. Sie werden somit verpflichtet, neben ihrer eigentlichen Entwicklungsarbeit umfassende Ressourcen für eine Methode einzusetzen, die für sie selbst keinen Nutzen bringt [Wieringa 1995, S. 16]. Dieses Ungleichgewicht zwischen Aufwand und Nutzen sowie der Eindruck, dass Traceability nicht kosteneffizient ist, haben fehlende Motivation und eine geringe Qualität des Tracelink-Modells zur Folge [Gotel und Finkelstein 1994, S. 97-98; Winkler und Pilgrim 2010, S. 545-546]. Es werden nur die absolut notwendigen Ressourcen sehr lokal und nicht systemübergreifend eingesetzt und damit Aspekte wie die Pflege des Tracelink-Modells bei Änderungen der Artefakte vernachlässigt [Gotel und Finkelstein 1994, S. 97-98; Winkler und Pilgrim 2010, S. 545]. Darüber hinaus ist der Aspekt, dass die *Entwickler durch die Modellierung der Tracelinks ihr Entwicklungswissen dokumentieren* und dadurch befürchten können, sich selbst überflüssig zu machen, nicht zu vernachlässigen [Sutinen et al. 2000, S. 7]. Dass die Traceability als Langzeitprojekt gesehen werden muss und auf Dauer dem ganzen Unternehmen hilft [Ramesh et al. 1997, S. 410], wird dabei ausgeblendet. Dezierte Teams, die sich nur um die Etablierung der Traceability kümmern, sind jedoch auch nicht empfehlenswert, da umfassendes Wissen über die Artefakte für die Modellierung der Tracelinks notwendig ist. Arkley und Riddle konnten zeigen, dass die Anzahl falscher Tracelinks in Projekten, in denen spezielle Traceability-Teams eingesetzt wurden, höher war als

in solchen, in denen die Tracelinks von Software-Entwicklern modelliert wurden [Arkley und Riddle 2005, S. 385-386].

Unabhängig davon, wer die Links modelliert, ist das Ergebnis immer subjektiv und nicht fehlerfrei (Stichwort: fehlende Tracelinks bzw. falsche Tracelinks) [Gotel und Finkelstein 1994, S. 98-99; Egyed und Grünbacher 2002, S. 163; Egyed et al. 2009, S. 256]. Aus Sicht der Nutzer der Traceability hat dies zur Folge, dass sie die *Qualität des Tracelink-Modells nicht einschätzen können und diesem nicht vertrauen* [Gotel und Finkelstein 1994, S. 97-98].

Dem stehen, wie bereits erwähnt, ein nicht unerheblicher *Aufwand und damit hohe Kosten gegenüber* [Winkler und Pilgrim 2010, S. 539]. So investiert das Department of Defense (DoD) der USA ca. 4 % ihrer Systementwicklungskosten in Traceability [Ramesh et al. 1997, S. 398]. Ansätze, die die Etablierung effizienter und durch Automatisierung oder Prozessunterstützung kostengünstiger machen sollen, befinden sich noch im Forschungsstadium [Ramesh und Jarke 2001, S. 44; Winkler und Pilgrim 2010, S. 545], und das Bewusstsein für die Vorteile der Traceability ist noch nicht ausreichend ausgeprägt [Arkley und Riddle 2005, S. 385-386; Mäder et al. 2009a, S. 1; Winkler und Pilgrim 2010, S. 545-546]. Zudem lässt sich das *Return on Investment (ROI) nicht einfach ermitteln* [Palmer 2000], und ein empirischer Nachweis, für welchen Anwender sich Traceability ab wann lohnt, ist nicht vorhanden [Winkler und Pilgrim 2010, S. 545-546]. Dadurch wird das Kosten/Nutzen-Verhältnis in der Industrie als sehr schlecht angesehen. Aus diesem Grund wird Traceability momentan nur in einem minimal geforderten Umfang zu Dokumentationszwecken praktiziert, wo es gesetzlich vorgeschrieben ist [Arkley und Riddle 2005, S. 385; Mäder et al. 2008, S. 1].

3.7.2 DARSTELLUNG DER HERAUSFORDERUNGEN AM BEISPIEL DER ENTWICKLUNG DES MECHATRONISCHEN AUßENSPIEGELS

Bei der Analyse des Vorgehens der Studenten während der Lehrveranstaltung „Virtual Engineering in Industry“ fällt auf, dass zwei unterschiedliche Rollen an der Entwicklung des mechatronischen Außenspiegels direkt bzw. indirekt beteiligt waren. Zum einen die Studenten selbst, die, abgesehen von wenigen Tätigkeiten im Bereich des Projektmanagements, als Entwickler tätig waren und jeweils für unterschiedliche Entwicklungsartefakte verantwortlich zeichneten. Und andererseits die Betreuer, die den Part des Entwicklungsmanagers übernahmen, in diesem Zusammenhang Design Reviews veranstalteten und die Entwicklung überwachten.

Durch diese personelle Aufteilung zeigten sich hinsichtlich der Etablierung einer durchgängigen Nachverfolgbarkeit deutliche Parallelen zu realen in der Literatur dokumentierten Entwicklungsprojekten (siehe bspw. [Ramesh und Jarke 2001]), weshalb

die zuvor vorgestellten Herausforderungen im Folgenden nochmals anhand der Entwicklung des mechatronischen Außenspiegels aufgezeigt werden sollen.

Während der Lehrveranstaltung spielten die technologischen Herausforderungen nur eine untergeordnete Rolle. Wie in Kapitel 2.1.5 beschrieben, wurde für die Entwicklung des mechatronischen Außenspiegels die integrierte Entwicklungsumgebung V6 von Dassault Systèmes verwendet, weshalb die Modellierung von Tracelinks zwischen den verwendeten Artefakten grundsätzlich möglich war. Aus technologischer Sicht zeigte sich darüber hinaus, dass eine automatisierte Erfassung der Tracelinks aufgrund der informellen Notation insb. der Anforderungen sowie der Tatsache, dass während des Projekts keine verbindlichen Regeln für die Benennung der Elemente vereinbart wurden, nicht möglich gewesen wäre. Abgesehen davon bietet die V6-Umgebung zum aktuellen Zeitpunkt keine Funktionalitäten zur Unterstützung des Modellierungsprozesses von Tracelinks, weshalb ohnehin eine manuelle Verknüpfung durchgeführt wurde.

Auf zwischenmenschlicher Ebene wurde deutlich, dass die Verteilung der Aufgaben und des Nutzens analog zur Beschreibung in Kapitel 3.7.1 war: Traceability wurde von den Entwicklungsmanagern (Betreuern) vorgeschrieben, um den Entwicklungsfortschritt verfolgen und die Traceability-Funktionalitäten der V6-Entwicklungsumgebung testen zu lassen. Die Entwickler (Studenten) hingegen mussten die Tracelinks zwischen den unterschiedlichen Artefakten modellieren, ohne einen direkten wahrgenommenen Nutzen davon zu haben. Theoretisch konnten die Tracelinks zwar genutzt werden, um während der Entwicklung bspw. bei Änderungen Auswirkungsanalysen durchzuführen. Praktisch wurde diese formale Vorgehensweise jedoch aufgrund fehlender Funktionalität in V6 und der vergleichsweise geringen Komplexität des Entwicklungsgegenstandes nicht angewendet. Somit war der Nutzen der Entwickler auf die Dokumentation für die Zertifizierung (Benotung) beschränkt. Die notwendigen Aufwände für die als Last empfundene Modellierung der Tracelinks mussten von den ohnehin knapp bemessenen Ressourcen im Rahmen des Projekts aufgebracht werden und fehlten bei der eigentlichen Entwicklung. Dies hatte zur Folge, dass das Thema Traceability vernachlässigt wurde und bspw. eine kontinuierliche Pflege des Tracelink-Modells nicht stattfand. Lediglich die Tracelinks zwischen Verhaltensmodell und Produktstruktur (sog. Ports) wurden umfassend betrachtet, da sie die Integration von Verhalten und Geometrie herstellen, die für die Validierung gefordert worden war und in Design Reviews vorgestellt werden musste.

Auch für die Entwicklungsmanager, die eigentlichen Nutznießer der Traceability, war diese Vorgehensweise der Entwickler bei der Modellierung und Pflege von Tracelinks problematisch. Da die Arbeiten am Traceability-Modell meilensteingetrieben waren und zudem nur mit den absolut notwendigen Ressourcen betrieben wurden, konnte

nicht zu jeder Zeit von der Aktualität des Traceability-Modells ausgegangen werden. Zudem sind die Tracelinks von der subjektiven Einschätzung der modellierenden Entwickler abhängig und selten fehlerfrei. Das bedeutet, dass zwischen abhängigen Elementen Tracelinks fehlten oder Tracelinks zwischen Elementen modelliert waren, bei denen keine Abhängigkeit bestand. Im Zusammenhang mit der zuvor genannten Vernachlässigung des Themas Traceability durch die Entwickler bedeutet dies, dass die Qualität des Traceability-Modells nicht eingeschätzt werden konnte und damit die verlässliche Beurteilung des Projektfortschritts bspw. durch eine Abdeckungsanalyse der Anforderungen nicht möglich war.

Dabei ist zu berücksichtigen, dass die Entwickler keinerlei methodische Unterstützung bei der Modellierung der Tracelinks erhielten. Lediglich eine Einführung in die Funktionalitäten zur manuellen Tracelinkerstellung wurde zu Beginn des Projektes im Rahmen einer Übung durchgeführt. Es wurde allerdings nicht weiter darauf eingegangen, zwischen welchen Artefakten oder zu welchem Zeitpunkt Tracelinks modelliert werden sollten. Auch wurde nicht die zu erreichende Granularität der Tracelinks diskutiert. Die Entscheidung zwischen welchen Artefakten, zu welchem Zeitpunkt und in welcher Granularität die durchgängige Nachverfolgbarkeit erreicht werden sollte, lag somit bei den Entwicklern. Diese entschieden sich, die Artefakte jeweils nach der Fertigstellung in Reihenfolge der Entstehung mit Tracelinks bis zur untersten Hierarchieebene zu verknüpfen (Anforderungsartefakt – Funktionsartefakt – Verhaltensartefakt – Produktstruktur). Auch wenn die Artefakte im Vergleich zu industriellen Maßstäben weniger komplex waren, so waren die dafür notwendigen Aufwände nicht zu unterschätzen: insgesamt existierten artefaktübergreifend 101 Elemente und theoretisch mussten 1304 Entscheidungen über die Existenz einer Abhängigkeit zwischen den Elementen getroffen werden. Diese Entscheidungen waren dabei nicht immer leicht zu treffen. Während bspw. ein Tracelink zwischen der Anforderung, die die Realisierung einer Außenspiegelheizung verlangt, und der Funktion „Außenspiegel beheizen“ vergleichsweise einfach zu modellieren war, war die Entscheidung, welche der modellierten Funktionen an der Umsetzung der Anforderung „elektronische Verstellbarkeit des Außenspiegels vorsehen“ beteiligt sind, weniger eindeutig. Gründe dafür liegen bspw. darin, dass die Anforderung durch mehrere Funktionen realisiert wurde, oder dass eine eher indirekte Beteiligung an der Realisierung vorlag. So hat bspw. die Funktion „Rückwärtsfahrt erkennen“ auf den ersten Blick keine Abhängigkeit zur Anforderung nach einer elektronischen Verstellbarkeit des Außenspiegels. Wird jedoch berücksichtigt, dass sich der Außenspiegel bei Rückwärtsfahrt automatisch verstellen soll, besteht Anlass, einen Tracelink zwischen diesen beiden Elementen zu modellieren. Solche Beispiele, ähnlich dem zuvor genannten, existieren zwischen allen Artefakten und erschweren die Entscheidungsfindung bei der Modellierung von

Tracelinks enorm, insbesondere wenn man, wie die Entwickler im beschriebenen Projekt, den späteren Verwendungszweck der Tracelinks nicht kennt.

Zusammenfassend kann somit gesagt werden, dass Entwickler einen nicht unerheblichen Aufwand betreiben mussten, um Tracelinks zu modellieren. Dabei handelte es sich um eine nicht-triviale Aufgabe, welche die umfassende Kenntnis der betrachteten Artefakte erforderte und bei der teilweise Entscheidungen getroffen werden mussten, deren Konsequenzen nicht abgesehen werden konnten. Hinzu kam, dass die Entwickler nicht die Nutznießer der so etablierten, durchgängigen Nachverfolgbarkeit waren. Jedoch galt auch für die Entwicklungsmanager, die im beschriebenen Projekt die Nutzer darstellten, dass das Thema durchgängige Nachverfolgbarkeit nicht unproblematisch war, da sie die Qualität des Tracelink-Modells nicht einschätzen konnten und eine uneingeschränkte Nutzung somit nicht möglich war.

3.8 ZUSAMMENFASSUNG UND DISKUSSION

In Kapitel 3 wurden die Grundlagen durchgängiger Nachverfolgbarkeit beschrieben. Dafür wurde zunächst auf den Bedarf eingegangen, der sich durch die Verschiebung von der Entwicklung rein mechanischer Systeme hin zu mechatronischen Systemen, die durch eine hohe Vernetzung geprägt sind, ergibt. Darüber hinaus stellt jedoch auch deren Entwicklung, an der unterschiedliche Disziplinen beteiligt sind, eine Herausforderung dar, da die disziplinübergreifenden Abhängigkeiten meist nur implizit bekannt und nicht allen Entwicklern bewusst sind [Aizenbud-Reshef et al. 2006, S. 515; Tudorache 2006, S. 3].

Im Anschluss wurde durchgängige Nachverfolgbarkeit als Lösungsansatz für die zuvor beschriebenen Herausforderungen vorgestellt, sowie deren Ursprung und Verbreitung näher betrachtet. Dabei wurde aufgezeigt, dass die Methode im Vergleich zur Entwicklung technischer Systeme in der Software-Entwicklung, bereits verbreitet ist, und auch zahlreiche positive Erfahrungen im Umgang mit Traceability gemacht wurden. Unabhängig vom Einsatzgebiet lassen sich dabei vier Phasen der Traceability unterscheiden die näher charakterisiert wurden: Planung, Erfassung, Verwendung und Pflege.

Bei der Anwendung dieser Methode zur Entwicklung technischer Systeme bestehen Herausforderungen, die vier Kategorien zugeordnet werden können: technologische, methodische, zwischenmenschliche und monetäre Herausforderungen. So stellen die informelle Notation der Entwicklungsartefakte, ihre Mehrsprachigkeit sowie die heterogene Toolandschaft, die durch spezialisierte Software geprägt ist, große technische Hürden bei der Etablierung einer durchgängigen Nachverfolgbarkeit dar. Darüber hinaus fehlen bislang die Berücksichtigung in den meisten Vorgehensmodelle

(vgl. Kapitel 2.1), Methoden zum effizienten Umgang mit der Traceability und dem Treffen der notwendigen Entscheidungen. Hinzu kommt, dass es keine Unterstützung bei der Festlegung der „angemessenen“ Granularität für die Modellierung von Tracelinks gibt, um ein Optimum zwischen Nutzen und Aufwand zu erreichen. Ferner ist der Aspekt, dass diejenigen, die für die Modellierung der Tracelinks verantwortlich zeichnen, selten auch die Nutzer der Nachverfolgbarkeit in späteren Entwicklungsphasen sind, eine große Herausforderung, welche zu mangelnder Qualität der Tracelink-Modelle führen kann. Abschließend bleibt zu erwähnen, dass dem Nutzen ein nicht unerheblicher Aufwand und damit hohe Kosten gegenübersteht.

Um diese Herausforderungen an den Belangen der an der Entwicklung beteiligten Ingenieure zu spiegeln, wurde das Beispiel des mechatronischen Außenspiegels analysiert und festgestellt, dass Entwicklungsingenieure mit der Modellierung von Tracelinks vor eine nicht-triviale, aufwändige Aufgabe gestellt werden, bei der schwere Entscheidungen getroffen werden müssen. Dem gegenüber stehen ein als gering wahrgenommener Nutzen und die Tatsache, dass die eigentlichen Nutznießer der durchgängigen Nachverfolgbarkeit andere sind. Diese konnten am Beispiel der Entwicklung des mechatronischen Außenspiegels als Entwicklungsmanager identifiziert werden, für die der Umgang mit dem Tracelink-Modell jedoch auch nicht unproblematisch ist, da sie die Qualität der Tracelinks nicht einschätzen können. Diese Herausforderungen führen dazu, dass durchgängige Nachverfolgbarkeit heute zumeist nur dort praktiziert wird, wo es gesetzlich vorgeschrieben ist [Arkley und Riddle 2005, S. 385; Mäder et al. 2008, S. 1] und dann meist nur zu Dokumentationszwecken und in einem Umfang, der minimal die Vorschriften erfüllt.

Trotzdem gibt es auch im Umfeld der Entwicklung mechatronischer Systeme bereits Software-Tools zur Erfassung, Pflege und Verwendung von Tracelinks. Eine Auswahl wurde in Kapitel 3.5 anhand der vier Kategorien Anforderungsmanagement-Software, PLM-Software, Analyse-Software sowie Integrationssoftware vorgestellt und ihre Eigenschaften hinsichtlich der durchgängigen Nachverfolgbarkeit mit Hilfe der in den Kapiteln 3.3 und 3.4 beschriebenen Kategorien bewertet. In der folgenden Tabelle 10 sind diese Kategorien (linke graue Spalte), deren Ausprägungen (weiße Zellen) sowie die Bewertungen der einzelnen Software-Tools (mittels Kürzel) zusammengefasst.

Bei der Betrachtung der verfügbaren Software-Tools zur Verwaltung von Tracelinks in Tabelle 10 fällt auf, dass insbesondere im Bereich der Modellierungsunterstützung wenige Lösungen für die Entwicklung mechatronischer Systeme verfügbar sind. Gleiches gilt für den Bereich der Pflege von Tracelink-Modellen, die von keinem der Werkzeuge unterstützt wird.

Die durchgängige Nachverfolgbarkeit bietet somit große Potenziale, um für die ein-

gangs erwähnten Herausforderungen bei der Entwicklung mechatronischer Systeme eine Hilfestellung zu bieten. Allerdings leidet sie in diesem Kontext unter vielschichtigen Akzeptanzproblemen. Vordergründig ist dafür das schlechte Kosten/Nutzen-Verhältnis des Ansatzes, welcher in den hohen Aufwänden und dem (gefühl) geringen Nutzen begründet liegt, verantwortlich. Um dieses Verhältnis zu verbessern, ergeben sich somit zwei Forschungsrichtungen: es gilt die Kosten zu reduzieren und den Nutzen zu steigern. Die vorliegende Arbeit fokussiert dabei auf die erste Forschungsrichtung, indem Methoden zur effizienten Modellierung von Tracelinks sowie deren Pflege konzipiert und evaluiert werden. Die konkreten Forschungsfragen, die dabei beantwortet werden sowie die Hypothesen, die zu diesem Zweck aufgestellt wurden, sind im folgenden Kapitel 4 zusammengefasst.

Akquisition der Artefakte	integrativ R V T N C M T	nicht-integrativ D M L				
Manipulierbarkeit der Artefakte	ja D V T	nein R M L N C M T				
Duplikation der Artefakte	original Daten D R V T N C M T	kopierte Daten M L				
Traceability Schema	ja R V M	nein D T L N M T				
Granularität	Artefakt-Ebene M T	Element-Ebene D R V M L N C M T	Parameter-Ebene T			
Arten von Datenstrukturen	Liste D R V T M L N C M T	Hierarchie D R V T M L N C M T	Poly-Hierarchie	netzartige Strukturen V T N C M T		
Modellierungsunterstützung	keine D R V T M L N	Regelbasiert C M T	Informationsrückführung C M T	Wizard M T	Template	
Art der Traceability	qualitativ D V T L N C M T	quantitativ M L				
Handhabung von Änderungen	Benachrichtigung D R V T L N M T	Änderungsübertragung T	keine M			
Zweck	Analyse D R V T M L N M T	Dokumentation D R V T L N M T	Fortschrittskontrolle R V T M T	Verifikation V T	Synthese V T	Synchronisation V T
Darstellung der Tracelinks	Matrix D R M L	Graph D R M L	Referenz D R V T			
Prozessphasen-Kontext	phasenintern D R V T L N C M T	phasenübergr. R V T M L N C M T				
Artefakt-Kontext	artefaktintern D V T L C M T	artefakt-übergreifen D R V T M L N C M T				
Semantik	typisiert R V N C M T	nicht typisiert D V T M L M T				
Speicherort der Tracelinks	verteilt R	zentral D V T M L N C M T				

Tabelle 10: Traceability-Eigenschaften der betrachteten Software-Tools (D – DOORS, R – Reqify, V – V6-Suite, T – Teamcenter, M – Metus, L – Loomeo. N – ToolNet, C – ConWeaver, MT – ModelTracer)

4 FORSCHUNGSFRAGEN UND HYPOTHESEN

Wie in Kapitel 3 dargestellt, bietet die durchgängige Nachverfolgbarkeit ein großes Potenzial für die Anwendung zur Entwicklung technischer Systeme. Ziel dabei ist es, Abhängigkeiten zwischen Entwicklungsartefakten explizit zu modellieren, um Inkonsistenzen zu vermeiden und die Entwicklung insgesamt transparenter zu gestalten. Allerdings bestehen zahlreiche Herausforderungen, die einem umfassenden Einsatz bislang noch entgegenstehen. Insgesamt wurden in Kapitel 3.7.1 zwölf Herausforderungen identifiziert, den Kategorien technologische, methodische, zwischenmenschliche und monetäre Herausforderungen zugeordnet und am Beispiel des mechatrischen Außenspiegels aus der Perspektive von Entwicklungsingenieuren und Entwicklungsmanagern reflektiert.

Auch wenn dies für eine industrielle Nutzung durchgängiger Nachverfolgbarkeit sinnvoll wäre – nicht alle diese Herausforderungen können im Rahmen der vorliegenden Arbeit adressiert werden. Aus diesem Grund wurde sich auf die folgenden methodischen Herausforderungen beschränkt, um Ingenieure mit Methoden zur Erstellung und Pflege durchgängiger Nachverfolgbarkeit zu unterstützen:

- 1 Fehlende methodische Unterstützung, um bei der Modellierung von Tracelinks den *Überblick zu behalten*.
- 2 Fehlende methodische Unterstützung, um den signifikanten Aufwand für die manuelle Modellierung zu bewältigen.

- 3 Fehlende methodische Unterstützung, um *korrekte Tracelinks zu modellieren* bzw. im Nachhinein falsche und fehlende zu identifizieren.

Diese Herausforderungen werden vor dem Hintergrund der beiden Phasen „Erfassung von Tracelinks“ (siehe auch Kapitel 3.4.1) sowie „Pflege von Tracelink-Modellen“ (siehe auch Kapitel 3.4.3) betrachtet. Konkret bedeutet dies, dass in Kapitel 4.1 eine Forschungsfrage vorgestellt wird, die sich der methodischen Unterstützung zur effizienten Modellierung von Tracelinks widmet und damit die zuvor beschriebenen Herausforderungen 1 und 2 adressiert. In den Kapiteln 4.2 sowie 4.3 werden anschließend Forschungsfragen formuliert, die sich mit der Sicherung der Qualität des Tracelink-Modells befassen und somit an Herausforderung 3 richten, indem Methoden erforscht werden, die eine Identifikation von falschen und fehlenden Tracelinks im Kontext der Entwicklung technischer Systeme ermöglichen.

Grundsätzlich ist anzumerken, dass keine dieser Forschungsfragen und Hypothesen den Anspruch erhebt, die formulierten Herausforderungen in Gänze zu bewältigen. Vielmehr handelt es sich um konkrete Teilaspekte, die erforscht werden, um einen Beitrag zur stärkeren Verbreitung durchgängiger Nachverfolgbarkeit zur Entwicklung technischer Systeme zu leisten.

Die Methoden, die in diesem Zusammenhang in dieser Arbeit entwickelt werden, richten sich an den Bedarfen der Entwicklung technischer Systeme aus. So werden im Folgenden die bereits vorgestellten Artefakte, die zur Entwicklung mechatronischer Systeme genutzt werden, hinsichtlich ihrer Strukturierungsansätze analysiert und die Erläuterung sowie Evaluation der Methoden mit Beispielen, die der Entwicklung mechatronischer Systeme entstammen, durchgeführt. Unabhängig davon sind die entwickelten Methoden jedoch nicht zwingend auf die Entwicklung technischer Systeme festgelegt. Vielmehr ist eine Übertragbarkeit auf die Softwareentwicklung (der die durchgängige Nachverfolgbarkeit ursprünglich entstammt) durchaus denkbar, wird allerdings im Rahmen der vorliegenden Arbeit nicht angestrebt.

4.1 METHODISCHE UNTERSTÜTZUNG ZUR EFFIZIENTEN MODELLIERUNG VON TRACELINKS

In der Phase der Erfassung von Tracelinks besteht u. a. die Herausforderung, dass es bei der heute üblichen manuellen Modellierung von Tracelinks ohne methodische Unterstützung für die Ingenieure schwierig ist, den Überblick zu behalten. Insbesondere bei der Analyse von Artefakten mit einer großen Anzahl von Elementen führt eine unstrukturierte Vorgehensweise dazu, dass nicht festgestellt werden kann, welche Elementkombinationen bereits auf Abhängigkeiten untersucht wurden. Hinzu kommt der signifikante Aufwand, der für die Modellierung aufgebracht werden muss. Die Forschungsfrage, die diese Herausforderungen adressiert, lautet:

Kann methodische Unterstützung helfen, den Aufwand bei der Modellierung von Tracelinks zu reduzieren? Wie muss diese ausgeprägt sein?

Sie adressiert den zuvor in Kapitel 3 beschriebenen Aspekt, dass bislang zu wenig effiziente Methoden für Erfassung von Tracelinks existieren. Die zugehörige Hypothese 1 macht sich die Eigenschaften bestimmter Arten von Hierarchien (siehe Kapitel 5.2.1.1) zunutze:

Die Eigenschaften von Inklusionshierarchien können genutzt werden, um die Anzahl zu treffender Entscheidungen bei der Modellierung von Tracelinks zu verringern.

Auf dieser Hypothese basierend wurde die Methode *EcoTracing* konzipiert, prototypisch implementiert und evaluiert. Alle notwendigen Grundlagen sowie die Beschreibung und Evaluierung der Methode ist Kapitel 5 zu entnehmen.

4.2 METHODISCHE UNTERSTÜTZUNG ZUR VERTEILTEN QUALITÄTSSICHERUNG

Bei der Pflege von Tracelink-Modellen besteht u. a. die Herausforderung, falsche Tracelinks zwischen Elementen, die keine Abhängigkeit zueinander aufweisen, zu identifizieren. Diese falsch positiven Tracelinks haben zur Folge, dass bspw. bei Auswirkungenanalysen Elemente fälschlicherweise berücksichtigt werden und somit zusätzliche Aufwände erzeugt werden. Um diese Tracelinks zu identifizieren wurde folgende Forschungsfrage formuliert:

Können Methoden des Crowdsourcings zur Steigerung der Qualität des Tracelink-Modells genutzt werden?

Das Crowdsourcing (deutsch: Schwarmauslagerung) ist eine relativ neue Strategie, die vor allen Dingen bei einer Vielzahl von Diensten im Internet zu finden ist [Estelles-Arolas und Gonzalez-Ladron-de-Guevara 2012, S. 189] und sich potenziell gut für die verteilte Sicherung der Qualität des Tracelink-Modells durch eine Vielzahl an Nutzern eignen könnte. Die zugehörige Hypothese 2, die in diesem Zusammenhang evaluiert wird, ist zweigeteilt:

Während der Interaktion mit dem Traceability-Modell fallen Nutzern falsch modellierte Tracelinks auf.

Die Anzahl der Nutzer hat einen Einfluss auf die Vollständigkeit der Ergebnisse.

Die Hypothese basiert auf der Annahme, dass den Nutzern des Tracelink-Modells während der Verwendung von Tracelinks (z. B. zur Durchführung von Auswirkungsanalysen) falsch positive Tracelinks auffallen und sie diese melden können. Der zweite Teil der Hypothese adressiert die Annahme, dass sich eine große Anzahl an Nutzern positiv auf die Anzahl identifizierter falsch positiver Tracelinks auswirkt.

Basierend auf dieser Hypothese wurde die Methode *TraceEvaluation* entwickelt und zum Zweck ihrer Evaluierung eine Studie durchgeführt. Die Beschreibung der Methode und das Ergebnis der Evaluation ist in Kapitel 6.3 zusammengefasst.

4.3 VERWENDUNG DES WISSENS VERGANGENER PROJEKTE ZUR QUALITÄTSSICHERUNG

Eine weitere Herausforderung bei der Pflege von Tracelink-Modellen ist es, Elementekombinationen zwischen denen eine Abhängigkeit besteht, aber kein Tracelink modelliert wurde, zu identifizieren. Diese falsch negativen Tracelinks haben zur Folge, dass bspw. bei Auswirkungsanalysen im Fall von Änderungen Elemente fälschlicherweise nicht berücksichtigt werden, was zu späten und damit teuren Änderungen führen kann. Um diese Tracelinks zu identifizieren wurde folgende Forschungsfrage formuliert:

Kann das Wissen, welches mithilfe von Tracelinks in vergangenen Projekten dokumentiert wurde, in aktuellen Projekten zur Steigerung der Qualität des Tracelink-Modells genutzt werden?

Die Frage ist, ob Wissen über Abhängigkeiten zwischen Elementen, welches in vergangenen Projekten dokumentiert wurde, wiederverwendet und zur Generierung von Vorschlägen für fehlende Tracelinks in aktuellen Projekten herangezogen werden kann. Die in diesem Zusammenhang formulierte Hypothese 3 postuliert, dass dies möglich ist:

Auf Basis von Tracelink-Modellen ähnlicher Vorgänger-Entwicklungsprojekte können relevante Vorschläge für fehlende Tracelinks in aktuellen Projekten generiert werden.

Als Einschränkung wird angegeben, dass es sich um inhaltlich ähnliche Projekte handeln muss und somit ähnliche Systeme entwickelt werden. Basierend auf dieser Hypothese wurde die Methode *TraceLegacy* konzipiert und implementiert. Die eigentliche Evaluation erfolgte mit Hilfe eines repräsentativen Beispiels. Alle Beschreibungen der Methode sowie deren Evaluation sind in Kapitel 6.4 zusammengestellt.

5 METHODEN ZUR EFFIZIENTEN MODELLIERUNG VON TRACELINKS

In Kapitel 5 liegt der Fokus auf Methoden, die Ingenieuren bei der Offline-Erfassung von Tracelinks helfen, indem der notwendige Aufwand für die manuelle Modellierung von Tracelinks reduziert wird. In diesem Zusammenhang werden zunächst in Kapitel 5.1 vorhandene Methoden des Stands der Technik vorgestellt. Da Hypothese 1 postuliert, dass sich Inklusionshierarchien für die angestrebte Aufwandsreduzierung eignen, werden in Kapitel 5.2 zunächst deren Eigenschaften erörtert. Anschließend wird für die bereits in Kapitel 2.2 beschriebenen Artefakte herausgearbeitet, ob und unter welchen Voraussetzungen die Hypothese für sie Anwendung finden kann. Um diese auf theoretischen Vorüberlegungen basierenden Grundlagen im Kontext der Entwicklung eines technischen Systems zu überprüfen, wurde anschließend eine Studie durchgeführt, deren Aufbau, Durchführung und Ergebnisse in Kapitel 5.3 beschrieben werden. Auf Basis der so zusammengestellten Voraussetzungen und Anforderungen wurde die Methode EcoTracing zur effizienten Erfassung von Tracelinks konzipiert, deren Funktionsprinzip und prototypische Implementierung in Kapitel 5.4 zusammengefasst werden. Die Wirksamkeit der Methode wurde in einer weiteren Studie nachgewiesen, die in Kapitel 5.4.5 erläutert und diskutiert wird.

Explizit nicht im Fokus sind Methoden, die auf Informationsrückführung zum Vorschlagen von Tracelinks basieren. Wie bereits in Kapitel 3.4.1 dargestellt, werden diese im

Rahmen dieser Arbeit dem Bereich der Pflege von Tracelink-Modellen zugeordnet und deshalb in Kapitel 6 betrachtet.

5.1 STAND DER TECHNIK UND FORSCHUNG

Im Stand der Technik und Forschung sind nur wenige Methoden vorhanden, um Entwickler bei der Erfassung von Tracelinks zu unterstützen. Zwar gibt es Empfehlungen, Tracelinks möglichst direkt von Entwicklern und nicht von dezidierten Qualitätsteams und darüber hinaus beim Feststellen einer Abhängigkeit erfassen zu lassen, wie dies jedoch gelebt werden soll, wird selten detailliert. So ist es heute insbesondere in der industriellen Anwendung üblich, Entwickler Abhängigkeitsmatrizen entweder allein oder im Rahmen von Workshops ausfüllen zu lassen [Stark 2011].

Egyed et al. beschreiben in [Egyed et al. 2009] die Methode „Value-based Trace Enhancement“, die zwar keine direkten Empfehlungen zur Modellierung gibt, jedoch den Aufwand zur Erfassung von Tracelinks in der Softwareentwicklung reduziert und damit Entwicklern eine Hilfestellung bietet. Die Methode geht von dem Grundsatz aus, dass bei der Modellierung von Tracelinks nicht alle Elemente eines Artefakts dieselbe Wichtigkeit für das zu entwickelnde System haben und somit nicht in derselben Granularität hinsichtlich des Vorhandenseins von Tracelinks analysiert werden müssen. Da es laut Egyed et al. wichtiger ist, vollständig als präzise zu sein, beginnt man zunächst damit, die Abhängigkeiten der betrachteten Artefakte in einer groben Granularität möglichst vollständig zu ermitteln. Eine Verfeinerung erfolgt nur in Bereichen der Artefakte, die eine hohe Wichtigkeit für die Entwicklung aufweisen.

Diese Wichtigkeit wird letztlich durch die spätere Verwendung der Tracelinks für unterschiedliche Zwecke bestimmt, die sich jedoch nicht einfach voraussagen lassen. Als guter Indikator für die Wichtigkeit eignen sich laut Egyed et al. die Prioritäten der Anforderungen, da sie die Gewichtung der beteiligten Stakeholder wiedergeben. Alternativ können Tracelinks auch manuell direkt Wichtigkeiten zugeordnet werden. Niedrig priorisierte Anforderungen werden somit grobgranular und hoch priorisierte feingranular nachverfolgt. Alternativ ist es auch möglich, die Traceability erst bei Bedarf zu verfeinern.

Durch diese Vorgehensweise ergeben sich Einsparungen, die sich jedoch nicht allgemeingültig quantifizieren lassen, da diese von der Struktur der betrachteten Artefakte abhängig sind. So ist ein Anteil von 50 % wichtiger Anforderungen nicht gleichbedeutend mit 50 % Aufwandsersparnis, da diese u. U. für die Definition großer Teile des zu entwickelnden Systems relevant sind. Egyed et al. berichten von einem Beispiel, bei dem 40 % der wichtigen Anforderungen für 50-80 % des Sourcecodes rele-

vant sind. Trotzdem waren in diesem Beispiel (durch Modifikation des Algorithmus zur Bestimmung der Wichtigkeit) Einsparungen von 30-70 % realisierbar.

Tilstra et al. beschreiben die Vorgehensweise „Distributed Modeling of Component DSM“ zum Ausfüllen von DSMs, mit deren Hilfe sich der dazu notwendige Aufwand reduzieren lässt [Tilstra et al. 2009]. Dabei werden die Elemente einer DSM in logische Gruppen (z. B. nach Modulen) aufgeteilt (siehe Abbildung 24, a) und deren interne Abhängigkeiten ermittelt. Zusätzlich wird diesen Gruppen jeweils ein Element mit dem Namen „extern“ hinzugefügt, um Abhängigkeiten über die Gruppengrenzen hinweg modellieren zu können (siehe Abbildung 24, b). Im Anschluss werden die Abhängigkeiten der Platzhalter-Gruppen untereinander in einer separaten sog. System-DSM ermittelt (siehe Abbildung 24, c), welche im darauffolgenden Schritt durch Ersetzen der Gruppen durch die beinhalteten Elemente wieder erweitert wird (siehe Abbildung 24, d). Abschließend müssen noch die „extern“-Elemente auf Abhängigkeiten überprüft werden, während die schraffierten Zellen in Abbildung 24, d) nicht erneut betrachtet werden müssen.

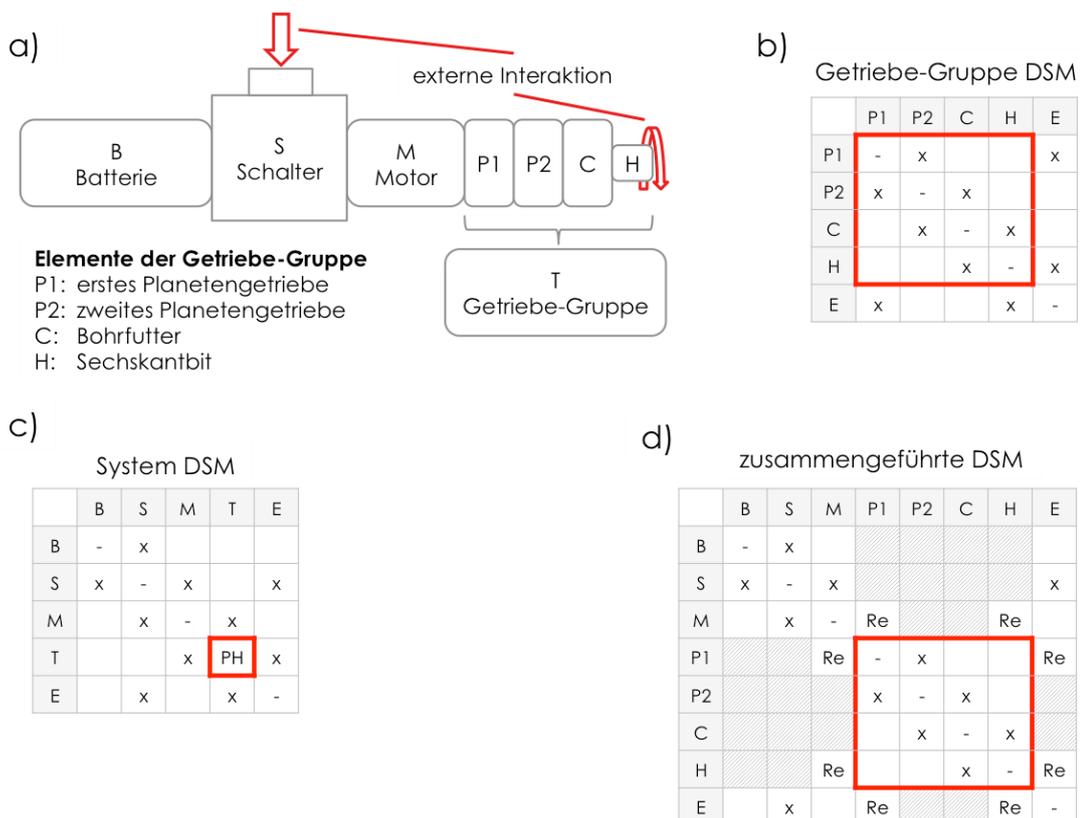


Abbildung 24: Methode "Distributed Modeling of Component DSM" zur Verringerung des Aufwands bei der Modellierung von Tracelinks (in Anlehnung an [Tilstra et al. 2009, S. 245])

Durch dieses Vorgehen wird eine vollständige DSM entwickelt ohne alle Elementkombinationen überprüfen zu müssen. In einem von den Autoren beschriebenen Beispiel ließen sich durch Anwendung der Methode so ca. 50 % des Aufwandes einspa-

ren, indem nicht alle 1056 sondern nur 508 Zellen der DSM auf Abhängigkeiten analysiert werden mussten [Tilstra et al. 2009, S. 245].

Einen sehr ähnlichen Ansatz wählen Biedermann et al. [Biedermann et al. 2010]. Sie beschreiben eine Methode, die Entwicklern Empfehlungen gibt, wie bei der Identifikation von Abhängigkeiten vorgegangen werden soll und sich die Aufwände bei dem Ausfüllen von Traceability Matrizen verringern lassen. Zu diesem Zweck wird zunächst zusätzlich zur eigentlich betrachteten, detaillierten Matrix (DSM) ein abstraktes System-Modell entwickelt und in einer DSM abgelegt. Im nächsten Schritt werden die Abhängigkeiten zwischen abstrakter und detaillierter DSM ermittelt (bspw. die Zugehörigkeiten von Bauteilen zu einem System) und in einer zusätzlichen DMM dokumentiert. In der abstrakten DSM werden dann die Abhängigkeiten innerhalb des Systems ermittelt und mit Hilfe der DMM auf die detaillierte DSM übertragen. Dies hat zur Folge, dass viele Einträge in der detaillierten DSM „vorausgefüllt“ werden. Diese müssen dann auf ihre Korrektheit überprüft und ggf. entfernt werden. Die erreichbaren Ersparnisse wurden im Rahmen einer vergleichenden Studie ermittelt und in Form einer Verringerung der notwendigen Zeit zur Erfassung der Abhängigkeiten um ca. 65 % angegeben [Biedermann et al. 2010, S. 310].

Die beschriebenen Methoden dienen der Unterstützung der Erfassung von Tracelinks und erfordern im Fall von Tilstra [Tilstra et al. 2009] und Biedermann [Biedermann et al. 2010] zusätzliche Verknüpfungen zwischen unterschiedlichen Abstraktionsebenen. Ausgehend von dieser Beobachtung wird im Folgenden untersucht, ob weitere Ersparnisse durch eine methodische Unterstützung möglich sind, die auf den zusätzlichen Arbeitsschritt der Verknüpfung von Abstraktionsebenen verzichtet und an Stelle dessen hierarchische Information verwendet. Zu diesem Zweck werden in Kapitel 5.2 Hierarchien, ihre Eigenschaften und Verwendung im Kontext der Entwicklung technischer Systeme diskutiert sowie wichtige Grundlagen zur Bewertung von Methoden zur Modellierungsunterstützung vorgestellt.

5.2 GRUNDLAGEN ZUR KONZEPTION UND BEWERTUNG DER METHODEN

Im vorliegenden Kapitel werden alle Grundlagen, die zur Konzeption einer Methode zur effizienten Erfassung von Tracelinks notwendig sind, zusammengefasst. Dazu wird in Kapitel 5.2.1 zunächst eine Analyse der Eigenschaften unterschiedlicher Arten von Hierarchien durchgeführt, um herauszuarbeiten, ob und unter welchen Bedingungen sich diese zur Konzipierung einer Methode zur Modellierungsunterstützung eignen. Anschließend wird dies auf die in Kapitel 2.2 vorgestellten Artefakte angewendet. Zusätzlich werden in Kapitel 5.2.2 unterschiedliche Granularitätsstufen der Traceability erläutert und Schlussfolgerungen gezogen, die für die Entwicklung des Algorithmus der Methode wichtig sind. Die mathematischen Grundlagen zur Berechnung des

Aufwandes bei der manuellen Erfassung von Tracelinks sind in Kapitel 5.2.3 beschrieben. In Kapitel 5.2.4 werden die Erkenntnisse abschließend kurz zusammengefasst und diskutiert.

5.2.1 STRUKTURANALYSE DER ARTEFAKTE DER ENTWICKLUNG MECHATRONISCHER SYSTEME

In Kapitel 2.2 wurden gängige Artefakte der Entwicklung mechatronischer Systeme vorgestellt und als Fazit herausgearbeitet, dass Abhängigkeiten zwischen den Elementen dieser Artefakte bestehen, die sich mit Hilfe durchgängiger Nachverfolgbarkeit abbilden lassen.

Im vorliegenden Kapitel werden diese hierarchischen Artefakte erneut aufgegriffen und hinsichtlich ihrer Struktur analysiert. Zu diesem Zweck werden in Kapitel 5.2.1.1 zunächst unterschiedliche Arten von Hierarchien vorgestellt, deren Eigenschaften charakterisiert und auf dieser Basis die Hypothese 1* entwickelt. Hintergrund dieser Hypothese ist, dass die Eigenschaften der verwendeten Hierarchien genutzt werden können, um eine Effizienzsteigerung bei der Erfassung von Tracelinks zu erzielen. Um festzustellen, auf welche der in Kapitel 2.2 vorgestellten Artefakte sich die Hypothese 1* anwenden lässt und welche Voraussetzungen dafür gelten, werden diese in Kapitel 5.2.1.2 hinsichtlich ihrer Strukturierungsansätze analysiert. Die Strukturierungsansätze sind dabei der Literatur entnommen und stellen den üblichen Aufbau der Artefakte und damit Empfehlungen für deren Strukturierung dar.

5.2.1.1 ARTEN VON HIERARCHIEN UND IHRE EIGENSCHAFTEN

Technische Systeme sind zumeist hierarchisch aufgebaut [van den Hamer und Lepoeter 1996, S. 9; VDI-Richtlinie 2206, S. 24; Lane 2006, S. 86-87]. Die Ursache dafür liegt in den zahlreichen Vorteilen, die sich daraus ergeben. So lassen sich bspw. in sich abgeschlossene Unterbaugruppen vormontieren, deren Entwicklung oder Fertigung auslagern bzw. deren Wiederverwendung forcieren. Letzteres wird bspw. in Form der modularen Quer- und Längsbaukästen in der Automobilindustrie bereits gelebt bzw. befindet sich aktuell in der Umsetzung. Darüber hinaus ergeben sich durch den hierarchischen oder modularisierten Aufbau der Systeme voneinander getrennte Entwicklungsaufgaben, die nur über deren Systemgrenzen miteinander in Kontakt stehen. Eine Verteilung auf mehrere Entwicklungsteams und die Begrenzung der Komplexität der einzelnen Entwicklungsaufgabe ist damit möglich.

Daher sind auch die meisten der zur Entwicklung dieser Systeme verwendeten Modelle bzw. Dokumente in Anlehnung an das System hierarchisch aufgebaut, wengleich sich die Arten der Hierarchien dabei unterscheiden können. Während das Ziel der Hierarchie bei Anforderungsdokumenten bspw. eine Sortierung sein kann, wer-

den Funktionsmodelle in unterschiedlichen Detaillierungs- oder Abstraktionsstufen aufgebaut, um schlussendlich elementare Funktionen zu identifizieren, die durch einfache Wirkprinzipien umgesetzt werden können [Pahl et al. 2007, S. 243].

Wie in Kapitel 3.7.1 dargestellt, mangelt es bei der Erstellung der durchgängigen Nachverfolgbarkeit an Methoden, mit denen eine effiziente Modellierung von Tracelinks unterstützt wird. Im Folgenden werden deshalb unterschiedliche Arten von Hierarchien vorgestellt und hinsichtlich ihrer Eigenschaften analysiert, um diese zwecks einer Effizienzsteigerung bei der Tracelink-Modellierung einzusetzen.

Nach Allen ist eine Hierarchie eine Sammlung von Elementen, die zueinander in asymmetrischen Beziehungen stehen. Diese Asymmetrie bezieht sich darauf, dass die Elemente in Ebenen angeordnet sind und die Beziehungen zwischen diesen von der höheren zu der niedrigeren gerichtet sind. Die Zuordnung der Elemente zu den Ebenen kann dabei nach beliebigen Kriterien erfolgen, die teilweise auch zeitgleich angewendet werden. So können höhere Ebenen bspw. aufgrund folgender Auswahl von Kriterien über niedrigeren Ebenen angeordnet sein [Allen 2011]:

- Sie sind der Kontext der unteren Ebenen.
- Sie stellen technische Randbedingungen für die unteren Ebenen dar.
- Sie verhalten sich langsamer, bei einer geringeren Frequenz, als untere Ebenen.
- Sie enthalten bzw. bestehen aus unteren Ebenen.

Grundsätzlich ist anzumerken, dass das Konzept der Hierarchie sehr allgemein ist und in einer Vielzahl von unterschiedlichen Wissenschaften Anwendung findet. Im Kontext des Systems Engineerings kommen sog. Inklusionshierarchien²⁰ mit ihren unterschiedlichen Ausprägungen zur Anwendung. Bei einer Inklusionshierarchie stellen die Elemente einer höheren Ebene einen Container für Elemente niedrigerer Ebenen dar (siehe bspw. Abbildung 25). Zur Verdeutlichung dieser Art von Hierarchie wird häufig das Beispiel der Matroschka-Puppen herangezogen, bei denen die äußere, größte Puppe eine kleinere enthält, die wiederum eine noch kleinere Puppe beinhaltet. Dieses Beispiel stellt jedoch einen Sonderfall der Inklusionshierarchie dar, da immer genau ein Element im nächst höheren enthalten ist [Simon 1973, S. 5]. Im Normalfall können in den übergeordneten Elementen eine beliebige Anzahl an untergeordneten Elementen beliebiger Art enthalten sein [Grobstein 1973, S. 32; Lane 2006, S. 84-85; Pumain 2006, S. 3-4].

²⁰ Auch verschachtelte Hierarchie genannt.

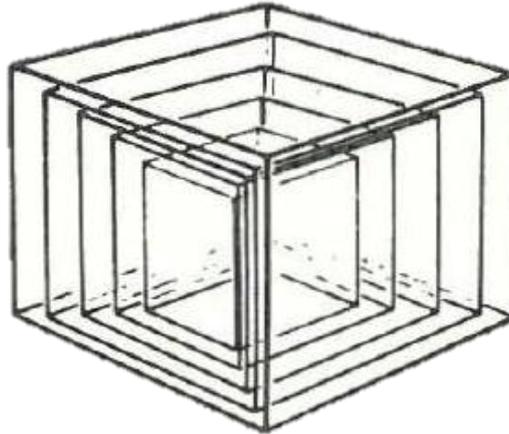


Abbildung 25: Beispiel einer Inklusionshierarchie [Grobstein 1973, S. 32]

Die bekanntesten und am häufigsten verwendeten Arten der Inklusionshierarchien sind die *subsumtive Inklusionshierarchie*²¹ und die *kompositionelle Inklusionshierarchie*. Mit Hilfe der subsumtiven Inklusionshierarchie können Objekte von allgemein bis spezifisch klassifiziert werden. Sie wird auch „is a“-Hierarchie genannt und entspricht der Generalisierung im Blockdefinitionsdiagramm der SysML [Weilkiens 2006, S. 158]. Ein anschauliches Beispiel dafür ist die Kategorisierung der Tierwelt, ausgehend von dem sehr abstrakten Element „Tier“, welches das Element „Säugetier“ enthält, welches wiederum die Elemente „Hund“, „Katze“ usw. enthält (siehe Abbildung 26).

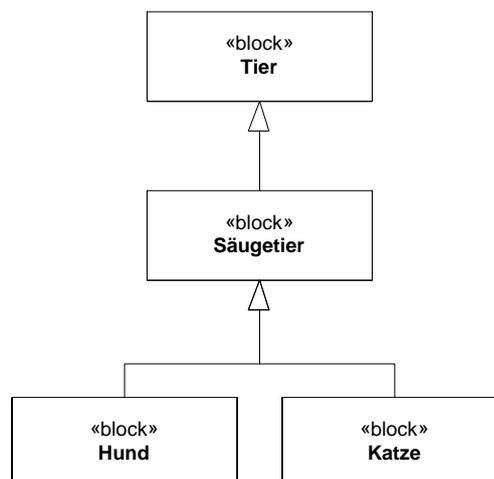


Abbildung 26: Kategorisierung der Tierwelt mit Hilfe einer subsumtiven Inklusionshierarchie

Die kompositionelle Inklusionshierarchie ist umgangssprachlich auch als „is part of“ oder „is composed of“-Hierarchie bekannt. Sie wird verwendet, um Systeme zu strukturieren [Lane 2006, S. 85-86; Pumain 2006, S. 4] und ist damit die am meisten verwendete Art der Hierarchie im Systems Engineering. Dabei wird ein System so lange

²¹ Auch Klassifikationshierarchie genannt.

in seine Subsysteme gegliedert, bis man zu einer Ebene gelangt, die sich nicht weiter unterteilen lässt bzw. nicht mehr sinngemäß ist. Im Blockdefinitionsdiagramm der SysML entspricht dies der Komposition bzw. Aggregation²² eines Systems aus seinen Bestandteilen [Weilkiens 2006, S. 155-157].

Beiden vorgestellten Arten von Inklusionshierarchien ist gemein, dass die hierarchischen Relationen, die verwendet werden, um die unterschiedlichen Ebenen der Hierarchien miteinander zu verbinden, transitiv sind [Hybertson 2009, S. 90]. In Kombination mit den ebenfalls transitiven Tracelinks [Egyed und Grünbacher 2002, S. 164] kann diese Eigenschaft bei der Modellierung von Tracelinks ausgenutzt werden. In Abbildung 27 ist die anhand der drei Elemente A1, B1 und B1.1 dargestellt: existiert ein Tracelink zwischen den Elementen A1 und B1.1 (durchgängiger Pfeil), so kann ein Tracelink unter Ausnutzung der Transitivität zwischen den Elementen A1 und B1 abgeleitet werden (gestrichelter Pfeil). Diese Eigenschaft kommt bspw. bei der Visualisierung von Traceability-Modellen zum Einsatz, wenn nicht alle Hierarchieebenen dargestellt, aber trotzdem alle Abhängigkeiten abgebildet werden sollen [Abma 2009, S. 69; Beier et al. 2012, S. 16].

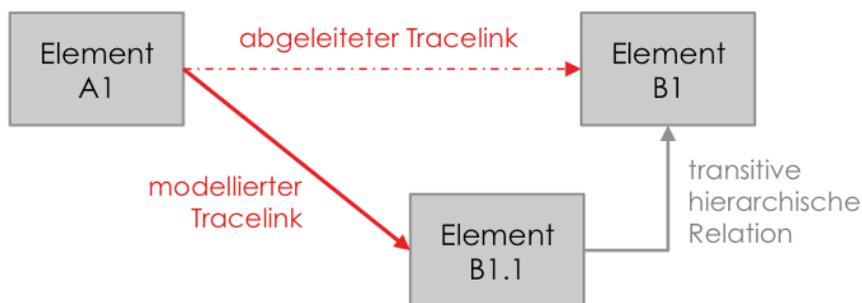


Abbildung 27: Ableitung von Tracelinks durch Ausnutzung der Transitivität

Im Kontext der Erfassung von Tracelinks kann diese Transitivität nun umgekehrt ausgenutzt werden, um die Entscheidung, keinen Tracelink zwischen A1 und B1 zu modellieren (da keine Abhängigkeit vorliegt) auf die Elementekombination von A1 und B1.1 zu übertragen²³. Diese Eigenschaft ist Basis für die Hypothese 1* auf deren Grundlage im Folgenden die EcoTracing-Methode entwickelt wird:

²² Der Unterschied zwischen Komposition und Aggregation besteht darin, dass bei ersterer die Elternelemente existenziell für ihre Kinder-elemente verantwortlich sind. Abgesehen davon verfügen sie über die gleichen Eigenschaften. Diese feine Unterscheidung gibt es bei der kompositionellen Inklusionshierarchie nicht, weshalb sowohl Komposition als auch Aggregation einer kompositionellen Inklusionshierarchie entsprechen.

²³ Diese Eigenschaft wird auch bei dem sog. Trace Enhancement ausgenutzt, bei dem zunächst nur auf hoher Ebene Tracelinks modelliert werden und erst bei Bedarf eine Detaillierung vorgenommen wird. Diese Detaillierung erfolgt dann nur zwischen den Kindern von Elementen, bei denen ein Tracelink vorliegt (siehe Kapitel 5.1) [Egyed et al. 2009, S. 249].

Weist ein Element in Artefakt B eine Abhängigkeit zu einem Element in Artefakte A auf, so müssen auch alle seine Elternelemente eine Abhängigkeit zu dem Element in Artefakt A bzw. (soweit vorhanden) seinen Elternelementen aufweisen.

Um diese theoretisch hergeleitete Hypothese empirisch für die Erfassung von Tracelinks zu validieren, wurde eine Studie durchgeführt. Der Aufbau, die Durchführung und die Ergebnisse dieser Studie sind in Kapitel 5.3 zusammengefasst.

Grundsätzlich ist es somit möglich, die Entscheidung über Abhängigkeiten auf abstrakter hierarchischer Ebene zu treffen und diese ebenfalls auf die zugehörigen Kinderelemente zu übertragen, um Aufwände bei der Modellierung von Tracelinks einzusparen. Allerdings sind diese Entscheidungen nicht trivial, da sie auf Basis der auf abstrakter Ebene zur Verfügung stehenden Informationen bzw. Eigenschaften der betrachteten Elternelemente getroffen werden müssen. Welche Informationen dies in Abhängigkeit der verwendeten Art der Hierarchie sind, wird im Folgenden erläutert.

So wird, wie zuvor beschrieben, ein Elternelement in einer kompositionellen Inklusionshierarchie durch die Kinderelemente der jeweils niedrigeren Ebene aufgebaut. Die Eigenschaften des Elternelements bestimmen sich aus denen der einzelnen Kinderelemente [Tudorache 2006, S. 55-56; Allen 2011] und deren Zusammenwirken. Dieser Zusammenhang wird „upward-causation“ genannt [Raia 2005, S. 297]. Bildlich gesprochen bestehen somit die Eigenschaften des Elternelements aus zwei Teilen: der Summe der Eigenschaften seiner Kinderelemente und den Eigenschaften, die sich aus deren Zusammenwirken ergeben. In Abbildung 28 ist dies auf der linken Seite schematisch für die kompositionelle Inklusionshierarchie dargestellt. Die Eigenschaften (dargestellt durch farbige Quadrate) der Elemente ergeben sich aus denen ihrer Kinderelemente sowie durch deren Zusammenspiel. Kennt man also die Gesamtfunktion „Spiegelglas verstellen“, mit der das Außenspiegelglas an die Position und Größe des Fahrers angepasst werden kann, kann man Rückschlüsse hinsichtlich der Teilfunktionen „Spiegelglas in x-Richtung verstellen“ und „Spiegelglas in y-Richtung verstellen“ ziehen und bereits auf Gesamtfunktionsebene entscheiden, dass bspw. eine elektronische Steuerung oder eine elektrische Versorgung gegeben sein muss²⁴. Bei der Abhängigkeitsanalyse eines Artefakts, welches mit einer kompositionellen Inklusionshierarchie strukturiert wurde, ist es somit (theoretisch) möglich, Entscheidungen über Abhängigkeiten auf Basis dieser aggregierten Eigenschaften des jeweiligen Elternelements zu treffen.

²⁴ Voraussetzung ist natürlich, dass es sich um einen elektrischen Außenspiegel handelt.

Im Kontext der Entwicklung eines technischen Systems stellt eine subsumptive Inklusionshierarchie hingegen eine Art Kategorisierung eines Artefakts dar. Allgemeine Elemente auf hoher Ebene werden durch spezifischere, untergeordnete Elemente detailliert. Ähnlich der Kategorisierung der Tierwelt (siehe Abbildung 26) können so bspw. spezifische Anforderungen hinsichtlich ihrer Hauptmerkmale „Geometrie“, „Energie“, „Fertigung“ usw. kategorisiert werden [Pahl et al. 2007, S. 220]. Bei dieser Art der Hierarchie stehen im Vergleich zur kompositionellen Inklusionshierarchie weniger Informationen auf abstrakter Hierarchieebene zur Verfügung, da die Eigenschaften der Elternelemente in der Hierarchie nicht aus denen der Kinderelemente aggregiert werden sondern durch deren Generalisierung lediglich der „gemeinsame Nenner“ der Eigenschaften abgebildet wird (in Abbildung 28 auf der rechten Seite durch das rote Quadrat dargestellt).

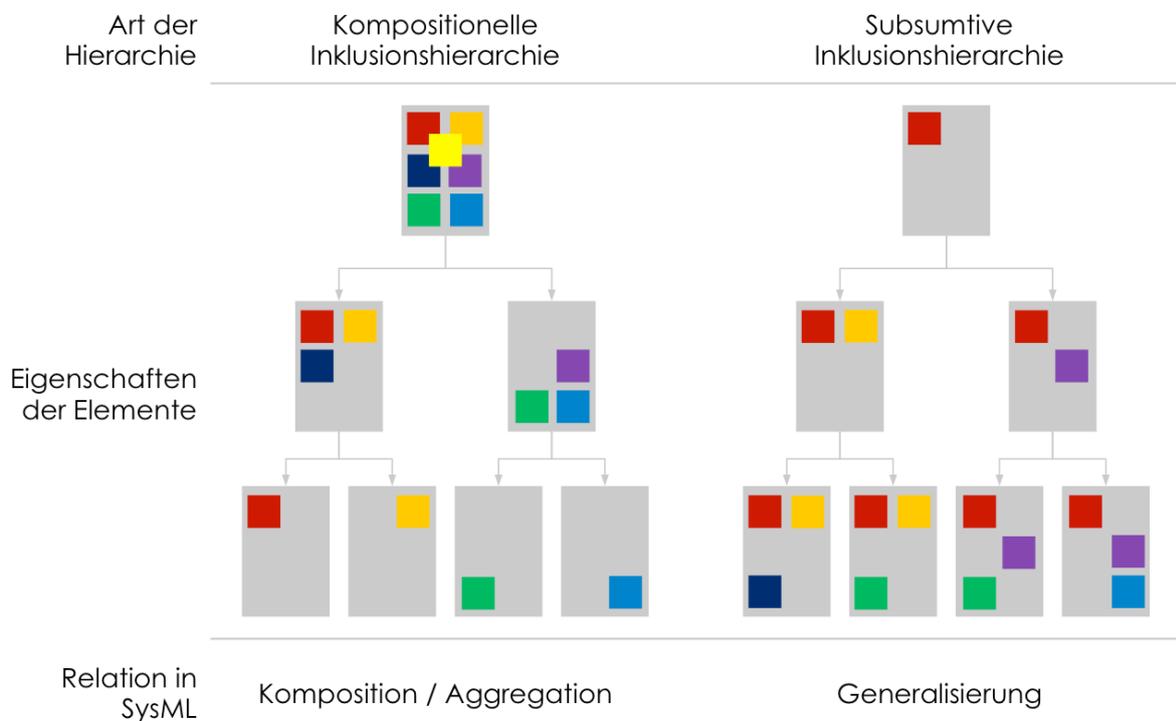


Abbildung 28: Schematische Darstellung der Zusammensetzung von Eigenschaften (visualisiert durch farbige Quadrate) in kompositionellen (links) und subsumptiven Inklusionshierarchien (rechts)

Bei einer Abhängigkeitsanalyse können anhand eines übergeordneten Elements somit nur Abhängigkeiten identifiziert werden, die sich aus diesem Nenner der Eigenschaften ergeben. Für die Identifikation aller weiteren Abhängigkeiten, die sich aus den konkretisierten Eigenschaften der Kinderelemente ergeben, ist deren detaillierte Kenntnis notwendig. So kann bspw. auf Basis der Kategorie „geometrische Anforderungen“ noch nicht entschieden werden, an welche Bauteile des Außenspiegels sich die enthaltenen Anforderungen richten.

Es gibt somit zwei Arten von Hierarchien, die während der Entwicklung technischer Systeme eine Rolle spielen: subsumtive und kompositionelle Inklusionshierarchien. In diesem Kapitel wurde herausgearbeitet, dass es im Fall der kompositionellen Inklusionshierarchie bei Kenntnis des Elternelements theoretisch möglich ist, auf dieser abstrakten Ebene Entscheidungen über Abhängigkeiten ganzer Hierarchieäste zu treffen. Bei der Abhängigkeitsanalyse subsumtiver Inklusionshierarchien ist hingegen die Kenntnis der Kinderelemente und deren Eigenschaften Voraussetzung.

5.2.1.2 ARTEFAKTE ZUR ENTWICKLUNG MECHATRONISCHER SYSTEME

Die bereits in Kapitel 2.2 kurz vorgestellten Artefakte werden im vorliegenden Kapitel erneut aufgegriffen und hinsichtlich ihres Strukturierungsansatzes analysiert. Ziel ist es zu beurteilen, welche Arten von Hierarchien zur Strukturierung üblicherweise genutzt werden, um die Voraussetzungen für die Anwendbarkeit der Hypothese 1* für das jeweilige Artefakt zu klären.

Anforderungsartefakte

Anforderungen werden, wie in Kapitel 2.2.1 beschrieben, in Anforderungsartefakten dokumentiert. Für den Aufbau dieser Artefakte gibt es dabei keine allgemeingültige Vorgabe. Vielmehr gibt es Empfehlungen für unterschiedliche hierarchische Strukturierungsansätze, die unternehmensspezifisch ausgeprägt werden müssen [Hood und Wiebel 2005, S. 63]. Um diese zu charakterisieren, können drei grundsätzliche Prinzipien verwendet werden [Yeh und Zave 1980, S. 1078]:

- Konkretisierung (Gegenteil: Abstraktion): aus allgemeinen Anforderungen werden spezielle abgeleitet
- Partitionierung (Gegenteil: Komposition): eine Anforderung wird in mehrere zerlegt
- Projektion (Gegenteil: Integration): eine Gesamtsicht auf das Produkt wird aus mehreren Teilsichten zusammengesetzt

Beim Prinzip der Konkretisierung werden zunächst sog. übergeordnete Anforderungen definiert²⁵, die das übergeordnete Ziel des Systems in seinem Umfeld beschreiben [NASA STI 2007, S. 41]. Diese Anforderungen werden anschließend Schritt für Schritt mit Hilfe von konkreten technischen²⁶, funktionalen und Leistungsanforderungen detailliert. Man spricht in diesem Zusammenhang von der Verfeinerung der An-

²⁵ Auch „High-Level Requirements“, „Erwartungen“ oder „Business Anforderungen“ genannt.

²⁶ Technische Anforderungen sind abgestimmte, validierte Anforderungen, die eine vollständige Beschreibung des Problems darstellen und zur Entwicklung des Lösungsansatzes genutzt werden können [NASA STI 2007, S. 41].

forderungen, die einer Generalisierung im Blockdefinitionsdiagramm der SysML entspricht (siehe Abbildung 29)²⁷. Entsprechend der Analyse in Kapitel 5.2.1.1 handelt sich bei diesem Strukturierungsansatz um eine subsumptive Inklusionshierarchie, da allgemeine Anforderungen detaillierteren übergeordnet sind. So kann bspw. die detaillierte Anforderung bzgl. der Leistung einer Heizfolie die allgemeinere Anforderung „Außenspiegelheizung vorsehen“ verfeinern.

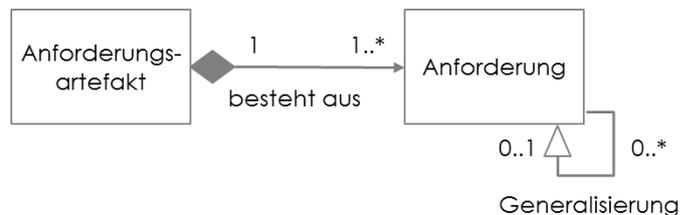


Abbildung 29: Strukturierung des Anforderungsartefakts mit Hilfe der Konkretisierung

Die Partitionierung wird hingegen durchgeführt, um eine Anforderung in mehrere zu zerlegen. So kann bspw. eine Anforderung an das Gesamtgewicht des Außenspiegels in mehrere Anforderungen zerlegt werden, die unterschiedlichen Systeme und Bauteile adressieren. Im Blockdefinitionsdiagramm lässt sich diese Abhängigkeit zwischen übergeordneter Anforderung und untergeordneten Anforderungen mit Hilfe der Komposition abbilden (siehe Abbildung 30). Diese Zerlegung führt somit zu einer kompositionellen Inklusionshierarchie (vgl. Kapitel 5.2.1.1).

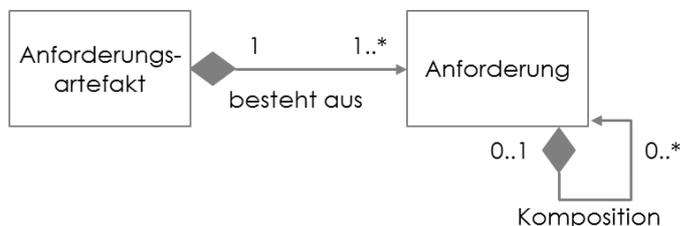


Abbildung 30: Strukturierung des Anforderungsartefakts mit Hilfe der Partitionierung

Bei der Projektion werden hingegen mehrere unterschiedliche Sichten auf das zu entwickelnde System gebildet, die sich zu einer Gesamtsicht zusammenführen lassen. Diese Sichten können losgelöst von dem zu entwickelnden System definiert werden, wie es bspw. beim FURPS-System (Functionality, Usability, Reliability, Performance und Supportability) [Weilkiens 2006, S. 41] oder bei der Strukturierung nach Hauptmerkmalen (z. B. Gestalt, Ergonomie, Fertigung usw.) umgesetzt ist [Pahl et al. 2007, S. 220]. Alternativ ist der Bezug zu dem zu entwickelnden System möglich, wenn dieses bereits hinreichend bekannt ist (z. B. bei Anpassungskonstruktionen). In diesem Fall werden

²⁷ Um den Abgleich mit den in Kapitel 5.2.1.1 beschriebenen Hierarchien zu ermöglichen wird für die Darstellung der Anforderungsartefakte in Abbildung 29 bis Abbildung 31 das Blockdefinitionsdiagramm und nicht das Anforderungsdiagramm der SysML verwendet.

häufig die Funktions-, System- oder Produktstruktur als Basis für Sichten und damit für die Benennung von Kategorien für die Strukturierung der Anforderungen verwendet [Pahl et al. 2007, S. 216].

Unabhängig von der Art stellen diese Sichten bzw. Kategorien dabei eine abstrakte Beschreibung der darin enthaltenen Anforderungen dar (z. B. „funktionale Anforderungen“ oder „Anforderungen an den Außenspiegel“), was mit Hilfe der Generalisierung abgebildet werden kann (siehe Abbildung 31).



Abbildung 31: Strukturierung des Anforderungsartefakts mit Hilfe der Projektion

Bei der Projektion handelt es sich somit um eine subsumtive Inklusionshierarchie, da die zu sortierenden Anforderungen einer Sicht bzw. Kategorie zugeordnet werden. Die Anforderung „Spiegelverstellung bei Rückwärtsfahrt“ kann über die „is-a“ Beziehung bspw. in die Kategorien „funktionale Anforderungen“ oder „Anforderungen an den Außenspiegel“ eingeordnet werden.

Zusammenfassend ist als Ergebnis der Analyse der Anforderungsartefakte wichtig zur Kenntnis zu nehmen, dass die Partitionierung von Anforderungen mit einer kompositionellen Inklusionshierarchie abgebildet wird. Sowohl die Konkretisierung als auch die Projektion werden hingegen in subsumtiven Inklusionshierarchien strukturiert. Entsprechend der Bewertung der subsumtiven Inklusionshierarchie in Kapitel 5.2.1.1 können somit für diese beiden Ansätze die Eigenschaften der Kinderelemente nicht umfassend auf übergeordneter Ebene bewertet werden und bei Abhängigkeitsanalysen ist die Kenntnis der Kinderelemente und ihrer Eigenschaften notwendig.

Funktionsartefakte

In Kapitel 2.2.2 wurden Funktionen als lösungsneutrale Beschreibungsform technischer Systeme eingeführt. Die Vorgehensweise zur Spezifikation der Funktionen ist dabei von abstrakt nach detailliert (top-down): die vergleichsweise abstrakten Gesamtfunktion wird schrittweise in ihre Teilfunktionen heruntergebrochen [Pahl et al. 2007, S. 242-248; Ponn und Lindemann 2008, S. 58]. Durch das Herunterbrechen werden die Funktionen immer konkreter bis Wirkprinzipien bzw. Lösungselemente für deren Erfüllung identifiziert werden können [Pahl et al. 2007, S. 255]. Die Frage der Granularität („Auf welcher Ebene hört man auf zu modellieren?“) ist dabei nicht einfach zu beantworten [Chakrabarti und Bligh 2001, S. 500]. So spielen bspw. die Art des zu entwickelnden Systems und der Entwicklung eine große Rolle. Während bei Neuentwicklungen eine höhere Granularität notwendig ist, sind bei evolutionären Weiterentwicklungen auch abstraktere Funktionsstrukturen für die Kommunikation der Ent-

wickler ausreichend. Genauso gibt es Unterschiede zwischen den Disziplinen. Sobald festgelegt wird, wie eine Funktion umgesetzt werden soll, kann es bspw. in der Softwareentwicklung notwendig sein, die Funktionen weiter herunterzubrechen, während die abstrakte Darstellung für die mechanische Konstruktion vollkommen ausreichend sein kann.

Solange die abstrakte Hauptfunktion überhaupt heruntergebrochen wird, stellen die Funktionsartefakte eine Verbindung zwischen der abstrakten Aufgabenstellung und den Detailentwicklungen der unterschiedlichen Disziplinen her [Erden et al. 2008, S. 147]. Ziel ist dabei, neben der lösungsneutralen Beschreibung des zu entwickelnden Systems, auch die Partitionierung der Entwicklungsaufgaben auf unterschiedliche Abteilungen bzw. Entwickler, was sich auf Basis der Teilfunktionen gut durchführen lässt [Leenders et al. 2007, S. 168; Pahl et al. 2007, S. 243].

Grundsätzlich gibt es drei unterschiedliche Strukturierungsansätze, um die Gesamtfunktion und ihre Teilfunktionen zu erfassen. Eine Listendarstellung eignet sich bspw. zur Dokumentation von relativ unabhängigen Funktionen oder zur Sammlung aller Funktionen als vorbereitende Vorstufe für die Erarbeitung anderer Darstellungen [Ponn und Lindemann 2008, S. 61]. Eine Erweiterung der Liste ist die Hierarchie, in der die Eltern-Kind-Beziehung der Gesamtfunktion und ihrer Teilfunktionen, aus denen sie besteht, abgebildet werden kann [Ponn und Lindemann 2008, S. 62]. Die Gesamtfunktion des Außenspiegels „Rückblick gewährleisten“ kann so übergeordnet zu seinen Teilfunktionen „Eis beseitigen“, „Rückblickrichtung verstellen“, „Außenspiegel steuern“ usw. dargestellt werden. Die Beziehung zwischen über- und untergeordneter Funktion wird im Funktionsartefakt, wie in Abbildung 32 dargestellt, mit einer Komposition abgebildet (die Gesamtfunktion besteht aus den Teilfunktionen), die einer kompositionellen Inklusionshierarchie entspricht (vgl. Kapitel 5.2.1.1).

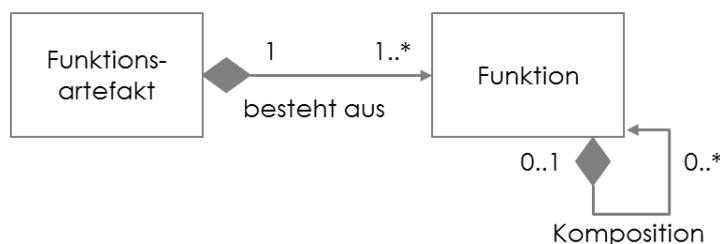


Abbildung 32: Strukturierung des Funktionsartefakts mit Hilfe einer Hierarchie

Dieser Strukturierungsansatz eignet sich zur Erfassung der Funktionen, da die Anzahl der dargestellten Hierarchieebenen frei gewählt und so leicht der gewünschte Abstraktionsgrad von abstrakter Gesamtfunktion bis zur detaillierten Teilfunktion eingestellt werden kann. Dies gilt, solange die Abhängigkeiten zwischen den Teilfunktionen der unterschiedlichen Hierarchieebenen eine untergeordnete Rolle spielen. Sind diese jedoch für die Analyse der Funktionen und damit für die Weiterentwicklung zur Lö-

sung wichtig, eignet sich die Funktionsstruktur, die häufig in Blockdiagrammen dargestellt wird, besser als die Hierarchie (siehe Abbildung 33).

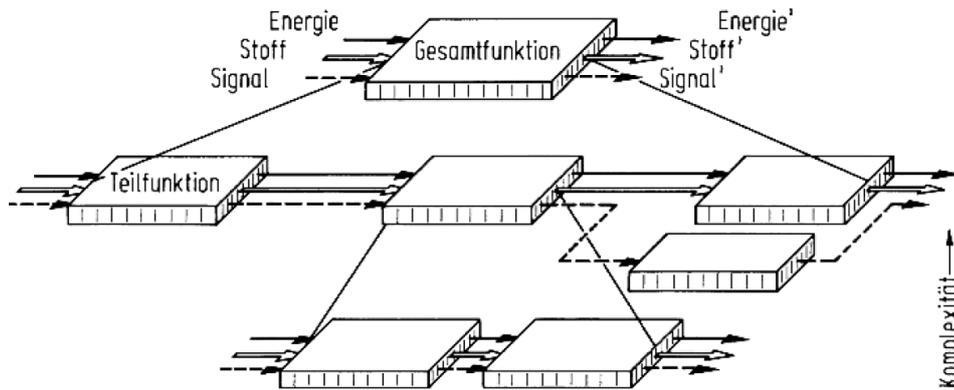


Abbildung 33: Dreidimensionale Blockdarstellung einer Funktionsstruktur [Pahl et al. 2007, S. 45]

Bei dieser Darstellungsform werden alle Funktionen, die sich auf einer hierarchischen Ebene befinden, sowie deren Abhängigkeiten dargestellt. Um die unterschiedlichen hierarchischen Ebenen gleichzeitig zu visualisieren, kann bspw. eine dreidimensionale Blockdarstellung wie in Abbildung 33 gewählt werden.

Hinsichtlich des Aufbaus des Funktionsartefakts bedeutet die Verwaltung von Funktionsstrukturen eine Erweiterung des in Abbildung 32 beschriebenen Modells um die Möglichkeit, Verknüpfungen zwischen den Funktionen abbilden zu können (siehe Abbildung 34).

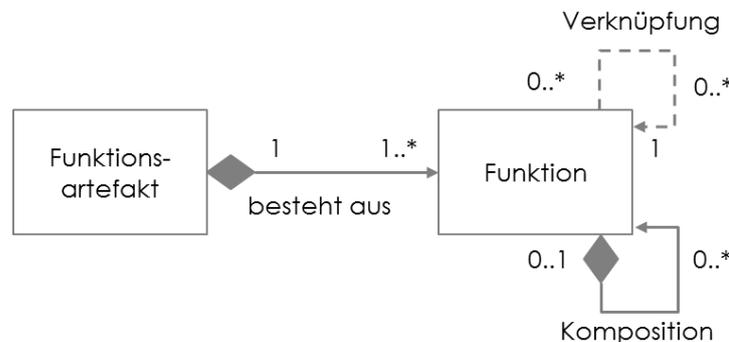


Abbildung 34: Strukturierung des Funktionsartefakts mit Hilfe einer Funktionsstruktur

Als wichtigste Erkenntnisse kann zusammengefasst werden, dass Funktionsartefakte der lösungsneutralen Beschreibung der Funktionen eines zu entwickelnden Systems dienen. Zu ihrer Strukturierung stehen drei Ansätze zur Verfügung: Liste, Hierarchie und Funktionsstruktur. Während die erste lediglich der Sammlung von Funktionen dient, werden die Hierarchie und die Funktionsstruktur Top-Down entwickelt, indem eine zunächst definierte Gesamtfunktion schrittweise in ihre Teilfunktionen heruntergebrochen wird. Bei der so entstehenden hierarchischen Struktur bestehen die übergeord-

neten aus den untergeordneten Funktionen, was einer kompositionellen Inklusionshierarchie entspricht.

Verhaltensartefakte

Wie in Kapitel 2.2.3 beschrieben, dienen Verhaltensartefakte der Simulation des Verhaltens von Systemen. In diesem Zusammenhang sind objektorientierte Modellierungsansätze in der Systementwicklung sehr verbreitet [Vajna 2009, S. 313]. Im Mittelpunkt dieser Ansätze steht eine textuelle Objektbeschreibungssprache [Janschek 2010, S. 132], mit deren Hilfe die Objekte beschrieben werden. Dazu werden Differenzial-, algebraische und diskrete Gleichungen genutzt [Vajna 2009, S. 314]. Darüber hinaus besitzen sie Parameter, die während der Simulation konstant sind, Variablen und Schnittstellen, über welche die Objekte miteinander kommunizieren [Drogies 2006, S. 75]. Dabei werden Objekte verwendet, um (Teil-)Systeme darzustellen. Mit Hilfe von Hierarchisierung können sie strukturiert und wiederum in ihre Subsysteme zerlegt werden. Dabei gilt, dass die Subsysteme möglichst unabhängig voneinander sein sollten und die Anzahl der Schnittstellen zwischen ihnen zu minimieren ist. Das Verhaltensmodell des Außenspiegels könnte so u. a. in die Systeme „Außenspiegelheizung“ und „Spiegelglasverstellung“ strukturiert werden, in denen das jeweilige Verhalten wiederum mit Objekten aus Bibliotheken nachgebildet wird. Die Kommunikation zwischen den Objekten erfolgt innerhalb der einzelnen Hierarchiestufen mittels Signal- und Energieflüssen, mit denen die Objekte bzw. (Teil-) Systeme miteinander verbunden werden. Hinsichtlich des Aufbaus von Verhaltensartefakten bedeutet dies, dass die abgebildeten Systemelemente mit Hilfe einer Komposition aufgebaut werden. Zwischen diesen Systemelementen bestehen jeweils auf einer Ebene Abhängigkeiten, die unterschiedlich ausgeprägt sein können (siehe Abbildung 35). Bei der so entstehenden Hierarchie handelt es sich um eine kompositionelle Inklusionshierarchie.

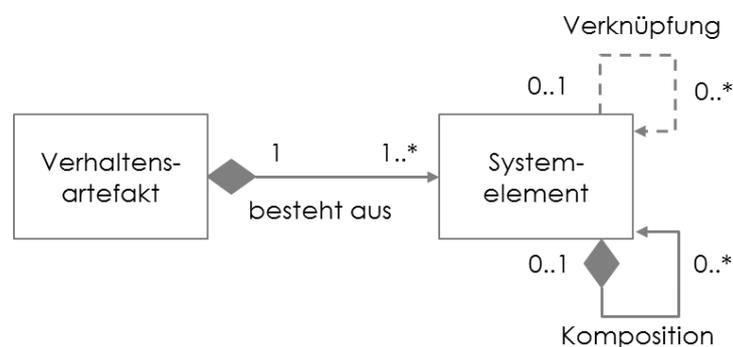


Abbildung 35: Strukturierung eines Verhaltensartefakts

Zusammenfassend ist für Verhaltensartefakte festzuhalten, dass diese hierarchisch aufgebaut werden. Die Struktur der Artefakte orientiert sich dabei an dem Aufbau des zu entwickelnden Systems, indem dieses in seine Subsysteme aufgeteilt und mit

Hilfe von Objekten (überwiegend aus Bibliotheken) nachgebildet wird. Bei der so entstehenden kompositionellen Inklusionshierarchie bestehen somit übergeordnete aus untergeordneten Objekten.

Produktstrukturartefakte

In Kapitel 2.2.4 wurde bereits beschrieben, dass Produkt- bzw. Systemstrukturen der Verwaltung von Systemen, Baugruppen sowie Bauteilen, Softwareelementen und Informationselementen dienen. Die Art und Weise der Gliederung - also welches Bauteil welcher Baugruppe bzw. welchem System zugeordnet wird - orientiert sich in der industriellen Anwendung am Zweck der Produktstruktur. Typische Ansätze, die auch in Kombination vorkommen können, sind u. a. die Strukturierung des Produkts nach:

- **Entwicklungsdisziplinen:** Produkte werden in Systeme unterteilt, die sich aufgrund ihrer Entwicklung voneinander unterscheiden. Diese können auf verschiedene Entwicklungsabteilungen aufgeteilt werden. In der Fahrzeugentwicklung kann so bspw. ein Kfz in die Systeme Rohkarosse, Interieur, Verkabelung, Antrieb, Unterboden und Fahrwerk unterteilt und verteilt entwickelt werden.
- **Funktionen:** Bei diesem Ansatz werden Bauteile, Baugruppen und Systeme nach ihren jeweiligen Funktionen klassifiziert und diesen zugeordnet [Eigner und Stelzer 2009, S. 28]. Dabei wirkt sich insbesondere die Tatsache, dass eine eindeutige Funktionszuordnung schwierig ist (entweder weil die Funktion eines Bauteils nicht direkt ersichtlich ist oder weil mehrere Funktionen unterstützt werden), als negativ aus.
- **Modularisierungsgesichtspunkten:** Bauteile werden den Baugruppen auf eine Art und Weise zugeordnet, dass sich möglichst in sich abgeschlossene Module ergeben, die nur eine geringe Anzahl Schnittstellen zu anderen Modulen aufweisen. Dieser Ansatz soll insbesondere bei komplexen Systemen zur verbesserten Übersicht und Transparenz beitragen [Vajna 2009, S. 206].
- **Fertigungsgesichtspunkten:** Bauteile werden nach ihrem Fertigungsverfahren zusammengefasst. Diese Art der Strukturierung findet insbesondere in der Produktionsplanung Anwendung [Eigner und Stelzer 2009, S. 28].
- **Montagegesichtspunkten:** Die Bauteile werden zu vormontierbaren Baugruppen zusammengefasst, die bei der späteren Montage zum Endprodukt zusammengefügt werden [Schuh et al. 2012, S. 118].

Somit ist es möglich, dasselbe Produkt nach unterschiedlichen Gesichtspunkten zu gliedern und zu jeweils unterschiedlich ausgeprägten Produktstrukturen zu gelangen. Dies ist in Abbildung 36 anhand der beispielhaften Strukturierung eines Außenspiegels dargestellt: im Fall a) wird der Außenspiegel hinsichtlich unter Montagegesichtspunkten strukturiert. Die Heizfolie wird in diesem Fall der Glaseinheit zugeordnet, da diese

auf das Spiegelglas aufgeklebt wird. In Fall b) wird der Außenspiegel hinsichtlich seiner Funktionen strukturiert. In diesem Fall wird die Heizfolie der Spiegelheizung zugeordnet.

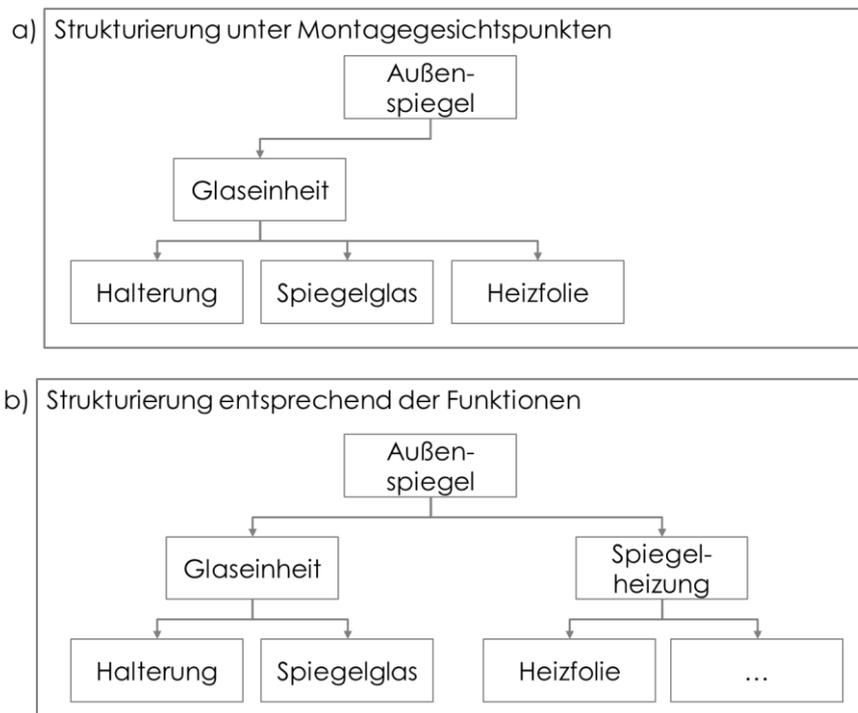


Abbildung 36: Ausschnitt einer beispielhaften Strukturierung eines Außenspiegels a) unter Berücksichtigung von Montagegesichtspunkten und b) entsprechend der Funktionen

Unabhängig von der Wahl des Strukturierungsansatzes wird die Produktstruktur immer aus Systemen, Baugruppen, sowie Bauteilen und Software- und Informationselementen aufgebaut. Erstere sind dabei die in der Hierarchie übergeordneten Knoten, wobei sie selbst wieder Systemen oder Baugruppen untergeordnet sein können. Bauteile liegen auf der niedrigsten hierarchischen Ebene. Die Beziehungen zwischen den einzelnen Ebenen der Hierarchie werden dabei mit einer Aggregation gebildet [Weilkiens 2006, S. 118] (siehe Abbildung 37). Die Hierarchie der Produktstruktur entspricht damit einer kompositionellen Inklusionshierarchie.

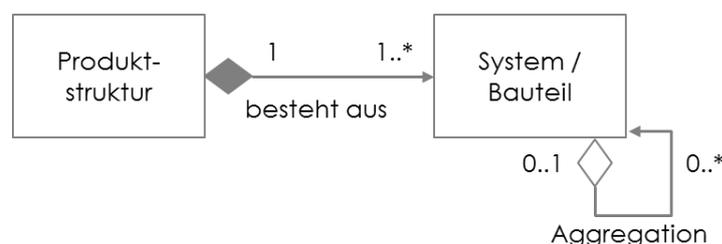


Abbildung 37: Schematischer Aufbau einer Produktstruktur

5.2.1.3 ZUSAMMENFASSUNG UND DISKUSSION

In Kapitel 5.2.1.1 wurden zunächst die zur Strukturierung von Artefakten verbreiteten subsumtive und die kompositionelle Inklusionshierarchie vorgestellt und ihre Eigenschaften vor dem Hintergrund effizienter Tracelink-Modellierung analysiert.

Als Ergebnis dieser Analyse wurde herausgearbeitet, dass bei Inklusionshierarchien aufgrund hierarchischer Transitivität Entscheidungen, die auf einer abstrakten Ebene für Elternelemente bzgl. Abhängigkeiten zu anderen Elementen getroffen werden, auf deren Kinderelemente übertragen werden können. Dies ist vorteilhaft, da auf diese Weise insbesondere bei der Entscheidung für die Nicht-Existenz einer Abhängigkeit viele Elementkombinationen vorab von der Analyse ausgeschlossen werden können. Diese Erkenntnis bildete die Grundlage für Hypothese 1*, die im Folgenden als Ausgangspunkt für die Entwicklung einer Methode zur effizienten Modellierung von Tracelinks dient (siehe Kapitel 5.4). Darüber hinaus wurde herausgearbeitet, dass in kompositionellen Inklusionshierarchien die Eigenschaften der Kinderelemente aufgrund der bestehenden Kompositions- bzw. Aggregationsbeziehungen auf Ebene des Elternelements abgeschätzt und damit Entscheidungen über Abhängigkeiten getroffen werden können. Bei subsumtiven Inklusionshierarchien kann diese Eigenschafts-Abschätzung hingegen aufgrund der vorliegenden Generalisierungsbeziehungen nicht ohne Kenntnis der Kinderelemente und deren Eigenschaften durchgeführt werden.

Anschließend wurden in Kapitel 5.2.1.2 die bereits in Kapitel 2.2 vorgestellten Artefakte einer Strukturanalyse unterzogen, um herauszuarbeiten mit welcher Art Hierarchie sie üblicherweise strukturiert werden und unter welchen Bedingungen sich die Hypothese 1* anwenden lässt. Eine Übersicht der Artefakte und deren Zuordnung zu den vorgestellten Arten der Hierarchien ist Tabelle 11 zu entnehmen.

	kompositionelle Inklusionshierarchie (Aggregation / Komposition)	subsumtive Inklusionshierarchie (Generalisierung)
Anforderungsartefakte	X	X
Funktionsartefakte	X	
Verhaltensmodelle	X	
Produktstrukturen	X	

Tabelle 11: Zuordnung von verwendeten Artefakten zur Art der Hierarchie

Grundsätzlich zeigte sich bei der Analyse, dass alle betrachteten Artefakte mit Inklusionshierarchien aufgebaut werden und Hypothese 1* damit für die betrachteten Artefakte seine Gültigkeit hat. Bei der Mehrzahl der Artefakte kommen kompositionelle Inklusionshierarchien zum Einsatz, bei denen die übergeordneten Elemente aus ihren Kinderelementen aufgebaut sind. In dieser Hinsicht die einzige Ausnahme stellt das

Anforderungsartefakt dar. Wird dieses mittels Projektion oder Konkretisierung strukturiert, bestehen Generalisierungsbeziehungen zwischen Eltern- und Kinderelementen. Ergebnis ist eine subsumtive Inklusionshierarchie, bei der Entscheidungen über Abhängigkeiten auf abstrakter Ebene nicht ohne die Kenntnis der Kinderelemente getroffen werden können.

Um die theoretisch hergeleitete Hypothese 1* sowie die beschriebenen Eigenschaften der Inklusionshierarchien hinsichtlich der auf abstrakter Ebene für eine Entscheidung zur Verfügung stehenden Informationen im Kontext der Entwicklung eines technischen Systems zu verifizieren, wurde eine Studie durchgeführt, deren Aufbau, Durchführung und Ergebnisse in Kapitel 5.3 beschrieben sind.

5.2.2 GRANULARITÄT IM KONTEXT DER TRACEABILITY

Wie in Kapitel 5.2.1 beschrieben, sind Artefakte im Systems Engineering aus Elementen aufgebaut, welche in beliebig viele hierarchische Ebenen strukturiert werden können. Mit Hilfe dieser Inklusionshierarchien lassen sich die Artefakte sortieren bzw. Detaillierungsgrade von abstrakt bis detailliert abbilden. Elemente können darüber hinaus mit Parametern beschrieben werden, was die detaillierteste Form darstellt. Eine Darstellung des Aufbaus von Artefakten sowie der Bedeutung der Begriffe „Granularität“, „Detaillierungsgrad“ und „hierarchische Ebene“ ist Abbildung 38 zu entnehmen.

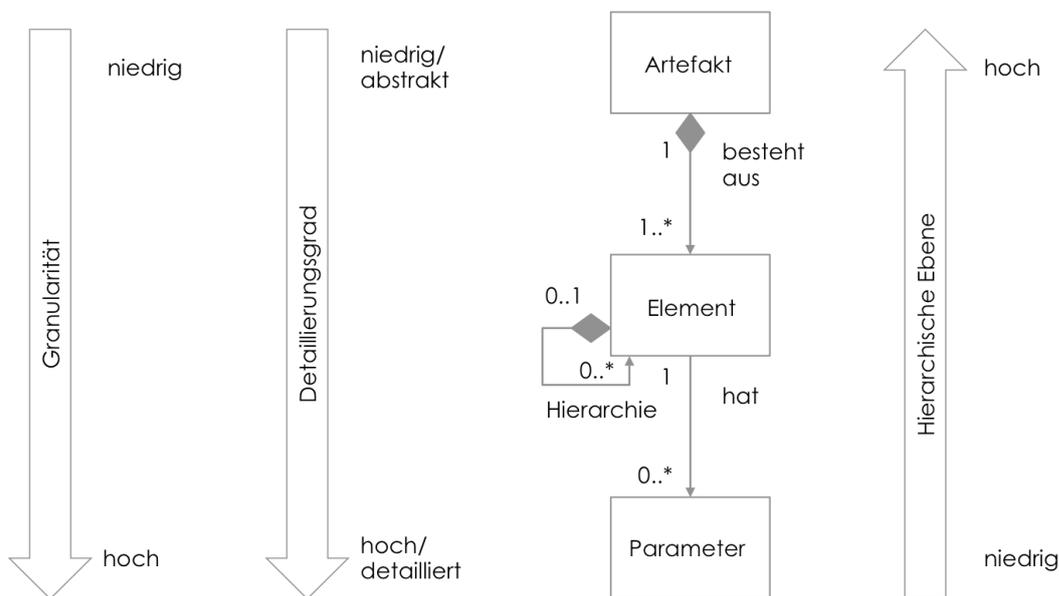


Abbildung 38: Aufbau von Artefakten im Systems Engineering und Einordnung der Begriffe "Granularität", "Detaillierungsgrad" und "hierarchische Ebene"

Im Kontext der durchgängigen Nachverfolgbarkeit stellt die Festlegung, zwischen welchen dieser Detaillierungsstufen bzw. in welcher Granularität Tracelinks modelliert werden, einen wichtigen Faktor zur Beeinflussung des Aufwand/Nutzen-Verhältnisses

dar (siehe auch Kapitel 3.4.4). Insgesamt gibt es fünf relevante Arten der Tracelink-Modellierung (siehe Abbildung 39). Diese werden im Folgenden kurz erläutert und anhand des schematischen Auszugs der beiden Artefakte „Anforderungsspezifikation“ und „Funktionshierarchie“ des mechatronischen Außenspiegels in Abbildung 40 veranschaulicht.

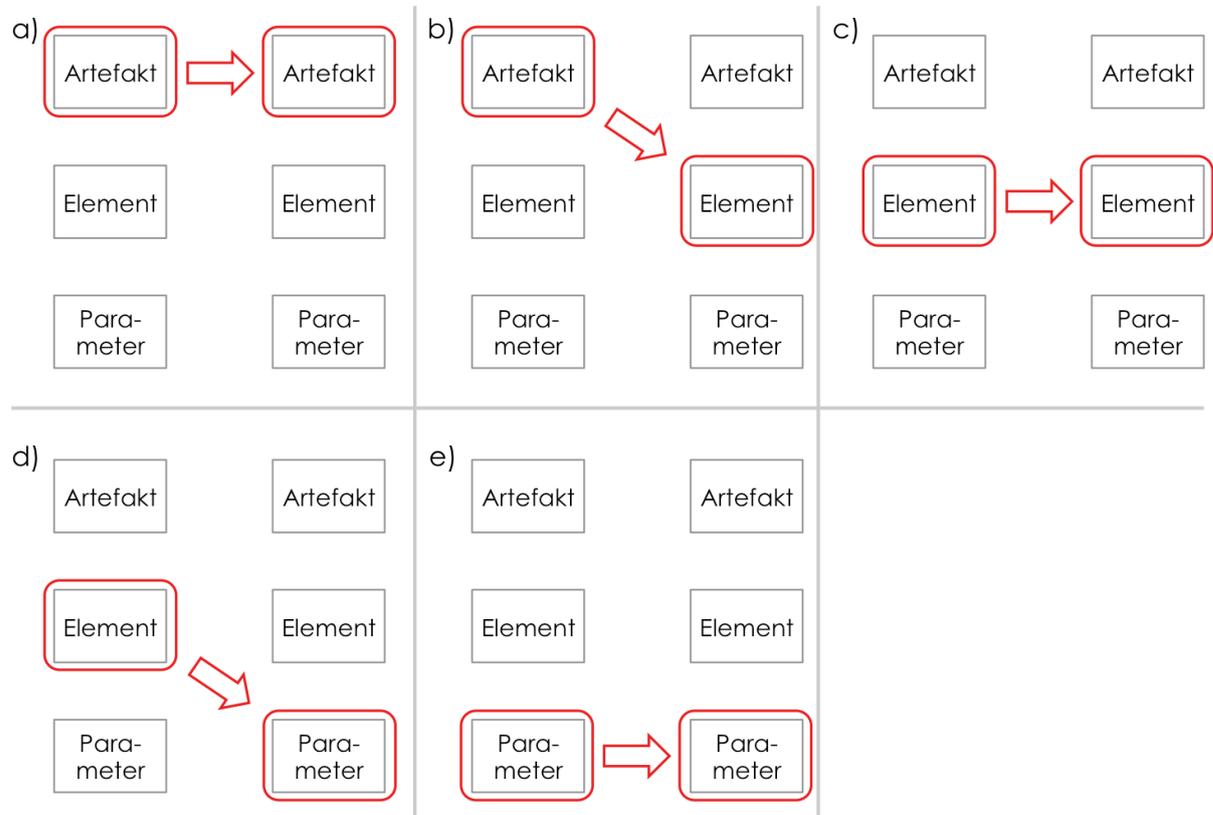


Abbildung 39: Granularitätsebenen bei der Modellierung von Tracelinks

- a) Die Modellierung von Tracelinks auf Artefakt-Ebene (Artefakt \leftrightarrow Artefakt) ist eine sehr abstrakte Form der Traceability (Abbildung 39, a). Sie wird bspw. genutzt, um auszudrücken, dass eine bestimmte Anforderungsspezifikation durch ein Funktionsmodell adressiert wird. Bei der Entwicklung eines Pkws könnte somit beschrieben werden, dass die Anforderungen der Anforderungsspezifikation „Außenspiegel“ durch die Funktionshierarchie „Außenspiegel“ erfüllt werden. Der Nutzen ist damit vergleichsweise gering, der Aufwand zur Modellierung von Tracelinks jedoch ebenfalls. Diese Detaillierungsstufe wird hauptsächlich genutzt, um Traceability-Schemata zu erstellen.
- b) Tracelinks zwischen Artefakten und Elementen (Artefakt \leftrightarrow Element) stellen die nächst detailliertere Stufe der Modellierung dar (Abbildung 39, b). Sie werden bspw. genutzt, um auszudrücken, dass ein Element relevant für die Ausarbeitung eines vollständigen Artefakts ist. Am Beispiel des Außenspiegels würde ein Tracelink zwischen der Anforderung 5.6 und der Funktionshierarchie „Außenspiegel“ bedeuten, dass die geforderte Außenspiegelheizung durch die Funk-

tionshierarchie realisiert wird. Dies kann bspw. sinnvoll sein, wenn mehrere Funktionsartefakte existieren (bspw. eine für den Einklappmechanismus und eine für die Beheizung). Der Aufwand zur Erfassung von Tracelinks hängt dabei stark davon ab, bis zu welcher hierarchischen Element-Ebene Tracelinks modelliert werden sollen.

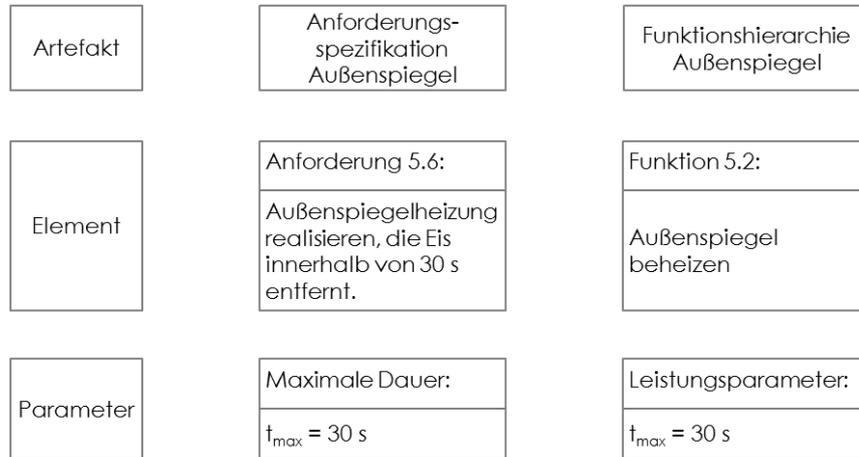


Abbildung 40: Auszug aus dem Anforderungs- und Funktionsartefakt des mechatronischen Außenspiegels

- c) Die Modellierung von Tracelinks zwischen Elementen unterschiedlicher Artefakte (Element \leftrightarrow Element) ist die häufigste Stufe der Modellierung (Abbildung 39, c). Auf dieser Stufe lässt sich bspw. ausdrücken, welche Anforderungen durch welche Funktionen oder welche Bauteile erfüllt werden müssen. Im Beispiel sagt ein Tracelink auf Elementebene aus, dass die Funktion „Außenspiegel beheizen“ zur Realisierung der Anforderung 5.6 beiträgt. Wie bei (Artefakt \leftrightarrow Element), hängt der Aufwand stark von der hierarchischen Struktur der Artefakte ab. Ist diese sehr tief und werden die Tracelinks bis zur untersten hierarchischen Ebene modelliert, ist die Anzahl der betrachteten Elemente und damit der Aufwand sehr groß.

- d) In der Stufe (Element «» Parameter) werden Elemente direkt mit bestimmten Parametern anderer Artefakte durch Tracelinks verknüpft (Abbildung 39, d). Dadurch können bspw. textuell beschriebene Anforderungen direkt auf Parameter in Funktions- oder Geometrie-Modellen bezogen werden und damit, im Fall von Änderungen, eine präzisere Identifikation betroffener Parameter durchgeführt werden. Die Anforderung 5.6, in der die maximale Zeit, in der Eis vom Außenspiegel entfernt werden soll, im Fließtext beschrieben ist, würde somit mit dem Leistungsparameter der Funktion 5.2 verknüpft. Insbesondere, wenn die Funktion über mehrere Parameter verfügt, ermöglicht diese Art der Modellierung die Identifikation spezifischer Parameter, bspw. bei Änderungen der textuellen Beschreibung der Anforderung. Der notwendige Aufwand ist höher als bei (Element «» Element), da die feingranularen Parameter in der Analyse berücksichtigt werden müssen.
- e) Die höchste Granularitätsstufe der Traceability wird erreicht, wenn Links zwischen einzelnen Parametern (Parameter «» Parameter) der Elemente modelliert werden (Abbildung 39, e). Es würde also der explizit modellierte Parameter der Anforderung $t_{\max} = 30 \text{ s}$ direkt mit dem Leistungsparameter der Funktion verknüpft werden. In Abhängigkeit der Realisierung im jeweiligen Software Tool lässt sich so bspw. der Parameter der Funktion, der während der Entwicklung der Funktionshierarchie ggf. variiert wird, mit dem der Anforderung abgleichen. Abweichungen lassen sich so direkt und ohne manuelle Überprüfung feststellen und automatisiert melden. Alternativ ist es möglich, dass nicht nur überwacht, sondern direkt geändert wird. Eine Änderung der Vorgabe in der Anforderung hätte somit direkt eine Änderung des Leistungsparameters der Funktion zur Folge. Nachteil dieser detailliertesten Form der Traceability-Modellierung ist der Aufwand, der für die Erfassung der Tracelinks notwendig ist. Letztlich muss jeder Parameter, der überwacht werden soll, per Hand verknüpft werden.

Anzumerken ist, dass eine geringe Granularität keine fehlenden Tracelinks zur Folge hat, da diese von den höheren Ebenen der Hierarchie auf die unteren vererbt werden [Sutinen et al. 2000, S. 12]. Das hat zur Folge, dass zahlreiche Elemente und Parameter einen Tracelink erben, die eigentlich keine Abhängigkeit aufweisen [Egyed et al. 2009, S. 248]. In Abbildung 41 ist dies beispielhaft dargestellt.

Besteht ein Tracelink von einem Artefakt A zu Artefakt B, so erben alle Kinder-elemente B1 und B2 ebenfalls einen Tracelink (Abbildung 41, a). Dadurch nimmt die Präzision, mit der betroffene Elemente und Parameter bei einer Auswirkungsanalyse identifiziert werden können, deutlich ab [Bianchi et al. 2000, S. 156; Egyed et al. 2009, S. 247]. Bei einer Auswirkungsanalyse müssten somit bei einer Änderung von A sowohl das abhängige Element B1 als auch das unabhängige Element B2 auf Auswirkungen

untersucht werden. Die Wahl einer höheren Granularität hätte indes zur Folge, dass ausgehend von Artefakt A nur das tatsächlich abhängige Element B1 mit einem Tracelink verknüpft wird (Abbildung 41, b). Bei Durchführung derselben Auswirkungsanalyse müsste somit nur B1 auf Auswirkungen untersucht werden.

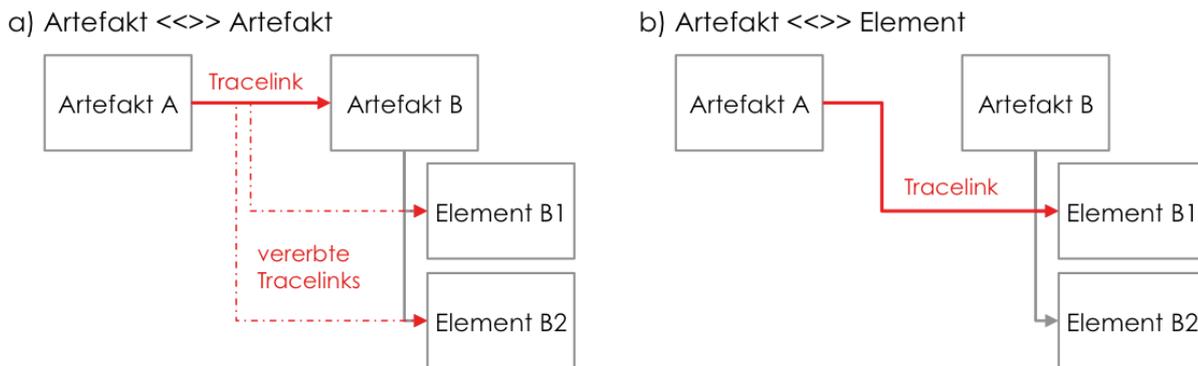


Abbildung 41: Folgen der Reduktion der Granularität auf die Auswertung von Tracelinks: a) durch eine niedrige Granularität werden viele Tracelinks zwischen Elementen vererbt, bei denen keine Abhängigkeit besteht. b) durch die Wahl einer höheren Granularität werden nur tatsächlich Abhängige Elemente verknüpft.

Eine allgemeingültige Aussage, welche Granularität der Tracelinks die „angemessene“ ist, kann nicht getroffen werden [Mäder et al. 2009a, S. 5], da diese von unterschiedlichen Aspekten abhängig ist. Unternehmensspezifische Vorgehensweisen und Methoden haben bspw. einen großen Einfluss auf die Strukturierung der Artefakte und damit auf den notwendigen Aufwand zur Erfassung der Tracelinks. Ähnlich verhält es sich mit individuellen Vorgehensweisen der Ingenieure, die beim Modellieren der Artefakte unterschiedliche Strukturierungsansätze wählen. Darüber hinaus werden während der Produktentwicklung in Abhängigkeit der Branche unterschiedliche Artefakte erstellt bzw. vom Gesetzgeber gefordert und diese wiederum sehr unterschiedlich aufgebaut. Den größten Einfluss auf die zu wählende Granularität eines Traceability-Ansatzes hat jedoch die geplante Verwendung der Tracelinks, welche sich zum Zeitpunkt der Erfassung auf mittel- und langfristige Sicht nur schwer voraussehen lässt [Bianchi et al. 2000, S. 156; Egyed et al. 2009, S. 257; Mäder et al. 2009a, S. 4].

Deshalb ist es sinnvoll, falls der Verwendungszweck nicht bekannt ist, die Modellierung der Tracelinks im Rahmen der zur Verfügung stehenden Ressourcen zunächst auf den hohen hierarchischen Element-Ebenen durchzuführen und insbesondere deren Korrektheit und Vollständigkeit zu berücksichtigen [Egyed et al. 2009, S. 242]. Dabei sollte beachtet werden, dass alle involvierten Artefakte auf einem ähnlichen Granularitätsniveau betrachtet werden. Eine Missachtung dieser Faustregel führt laut Mäder et al. zu einer großen Anzahl von Tracelinks, die nicht mehr Aussagekraft bieten, als das Verknüpfen auf einer hohen hierarchischen Ebene [Mäder et al. 2009a, S. 4]. Anschließend können beispielsweise Methoden wie das „Value-based Trace

Enhancement“ von Egyed et al. [Egyed et al. 2009] o. ä. [Ramesh und Edwards 1993, S. 257; Bianchi et al. 2000, S. 157; Ramesh und Jarke 2001, S. 19] gezielt angewendet werden, um die Traceability ziel- und bedarfsgerecht zu verfeinern.

Hinsichtlich der Methoden zur effizienten Modellierung von Tracelinks gilt es somit festzuhalten, dass eine flexible Unterstützung der unterschiedlichen Granularitätsebenen gewährleistet werden sollte. Insbesondere bei der Modellierung auf der (Element «» Element)-Ebene, die in Abhängigkeit der Strukturierung der Artefakte sehr umfangreich sein kann, sollte die Modellierungstiefe frei wählbar sein.

5.2.3 ERMITTLUNG DES AUFWANDS BEI DER MANUELLEN ERFASSUNG VON TRACELINKS

Wie in Kapitel 3.4.1 beschrieben, kann bei der Erfassung von Tracelinks zwischen einer Online- und einer Offline-Erfassung unterschieden werden. Während erstere parallel zur Modellierung der Artefakte durch den Anwender durchgeführt wird, findet letztere im Anschluss an die Modellierung der Artefakte statt. Da es sich bei der im Kapitel 5.3 vorgestellten Methode EcoTracing um eine Offline-Methode handelt, wird diese Vorgehensweise hier als Grundlage für die Ermittlung des Aufwandes verwendet.

Bei der Offline-Erfassung definiert sich der Aufwand für die Erfassung vorwiegend über die Anzahl der Entscheidungen (ob eine Abhängigkeit zwischen zwei Elementen besteht oder nicht), die bei der Analyse getroffen werden müssen, sowie die benötigte Zeit. Vereinfachend wird dabei angenommen, dass die Erfassung nach dem u. a. von Maurer, Lindemann et al. und Biedermann et al. beschriebenen Schema in Form von Workshops, bei denen alle Elementkombinationen überprüft werden, durchgeführt wird (siehe Abbildung 42) [Maurer 2007, S. 98; Lindemann et al. 2009, S. 79; Biedermann et al. 2010, S. 309].

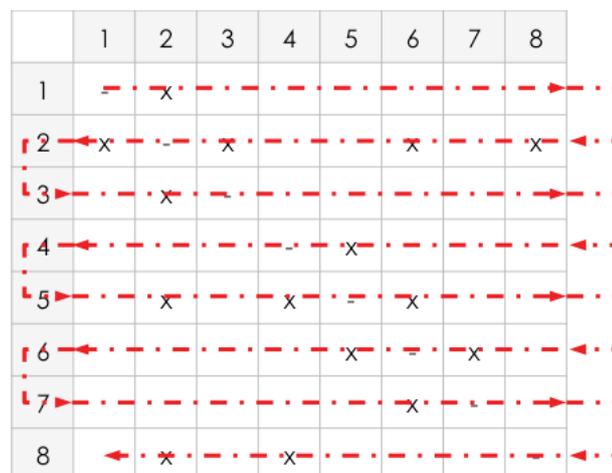


Abbildung 42: Vorgehensweise zur Erfassung von Abhängigkeiten am Beispiel einer DSM (in Anlehnung an [Maurer 2007, S. 98])

Diese strukturierte Vorgehensweise wird von den Autoren empfohlen, da so keine Elementkombinationen übergangen werden. Neben diesen Quellen aus dem akademischen Bereich wird diese Vorgehensweise auch im industriellen Kontext angewendet, wie im Rahmen einer Studie zu den Herausforderungen moderner Produktentstehungsprozesse durch das Fraunhofer IPK festgestellt wurde [Stark 2011].

Im Folgenden wird die Berechnung anhand eines abstrakten Beispiels durchgeführt (Abbildung 43). Dabei werden die Artefakte A und B betrachtet, die jeweils aus m bzw. n Elementen hierarchisch aufgebaut sind.

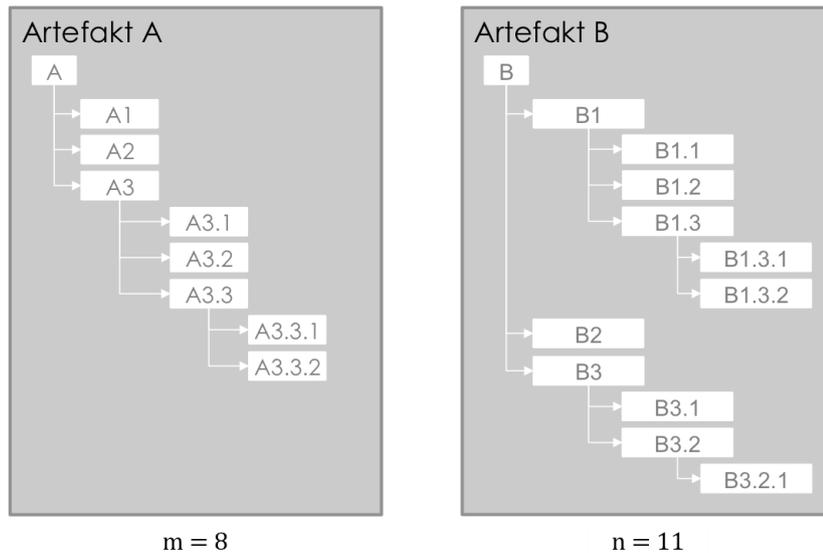


Abbildung 43: Abstraktes Beispiel zur Ermittlung des Aufwands

Für dieses Beispiel ergibt sich die Anzahl notwendiger Entscheidungen E_M für die Artefakte mit m bzw. n Elementen, nach der oben beschriebenen Vorgehensweise, somit zu:

$$E_M = m * n . \quad (1)$$

Der notwendige zeitliche Aufwand T_M kann damit, bei einer gemittelten Bearbeitungs- bzw. Entscheidungszeit von t_\emptyset , wie folgt berechnet werden:

$$T_M = m * n * t_\emptyset . \quad (2)$$

Werden nicht nur zwei sondern x Artefakte betrachtet und miteinander verknüpft, ergibt sich die Anzahl notwendiger Artefaktkombinationen o nach folgender Formel:

$$o = \sum_{i=1}^{x-1} i \quad (3)$$

Somit lassen sich die Anzahl notwendiger Entscheidungen und der zeitliche Aufwand folgendermaßen berechnen:

$$E_M = \sum_{i=1}^o m_i * n_i \quad (4)$$

bzw.

$$T_M = \sum_{i=1}^o (m_i * n_i) * t_{\emptyset} . \quad (5)$$

Für das Beispiel in Abbildung 43 ergeben sich damit, unter Berücksichtigung der Element-Anzahlen $m = 8$ und $n = 11$ und einer geschätzten mittleren Bearbeitungszeit von $t_{\emptyset} = 2$ s, folgende Werte:

$$E_M = m * n = 8 * 11 = 88 \quad (6)$$

bzw.

$$T_M = m * n * t_{\emptyset} = 8 * 11 * 2s = 176 s . \quad (7)$$

5.2.4 ZUSAMMENFASSUNG UND SCHLUSSFOLGERUNGEN

In Kapitel 5.2.1 wurde zunächst eine Betrachtung hierarchischer Artefakte durchgeführt. Bei dieser Analyse wurde herausgearbeitet, dass es aufgrund der Transitivität hierarchischer Relationen möglich ist, Entscheidungen, die über Abhängigkeiten von Elternelementen getroffen werden, auf deren Kinderelemente zu übertragen. Dieses Erkenntnis bildete die Grundlage für Hypothese 1*, die im Folgenden als Ausgangspunkt für die Entwicklung einer Methode zur effizienten Modellierung von Tracelinks dient (siehe Kapitel 5.4). Da die Analyse zusätzlich ergab, dass die Art der Hierarchie Auswirkungen auf die Erfassung von Tracelinks hat, wurden anschließend die bereits in Kapitel 2.2 vorgestellten Artefakte wieder aufgenommen und einer Strukturanalyse unterzogen. Dabei wurde analysiert mit welcher Art Hierarchie sie üblicherweise strukturiert werden und unter welchen Voraussetzungen die Abhängigkeitsanalyse durchgeführt werden kann. Um diese theoretischen Ausführungen zusätzlich im Kontext der Entwicklung eines technischen Systems zu verifizieren wurde eine Studie durchgeführt, die im Folgenden Kapitel 5.3 beschrieben ist.

Darüber hinaus wurde in Kapitel 5.2.2 die Granularität im Kontext der durchgängigen Nachverfolgbarkeit betrachtet. Hintergrund dieser Betrachtung war, dass der Aufwand bei der Erfassung von Tracelinks maßgeblich von der Granularität der modellierten Tracelinks abhängt. Diese hängt wiederum u. a. von dem Verwendungszweck der Tracelinks ab, der in den frühen Phasen der Systementwicklung meist noch nicht final definiert ist. Eine Methode zur effizienten Modellierung von Tracelinks sollte somit die Wahl der Granularität dem Nutzer überlassen, um zunächst eine niedrige Granularitätsstufe und zu einem späteren Zeitpunkt eine bedarfsgerechte Verfeinerung in eine höhere Granularitätsstufe zu ermöglichen:

Anforderung 1: Die Granularität der Tracelinks muss während der Erfassung frei wählbar sein.

5.3 EVALUATION DER HYPOTHESE 1* IM KONTEXT DER SYSTEMENTWICKLUNG

In Kapitel 5.2.1 wurde durch die Analyse der Eigenschaften unterschiedlicher Arten von Hierarchien sowie deren Zuordnung zu den Artefakten der Systementwicklung die Hypothese 1* abgeleitet, die für die später vorgestellte Methode EcoTracing essentiell ist. Um die Anwendbarkeit der Hypothese im Kontext der Systementwicklung abzusichern, wurde eine Studie mit 17 Probanden durchgeführt und die zugehörige Nullhypothese überprüft. Sie lautet:

Weist ein Element in B eine Abhängigkeit zu einem Element in A auf, so darf keines seiner Elternelemente eine Abhängigkeit zu dem Element in A bzw. (soweit vorhanden) seinen Elternelementen aufweisen.

Der Aufbau und die Durchführung der Studie werden im folgenden Kapitel 5.3.1 vorgestellt. Die Beschreibung der Auswertung und Diskussion der Ergebnisse sind im Kapitel 5.3.2 zu finden.

5.3.1 AUFBAU UND DURCHFÜHRUNG DER STUDIE

Um die theoretisch hergeleitete Hypothese empirisch im Kontext der Systementwicklung überprüfen zu können und damit die Nullhypothese zu falsifizieren, ist es notwendig, anhand von Beispielartefakten Abhängigkeitsuntersuchungen getrennt voneinander auf einer niedrigen und einer hohen hierarchischen Ebene durchzuführen. Anschließend wird analysiert, ob sich Widersprüche zwischen den Tracelinks dieser Ebenen ergeben. Ein Widerspruch (der die Nullhypothese stützt) liegt vor, wenn auf niedriger hierarchischer Ebene ein Tracelink zwischen zwei Elementen modelliert wurde, deren Elternelemente aber keinen Tracelink zueinander aufweisen.

Der Aufbau der Studie sah vor, dass Probanden anhand mehrerer Beispiele manuelle Abhängigkeitsuntersuchungen durchführen. Die gewählten Beispielartefakte waren Funktionshierarchien und Produktstrukturen (strukturiert als kompositionelle Inklusionshierarchien), die über eine Gesamtfunktions- bzw. Baugruppenebene und eine Teilfunktions- bzw. Bauteilebene verfügten (siehe Anhang A). Diese Abstraktionsebenen wurden getrennt voneinander bearbeitet und zudem die hierarchischen Zusammenhänge zwischen den Ebenen (also welche Teilfunktionen zu welcher Gesamtfunktion und welche Bauteile zu welcher Baugruppe gehören) bei der Bearbeitung der Aufgabenstellung nicht dargestellt, um eine unabhängige Bewertung der Elementekombinationen zu erreichen.

Bei der Auswertung wurde nach Abschluss der Modellierung überprüft, ob die jeweiligen Elternelemente derjenigen Elemente, die über einen Tracelink verknüpft wurden, ebenfalls über einen modellierten Tracelink verfügen (siehe Abbildung 44). Sollte dies nicht der Fall sein, wurde der Widerspruch, der die Nullhypothese stützt, erfasst und gesondert mit dem Probanden diskutiert, um die Gründe für die Entstehung des Widerspruchs zu erörtern.

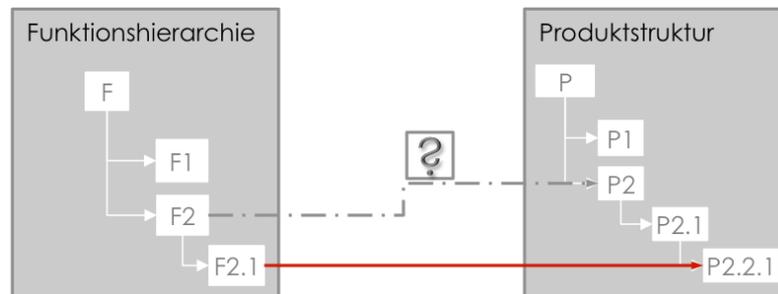


Abbildung 44: Bei der Auswertung wird überprüft, ob auf höchster hierarchischer Ebene ein Tracelink vorhanden ist (dargestellt durch die graue gestrichelte Linie), wenn deren Kilderelemente einen Tracelink aufweisen (dargestellt durch die rote durchgängige Linie). Ist dem nicht so, wird ein Widerspruch registriert.

Zum Zweck der Erfassung von Entscheidungen bzgl. der Existenz von Abhängigkeiten zwischen den Elementen der Artefakte wurden Excel-Matrizen verwendet (siehe Abbildung 45). Die Visualisierung der Systeme und Baugruppen bzw. Bauteile²⁸ erfolgte dabei in den Zeilen links der Matrix. Die Funktionen wurden, entgegen der üblichen Darstellung in Matrizen, bei der die Beschriftungen der Spalten um 45° oder 90° gedreht werden, ebenfalls waagrecht angeordnet, um eine gute Lesbarkeit zu gewährleisten. Um trotzdem eine eindeutige Zuordnung der Funktionen zu der gemeinsamen Schnittzelle mit dem Produktstruktur-Element zu erreichen, wurden beide Elemente bei Auswahl der Zelle gelb hinterlegt (siehe Abbildung 45).

Die Navigation zwischen den Zellen der Matrix und damit zwischen den Elementen der Artefakte erfolgte mit den Pfeiltasten: „Pfeil nach rechts“ → und „Pfeil nach links“ ←. Bei Erreichen des rechten Rands der Matrix sprang die ausgewählte Zelle automatisch, durch Betätigen der Taste „Pfeil nach rechts“, in die erste Zelle der darauffolgenden Zeile. Die Probanden wurden angewiesen nur diese Tasten und nicht die Maus zu nutzen, sodass (abgesehen von nachträglichen Korrekturen) überwiegend von der gleichen Bearbeitungsreihenfolge der unterschiedlichen Probanden ausgegangen werden kann.

²⁸ In Abhängigkeit, ob die Analyse auf hoher oder niedriger hierarchischer Ebene der Artefakte durchgeführt wird, werden links der Matrix entweder die Systeme und Baugruppen oder die Bauteile dargestellt.

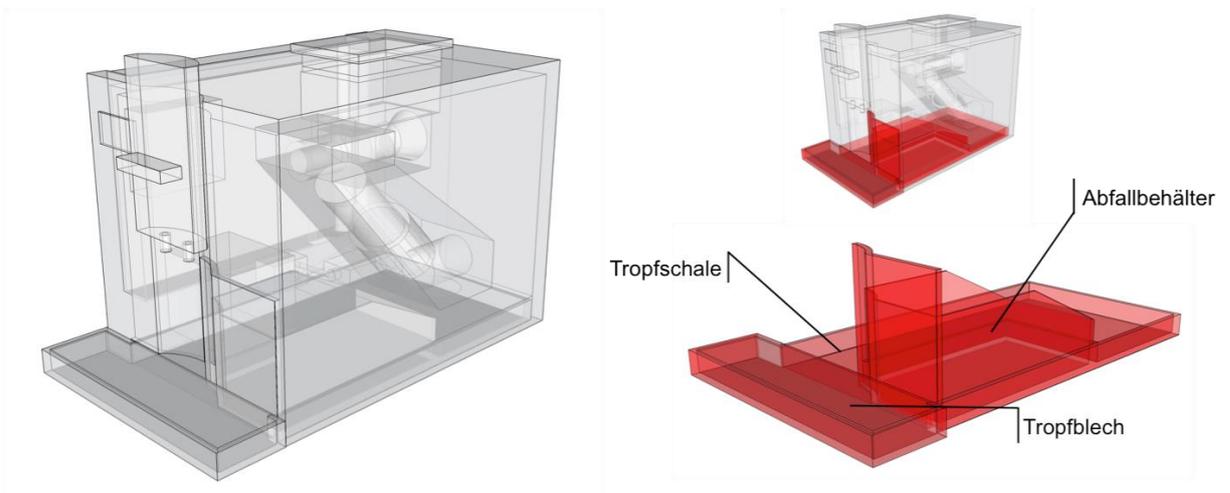


Abbildung 46: Schematische Darstellung des Kaffeevollautomaten (links: Übersicht, rechts: Darstellung der Baugruppe Auffangeinheit mit beschrifteten Einzelteilen)

Die Funktionsstruktur enthielt sieben Funktionen auf höchster hierarchischer Ebene, die in insgesamt 17 Teilfunktionen zerlegt wurden. Eine Übersicht aller Bauteile inkl. visueller Darstellung sowie der vollständigen Funktionshierarchie ist Anhang A zu entnehmen.

Im Fall des Fahrrad-Beispiels beinhaltete die Produktstruktur acht Baugruppen und Systeme auf höchster hierarchischer Ebene, die aus 26 Bauteilen aufgebaut waren. Das Fahrrad als Übersichtsabbildung sowie das Antriebssystem sind in Abbildung 47 schematisch dargestellt.

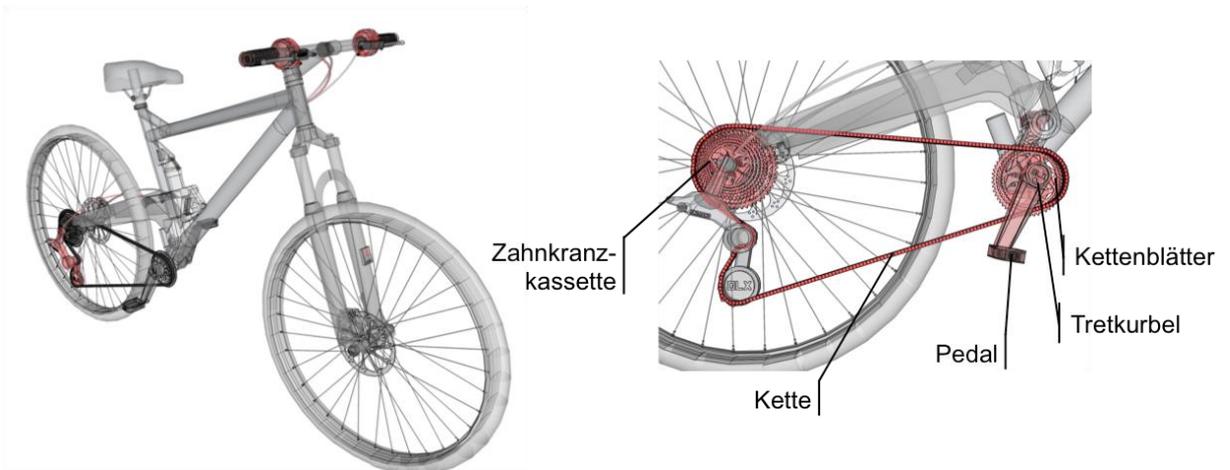


Abbildung 47: Schematische Darstellung des Fahrrads (links: Übersicht, rechts: Darstellung der Antriebssystems mit beschrifteten Einzelteilen)

Die zugehörige Funktionsstruktur beinhaltete vier Funktionen auf höchster hierarchischer Ebene und 11 Funktionen auf niedrigster hierarchischer Ebene. Eine Übersicht

aller Bauteile inkl. visueller Darstellung sowie der vollständigen Funktionshierarchie ist in Anhang A zu finden.

Bei der Bearbeitung beider Beispiele wurden die beiden Granularitätsebenen (Baugruppen «» Gesamtfunktionen und Bauteile «» Teilfunktionen) nacheinander durch die Probanden analysiert. Dabei wurden alle Elementekombinationen durch Bearbeitung aller Zellen der Matrix überprüft und die Entscheidungen für Existenz bzw. Nicht-Existenz einer Abhängigkeit durch eine „1“ bzw. „0“ in der jeweiligen Zelle dokumentiert. Um bei der späteren Auswertung eventuelle Lerneffekte ausschließen zu können, wurde sowohl die Bearbeitungsreihenfolge der Beispiele als auch die der beiden Granularitätsebenen über die Probanden alterniert.

Im Anschluss an die vollständige Bearbeitung der Beispiele wurden die Matrizen hinsichtlich der vorkommenden Widersprüche zwischen den Abstraktionsebenen analysiert und diese mit den Probanden diskutiert. Ziel dieser Diskussion war es, den Grund für die Entscheidungen, einen Tracelink auf niedriger hierarchischer Ebene, aber keinen zwischen deren Elternelementen auf hoher hierarchischer Ebene zu modellieren, zu verstehen. Dabei war insbesondere die Kategorisierung der Widersprüche von Interesse, um eine Einordnung hinsichtlich der Hypothese zu ermöglichen.

Die Eigenschaften der Studie sind noch einmal zusammenfassend in Tabelle 12 dargestellt.

Aufgabenstellung	Identifikation von Abhängigkeiten und manuelle Modellierung von Tracelinks auf zwei Granularitätsebenen für drei Beispiele. Es handelt sich dabei um eine realistische Aufgabenstellung, die an industrielle Aufgabenstellungen angelehnt ist, sich jedoch in Ausprägung der Artefakte hinsichtlich Umfang und Komplexität unterscheidet
Analyseeinheit	Getroffene Entscheidungen auf beiden Granularitätsebenen
Analyse-methode	Überprüfung, ob für alle Elemente der niedrigsten hierarchischen Ebene der Produktstruktur, die einen Tracelink zu einer Teilfunktion der niedrigsten hierarchischen Ebene der Funktionshierarchie haben, ebenfalls Tracelinks zwischen deren Elternelementen der höchsten hierarchischen Ebene vorhanden sind.
Methode zur Datenerfassung	Direkte Erfassung der Entscheidungen der Probanden in den Matrizen; Interview der Probanden, um die Gründe für Entscheidungen zu erfassen
Anzahl der Teilnehmer	17
Charakterisierung der Teilnehmer	Studentische Hilfskräfte und wissenschaftliche Mitarbeiter aus dem Bereich Virtuelle Produktentstehung des Fraunhofer IPK und dem Fachgebiet Industrielle Informationstechnik der Technischen Universität Berlin; Grundsätzliche Kenntnisse über Artefakte, deren Entwicklung und Abhängigkeiten zwischen diesen vorhanden

Untersuchungs- objekt	<p>Zwei Artefakte eines Hauses (Einführungsbeispiel):</p> <ul style="list-style-type: none"> - Funktionshierarchie mit 10 Funktionen auf zwei Hierarchieebenen - Produktstruktur mit 12 Baugruppen und Bauteilen auf zwei Hierarchieebenen <p>Zwei Artefakte eines Kaffeevollautomaten:</p> <ul style="list-style-type: none"> - Funktionshierarchie mit 21 Funktionen auf zwei Hierarchieebenen - Produktstruktur mit 51 Baugruppen und Bauteilen auf zwei Hierarchieebenen <p>Zwei Artefakte eines Fahrrads</p> <ul style="list-style-type: none"> - Funktionshierarchie mit 15 Funktionen auf zwei Hierarchieebenen - Produktstruktur mit 40 Baugruppen und Bauteilen auf zwei Hierarchieebenen
--------------------------	--

Tabelle 12: Eigenschaften der Studie zur Überprüfung der Hypothese 1*

5.3.2 AUSWERTUNG UND DISKUSSION DER ERGEBNISSE

Die Ergebnisse der einzelnen Probanden sind für die drei Beispiele Haus, Kaffeevollautomat und Fahrrad in Tabelle 13 zusammengefasst. Für das Einführungsbeispiel des Hauses bewegt sich die Anzahl an Widersprüchen zwischen 0 und 2, was bezogen auf die Gesamtzahl möglicher Widersprüche Werten von 0 % bis 10 % entspricht. Beim Kaffeevollautomaten liegt die Anzahl der Widersprüche zwischen 2 und 10 bzw. 4 % und 18 %. Beim Fahrrad ist die Anzahl der Widersprüche mit 0 bis 7 geringer als beim Kaffeevollautomaten. Prozentual gesehen liegt das Fahrrad-Beispiel zwischen 0 % und 22%, wobei der Maximalwert damit sogar höher ist, als beim Kaffeevollautomat-Beispiel.

Bei der Betrachtung der Mittelwerte der Widersprüche fällt auf, dass das Haus- und das Fahrrad-Beispiel mit 4,7 % und 5 % in etwa auf einem Niveau liegen, während der Mittelwert des Kaffeevollautomaten mit 9 % fast doppelt so hoch ist. Damit wird die Nullhypothese im Haus-Beispiel in 95,3 %, im Fahrrad-Beispiel in 95 % und im Kaffeevollautomaten-Beispiel in 91 % der Fälle falsifiziert. In der überwiegenden Anzahl der Fälle weisen die Elternelemente somit einen Tracelink auf, wenn deren Kinderelemente ebenfalls einen aufweisen. Dabei war ein Wert von 100 % aufgrund der nicht eindeutigen Strukturierung der Beispielartefakte und des subjektiv unterschiedlichen Produktverständnisses der Probanden nicht zu erwarten.

Allerdings zeigt sich bei näherer Betrachtung, dass eine Korrelation zwischen dem Mittelwert der Selbsteinschätzung²⁹ bzgl. des Produktverständnisses der Probanden

²⁹ Die Selbsteinschätzung ist ein Wert zwischen 0 (geringe Kenntnis) und 10 (umfassende Kenntnis), mit dem sich die Probanden selbst hinsichtlich ihrer Kenntnis der jeweiligen Beispiele bewertet haben.

und dem Mittelwert der genormten Anzahl an Widersprüchen³⁰ besteht, die mit einem Korrelationskoeffizienten von -0,98 bestimmt werden kann. Das heißt, es besteht ein Zusammenhang zwischen der Kenntnis der Beispiele und der Anzahl an Widersprüchen, die sich bei der Abhängigkeitsanalyse durch die Bewertung der Probanden ergeben: je besser das Beispielprodukt bekannt ist, desto weniger Widersprüche ergeben sich zwischen den Bearbeitungsebenen. In Abbildung 48 sind die Werte der Selbsteinschätzung sowie die Kehrwerte der genormten Anzahl an Widersprüchen³¹ über den drei Beispielprodukten dargestellt.

	Haus		Kaffeefullautomat		Fahrrad	
	Wider- sprüche (Anzahl)	Wider- sprüche (Prozent)	Wider- sprüche (Anzahl)	Wider- sprüche (Prozent)	Wider- sprüche (Anzahl)	Wider- sprüche (Prozent)
Proband 1	1	5,0 %	6	11,0 %	4	13,0 %
Proband 2	1	5,0 %	8	14,0 %	2	6,0 %
Proband 3	0	0,0 %	4	7,0 %	2	6,0 %
Proband 4	0	0,0 %	4	7,0 %	0	0,0 %
Proband 5	2	10,0 %	10	18,0 %	5	16,0 %
Proband 6	2	10,0 %	3	5,0 %	0	0,0 %
Proband 7	1	5,0 %	9	16,0 %	0	0,0 %
Proband 8	1	5,0 %	4	7,0 %	0	0,0 %
Proband 9	1	5,0 %	3	5,0 %	1	3,0 %
Proband 10	0	0,0 %	2	4,0 %	0	0,0 %
Proband 11	2	10,0 %	6	11,0 %	0	0,0 %
Proband 12	1	5,0 %	2	4,0 %	0	0,0 %
Proband 13	1	5,0 %	4	7,0 %	4	13,0 %
Proband 14	1	5,0 %	5	9,0 %	0	0,0 %
Proband 15	1	5,0 %	6	11,0 %	0	0,0 %
Proband 16	1	5,0 %	7	12,0 %	2	6,0 %
Proband 17	0	0,0 %	6	11,0 %	7	22,0 %
Mittelwert	0,94	4,7 %	5,24	9,0 %	1,59	5,0 %
Summe	16		89		27	

Tabelle 13: Ergebnis der Studie zur Überprüfung der Hypothese 1* (Erläuterung, was als Widerspruch gezählt wird, in Abbildung 44)

Diese Erkenntnis wird auch durch die Nachbesprechungen, in denen die Widersprüche diskutiert wurden, bestätigt. Die Widersprüche ergaben sich in der Mehrzahl der Fälle dadurch, dass den Probanden die Baugruppen mit ihren Bauteilen oder die Funktionsstrukturen nicht ausreichend bekannt waren. Im Folgenden wird daher für

³⁰ Dabei wird die Anzahl an Widersprüchen auf die maximal mögliche Anzahl an Widersprüchen bezogen, um vergleichbare Werte zu erlangen.

³¹ Da es sich um eine negative Korrelation handelt wurde für die Darstellung der Kehrwert gewählt, um für eine eingängigere Darstellung einen parallelen Verlauf der Werte zu erreichen.

die beiden Beispielprodukte Fahrrad und Kaffeefullautomat näher auf einige häufig auftretende Widersprüche eingegangen und deren Entstehung diskutiert.

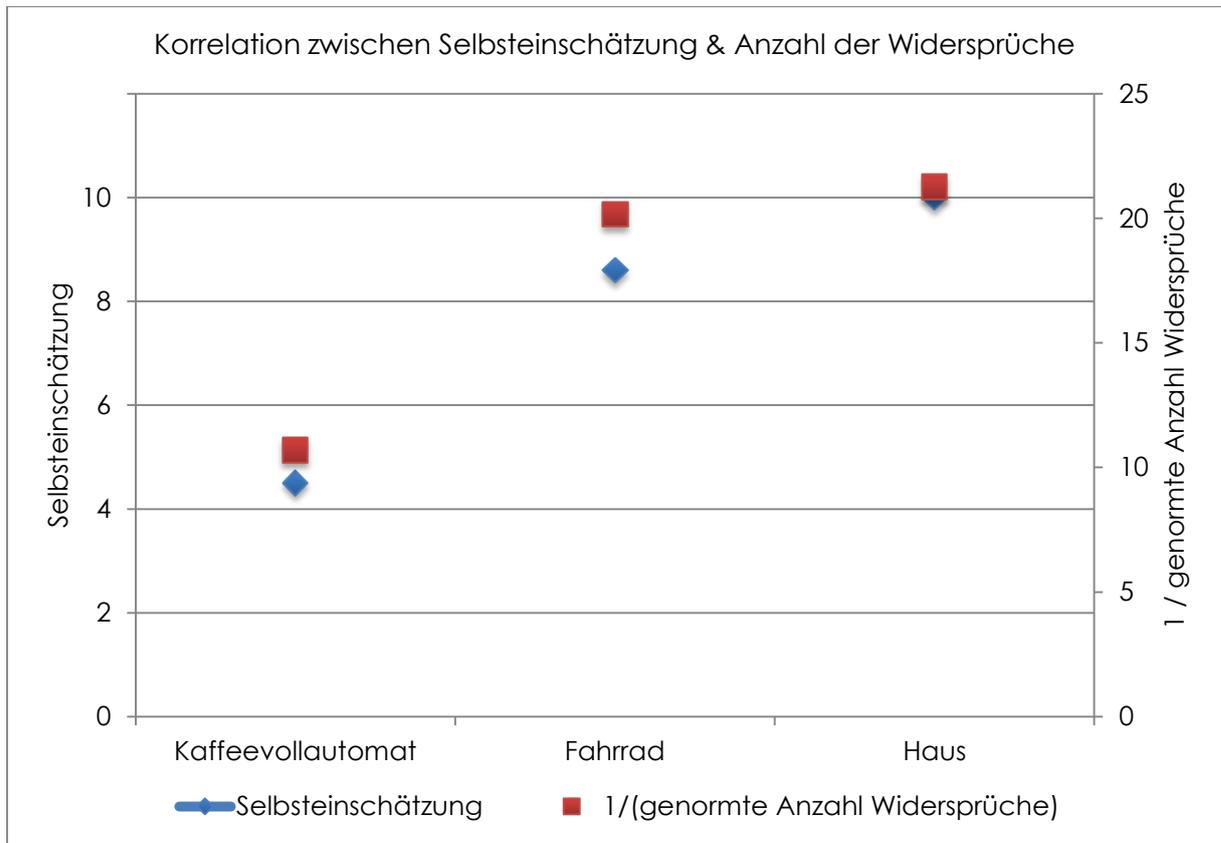


Abbildung 48: Korrelation zwischen der Selbsteinschätzung und der Anzahl der Widersprüche

In Tabelle 14 sind die kumulierten Widersprüche, die bei der Analyse des Fahrrad-Beispiels aufgetreten sind, in Bezug zur Elementekombination, bei der sie aufgetreten sind, dargestellt. Dabei fallen insbesondere die vier Kombinationen „Antriebseinheit / Variable Übersetzung realisieren“, „Vorderrad / Bremsen realisieren“, „Vorderrad / Lenken ermöglichen“ sowie „Hinterrad / Lenken ermöglichen“ mit jeweils drei bis vier Nennungen auf. Der Widerspruch, der bei der ersten Kombination aufgetreten ist, konnte in den Nachbesprechungen bei allen Fällen auf die unerwartete Strukturierung der Baugruppen zurückgeführt werden, die durch den Autor dieser Arbeit vorgenommen wurde. Dabei wurde die Zahnkranzkassette, die sowohl eine Teilfunktion bei der Realisierung des Antriebs als auch bei der variablen Übersetzung übernimmt, der Baugruppe Antriebseinheit zugeordnet. Bei der Analyse auf Baugruppenebene war diese Zuordnung den Probanden nicht bewusst, weshalb die Antriebseinheit nicht mit der Funktion „Variable Übersetzung realisieren“ in Verbindung gebracht wurde. Hätten die Probanden das Beispiel selber entwickelt und somit selber über die Strukturierung des Produkts entschieden, oder wären die Bestandteile der Baugruppe einsehbar gewesen, wäre dieser Widerspruch laut Aussage der Probanden nicht aufgetreten.

					Antrieb realisieren
					Bremsen realisieren
					Lenken ermöglichen
					Variable Übersetzung realisieren
Rahmen	1	2	2	2	
Lenkung		1		1	
Antriebseinheit				4	
Schaltung		1			
Bremssystem					
Sitzeinheit					
Vorderrad	2	3	3		
Hinterrad	2	3			

Tabelle 14: Kumulierte Widersprüche des Fahrrad-Beispiels

Die anderen drei Widersprüche, in die die Baugruppen Vorder- und Hinterrad involviert sind, ergeben sich hingegen nicht aus der fehlenden Kenntnis des Beispiels. Hier änderte sich vielmehr das Verständnis der Probanden, wie ein Bauteil zu der Erfüllung einer Funktion beiträgt, während der Studie. Während vorher auf der abstrakten Bewertungsebene nur Abhängigkeiten zwischen Funktionen und ihren tatsächlichen Funktionsträger dokumentiert wurden, zeigte sich bei den Rädern eine Verschiebung hin zu einem Kraftflussdenken. Bei dieser Betrachtungsweise sind dann bspw. bei der Funktion „Bremsen ermöglichen“ nicht mehr nur das eigentliche Bremssystem, sondern auch die Nabe, die Speichen, die Felge und die Reifen beteiligt, da sie zur Übertragung der Bremskräfte beitragen.

Tabelle 15 beinhaltet die aufgetretenen Widersprüche des Kaffeevollautomaten-Beispiels. Der am häufigsten aufgetretene Widerspruch ist der zwischen der Elementekombination „Wasserbehälter / Benutzer warnen“. Er liegt darin begründet, dass ein Wasserstandssensor in der Baugruppe „Wasserbehälter“ integriert ist, den ca. 80 % der Probanden nicht dort, sondern eher im elektrischen System vermutet hätten.

Gleiches gilt für die Widersprüche mit der Funktion „Umwelteinflüsse fernhalten“ in Kombination mit den Baugruppen „Bohnspeicher“, „Wasserbehälter“, „Mahleinheit“ und „Brüheinheit“, die entweder über einen Deckel verfügen oder über ein Gehäuse, welche bei der Analyse der Baugruppen auf Abhängigkeiten vergessen wurden. Grund dafür war, dass die Probanden die Realisierung dieser Funktion allein beim Gehäuse gesehen und somit auf abstrakter Ebene nicht weiter berücksichtigt haben.

Bei den beiden Elementekombinationen „Mahleinheit / Kaffee zubereiten“ und „Fluidsystem / Kaffee zubereiten“ ist hingegen nicht die fehlende Kenntnis der Produktstruktur sondern der Funktionshierarchie ausschlaggebend für die Entstehung der Widersprüche. Bei der Nachbesprechung stellte sich hier heraus, dass die Probanden eine sehr unterschiedliche Vorstellung von der Funktion „Kaffee zubereiten“ hatten. Während der eigentliche Beispielaufbau vorsah, dass alle Teilfunktionen ab „Drücken

des Kaffeebezugsknopfes“ beinhaltet sind („Bohnen mahlen“, „Kaffeepulver verdichten“ usw.), verstanden viele der Probanden lediglich das Versetzen des Kaffeepulvers mit Wasser unter dieser Funktion.

							Bohnen aufbewahren
							Umwelteinflüsse fernhalten
							Abfall sammeln
							Frischwasser bereitstellen
							Kaffee zubereiten
							Zubereitung steuern
							Benutzer warnen
Gehäuse			2	1		2	
Bohnspeicher	11						
Auffangeinheit	2					3	
Wasserbehälter	6				2	14	
Mahleinheit	7			4			
Brüheinheit	11						
Fluidsystem	2	2	8	6	3		
Elektrisches System				1	2		

Tabelle 15: Kumulierte Widersprüche des Kaffeevollautomaten-Beispiels

Wie bereits erwähnt wurde die Hypothese 1* im Rahmen der Studie in 95,3 % (Haus-Beispiel), 95 % (Kaffeevollautomaten-Beispiel) und 91 % der Fälle (Fahrrad-Beispiel) bestätigt. In den verbleibenden 4,7 % bis 9 % der Fälle, in denen ein Widerspruch zur Hypothese und somit eine Bestätigung der Nullhypothese auftritt, können zwei grundsätzliche Arten unterschieden werden: solche, die sich aus der nicht ausreichenden Kenntnis der in den Beispielprodukten verwendeten Funktions- und Produktstrukturen ergeben und solche, die auf fälschlicherweise modellierten Tracelinks basieren, bei denen eigentlich keine Abhängigkeit vorliegt (wenn z. B. der Kraftfluss als Bewertungsgrundlage angewendet wird). Es trat kein Fall auf, welcher der anfangs formulierten Hypothese 1* tatsächlich widerspricht.

Die Hypothese 1* kann somit für die Entwicklung einer Methode zur effizienten Modellierung von Tracelinks verwendet werden. Entgegen der Ausführungen in Kapitel 5.2.1.1 ist es jedoch für alle Hierarchiearten notwendig, dass die Anwender der Methode über umfassende Kenntnisse der zu analysierenden Artefakte verfügen. Für die Erfüllung dieser Voraussetzung spricht im industriellen Umfeld, dass sich die Strukturierung der Artefakte nicht wie in der Studie an den Vorstellungen einer Person orientiert, sondern an festen Regeln und Vereinbarungen, die den Entwicklern bekannt sind. Zusätzlich wird die Methode nicht durch unbeteiligte Personen, sondern durch Experten angewendet, die umfassenden Kenntnisse der Systeme, deren Strukturierung sowie deren Funktionen haben.

Es lassen sich somit zwei weitere Anforderungen für die Umsetzung der Methode EcoTracing formulieren:

- Anforderung 2: Eine Methode, die die transitiven Eigenschaften von Inklusionshierarchien zur effizienten Modellierung von Tracelinks nutzt, kann nur von Anwendern mit umfassenden Kenntnissen der zu analysierenden Artefakte genutzt werden.
- Anforderung 3: Eine Methode, die die transitiven Eigenschaften von Inklusionshierarchien zur effizienten Modellierung von Tracelinks nutzt, muss so realisiert werden, dass sich die Anwender jederzeit über den Kontext (Identifikation der Eltern- und Kinderelemente) eines Elements informieren können.

5.4 ECOTRACING – METHODE ZUR EFFIZIENTEN MODELLIERUNG VON TRACELINKS

Die im Folgenden beschriebene Methode EcoTracing basiert auf der in Kapitel 5.3 evaluierten Hypothese 1*. Sie nutzt den hierarchischen Aufbau der im Systems Engineering verwendeten Artefakte, um während der Erfassung von Tracelinks Entscheidungen über Abhängigkeiten eines betrachteten Elternelements auf seine Kinderelemente zu übertragen. Ziel ist es dabei die Anzahl der notwendigen Entscheidungen zu reduzieren.

Die Methode EcoTracing wurde durch den Autor dieser Arbeit entwickelt und im Rahmen der Studienarbeit von Michel Bornath prototypisch implementiert [Bornath 2011a]. Die Ergebnisse wurden auf der International Conference on Engineering Design vorgestellt [Stark und Figge 2011] und in [Beier et al. 2011] sowie [Königs et al. 2012] veröffentlicht.

5.4.1 FUNKTIONSPRINZIP DER METHODE ECOTRACING

Bei der Methode EcoTracing handelt es sich um einen Top-Down-Ansatz, bei dem die zu untersuchenden Artefakte ausgehend von hohen hierarchischen Ebenen analysiert werden. Die Detaillierungsstufe, auf der die Elemente auf Abhängigkeiten untersucht werden, wird schrittweise erhöht, bis die benötigte Granularität der Tracelinks erreicht ist. Dies bringt zum einen den bereits erwähnten Vorteil mit sich, dass bereits auf hoher hierarchischer Ebene Entscheidungen über Abhängigkeiten getroffen und somit der Aufwand zur Erfassung von Tracelinks reduziert werden kann. Zum anderen ermöglicht dieser Ansatz eine Vorgehensweise, wie sie in Kapitel 5.2.4 als Anforderung 1 formuliert wurde: Tracelinks können zunächst auf einer groben Granularitätsstufe und erst bei Bedarf in einzelnen Teilbereichen der Artefakte detaillierter modelliert werden.

Dabei nutzt die Methode EcoTracing die transitiven Eigenschaften von Inklusionshierarchien, um den Aufwand bei der Erfassung von Tracelinks zu reduzieren. Abbildung 49 stellt diese Vorgehensweise am abstrakten Beispiel einer Anforderungsspezifikation und einer Funktionshierarchie dar. Ausgehend von Anforderung R1 wird überprüft, ob zur Funktion F2 eine Abhängigkeit besteht. Da dies nicht der Fall ist wird kein Tracelink modelliert (dargestellt durch das Kreuz in Abbildung 49, linke Seite). Basierend auf dieser Entscheidung wird die hierarchische Strukturierung der Funktionshierarchie ausgenutzt, um diese Festlegung auch auf F2.1 sowie F2.2 zu übertragen (dargestellt durch die Kreuze in Abbildung 49, rechte Seite). Die Überprüfung der Kombinationen mit R1 können eingespart werden.



Abbildung 49: Funktionsprinzip der Methode EcoTracing

5.4.2 ALGORITHMUS DER METHODE ECOTRACING

Der Top-Down Algorithmus der Methode EcoTracing wird im Folgenden anhand der beiden generischen Artefakte A und B erläutert. Ausgehend von Artefakt A wird zunächst das Element A1 in Kombination mit den Elementen der höchsten hierarchischen Ebene des Artefakts B (B1, B2 und B3) auf Abhängigkeiten untersucht (dargestellt durch Fragezeichen in Abbildung 50).

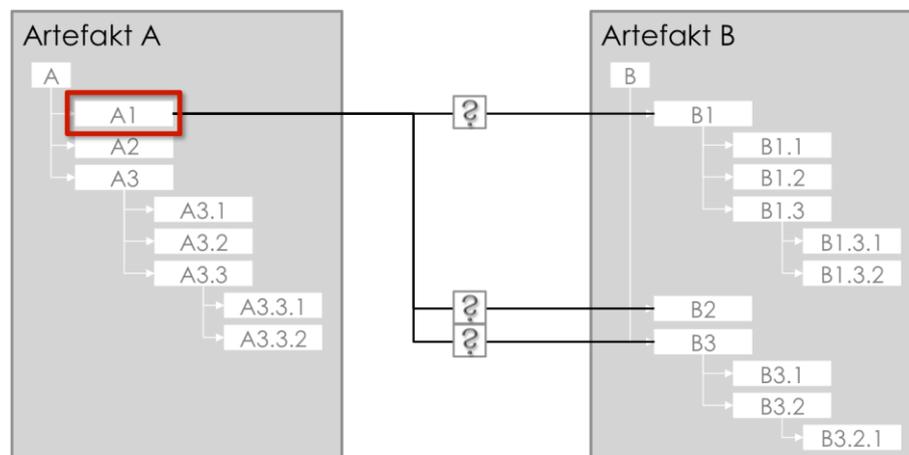


Abbildung 50: Untersuchung der Artefakte auf Abhängigkeiten zwischen den Elementen

Werden zwischen A1 und einem der Elemente Bx keine Abhängigkeiten festgestellt, so wird dies jeweils durch Markierung der Elementkombination mit einem „X“ doku-

mentiert. Wie in Kapitel 5.4.1 beschrieben, können aus dieser Festlegung bereits auf dieser Betrachtungsebene die Abhängigkeit zwischen A1 und einem der Kinderelemente von B1 ausgeschlossen werden und brauchen deshalb nicht mehr analysiert werden (dargestellt durch Kreuze in Abbildung 51). Je nach Tiefe der Hierarchie können hierdurch eine große Anzahl von Kombinationen noch vor ihrer Betrachtung ausgeschlossen werden.

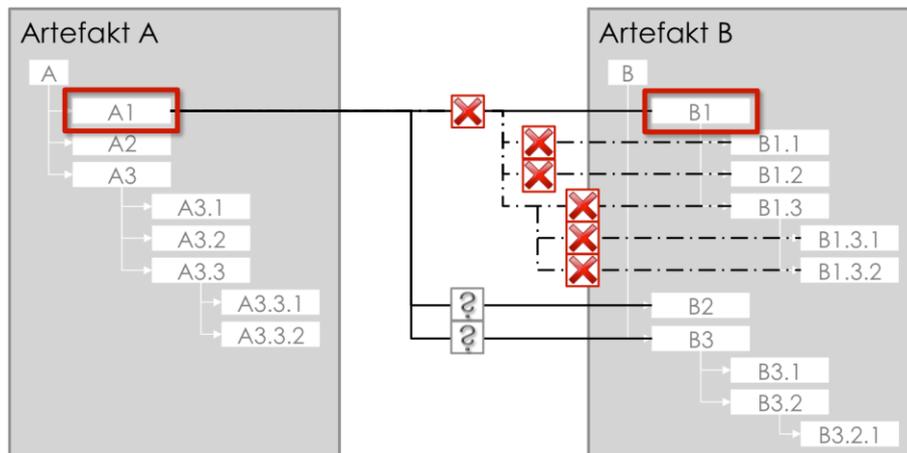


Abbildung 51: Markierung von Elementkombinationen bei nicht vorhandener Abhängigkeit

Wird zwischen den betrachteten Elementen eine Abhängigkeit festgestellt, wie in Abbildung 52 durch den Haken auf der Verbindungslinie zwischen A1 und B3 visualisiert, so wird ein Tracelink zwischen diesen modelliert. Zeitgleich werden alle Kombinationen von A1 mit den direkten Kinderelementen von B3 für eine spätere Untersuchung vorgemerkt (dargestellt durch eine Fahne in Abbildung 52), da die Wahrscheinlichkeit, dass mindestens eines der Elemente eine Abhängigkeit zu A1 aufweist, sehr groß ist. Die Markierung dient dabei dem Zweck, die Erfassung der Tracelinks abbrechen, zu einem beliebigen späteren Zeitpunkt wieder aufnehmen und die bisherige Analyse nachvollziehen zu können. Insgesamt lassen sich mit Hilfe der unterschiedlichen Markierungen folgende Zustände von Elementkombinationen unterscheiden:

- Elementkombinationen, welche bereits untersucht wurden, aber keine Abhängigkeit zwischen ihnen festgestellt werden konnte, 
- Elementkombinationen, welche noch nicht auf Abhängigkeiten zueinander untersucht wurden und 
- Elementkombinationen, bei denen eine Abhängigkeit wahrscheinlich ist, eine detaillierte Untersuchung jedoch noch aussteht. 

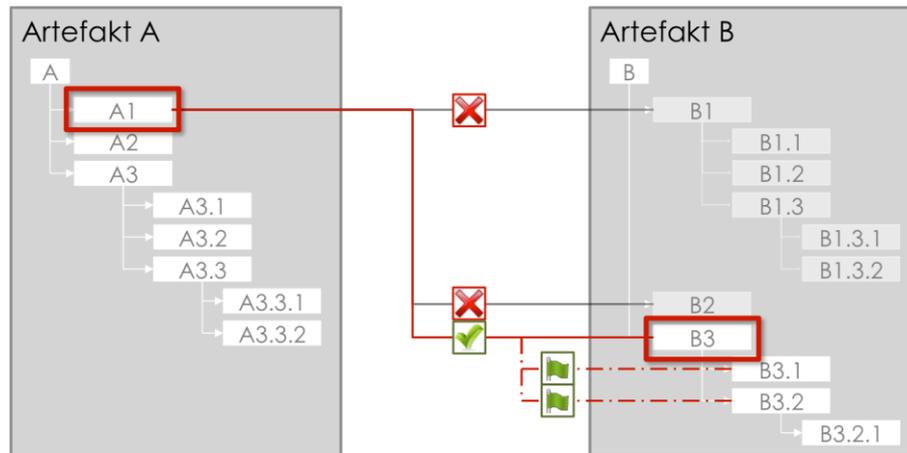


Abbildung 52: Modellierung von Tracelinks und Markierung der Kinder-Elemente für eine spätere Untersuchung

Sobald ausgehend von A1 alle Kombinationen mit Elementen der höchsten hierarchischen Ebene von Artefakt B analysiert wurden, werden zunächst alle Geschwisterelemente³² von A1 in gleicher Weise untersucht. Erst im nächsten Schritt werden die tieferen hierarchischen Ebenen der beiden Artefakte auf Abhängigkeiten analysiert. Diese Vorgehensweise ermöglicht es dem Anwender, beide Artefakte zunächst vollständig auf abstraktem Niveau auf Abhängigkeiten zu untersuchen und die Granularität im Anschluss schrittweise zu erhöhen (wie in Anforderung 1 gefordert). Eine detaillierte Darstellung des beschriebenen Algorithmus ist in Abbildung 53 zu finden. Dabei werden die betrachteten Artefakte mit A und B bezeichnet. Die Hierarchieebene wird mit Hilfe des Hierarchie-Indexes A_H bzw. B_H ausgedrückt, wobei die oberste Ebene mit dem Index 1 bezeichnet wird. Elemente, die auf derselben Hierarchieebene liegen (auch Geschwisterelemente genannt) werden mit dem Geschwister-Index durchnummeriert.

Ergänzend zu diesem Ablauf wurde in der prototypischen Implementierung (siehe Kapitel 5.4.3) ein weiterer, leicht abgewandelter Algorithmus umgesetzt. Dieser sieht vor, dass ausgehend von A1 eine Erfassung der Tracelinks direkt bis in die niedrigste Granularitätsstufe von Artefakt B durchgeführt wird. Diese Vorgehensweise bietet sich an, wenn ohnehin eine hohe Granularität angestrebt wird, da sich der Anwender auf die Abhängigkeiten zwischen zwei Elementen und deren Kinderelementen konzentrieren kann und der Kontext nicht ständig geändert wird.

³² Als Geschwisterelemente werden Elemente bezeichnet, die auf einer hierarchischen Ebene liegen.

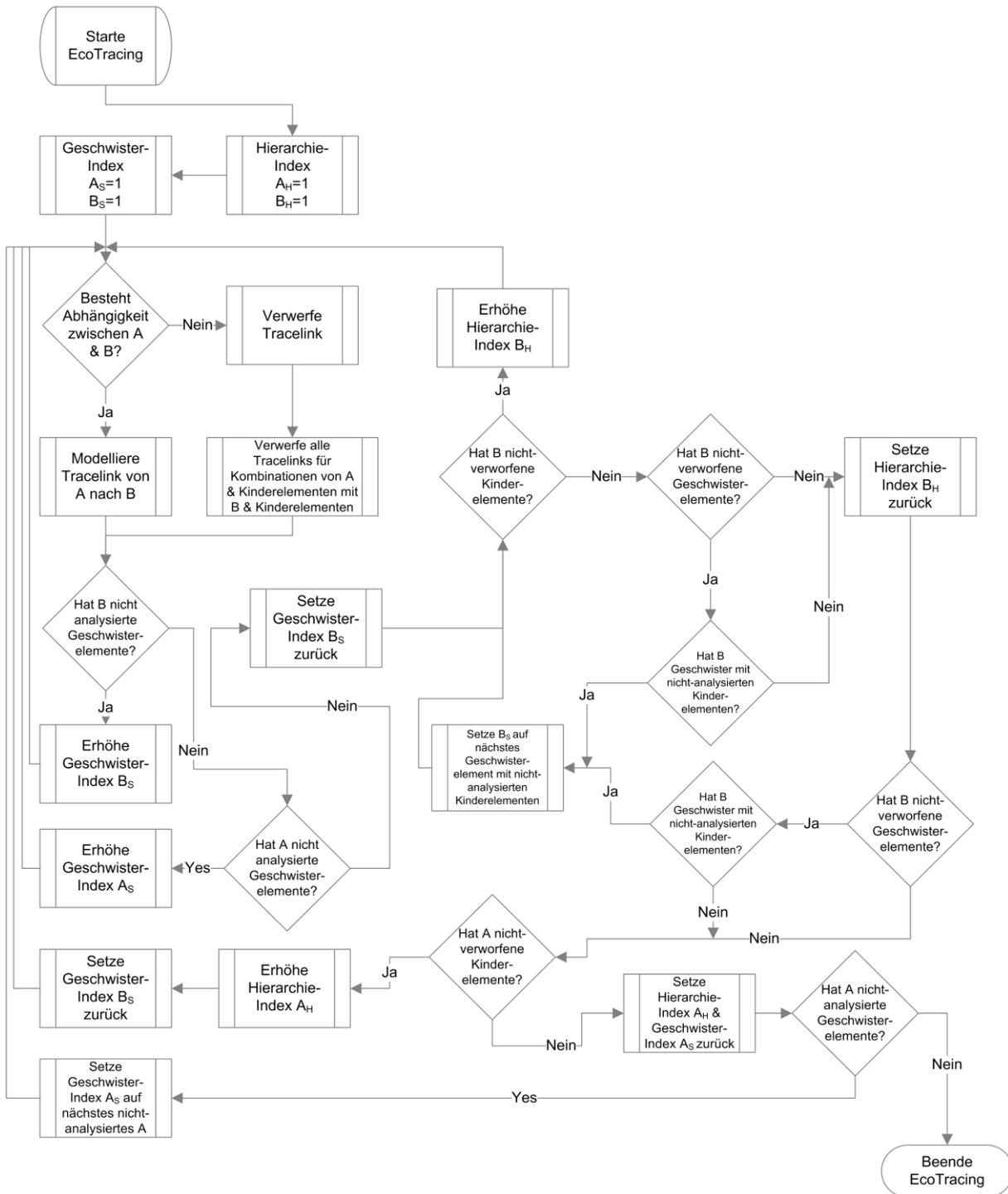


Abbildung 53: Algorithmus der EcoTracing Methode (zunächst Erfassung der Tracelinks auf hoher hierarchischer Ebene und folgend schrittweise Erhöhung der Granularität).

5.4.3 PROTOTYPISCHE IMPLEMENTIERUNG DER METHODE ECOTRACING

Bei der Methode EcoTracing handelt es sich um eine sog. Offline-Methode zur Erfassung von Tracelinks. Ihre Anwendung erfolgt damit separat und nicht während der Erstellung der Artefakte. Im Vergleich zur manuellen Online-Erfassung liegt der Vorteil der Offline-Erfassung darin, dass Tracelinks nicht während der Modellierung der Artefakte erfasst werden, sondern im Nachgang strukturiert vorgegangen wird (siehe Abbildung 54). Zwar wirkt die Online-Erfassung zunächst intuitiver, da die Entwickler die Tracelinks sofort bei Auftreten einer Abhängigkeit modellieren, allerdings kann durch diese unstrukturierte Vorgehensweise eine Vollständigkeit der Abhängigkeitsanalyse nicht garantiert werden³³.

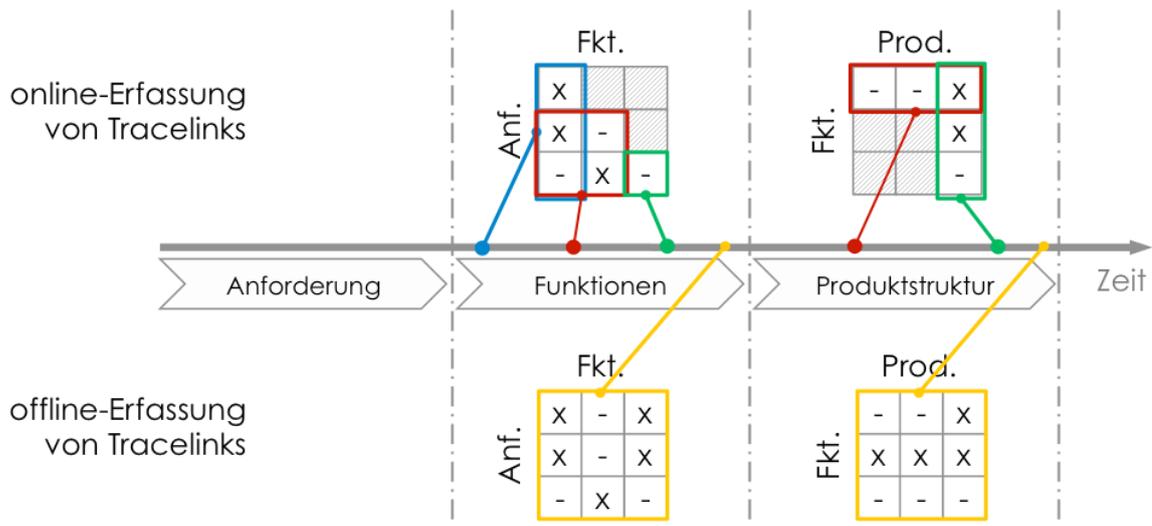


Abbildung 54: Vergleich der Online- mit der Offline-Erfassung von Tracelinks: kontinuierliche, jedoch unvollständige Online-Analyse gegenüber einer vollständigen Offline-Analyse zu bestimmten Zeitpunkten im Entwicklungsprozess (hier dargestellt: zum Ende der jeweiligen Phase)

Um EcoTracing als Offline-Methode jedoch auch während der Entwicklung anwenden zu können sind zwei Anwendungsfälle zu unterscheiden:

Fall 1: Artefakte haben einen finalen Zustand: Wenn beide zu untersuchenden Artefakte bereits einen finalen Zustand haben und damit nicht mehr bzw. nur noch wenig bearbeitet werden, kann EcoTracing ohne Einschränkungen zur Abhängigkeitsanalyse eingesetzt werden.

³³ Es ist grundsätzlich möglich, EcoTracing als offline-Methode mit der online-Erfassung zu kombinieren. In diesem Fall würden während der Erstellung der Artefakte online Tracelinks modelliert und abschließend eine offline-Erfassung durchgeführt, um die Vollständigkeit garantieren zu können. Bei dieser Analyse können die online erfassten Tracelinks berücksichtigt werden, jedoch müssten alle Element-Kombinationen, die keinen Tracelink aufweisen nachträglich hinsichtlich vorhandener Abhängigkeiten analysiert werden.

Fall 2: Artefakte haben einen Zwischenzustand: Auch wenn eins der beiden oder beide Artefakte noch keinen finalen Zustand haben, ist die Anwendung der Methode EcoTracing möglich. In diesem Fall sollte der EcoTracing-Algorithmus so konfiguriert werden, dass die Abhängigkeitsanalyse Ebenen-weise von abstrakt nach detailliert durchgeführt wird (siehe Kapitel 5.4.2, Seite 121). Die Analyse wird dann bis zu einer beliebigen Granularität durchgeführt, die nur von der bis dahin fertiggestellten Detaillierungstiefe der Artefakte begrenzt wird. Zu einem späteren Zeitpunkt, wenn die Artefakte weiterentwickelt und somit ein neuer Zwischenstand erreicht wurde, kann die Analyse an der gleichen Stelle wieder aufgenommen werden, da allen Elementekombinationen ein Analyse-Status zugeordnet wird. Grundlegende Voraussetzung für die Anwendung von EcoTracing ist jedoch, dass die Artefakte Ebenen-weise von abstrakt nach detailliert entwickelt werden und sich nach Erstellung der Elemente keine gravierenden Änderungen mehr ergeben. Sollten doch noch Änderungen auf abstrakter Ebene (bspw. durch Hinzufügen eines Elements) in einem der Artefakte auftreten, so muss dieses nachträglich in Kombination mit allen Elementen des anderen Artefakts untersucht werden. Gleiches gilt für ein Elternelement, wenn dessen Kinderelemente gravierend geändert werden und sich dadurch seine Eigenschaften ebenfalls ändern. So müsste bspw. das Modul „Außenspiegelverstellung“ in einem Verhaltensmodell erneut auf Abhängigkeiten zu anderen Artefakten untersucht werden, wenn dessen Wirkprinzip von manueller auf elektrische Verstellung geändert und damit viele seiner Systemelemente ausgetauscht werden würden.

Vor dem Hintergrund dieser beiden Anwendungsfälle wurde die prototypische Implementierung der Methode EcoTracing realisiert. Sie wurde als Plug-In in den in Kapitel 3.6.4.3 vorgestellten ModelTracer als Wizard (genannt: EcoTracer), der den Anwender strukturiert durch den Prozess der Erfassung von Tracelinks führt, integriert.

Das EcoTracing beginnt mit der Auswahl der zu analysierenden Artefakte über zwei Drop-Down-Menüs (Abbildung 55, Nr. 1). Die so ausgewählten Artefakte werden anschließend in einer reduzierten Darstellung visualisiert (Abbildung 55, Nr. 2). Elemente, die bereits analysiert wurden und deren Abhängigkeit bereits zuvor ausgeschlossen wurde, werden nicht angezeigt, was die Übersichtlichkeit durch Reduzierung der Anzahl angezeigter Elemente für den Anwender erhöht. In den so dargestellten Artefakten können im Anschluss die Bereiche, die im Rahmen der Tracelink-Erfassung analysiert werden sollen, durch Mehrfachauswahl eingegrenzt werden. Dabei ist es möglich, die Auswahl auf beliebigen Hierarchieebenen zu treffen. Diese Auswahl ermöglicht es, bspw. nur eine bestimmte Elementekombination und alle ihre Kinderelemente direkt bis in eine hohe Granularität zu untersuchen und Verantwortlichkeiten für die Verknüpfung bestimmter Artefaktbereiche zu berücksichtigen.

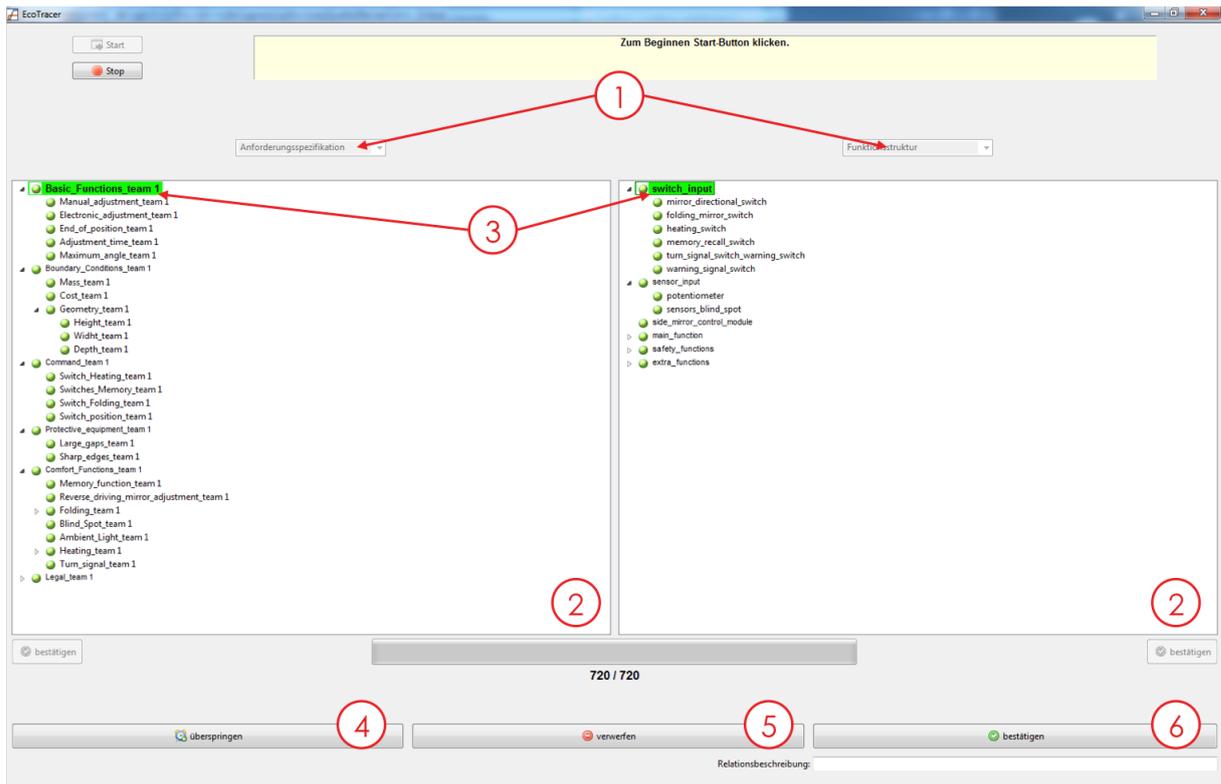


Abbildung 55: Graphical User Interface (GUI) des EcoTracers

Im Anschluss an die Bereichsauswahl beginnt die Erfassung der Tracelinks. Die aktuell betrachtete Elementkombination wird grün hervorgehoben (Abbildung 55, Nr. 3). Um sich dabei den Kontext des betrachteten Elements vergegenwärtigen zu können, ist es möglich, die Hierarchiebäume auf- bzw. zuzuklappen³⁴. Für jede der so betrachteten Elementkombination hat der Anwender drei Möglichkeiten zur Auswahl:

- „Überspringen“: Es wird aktuell keine Entscheidung für die Kombination getroffen (Abbildung 55, Nr. 4). Dies überträgt sich auch auf deren Kinderelemente, die in der aktuellen Analysesitzung nicht mehr zur Überprüfung vorgeschlagen werden.
- „Verwerfen“: Es wird bestätigt, dass keine Abhängigkeit zwischen der aktuellen Elementkombination vorhanden ist (Abbildung 55, Nr. 5). Somit wird kein Tracelink modelliert. Diese Aussage überträgt sich auch auf deren Kinderelemente, die weder in der aktuellen Analysesitzung, noch zu einem späteren Zeitpunkt überprüft werden müssen.

³⁴ Diese Funktionalität adressiert die Anforderung 3 aus Kapitel 5.3.2, in der gefordert wird, dass man jederzeit Kinder- und Elternelemente visualisieren können muss, um bspw. abschätzen zu können, welche Bauteile in einer Baugruppe enthalten sind.

- „Bestätigen“: Es wird eine Abhängigkeit zwischen der hervorgehobenen Elementekombination bestätigt und ein Tracelink modelliert (Abbildung 55, Nr. 6). Zusätzlich besteht die Möglichkeit diesem Tracelink im Feld „Relationsbeschreibung“ eine Beschreibung (z. B. den Grund für den Tracelink oder Typ des Tracelinks) hinzuzufügen. Alle Kinderelemente werden im weiteren Verlauf der Sitzung zur Analyse vorgeschlagen.

Diese Status werden separat für jede Artefakt-Kombination gespeichert. Die gewählte Datenstruktur entspricht einer Matrix, da so für jedes Elementepaar ein Status gespeichert werden kann. Dabei wird für jedes Elementepaar zunächst ein Status „initial“ vergeben, um zu einem späteren Zeitpunkt feststellen zu können, welche Kombinationen bereits vom Anwender analysiert wurden und wo sich der Wiedereinstiegspunkt für die Fortsetzung der Analyse befindet. Während für die Funktionalität „Überspringen“ kein Status vergeben wird, werden „Verworfen“ und „Bestätigt“ den Kombinationen zugeordnet. Dabei wird der erstgenannte auch auf die Kinderelemente übertragen und bei letztgenanntem den jeweiligen Kinderelementen der Status „möglich“ zugeordnet. Zusätzlich werden die Tracelinks, die durch das Bestätigen einer Abhängigkeit erstellt werden, in die Datenbank des ModelTracers eingetragen.

Sobald eine der drei Möglichkeiten durch den Anwender gewählt wird, springt der Algorithmus zur nächsten Elementekombination. Der strukturierte Ablauf der Erfassung erfolgt dabei wie in Kapitel 5.4.2 beschrieben. Dadurch können keine Elementekombinationen vergessen werden, was im Vergleich zu der manuellen Analyse von Artefakten ein großer Vorteil ist. Der Fortschritt der Erfassung wird als Verhältnis von bereits analysierten zu insgesamt zu analysierenden Kombinationen mittels eines Fortschrittsbalkens visualisiert. Letztere Zahl ergibt sich dabei aus der Multiplikation der Anzahl der Elemente der beiden Artefakte (siehe Formel (1) aus Kapitel 5.2.3).

5.4.4 ERMITTLUNG DES AUFWANDS BEI DER ERFASSUNG VON TRACELINKS MIT ECO TRACING

Die Ermittlung der Anzahl notwendiger Entscheidungen mit Hilfe von EcoTracing unterscheidet sich von der bei manueller Vorgehensweise (vgl. Kapitel 5.2.3), da nicht alle Elementekombinationen berücksichtigt werden müssen. Die folgenden Erläuterungen gelten für Artefakte mit zwei Hierarchieebenen, das Prinzip ist jedoch auf Artefakte mit beliebigen Hierarchietiefen übertragbar. Um die Anzahl notwendiger Entscheidungen bei der Anwendung von EcoTracing zu ermitteln, muss zunächst auf der höheren Hierarchieebene überprüft werden, ob ein Tracelink zwischen einem Elementepaar vorliegt. Ist dies der Fall, müssen diese und alle Elementekombinationen ihrer Kinderelemente zur Anzahl notwendiger Entscheidungen addiert werden. Liegt keine Abhängigkeit vor, ist nur die Entscheidung, für dieses Elementepaar keinen

Tracelink zu modellieren, notwendig. Für dessen Kinder Elemente lautet die automatische Entscheidung, alle Abhängigkeiten auszuschließen.

E_k sei die Anzahl der Entscheidungen für eine Kombination der Elternelemente A_a und B_b , während p_a und q_b für die jeweilige Anzahl der Kinder Elemente dieser Elternelemente (inkl. dem Elternelement selbst) stehen. Die Indizes a und b entsprechen der Anzahl der Elemente auf oberster Hierarchieebene und laufen von 1 bis r bzw. s (siehe Abbildung 56).

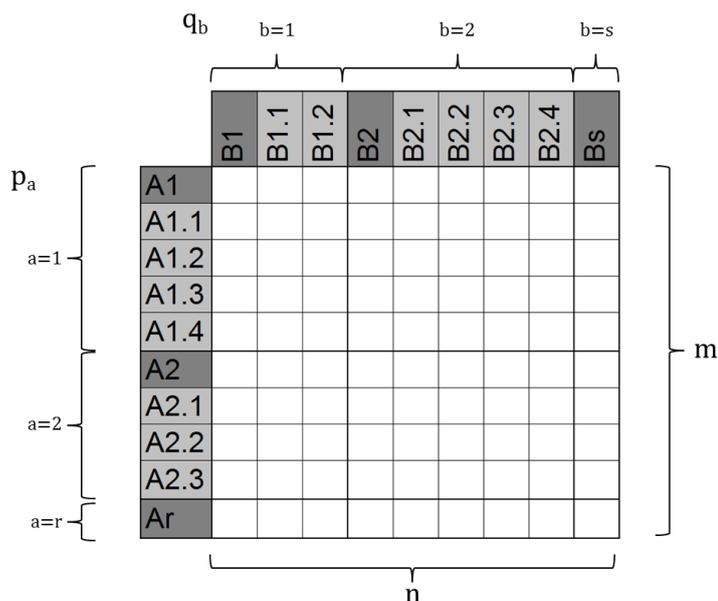


Abbildung 56: Notwendige Parameter zur Bestimmung des Aufwands unter Verwendung von EcoTracing

Die Anzahl notwendiger Entscheidungen mittels EcoTracing E_{ET} ergibt sich somit zu:

$$E_{ET} = \sum_{k=1}^{r \cdot s} E_k \quad (8)$$

$$E_k = \begin{cases} p_a \cdot q_b, & (A_a, B_b) = 1 \\ 1, & (A_a, B_b) = 0 \end{cases} \quad \text{für } a = 1 \dots r \text{ und } b = 1 \dots s \quad (9)$$

Die Unterscheidung in Formel (9) ist notwendig, da bei Existenz einer Abhängigkeit der Elternelemente zusätzlich alle Kombinationen der Kinder Elemente überprüft werden und somit Entscheidungen getroffen werden müssen. Besteht jedoch keine Abhängigkeit zwischen diesen Elementen, so muss nur bei den Elternelementen die Entscheidung getroffen werden, keinen Tracelink zu modellieren. Alle weiteren Entscheidungen entfallen.

In Abbildung 57 ist ein Beispiel in einer Matrix dargestellt, anhand dessen die Berechnung des Aufwands exemplarisch erläutert werden soll. Die modellierten Tracelinks

auf höchster hierarchischer Ebene zwischen A1 und B1 sowie A2 und B2 sind durch ein „x“ dargestellt, während zwischen A1 und B2 sowie A2 und B1 keine Abhängigkeit besteht und daher kein Tracelink modelliert wurde (dargestellt durch ein „-“).

		q ₁			q ₂				
		B1	B1.1	B1.2	B2	B2.1	B2.2	B2.3	B2.4
p ₁	A1	x			-				
	A1.1								
	A1.2		E ₁				E ₂		
	A1.3								
	A1.4								
p ₂	A2	-			x				
	A2.1		E ₃				E ₄		
	A2.2								
	A2.3								

Abbildung 57: Beispiel zur Erläuterung der Ermittlung der Anzahl der Entscheidungen unter Verwendung von EcoTracing (Elementkombinationen, bei denen eine Entscheidung getroffen werden muss, sind schraffiert dargestellt)

Die Anzahl notwendiger Entscheidungen für den Fall, dass EcoTracing verwendet wird, beträgt somit:

$$E_1 = p_1 \cdot q_1 = 5 \cdot 3 = 15 \quad (10)$$

$$E_2 = 1 \quad (11)$$

$$E_3 = 1 \quad (12)$$

$$E_4 = p_2 \cdot q_2 = 4 \cdot 5 = 20 \quad (13)$$

$$E_{ET} = \sum_{k=1}^4 E_k = 15 + 1 + 1 + 20 = 37 \quad (14)$$

Für das vorliegende Beispiel müssen somit unter Anwendung von EcoTracing 37 Entscheidungen getroffen werden.

5.4.5 EVALUATION DER METHODE ECOTRACING

Um die Aufwandsersparnis der Methode EcoTracing zu ermitteln und damit die Wirksamkeit der Hypothese 1 zu überprüfen, wurde eine Studie mit mehreren Probanden durchgeführt. Hypothese 1 zielt auf die Ausnutzung der Eigenschaften von Inklusionshierarchien, um Einsparungen bei der Erfassung von Tracelinks zu erreichen (siehe Kapitel 4.1).

Der Aufbau und die Durchführung der Studie werden im folgenden Kapitel 5.4.5.1 vorgestellt. Die Beschreibung der Auswertung und Diskussion der Ergebnisse befindet sich im Kapitel 5.4.5.2.

5.4.5.1 AUFBAU UND DURCHFÜHRUNG DER STUDIE

Zur Überprüfung der Hypothese 1 wurde eine Studie mit 19 Probanden am Produktionstechnischen Zentrum Berlin durchgeführt. Die Teilnehmer waren studentische Hilfswissenschaftler und wissenschaftliche Mitarbeiter, denen das grundsätzliche Themengebiet der Abhängigkeiten zwischen unterschiedlichen Artefakten in der Systementwicklung bekannt war.

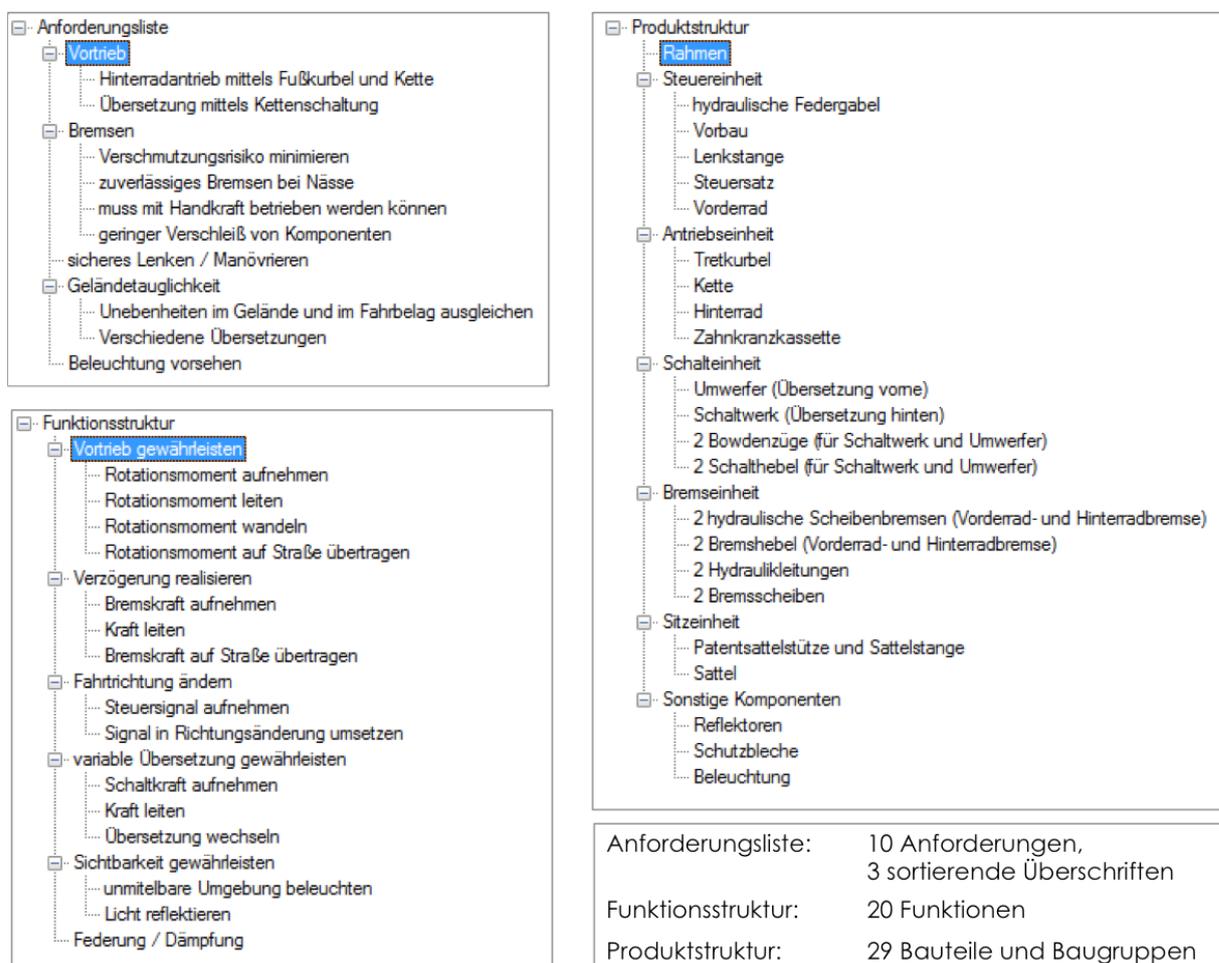


Abbildung 58: Übersicht der Artefakte des Fahrradbeispiels

Die im Rahmen der Studie verwendeten drei Artefakte waren das Anforderungsmodell, die Funktionsliste und die Produktstruktur eines Fahrrads, die in Vorbereitung der Studie erstellt wurden. Während das Anforderungsmodell als subsumtive Inklusionshierarchie strukturiert war, waren die Funktionsliste und die Produktstruktur als kompositionelle Inklusionshierarchie aufgebaut.

Das Anforderungsmodell beinhaltete 10 Anforderungen und drei Kategorien auf zwei Hierarchieebenen, die Funktionsliste 20 Funktionen auf zwei Hierarchieebenen und die Produktstruktur 29 Bauteile und Baugruppen ebenfalls auf zwei Hierarchieebenen. Eine Übersicht der Artefakte ist in Abbildung 58 abgebildet.

Zu Beginn der Studie erhielten die Probanden eine Einweisung zum Thema Traceability und der Funktionsweise der prototypischen Implementierung der Methode EcoTracing, welche im Rahmen der Studie für die Abhängigkeitsanalyse am PC verwendet werden sollte. Im Anschluss erhielten sie die Aufgabenstellung, welche aus drei Teilaufgaben bestand, wobei sich nur die erste auf die Methode EcoTracing bezog (siehe Anhang B). Bei den anderen beiden handelte es sich um zwei Auswirkungsanalysen, von denen eine der Evaluierung der Methode TraceEvaluation diente (siehe Kapitel 6.3.3). Unter Anwendung des EcoTracing-Algorithmus sollten die Probanden die drei Artefakte auf Abhängigkeiten untersuchen und entsprechend Tracelinks bis zur höchsten Granularitätsstufe modellieren. Das dabei entstehende Traceability-Modell wurde für jeden Probanden einzeln gespeichert, sodass eine individuelle Auswertung möglich war. Während der Durchführung der Studie konnten die Probanden Fragen an den Versuchsleiter stellen.

In Tabelle 16 sind die Eigenschaften der Studie in Tabellenform als Steckbrief zusammengefasst.

Analyseeinheit	Anzahl der notwendigen Entscheidungen bei der Analyse der Artefakte auf Abhängigkeiten unter Verwendung der Methode EcoTracing im Vergleich zur manuellen Vorgehensweise (vgl. dazu Kapitel 5.2.3)
Analysemethoden	Vergleich der Anzahl der Entscheidungen für EcoTracing und für die manuelle Vorgehensweisen pro Proband
Methode zur Datenerfassung	Nachträgliche Analyse der Abhängigkeitsmatrizen, die durch das EcoTracing generiert werden
Aufgabenstellung	Identifikation von Abhängigkeiten und Modellierung von Tracelinks unter der Verwendung von EcoTracing; realistische Aufgabenstellung, die an industrielle Aufgabenstellungen angelehnt ist, sich jedoch in Ausprägung der Artefakte hinsichtlich Umfang und Komplexität unterscheidet
Durchschnittliche Dauer der Studie	ø 32 Minuten 10 Sekunden
Anzahl der Teilnehmer	19
Charakterisierung der Teilnehmer	Studentische Hilfskräfte und wissenschaftliche Mitarbeiter aus dem Bereich Virtuelle Produktentstehung des Fraunhofer IPK und dem Fachgebiet Industrielle Informationstechnik der Technischen Universität Berlin; Grundsätzliche Kenntnisse über Artefakte, deren Entwicklung und Abhängigkeiten zwischen diesen vorhanden

Untersuchungsobjekt	Drei Artefakte eines Fahrrads: <ul style="list-style-type: none"> - Anforderungsmodell mit 10 Anforderungen und 3 sortierenden Überschriften mit zwei Hierarchieebenen, - Funktionsliste mit 20 Funktionen auf zwei Hierarchieebenen, - Produktstruktur mit 29 Bauteilen und Baugruppen auf zwei Hierarchieebenen.
---------------------	---

Tabelle 16: Eigenschaften der Studie zur Überprüfung der Hypothese 1

5.4.5.2 AUSWERTUNG DER STUDIE

Im Anschluss an die Durchführung der Studie wurden die erstellten Traceability-Modelle nach Excel exportiert (siehe Tabelle 17).

	A1.1 Vortrieb	A1.1.1 Hinterradantrieb mittels Fußku...	A1.1.2 Übersetzung mittels Kettenscha...	A1.2 Bremsen	A1.2.1 Verschmutzungsrisiko minimieren	A1.2.2 zuverlässiges Bremsen bei Nässe	A1.2.3 muss mit Handkraft betrieben w...	A1.2.4 geringer Verschleiß von Compon...	A1.3 sicheres Lenken / Manövrieren	A1.4 geländetaugliche Federung und Üb...	A1.4.1 Unebenheiten im Gelände und im...	A1.4.2 Verschiedene Übersetzungen	A1.5 Beleuchtung vorsehen
F1.1 Vortrieb gewährleisten	1	1	1										
F1.1.1 Fußkraft aufnehmen	1	1	1										
F1.1.2 Antriebskraft leiten	1	1	1										
F1.1.3 Kraft / Moment wandeln	1	1	1										
F1.1.4 Rotationsmoment auf Straße übe...	1	1	1										
F1.2 Verzögerung realisieren				1		1	1						
F1.2.1 Handkraft aufnehmen				1		1	1						
F1.2.2 Kraft leiten				1		1							
F1.2.3 Bremskraft auf Hinterrad / Str...				1		1	1						
F1.3 Fahrtrichtung ändern									1				
F1.3.1 Steuersignal aufnehmen									1				
F1.3.2 Richtungsänderung ausführen									1				
F1.4 variable Übersetzung gewährleisten	1	1	1							1	1	1	
F1.4.1 Fingerkraft aufnehmen										1	1		
F1.4.2 Kraft leiten	1	1	1							1	1	1	
F1.4.3 Übersetzung wechseln	1	1	1							1		1	
F1.5 Sichtbarkeit gewährleisten													1
F1.5.1 unmittelbare Umgebung beleuchten													1
F1.5.2 Licht reflektieren													1
F1.6 Federung / Dämpfung									1	1	1		

Tabelle 17: Darstellung der durch Proband 1 modellierten Tracelinks für die Artefaktkombination (Anforderungen «» Funktionen)

Die Darstellungsform ist an Multi Domänen Matrizen (MDM) angelehnt, weshalb für die drei Artefakte Anforderungsmodell, Funktionsliste und Produktstruktur pro Proband

jeweils drei Matrizen erstellt wurden. Dabei wird jeweils ein Artefakt in den Zeilen und eins in den Spalten eingetragen und die Existenz eines ungerichteten Tracelinks durch eine „1“ in deren gemeinsamer Zelle dokumentiert.

Mit Hilfe dieser Matrizen lassen sich unter Berücksichtigung der Berechnungsvorschriften aus den Kapiteln 5.2.3 und 5.4.4 einige für das Traceability-Modell charakteristische Größen ableiten. Diese wurden in Form der „Anzahl Entscheidungen“ und der „Ersparnis“ (im Vergleich zur manuellen Vorgehensweise) pro Proband und Artefaktkombination berechnet. Das Ergebnis ist in Tabelle 18 dargestellt.

Bei der Analyse der erfassten Daten fällt auf, dass die Probanden sehr unterschiedliche Umfänge an Entscheidungen zu treffen hatten. Die maximale durch Proband 02 getroffene Anzahl betrug 830 und die minimale durch Proband 05 getroffene Anzahl 360. Die anderen 17 Probanden liegen überwiegend, mit der Anzahl der von ihnen getroffenen Entscheidungen, im Bereich des Mittelwerts von ca. 500 Entscheidungen.

	Anzahl Entscheidungen mit EcoTracing				Ersparnis mit EcoTracing (im Vergleich zur manuellen Vorgehensweise)			
	A - F	A - P	F - P	Σ	A - F	A - P	F - P	Σ
Proband 01	91	144	197	432	65,0 %	61,8 %	66,0 %	64,5 %
Proband 02	169	319	342	830	35,0 %	15,4 %	41,0 %	31,8 %
Proband 03	215	157	149	521	17,3 %	58,4 %	74,3 %	57,2 %
Proband 04	152	197	199	548	41,5 %	47,7 %	65,7 %	55,0 %
Proband 05	83	99	178	360	68,1 %	73,7 %	69,3 %	70,4 %
Proband 06	166	150	191	507	36,2 %	60,2 %	67,1 %	58,3 %
Proband 07	97	133	288	518	62,7 %	64,7 %	50,3 %	57,4 %
Proband 08	100	214	221	535	61,5 %	43,2 %	61,9 %	56,0 %
Proband 09	110	207	196	513	57,7 %	45,1 %	66,2 %	57,8 %
Proband 10	94	132	163	389	63,8 %	65,0 %	71,9 %	68,0 %
Proband 11	91	140	189	420	65,0 %	62,9 %	67,4 %	65,5 %
Proband 12	91	112	162	365	65,0 %	70,3 %	72,1 %	70,0 %
Proband 13	137	146	199	482	47,3 %	61,3 %	65,7 %	60,4 %
Proband 14	83	213	196	492	68,1 %	43,5 %	66,2 %	59,6 %
Proband 15	101	124	186	411	61,2 %	67,1 %	67,9 %	66,2 %
Proband 16	129	228	371	728	50,4 %	39,5 %	36,0 %	40,2 %
Proband 17	80	167	174	421	69,2 %	55,7 %	70,0 %	65,4 %
Proband 18	118	95	197	410	54,6 %	74,8 %	66,0 %	66,3 %
Proband 19	197	157	230	584	24,2 %	58,4 %	60,3 %	52,0 %
Mittelwerte	121,3	164,9	212,0	498,2	53,4 %	56,2 %	63,4 %	59,1 %
Maximal	215	319	371	830	69,2 %	74,8 %	74,3 %	70,4 %
Minimal	80	95	149	360	17,3 %	15,4 %	36,0 %	31,8 %

Tabelle 18: Studienergebnisse für die Anzahl der Entscheidungen und die Ersparnis an Entscheidungen durch Anwendung von EcoTracing pro Proband zwischen Anforderungen (A), Funktionen (F) und Produktelementen (P)

Es wird deutlich, dass die Modellierung von Tracelinks mit EcoTracing und damit die Anzahl der zu treffenden Entscheidungen sehr stark von der subjektiven Interpretation der Artefakte durch die Probanden beeinflusst wird. Zur näheren Analyse wurden daher in Tabelle 19 die modellierten Tracelinks aller 19 Probanden am Beispiel der Anforderungen und Funktionen aggregiert³⁵. Eine 19 in einer Zelle bedeutet somit, dass alle Probanden für diese Elementekombination einen Tracelink modelliert haben. Über die Existenz dieser Abhängigkeit herrschte somit Einigkeit und die Wahrscheinlichkeit, dass zwischen dieser Elementekombination tatsächlich ein Tracelink sinnvoll ist, ist groß. Umgekehrt gilt für die Nicht-Existenz einer Abhängigkeit: hat keiner der Probanden bei einem Elementepaar einen Tracelink modelliert, ist die Nicht-Existenz der Abhängigkeit sehr wahrscheinlich (dargestellt durch eine grüne Färbung der Zelle in Tabelle 19). Eine 9 hingegen bedeutet, dass 9 Probanden für diese Elementekombination einen Tracelink modelliert haben, jedoch auch 10 Probanden keine Abhängigkeit zwischen diesen Elementen gesehen haben. Die Existenz einer Abhängigkeit ist somit nicht sicher (dargestellt durch eine rote Färbung der Zelle in Tabelle 19). Werte, die zwischen den genannten liegen, werden entsprechend ihrer Nähe zu 9 in Abstufungen der Farben Grün, Rot und Gelb dargestellt. Unabhängig von der Studie wurde das Beispielprodukt auch vom Autor dieser Arbeit hinsichtlich vorhandener Abhängigkeiten untersucht. Tracelinks, die bei dieser Analyse modelliert wurden, sind in Tabelle 19 durch rote Zahlen dargestellt.

Um die Entscheidungen für oder gegen einen Tracelink und damit die Bewertung, ob eine Abhängigkeit zwischen Anforderung und Funktion besteht, näher zu betrachten, werden im Folgenden einige exemplarische Elementekombinationen beschrieben und deren Eigenschaften diskutiert:

„F1.1 Vortrieb gewährleisten“ | „A1.1 Vortrieb“: Zwischen der Anforderung „A1.1 Vortrieb“ und der Funktion „F1.1 Vortrieb gewährleisten“ sowie allen Kinderelementen der Funktion (F1.1.1 – F1.1.4) wurde durch alle Probanden eine Abhängigkeit identifiziert. Wie der Tabelle 19 zu entnehmen ist, wurde auch durch den Autor an dieser Stelle ein Tracelink modelliert. Begründet ist dies darin, dass die Funktion und ihre Teilfunktionen den Vortrieb des Fahrrads adressieren. Hilfreich ist dabei auch die teilweise übereinstimmende Bezeichnung der Elemente. Bei dem Kinderelement der Anforderung „A1.1.2 Übersetzung mittels Kettenschaltung realisieren“ zeigen sich dagegen große Unterschiede in der Bewertung durch die Probanden. Aus Sicht des Autors ist zu keinem der Funktionselemente ein Tracelink gerechtfertigt, da sich die Forderung nach einer Kettenschaltung nicht an die lösungsneutrale Funktion „F1.1 Vortrieb gewährleisten“, sondern an „F1.4 variable Übersetzung gewährleisten“ richtet. Dass

³⁵ Die Tabellen der anderen Artefaktkombinationen sind Anhang B zu entnehmen.

trotzdem durch ca. 50% der Probanden eine Abhängigkeit identifiziert wurde, liegt wahrscheinlich daran, dass nicht die lösungsneutrale Funktion bewertet wurde, sondern bereits eine Vorfixierung auf die späteren Lösungselemente Kettenblätter und Zahnkranzkassette stattgefunden hat, die beide neben der Realisierung des Vortriebs ebenfalls an der durch die Anforderung spezifizierten Funktion beteiligt sind.

	A1.1 Vortrieb	A1.1.1 Hinterradantrieb mittels Fußku...	A1.1.2 Übersetzung mittels Kettenscha...	A1.2 Bremsen	A1.2.1 Verschmutzungsrisiko minimieren	A1.2.2 zuverlässiges Bremsen bei Nässe	A1.2.3 muss mit Handkraft betrieben w...	A1.2.4 geringer Verschleiß der Kompon...	A1.3 sicheres Lenken / Manövrieren	A1.4 geländetaugliche Federung und Üb...	A1.4.1 Unebenheiten im Gelände und im...	A1.4.2 Verschiedene Übersetzungen	A1.5 Beleuchtung vorsehen
F1.1 Vortrieb gewährleisten	19	18	15	4			1	2	6	5	2	2	2
F1.1.1 Fußkraft aufnehmen	19	18	8	1					3	3	1	1	
F1.1.2 Antriebskraft leiten	19	18	11						3	4	1	1	
F1.1.3 Kraft / Moment wandeln	19	17	13	3				1	3	5	1	2	1
F1.1.4 Rotationsmoment auf Straße übe...	19	15	8	4				1	6	5	1	1	
F1.2 Verzögerung realisieren	5	1	1	19	4	17	18	11	15	4	3	1	
F1.2.1 Handkraft aufnehmen	3	1	1	18		9	19	4	11	2	1		
F1.2.2 Kraft leiten	3			19	2	14	10	6	10	4	2	1	
F1.2.3 Bremskraft auf Hinterrad / Str...	3			19	4	17	12	10	12	4	3		
F1.3 Fahrtrichtung ändern	3	2	1	4	1	3	4	1	19	6	4	3	
F1.3.1 Steuersignal aufnehmen	2	1		1					19	2	2	1	
F1.3.2 Richtungsänderung ausführen	3	1		1		1	1		19	5	3	3	
F1.4 variable Übersetzung gewährleisten	16	7	16	2			1	1	7	16	5	15	
F1.4.1 Fingerkraft aufnehmen	5		4	2			1		6	9	4	6	
F1.4.2 Kraft leiten	12	3	12	2			1		5	12	5	10	
F1.4.3 Übersetzung wechseln	14	5	14	1				1	7	16	4	15	
F1.5 Sichtbarkeit gewährleisten	3			2	1	1			8	1			19
F1.5.1 unmittelbare Umgebung beleuchten	2			2	1	1			8	1			18
F1.5.2 Licht reflektieren	1			1					4	1			19
F1.6 Federung / Dämpfung	4	1		7	2	6	2	6	12	19	18	5	

Tabelle 19: Aggregierte Anzahl modellierter Tracelinks zwischen Anforderungen und Funktionen (fette Zahlen markieren eine Zelle in der ein Tracelink durch den Autor modelliert wurde)

„F1.2 Verzögerung realisieren“ | „A1.2 Bremsen“: Auch bei dieser Elementkombination sind sich alle Probanden zunächst auf höchster Ebene einig und sehen eine Abhängigkeit zwischen der Forderung nach einer Bremse und der Funktion das Fahrrad zu verzögern. Auch in Bezug auf die Teilfunktionen in Kombination mit der Anforderung „A1.2 Bremsen“ stimmen die Bewertungen der Probanden überein. Die Elementkombination der Funktion „F1.2 Verzögerung realisieren“ und der Anforderung

„A1.2.4 geringer Verschleiß der Komponenten“ stellt hingegen einen interessanten Fall dar³⁶. Bei der Anforderung handelt es sich um eine nicht-funktionale Anforderung, deren Forderung sich somit zunächst nicht an die lösungsneutrale Funktion, sondern an die Bauteile in der Produktstruktur richtet. Allerdings ist das Produkt „Fahrrad“ mit seinen Komponenten den Probanden hinreichend bekannt und immer ähnlich aufgebaut, sodass für die Umsetzung der Funktion „F1.2 Verzögerung realisieren“ implizit die Bauteile „Bremsscheibe“ und „Bremsklötze“ vermutet werden. Durch diese gedankliche Verbindung der tatsächlichen Realisierung, an die sich die Anforderung A1.2.4 richtet, und der Funktion, modellierten ca. 50 % der Probanden (wie auch der Autor) einen Tracelink zwischen diesen Elementen.

„F1.4 variable Übersetzung gewährleisten“ | „A1.4 geländetaugliche Federung und Übersetzung“: Bei dieser Kombination zeigt sich, dass die Übereinstimmung der Probanden vergleichsweise hoch ist, solange sprechende Elementnamen, die direkt Assoziationen hervorrufen, bzw. einzelne Worte wiederverwendet werden (in diesem Fall „Übersetzung“). So sind alle Kombinationen der Funktionen „F1.4 variable Übersetzung gewährleisten“ und „F1.4.3 Übersetzung wechseln“ mit den Anforderungen „A1.4 geländetaugliche Federung und Übersetzung“ und „A1.4.2 verschiedene Übersetzungen“ jeweils von 15 bis 16 Probanden als abhängig bewertet worden. Sobald jedoch detaillierte Teilfunktionen betrachtet werden, deren Bezeichner keine eindeutigen Assoziationen bei den Probanden hervorrufen („F1.4.1 Fingerkraft aufnehmen“ und „F1.4.2 Kraft leiten“), ergibt sich ein unentschiedenes Bild bei den Bewertungen.

Trotz dieser subjektiven Einflüsse bei der Tracelink Modellierung, die sehr stark von der Interpretation der Artefakte durch die Probanden abhängt, zeigt sich, dass bei ca. 80 % der Elementkombinationen viele (16-17) bis sehr viele (18-19) Probanden übereinstimmen und somit einen bzw. eben keinen Tracelink modelliert haben (siehe Abbildung 59).

Diese unterschiedlichen Interpretationen wirken sich natürlich auf die zu erreichende Aufwandsersparnis bei der Anwendung von EcoTracing aus, deren Ermittlung das eigentliche Ziel der Studie darstellte. Die Anzahl der Entscheidungen, welche die 19 Probanden treffen mussten, um die drei Artefakte auf Abhängigkeiten zu untersuchen, liegen zwischen minimal 360 und maximal 830 (siehe Tabelle 18). Die erreichbare Aufwandsersparnis liegt zwischen minimal 31,8 % und maximal 70,4 % bei einer Standardabweichung von 9,5 %. Der Mittelwert der Anzahl der Entscheidungen liegt bei ca. 500, was bei einer maximal möglichen Anzahl von Entscheidungen (wie bei

³⁶ Dieser Fall taucht auch bei anderen Elementkombinationen auf, soll aber anhand dieses Beispiels exemplarisch erläutert werden.

der manuellen Vorgehensweise) von 1217 einer Aufwandsersparnis von ca. 60 % entspricht (siehe Abbildung 60).

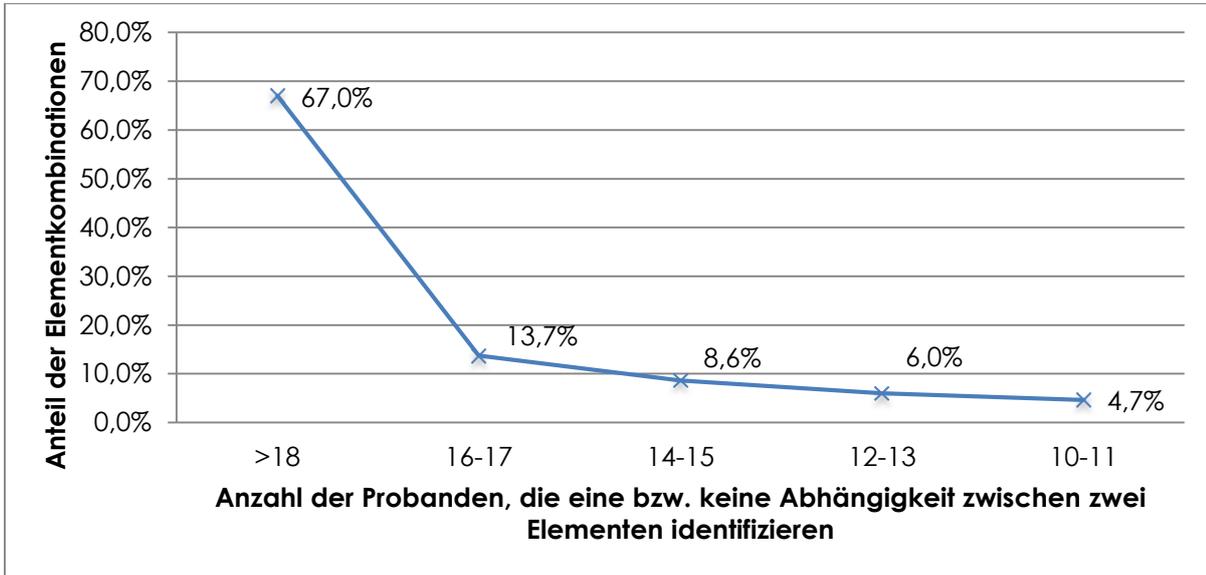


Abbildung 59: Häufigkeit in der unterschiedliche Anzahlen von Probanden übereinstimmen und somit einen bzw. keinen Tracelink modelliert haben

Bei der näheren Betrachtung der modellierten Tracelinks von Proband 02 und Proband 16, die bei der Berechnung der erreichten Ersparnis als Ausreißer auffallen, zeigt sich, dass bei der Bewertung, ob eine Abhängigkeit besteht, ähnliche Schwierigkeiten bei der Interpretation des Konzepts „Tracelink“ auftraten, wie schon in Kapitel 5.3.2 beschrieben. So wurden nicht die eigentliche Funktionserfüllung oder Anforderungsrealisierung zur Bewertung herangezogen, sondern darüber hinaus andere Maßstäbe wie geometrische Nähe oder Kraftfluss verwendet.

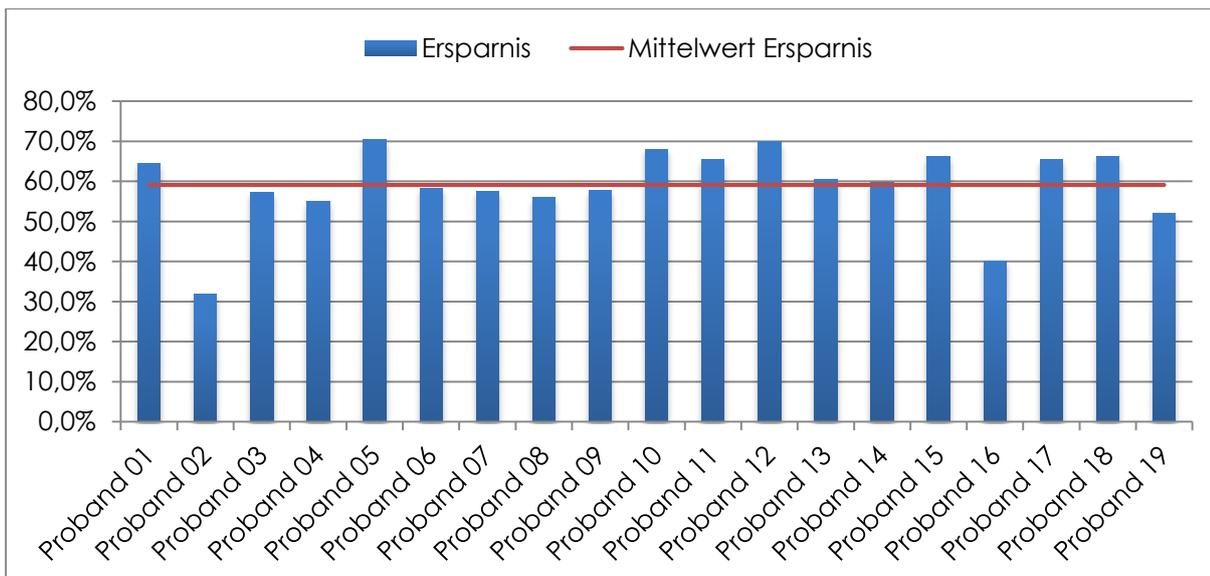


Abbildung 60: Ersparte Entscheidungen durch EcoTracing im Vergleich zur manuellen Methode (in %)

Das führte bei den Probanden dazu, dass bspw. der Rahmen des Fahrrads an der Erfüllung aller Anforderungen beteiligt ist. Gleiches gilt für die Anforderung „A1.1 Vortrieb“, die mit allen Bauteilen und Baugruppen der Produktstruktur verknüpft wurde. Streicht man diese beiden Probanden folglich als Ausreißer ergibt sich eine mittlere Aufwandsersparnis von ca. 62 % und eine Standardabweichung von ca. 5,4 %.

5.5 ZUSAMMENFASSUNG UND DISKUSSION

Das Kapitel 5 behandelt die Forschungsfrage, ob eine methodische Unterstützung zur Modellierung von Tracelinks dabei helfen kann, den dafür notwendigen Aufwand zu reduzieren (siehe Kapitel 4.1). Die vor dem Hintergrund dieser Fragestellung formulierte Hypothese 1 geht davon aus, dass die Eigenschaften von Inklusionshierarchien genutzt werden können, um die Anzahl zu treffender Entscheidungen bei der Modellierung von Tracelinks zu verringern. Um die Forschungsfrage zu beantworten und die Hypothese überprüfen zu können, wurden in Kapitel 5.1 der aktuelle Stand der Technik zusammengefasst und in Kapitel 5.2 notwendigen Grundlagen erörtert. Dabei wurde ein besonderer Fokus auf unterschiedliche Arten von Hierarchien, deren Eigenschaften sowie deren Abbildung auf gängige Artefakte der Systementwicklung gelegt. Ergebnis dieser Analyse war die konkretisierte Hypothese 1*, die die Eigenschaften kompositioneller Hierarchien charakterisiert und damit die Grundlage für die Methode EcoTracing darstellt. Um diese theoretisch hergeleitete Hypothese 1* empirisch zu evaluieren, wurde eine Studie durchgeführt, bei der die Hypothese unter einer Voraussetzung bestätigt werden konnte. Diese Voraussetzung, die aus der ermittelten Korrelation zwischen dem Wissen der Probanden und der Anzahl auftretender Widersprüche, sowie den Ergebnissen der Nachbesprechungen, abgeleitet wurde, ist eine umfassende Kenntnis der zu analysierenden Artefakte hinsichtlich der vorhandenen Elemente und deren Strukturierung durch die Anwender.

Vor diesem Hintergrund wurde das Funktionsprinzip der Methode EcoTracing erläutert und der gewählte Algorithmus zur Abarbeitung von Elementekombinationen bei der Abhängigkeitsanalyse detailliert beschrieben. Dieser wurde in einer prototypischen Implementierung realisiert, die für die anschließende Evaluation der Leistungsfähigkeit der Methode mit Hilfe einer weiteren Studie benötigt wurde. Bei dieser Studie wurde eine erreichbare Aufwandsersparnis von durchschnittlich ca. 60 % ermittelt, die mit der Methode im Vergleich zur manuellen Vorgehensweise erreichbar ist.

Der Vergleich mit dem in Kapitel 5.1 beschriebenen Stand der Technik zeigt, dass EcoTracing bei der möglichen Ersparnis ähnliche oder bessere Werte als die dort beschriebenen Methoden ermöglicht. Tilstra et al. ermitteln die mögliche Ersparnis durch Anwendung ihrer Methode anhand eines Schraubenziehers, den sie selbstständig gruppieren und anschließend die Anzahl zu analysierender Zellen (entspricht

der Anzahl an Entscheidungen) mit der maximal möglichen Anzahl an Entscheidungen ins Verhältnis setzen. Da ihre Methode sehr abhängig von der Gruppierung der Bauteile ist, führen Sie eine Studie durch, in deren Rahmen das gleiche Beispiel mittels eines Zufallsgenerators gruppiert und die möglichen Ersparnisse automatisiert ermitteln werden. Dabei zeigt sich, dass eine Ersparnis von maximal 50 % für das gegebene Beispiel erreichbar ist (anstelle von 1056 müssen nur 508 Zellen analysiert werden) [Tilstra et al. 2009, S. 245].

Auch Biedermann et al. haben eine Fallstudie durchgeführt, um die erreichbare Ersparnis ihrer Methode zu ermitteln. Gegenstand dieser Studie war die Struktur einer Montagezelle, die auf Abhängigkeiten untersucht wurde. Als Ergebnis werden ca. 65 % Zeitersparnis bei der Analyse angegeben, wobei unterschiedliche Dauern für die Diskussion der Elementepaare in den unterschiedlichen Matrizen zugrunde gelegt werden (72 Sekunden pro Entscheidung in der abstrakten DSM, 1,8 Sekunden pro Entscheidung in der DMM und 36 Sekunden pro Entscheidung in der detaillierten DSM) [Biedermann et al. 2010, S. 310]. Da diese Annahmen insbesondere bei der Ermittlung von Abhängigkeiten in der DMM (deren Erstellung im Vergleich zur manuellen Vorgehensweise als zusätzliche Arbeit notwendig ist) als zu optimistisch angesehen werden und zudem eine Vergleichbarkeit auf Basis der Anzahl der Entscheidungen leichter möglich ist, ist es notwendig die Angaben umzurechnen. Damit ergibt sich, dass unter Anwendung der Methode von Biedermann et al. 3519 Entscheidungen getroffen werden müssen, was im Vergleich zur manuellen Vorgehensweise einer Ersparnis von ca. 33 % entspricht.

Als Ergebnis des Kapitels 5 kann die Hypothese 1 bestätigt werden: die Eigenschaften von Inklusionshierarchien können genutzt werden, um die Anzahl zu treffender Entscheidungen bei der Modellierung von Tracelinks zu verringern. Diese Art von Hierarchien ist sehr weit verbreitet in Artefakten, die üblicherweise in der Systementwicklung genutzt werden. Voraussetzung ist jedoch, dass die Anwender der Methode ein umfassendes Wissen über die zu analysierenden Artefakte und deren Aufbau haben. Für die Erfüllung dieser Voraussetzung spricht im industriellen Umfeld, dass sich die Strukturierung der Artefakte an festen Regeln orientiert, die den Entwicklern bekannt sind. Zusätzlich sollte die Methode durch Experten angewendet werden, die umfassende Kenntnisse der Systeme, deren Strukturierung sowie deren Funktionen haben.

Unabhängig von der Methode EcoTracing konnte durch die durchgeführten Studien nachgewiesen werden, dass die Modellierung von Tracelinks sehr stark durch die subjektive Interpretation der inhaltlichen Bedeutung der Elemente und des Konzepts „Tracelink“ durch den Anwender geprägt ist. So konnte festgestellt werden, dass Abhängigkeiten häufiger als solche identifiziert werden, wenn die betroffenen Elemente gleiche oder synonyme Worte in ihren Bezeichnern aufweisen. Darüber hinaus wer-

den nicht immer dieselben Maßstäbe bei der Bewertung von Abhängigkeiten verwendet. Verbreitet vorgekommen ist in den Studien bspw. neben der eigentlichen Funktionserfüllung, Kraftflüsse oder geometrische Nähe als Indikator für oder gegen die Entscheidung, einen Tracelink zu modellieren, zu verwenden. Auch ist der Umgang mit nicht-funktionalen Anforderungen bei der Abhängigkeitsanalyse zwischen Anforderungs- und Funktionsartefakten nicht einfach und wurde durch die Probanden unterschiedlich interpretiert.

Dieser Aspekt der Modellierung von Tracelinks wurde bislang nicht berücksichtigt und birgt große Forschungspotenziale. Ziel muss es in diesem Zusammenhang sein, die Entwickler als Anwender der Methoden bei dem Treffen der einzelnen Entscheidungen zu unterstützen. So könnte bspw. die Strukturierung von Anforderungen nach funktionalen Aspekten bei der Abhängigkeitsanalyse zwischen Anforderungs- und Funktionsartefakten eine große Hilfe darstellen. Ebenfalls vielversprechend könnte hinsichtlich einer optimierten Abhängigkeitsanalyse die detailliertere Untersuchung unterschiedlicher Formen der Elementbeschreibung sein, um bei den Anwendern eine klarere Vorstellung von deren Inhalt hervorzurufen und somit die eindeutigere Ableitung von Tracelinks zu ermöglichen.

Trotzdem stellt die Methode EcoTracing einen deutlichen Fortschritt zum Stand der Technik dar. In Bezugnahme auf die in Kapitel 4 beschriebenen Herausforderung 1 hilft sie dem Entwickler aufgrund der Umsetzung als Wizard in Kombination mit der strukturierten Abarbeitung aller relevanten Elementkombinationen den Überblick bei der Erfassung von Tracelinks zu behalten. Damit gelingt es eine Traceability-Analyse durchzuführen, deren Vollständigkeit im Vergleich zu sonst üblichen Online-Erfassungsmethoden garantiert werden kann. Darüber hinaus wurde dargestellt, dass die Anzahl zu treffender Entscheidungen durch Anwendung von EcoTracing im Mittel um ca. 60 % reduziert werden kann. EcoTracing adressiert damit Herausforderung 2, indem der signifikante Aufwand für die manuelle Modellierung von Tracelinks verringert wird.

6 METHODEN ZUR PFLEGE VON TRACELINK-MODELLEN

Wie in Kapitel 3.4.3 dargestellt, ist es notwendig, Abhängigkeitsnetze kontinuierlich zu pflegen, um deren Qualität hoch, also die Zahl enthaltener Fehler niedrig, zu halten. Dies ist insbesondere vor dem Hintergrund der späteren Nutzung der Abhängigkeitsnetze für unterschiedliche Anwendungen wie z. B. Auswirkungsanalysen essentiell, da diese andernfalls nicht korrekt durchgeführt werden können.

Die Fehler im Tracelink-Modell können auf unterschiedliche Arten entstehen. Zum einen können während der manuellen Erfassung der Tracelinks Fehler gemacht werden. Dabei ist zu berücksichtigen, dass es sich bei der Entscheidung, einen Tracelink zu modellieren, um eine sehr individuelle Einschätzung handelt, die auf der subjektiven Interpretation der betrachteten Elemente beruht und damit ein „korrektes“ Tracelink-Modell nicht existiert (siehe Diskussion in Kapitel 5.5). Zum anderen führen auch die dynamischen Weiterentwicklungen und Änderungen der Artefakte dazu, dass Abhängigkeiten zwischen Elementen hinzukommen oder wegfallen [Egyed und Grünbacher 2002, S. 169].

Auch wenn die Fehler im Traceability-Modell unterschiedliche Ursachen haben und unter Umständen gar nicht eindeutig als Fehler identifiziert werden können, sondern eher auf unterschiedlichen Interpretationen der Elemente und Artefakte beruhen, sind Methoden notwendig, um Hinweise auf wahrscheinliche Fehler zu geben und somit die Möglichkeit für deren Beseitigung zu geben.

Wie bereits in Kapitel 3.4.1 erwähnt, werden auch Methoden, die sich mit der (teil-)automatisierten Identifikation von Abhängigkeiten und der anschließenden Modellierung von Tracelinks befassen, im Rahmen dieser Arbeit der Pflege von Tracelink-Modellen zugeordnet, da eine Vollständigkeit nicht sinnvoll bewertet werden kann.

Diese und andere Methoden werden in Kapitel 6.1 als Stand der Technik beschrieben, kategorisiert und auf dieser Basis Defizite (und damit potenzielle Forschungsrichtungen) aufgedeckt. Im Folgenden werden die beiden Methoden *TraceEvaluation* (siehe Kapitel 6.3) und *TraceLegacy* (siehe Kapitel 6.4) vorgestellt, die diese Defizite adressieren und über den aktuellen Stand der Technik hinausgehen. Die für die Entwicklung der Methoden sowie deren Evaluation notwendigen Grundlagen werden zuvor in Kapitel 6.2 beschrieben. Die abschließende Diskussion der Methoden erfolgt in Kapitel 6.5.

6.1 STAND DER TECHNIK UND FORSCHUNG

Die Methoden zur (teil-)automatisierten Pflege von Tracelink-Modellen basieren darauf, dass unterschiedliche Informationsquellen hinsichtlich bestimmter Kriterien analysiert werden. Anhand der dazu verwendeten Analysealgorithmen lassen sich die Methoden zur Pflege von Tracelink-Modellen kategorisieren. Dabei werden zwei grundsätzliche Kategorien unterschieden: Methoden, die auf Regeln und Methoden, die auf linguistischen Analysen bzw. Informationsrückgewinnung³⁷ (IR) basieren [Winkler und Pilgrim 2010, S. 550]. Erstere nutzen Regeln, die definieren, wann eine Abhängigkeit zwischen zwei Elementen wahrscheinlich bzw. unwahrscheinlich ist. Häufig werden die Strukturen der Artefakte und/oder Tracelink-Modelle analysiert und so potenzielle Abhängigkeiten identifiziert. Methoden der zweiten Kategorie wenden hingegen bspw. linguistische Untersuchungen der textuellen Beschreibungen der Elemente an und erkennen so, bspw. durch wiederkehrende Worte in unterschiedlichen Elementen, Ähnlichkeiten.

Das Ergebnis der Analyse sind in jedem Fall Empfehlungen für fehlende Tracelinks zwischen Elementen, bei denen eine Abhängigkeit vorliegt (*falsch negative Tracelinks*) bzw. für fälschlicherweise modellierte Tracelinks zwischen Elementen, bei denen keine Abhängigkeit vorhanden ist (*falsch positive Tracelinks*) (vgl. auch Kapitel 3.4.3). Dabei stellen alle hier vorgestellten Algorithmen Ansätze zur Unterstützung der Nutzer dar, denen letztlich die Entscheidung über das Vorhandensein einer Abhängigkeit obliegt [Hayes et al. 2006, S. 5-6]. Eine vollautomatisierte Pflege ist bislang noch nicht möglich, da die Algorithmen nicht alle falsch positiven bzw. falsch negativen Trace-

³⁷ Informationsrückgewinnung ist im Englischen unter dem Begriff „information retrieval“ bekannt.

links identifizieren können [Lucia et al. 2007, S. 44] und sich die Abhängigkeiten darüber hinaus häufig nicht nur aus den einzelnen Elementen, sondern zusätzlich deren Kontext ergeben, welcher noch nicht zuverlässig automatisiert erfasst werden kann. Das Ziel der Methoden zur Pflege von Tracelink-Modellen ist somit nicht, den Entwickler zu ersetzen, sondern ihm vielmehr eine Unterstützung bei den zahlreichen Entscheidungsprozessen während der Pflege zu bieten bzw. ihn durch gezielte Vorschläge zu unterstützen.

Ferner ist anzumerken, dass es bislang noch nicht „die Methode“ zur Pflege von Tracelink-Modellen gibt, die für alle Anwendungsbereiche geeignet ist. Es wird vielmehr in absehbarer Zukunft dabei bleiben, dass mehrere Methoden Teile eines gemeinsamen Puzzles darstellen, welches in Kombination eine möglichst gute Qualität des Tracelink-Modells sicherstellt [Aizenbud-Reshef et al. 2006, S. 523].

In den Kapiteln 6.1.1 und 6.1.2 werden, strukturiert nach den zuvor genannten Kategorien, einige Methoden vorgestellt, um einen grundsätzlichen Überblick über den Stand der Technik zu geben. Anschließend werden sie in Kapitel 6.1.3 anhand ihrer Eigenschaften charakterisiert³⁸, Defizite aufgezeigt und auf dieser Basis mögliche Weiterentwicklungsmöglichkeiten in diskutiert.

6.1.1 REGELBASIERTE METHODEN

Regelbasierte Methoden analysieren die Strukturen der Artefakte und/oder des zugehörigen Tracelink-Modells, mit dem Ziel dessen Qualität zu verbessern [Winkler und Pilgrim 2010, S. 550]. Dabei kommen Regeln zur Anwendung, die häufig auf Erfahrungen basieren und definieren, welche Bedingungen für einen bestimmten Elementtyp erfüllt sein müssen, damit dieser in Bezug auf seine modellierten Tracelinks als fehlerfrei gelten kann. So kann bspw. eine Regel definiert werden, die aussagt, dass jede funktionale Anforderung mindestens einen Tracelink zu einer Funktion in der Funktionsliste bzw. -struktur aufweisen muss (Methode 1). Existiert dieser Tracelink nicht, kann davon ausgegangen werden, dass entweder die Anforderung noch nicht in dem Funktionsartefakt berücksichtigt wurde oder aber, dass ein Fehler im Tracelink-Modell vorliegt. Die Auswertung des Traceability-Modells würde somit solche Anforderungen identifizieren, die keinen Tracelink aufweisen und einer erneuten Überprüfung unterzogen werden sollten [Spanoudakis et al. 2004, S. 112]. Eine Erweiterung dieser Methode wird durch Eben et al. beschrieben. Dabei ist nicht nur eine direkte Verknüpfung zu den Funktionen, sondern darüber hinaus zu den anderen verwen-

³⁸ Um diese Charakterisierung zielgerichtet durchführen zu können werden alle folgend vorgestellten Methoden durchnummeriert.

ten Artefakten bis hin zur Produktstruktur erforderlich, um die Bedingung der Fehlerfreiheit zu erfüllen (Methode 2) [Eben et al. 2010, S. 1061].

Ein weiteres Beispiel für die Definition einer Regel wird von Winkler und Pilgrim genannt, welche bei der Verwendung von transitiven Tracelinks angewendet werden kann [Winkler und Pilgrim 2010, S. 550]. Gesetzt den Fall, dass ein Element des Artefakts A mit einem Element des Artefakts B verknüpft ist, welches wiederum mit einem Element in Artefakt C verknüpft ist, kann automatisch ein Tracelink von dem Element in Artefakt A zu dem in C vorgeschlagen bzw. modelliert werden (Methode 3) [Winkler und Pilgrim 2010, S. 550].

Auch Maurer beschreibt Methoden, die auf der Transitivität der Tracelinks basieren, um im Kontext von Abhängigkeitsmatrizen artefaktinterne Tracelinks aus artefaktübergreifenden abzuleiten (Methode 4). Konkret geht es um die Erstellung von Design Structure Matrices (DSM) auf Basis der Informationen in Domain Mapping Matrices (DMM). Er identifiziert sechs unterschiedliche Regeln, die sich ergeben, da die Richtung der Tracelinks berücksichtigt wird [Maurer 2007, S. 95]. Eine Übersicht der Regeln ist in Abbildung 61 zu sehen. Eine nähere Erläuterung der Regeln anhand übersichtlicher Beispiele ist in [Maurer 2007] zu finden.

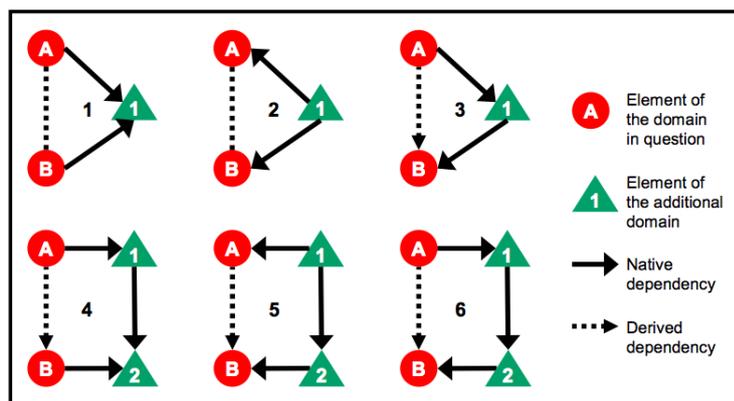


Abbildung 61: Regeln zur Ableitung von Abhängigkeiten nach [Maurer 2007, S. 95]

Egyed und Grünbacher schlagen eine Methode vor, die laut dem angeführten Beispiel im Rahmen der Softwareentwicklung angewendet, jedoch mit Hilfe einiger Anpassungen auch auf andere Einsatzbereiche übertragen werden kann. Analysiert werden dabei die Tracelinks zwischen Anforderungen und Bereichen im Quellcode einer Software. Die Hypothese der Autoren lautet dabei, dass für den Fall, dass eine Anforderung A auf einen Bereich C_A im Code und eine zweite Anforderung B auf einen Bereich C_B im Code verweist, und diese beiden Bereiche eine Überlappung aufweisen (und somit zumindest teilweise identisch sind), eine Abhängigkeit zwischen den Anforderungen A und B möglich ist. Dabei gilt, je größer die Überlappung zwischen den beiden Bereichen, desto größer die Wahrscheinlichkeit, dass eine Abhän-

gigkeit besteht und somit ein Tracelink modelliert werden sollte (Methode 5) [Egyed und Grünbacher 2002, S. 169].

6.1.2 LINGUISTISCHE UND IR-METHODEN

Linguistische Methoden analysieren im Gegensatz zu den regelbasierten Methoden, nicht die Strukturen der Artefakte bzw. des Tracelink-Modells, sondern die textuellen Bezeichner der Elemente der Artefakte. Während die linguistischen Methoden über vergleichsweise einfache Vergleichsalgorithmen verfügen, sind Methoden der Informationsrückgewinnung in der Lage unstrukturierte Fließtexte zu analysieren und Schlüsselbegriffe zu extrahieren [Lucia et al. 2007, S. 11-12; Manning et al. 2009, S. 1]. Sie können damit bspw. auch zur Analyse textueller Anforderungsspezifikationen, und nicht nur deren Anforderungsbezeichner, zum Einsatz kommen [Winkler und Pilgrim 2010, S. 551]. Im Kontext der durchgängigen Nachverfolgbarkeit können die so extrahierten Schlüsselwörter eines Artefakts als Suchanfrage an ein anderes Artefakt dienen und somit die Ähnlichkeit zwischen ihnen berechnet werden. Ist diese zwischen spezifischen Bereichen oder Elementen hoch, wird die Empfehlung ausgegeben, einen Tracelink zu modellieren [Lucia et al. 2007, S. 11-12].

Eine vergleichsweise einfache linguistische Methode wurde von Kaindl et al. vorgestellt (Methode 6). Dabei werden Texte mittels Stringabgleich nach gemeinsamen Wörtern durchsucht und bei Übereinstimmung Tracelinks modelliert [Kaindl et al. 1999]. Im Kontext der Systementwicklung und den zuvor in Kapitel 2.2 vorgestellten Artefakten würde dies bedeuten, dass bspw. die Anforderungen und Funktionsbeschreibungen analysiert und bei übereinstimmenden Worten ein Tracelink vorgeschlagen werden würde.

Eine Erweiterung dieses Ansatzes stellt die von Spanoudakis et al. beschriebene Methode dar (Methode 7) [Spanoudakis et al. 2004]. Sie basiert auf der Verwendung von sog. RTOM-Regeln (Requirements-to-object-model), um Tracelinks zwischen Anforderungsspezifikationen und weiteren Artefakten zu modellieren. Im Unterschied zu der Methode von Kaindl et al. werden die Texte jedoch nicht nur hinsichtlich einzelner Wörter analysiert, sondern mit Hilfe des sog. CLAWS-Algorithmus auch deren grammatikalische Funktion in der Satzstruktur identifiziert [Lancaster University - Centre for Computer Corpus Research on Language 2013]. Die Texte werden zwar nicht hinsichtlich ihrer Semantik untersucht, allerdings lassen sich aussagekräftige Bereiche in Sätzen anhand der grammatikalischen Funktionen der zugehörigen Wörter identifizieren. Anschließend wird der Stringabgleich auf diese relevanten Bereiche eingegrenzt, um die Relevanz der Tracelinkvorschläge sicherzustellen [Spanoudakis et al. 2004, S. 106].

Cerbah und Euzenat verwenden hingegen eine Art Glossar, in dem technische Begriffe dokumentiert sind, um den Stringabgleich auf relevante Worte zu beschränken (Methode 8). Mittels dieser technischen Begriffe können formale und informale Artefakte durchsucht und, bei Existenz in beiden Artefakten, ein Tracelink vorgeschlagen werden (siehe Abbildung 62) [Cerbah und Euzenat 2001].

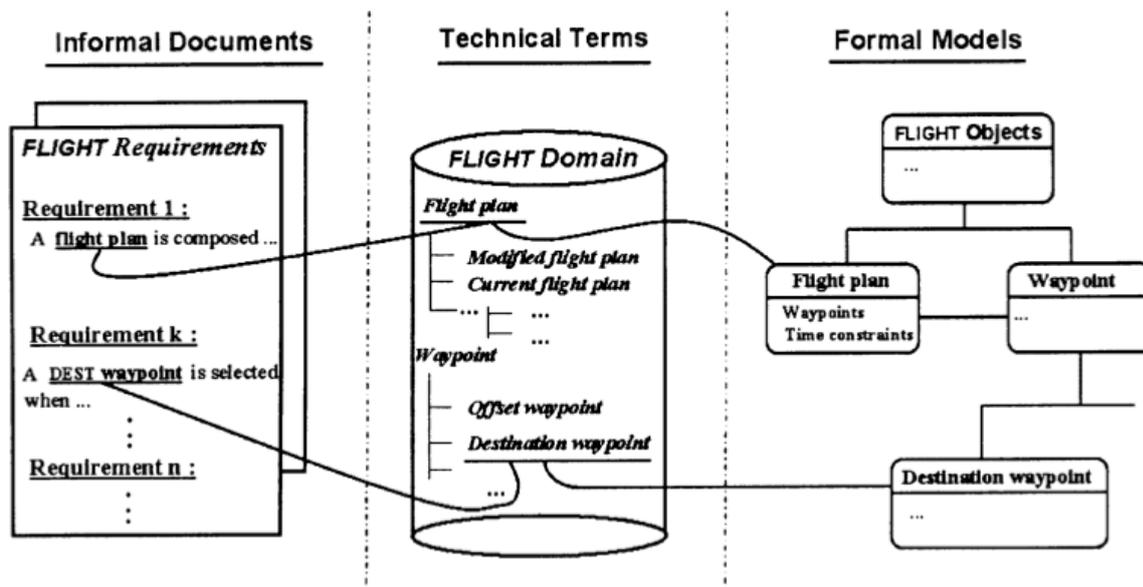


Abbildung 62: Verwendung eines Glossars, um wichtige Begriffe in textuellen Beschreibungen identifizieren zu können [Cerbah und Euzenat 2001, S. 33]

Eine von Antoniol et al. beschriebene Methode basiert auf Informationsrückgewinnung um Abhängigkeiten zwischen Fließtext, wie er bspw. in Anforderungen vorkommt, und Quellcode zu identifizieren (Methode 9). Dabei werden die Fließtexte vorbereitet (Ersetzen von Groß- durch Kleinbuchstaben, Entfernung von Stoppwörtern und Konvertierung von Plural- in Singular-Formen) und indiziert. Gleiches wird auch für die Bezeichner der Klassen durchgeführt. Die Berechnung der Ähnlichkeit zwischen Suchanfrage und den indizierten Dokumenten erfolgt dann mit Hilfe eines Vektorraum-IR-Modells, bei dem der Vektorraum durch die in den Bezeichnern und Dokumenten enthaltenen Worte aufgespannt wird. Das Ergebnis ist für jede Klasse eine Liste anhand ihrer Ähnlichkeit sortierter Dokumente, zu denen ein Tracelink potenziell sinnvoll wäre [Antoniol et al. 2000].

Auch Hayes et al. verwenden unterschiedliche IR-Algorithmen, um Vorschläge für potenzielle Abhängigkeiten zu generieren (Methode 10). Ihr Fokus liegt dabei auf dem Vorschlagen von Tracelinks zwischen Anforderungsspezifikationen unterschiedlichen Detaillierungslevels. Zu diesem Zweck haben sie das Werkzeug RETRO (REquirements TRacing On-target) entwickelt, mit dessen Hilfe die Anforderungen indiziert und mit unterschiedlichen IR-Algorithmen auf Ähnlichkeiten zueinander analysiert werden. Die Grundlage bildet das Vektorraum-IR-Modell, welches auch von Antoniol

et al. verwendet wird (siehe Methode 9). In RETRO wird es um einen Thesaurus sowie das sog. Latent Semantic Indexing ergänzt. Letzteres erlaubt Analysen, die über einen Wortvergleich hinausgehen und die Ähnlichkeit zweier Vektoren mithilfe des Kosinus ermitteln [Hayes et al. 2006].

6.1.3 ZUSAMMENFASSUNG UND DISKUSSION

In den vorangegangenen Kapiteln 6.1.1 und 6.1.2 wurden insgesamt 10 Methoden vorgestellt, mit denen Artefakte und Tracelink-Modelle analysiert werden können, um Vorschläge für Tracelinks zu generieren und somit zur Qualitätssicherung des Tracelink-Modells beizutragen. In der folgenden Tabelle 20 werden diese Methoden anhand ihrer Eigenschaften unterschiedlichen Kategorien zugeordnet. Mit der ersten Kategorie wird charakterisiert, welche Art von Fehlern im Tracelink-Modell mit der jeweiligen Methode identifiziert werden kann (falsch positive und/oder falsch negative Tracelinks). Die zweite Kategorie adressiert die Art der Informationsquelle, die durch die Algorithmen analysiert wird, um Abhängigkeiten zu identifizieren (das Artefakt inkl. seiner Elemente, die bereits modellierten Tracelinks im Tracelink-Modell oder, wenn beides analysiert wird, das Traceability-Modell). In der dritten Kategorie wird bewertet, ob als Informationsquelle nur aktuelle Artefakte und Tracelink-Modelle herangezogen oder, ob (zusätzlich) Informationen vergangener Projekte zur Ermittlung von Vorschlägen für Tracelinks verwendet werden (entstammt die Informationsquelle dem aktuellen oder einem vorherigen Projekt).

Methode	Identifikation von falsch ...		Art der analysierten Informationsquelle		Aktualität der analysierten Informationsquelle	
	positiven Tracelinks	negativen Tracelinks	Artefakt	Tracelink-Modell	aktuelles Projekt	vorheriges Projekt
1		x	x	x	x	
2		x	x	x	x	
3		x		x	x	
4		x		x	x	
5		x	x	x	x	
6		x	x		x	
7		x	x		x	
8		x	x		x	
9		x	x		x	
10		x	x		x	

Tabelle 20: Kategorisierung der Methoden des Stands der Technik

Bei der Betrachtung der Tabelle 20 wird deutlich, dass alle vorgestellten regelbasierten Methoden (1-5) die Existenz eines Tracelink-Modells voraussetzen, da die Regeln

die Struktur der Elemente im Traceability-Modell (bestehend aus Artefakten und Tracelink-Modell) überwachen. Es werden somit bei der regelbasierten Analyse immer die Tracelinks und bei den Methoden 1, 2 und 5 zusätzlich die Elemente der Artefakte sowie deren hierarchischen Verknüpfungen berücksichtigt. Bei den linguistischen bzw. IR-Methoden (6-10) werden hingegen immer die textuellen Bezeichner bzw. Beschreibungen der Elemente analysiert, weshalb die für die Analyse notwendigen Informationen in allen Fällen den Artefakten entnommen werden und das Tracelink-Modell nicht berücksichtigt wird.

Was übergreifend für alle vorgestellten Methoden gilt ist, dass sie sich maßgeblich für die Identifikation von falsch negativen Tracelinks eignen. Tracelinks, die modelliert wurden, obwohl keine Abhängigkeit zwischen Elementen besteht, werden durch die Methoden in ihrer beschriebenen Umsetzung nicht erkannt. Auch wenn diese Fehler, wie in Kapitel 3.4.3 beschrieben, leichter zu identifizieren sind, ist eine Unterstützung bei deren Identifikation im Kontext der Qualitätssicherung des Tracelink-Modells notwendig. Im Folgenden Kapitel 6.2 werden daher die Grundlagen des Konzepts *Crowdsourcing*³⁹ erörtert, da es sich potenziell gut für die Unterstützung eben dieses Prozesses eignet.

Weiter fällt auf, dass alle Analyse-Methoden ihre Informationen lediglich aus aktuellen Projekten beziehen. Das hat bei den linguistischen und IR-Methoden den Nachteil, dass die Elemente der analysierten Artefakte überlappende Inhalte in Form von Wörtern, Synonymen oder Beschreibungen aufweisen müssen, da Abhängigkeiten sonst nicht erkannt werden können [Winkler und Pilgrim 2010, S. 552]. Dies ist, insbesondere wenn unterschiedliche Arten von Artefakten hinsichtlich ihrer Abhängigkeiten zueinander analysiert werden, nicht immer gegeben. So kann sich bspw. das verwendete Vokabular, welches bei der Anforderungsspezifikation verwendet wird, von dem bei der Benennung der Bauteile und Baugruppen in der Produktstruktur unterscheiden und somit die Analyse-Methoden keine Abhängigkeit feststellen⁴⁰. Für diesen Fall kann es sinnvoll sein, auf Artefakte vergangener Projekte und damit auf darin gespeichertes Wissen zuzugreifen.

An diesen beiden vom Stand der Technik noch nicht adressierten Aspekten orientieren sich die im Folgenden beschriebenen Methoden *TraceEvaluation* und *TraceLegacy*, um ein Puzzle-Stück zur Sicherung der Qualität des Tracelink-Modells beizutragen.

³⁹ Deutsch: Schwarmauslagerung

⁴⁰ Eine Ausnahme stellt in diesem Zusammenhang die Methode 10 dar, da ein Thesaurus verwendet wird und somit auch verwandte Begriffe erkannt werden können.

6.2 GRUNDLAGEN ZUR KONZEPTION UND BEWERTUNG DER METHODEN

In Kapitel 6.2 werden Grundlagen erläutert, die für die Entwicklung und die Bewertung der Methoden TraceEvaluation und TraceLegacy essentiell sind. Dabei ist das Kapitel 6.2.1 die Basis für die Crowdsourcing-Methode TraceEvaluation, bei der das Wissen der Mitarbeiter genutzt wird, um fehlerhafte Tracelinks zu identifizieren. In Kapitel 6.2.2 werden die beiden Messgrößen Trefferquote und Präzision eingeführt, die für die Bewertung der beiden Methoden im Rahmen der durchgeführten Evaluationen herangezogen werden.

6.2.1 CROWDSOURCING

Crowdsourcing ist eine relativ neue Strategie, die vor allen Dingen bei einer Vielzahl von Diensten im Internet zu finden ist [Estelles-Arolas und Gonzalez-Ladron-de-Guevara 2012, S. 189]. Wie häufig bei neu aufkommenden Themenbereichen verbreitet, gibt es auch in diesem Bereich aktuell noch Diskussionen über eine eindeutige Namensgebung [Doan et al. 2011, S. 86] und die beinhalteten Aspekte [Estelles-Arolas und Gonzalez-Ladron-de-Guevara 2012, S. 189]. So werden neben dem Begriff Crowdsourcing auch zahlreiche andere synonym verwendet: *peer production*, *user-powered systems*, *user-generated content*, *collaborative systems*, *cummunity systems*, *social systems*, *social search*, *social media*, *collective intelligence*, *wikinomics*, *crowd wisdom*, *smart mobs*, *mass collaboration* und *human computation* [Doan et al. 2011, S. 86]. Hinsichtlich der genauen Definition stellen insbesondere die sehr unterschiedlichen Ausprägungen des Crowdsourcings die Wissenschaftler vor eine Herausforderung [Estelles-Arolas und Gonzalez-Ladron-de-Guevara 2012, S. 189]. Typische Beispiele für Crowdsourcing unterschiedlicher Ausprägungen sind Wikipedia [Wikimedia Foundation Inc 2013], Linux [Linux.org 2013], Yahoo! Anwers [Yahoo! Inc. 2013], Amazon Mechanical Turk [Amazon.com 2013], bei denen unterschiedlich komplexe Aufgabenstellungen an den Schwarm ausgelagert werden. Eine weit verbreitete Definition, die auch im Rahmen dieser Arbeit angewendet wird, lautet:

„We say that a system is a CS [Crowdsourcing] system, if it enlists a crowd of humans to help solve a problem defined by the system owners [...].“ [Doan et al. 2011, S. 87]

Demnach ist Crowdsourcing ein System, mit dessen Hilfe vom Prozesseigner definierte Aufgaben- oder Problemstellungen durch eine größere Anzahl an Menschen bearbeitet werden. Der Grundsätzliche Aufbau dieses Systems ist in Abbildung 63 (grauer Kasten) abgebildet.

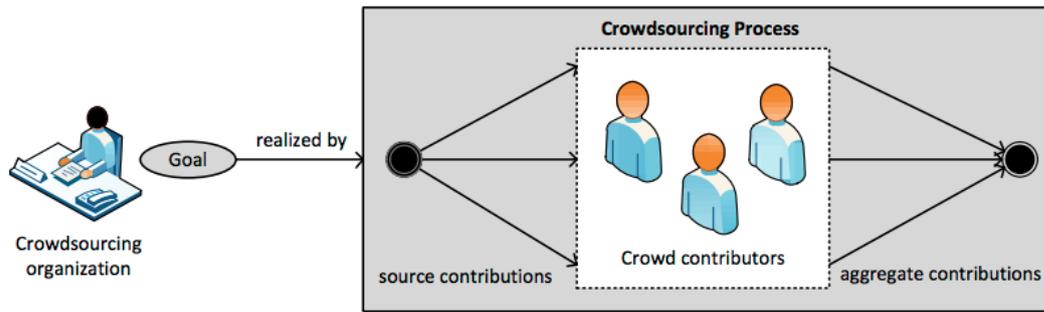


Abbildung 63: Typisches Crowdsourcing System [Geiger et al. 2011, S. 2]

Den Ausgangspunkt stellt dabei eine Organisation oder Person (allg.: Auftraggeber) dar, die eine bestimmte Aufgabenstellung definiert, die durch das Crowdsourcing-System bearbeitet werden soll. Diese Aufgabenstellung wird im Folgenden in Sub-Aufgabenstellungen aufgeteilt, die an die Mitwirkenden (die Crowd) verteilt und von diesen bearbeitet werden. Im Anschluss müssen deren Beiträge, welche Sub-Ergebnisse darstellen, zu einem Gesamtergebnis aggregiert werden, welches wiederum die in der Aufgabenstellung definierte Fragestellung beantwortet [Geiger et al. 2011, S. 1-2; Jones 2013, S. 133].

Unabhängig vom konkreten Crowdsourcing-System stellen sich vier typische Fragestellungen, die laut Jones für eine erfolgreiche Umsetzung beantwortet werden müssen [Jones 2013, S. 133]:

- Wie werden die Mitwirkenden rekrutiert?
- Was können die Mitwirkenden zur Gesamtaufgabenstellung beitragen?
- Wie können die Teilergebnisse aggregiert werden?
- Wie wird mit Missbrauch umgegangen?

Geiger et al. haben diese Fragen aufgegriffen und eine umfassende Literaturrecherche durchgeführt, um die Eigenschaften eines Crowdsourcing-Systems herauszuarbeiten und damit eine Basis für die Kategorisierung existierender, aber auch Entwicklung neuer Crowdsourcing-Systeme bereitzustellen (siehe Abbildung 64) [Geiger et al. 2011, S. 5-6].

Die so identifizierten vier Kategorien *Vorauswahl der Mitwirkenden*, *Zugänglichkeit der einzelnen Teilergebnisse*, *Aggregation der Teilergebnisse* und *Entlohnung für Teilergebnisse* werden im Folgenden näher erläutert.

Vorauswahl der Mitwirkenden. Diese Kategorie adressiert den Pool, aus dem die Mitwirkenden rekrutiert werden. Geiger et al. identifizieren insgesamt vier unterschiedliche Arten der Vorauswahl [Geiger et al. 2011, S. 6]. So kann ein bestimmtes Vorwissen für die Bearbeitung der Aufgabenstellungen notwendig sein. In diesem Fall würden die Mitwirkenden *qualifikationsbasiert* ausgewählt, wobei die Qualifikation bspw.

mit Hilfe eines Tests vor der Bearbeitung überprüft werden kann. Eine Vorauswahl ist hingegen *kontextspezifisch*, wenn sie aufgrund des Kontextes der Mitwirkenden getroffen wird [Corney et al. 2010, S. 298; Rouse 2010, S. 5; Estelles-Arolas und Gonzalez-Ladron-de-Guevara 2012, S. 193-194]. Dies kann bspw. der Fall sein, wenn Geheimhaltungsstufen eingehalten werden müssen und daher nur Mitarbeiter einer speziellen Firma im Crowdsourcing-System mitwirken können. Darüber hinaus kann es vorkommen, dass sowohl die Qualifikation als auch der Kontext zur Auswahl der Mitwirkenden herangezogen wird oder aber keine Vorauswahl getroffen wird [Geiger et al. 2011, S. 6].

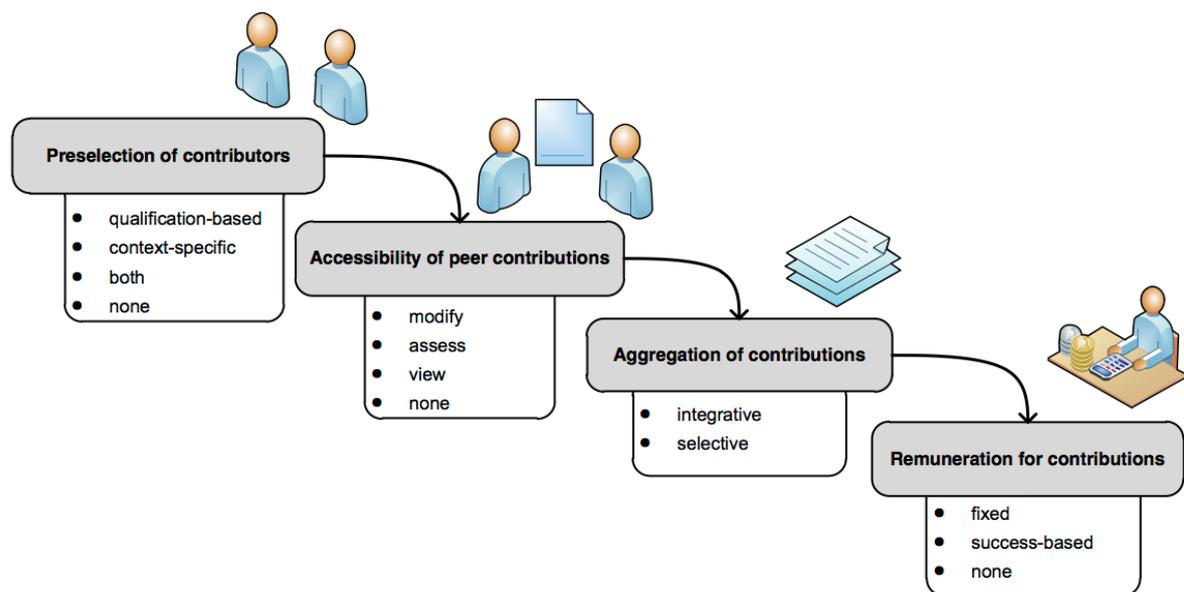


Abbildung 64: Taxonomie zur Beschreibung der Eigenschaften von Crowdsourcing-Systemen [Geiger et al. 2011, S. 6]

Zugänglichkeit der einzelnen Teilergebnisse. Diese Kategorie beschreibt die Form der Zusammenarbeit der einzelnen Mitwirkenden anhand der Zugänglichkeit der Teilergebnisse für andere. *Keine* bedeutet, dass die Mitwirkenden isoliert voneinander die Aufgabenstellungen bearbeiten und die Ergebnisse anderer nicht berücksichtigt werden können. *Sehen* hingegen heißt, dass die Mitwirkenden die Ergebnisse anderer zu sehen bekommen und bei ihrer Bearbeitung der Aufgabenstellung berücksichtigen können. Werden in der Umsetzung des Crowdsourcing-Systems, neben dem Sehen der anderen Beiträge, noch Funktionen zum Kommentieren oder Bewerten dieser bereitgestellt, ordnet man diesem System das Attribut *beurteilen* zu. Die stärkste Form der Interaktion zwischen den Mitwirkenden liegt bei *ändern* vor, da hier Beiträge anderer geändert oder gelöscht werden können [Geiger et al. 2011, S. 7].

Aggregation der Teilergebnisse. Die von den Mitwirkenden erzielten Teilergebnisse müssen hinsichtlich ihrer Korrektheit bewertet, sortiert und zusammengeführt werden, um das anvisierte Gesamtergebnis zu erhalten [Doan et al. 2011, S. 95]. Hier kommt

es stark auf den Beitrag der Mitwirkenden an. Während z. B. bei der Wikipedia die Integration und Kontrolle der Ergebnisse durch die Mitwirkenden selbst durchgeführt werden, sind eher lose Kopplungen, bei denen die Mitwirkenden von den Ergebnissen der anderen wissen, die Regel. In diesen Fällen muss die Aggregation durch den Auftraggeber durchgeführt werden. Grundsätzlich lassen sich aber zwei Ausprägungen unterscheiden. Bei der *integrativen* Aggregation werden alle Teilergebnisse der Mitwirkenden genutzt, solange sie nicht gesetzte Qualitätsstandards verfehlen. Bei einer *selektiven* Aggregation wird dagegen eine Auswahl getroffen und eine Vielzahl an Ergebnissen verworfen [Geiger et al. 2011, S. 7].

Entlohnung für Teilergebnisse. Die Entlohnung für die Erbringung von Teilergebnissen stellt einen wichtigen Aspekt von Crowdsourcing Systemen dar. Dabei werden von Geiger et al. die drei Ausprägungen *fixiert*, *erfolgsbasiert* und *keine* unterschieden. Während bei der ersten Ausprägung die Entlohnung vor Bearbeitung festgelegt und unabhängig vom Ergebnis ausbezahlt wird, spielt die Qualität des Ergebnisses bei der erfolgsbasierten Ausprägung eine entscheidende Rolle [Geiger et al. 2011, S. 7-8]. Wird keine Entlohnung bereitgestellt, müssen entweder Freiwillige gefunden oder aber bspw. Mitarbeiter einer Firma zum Mitwirken verpflichtet werden [Doan et al. 2011, S. 94].

Diese vier Kategorien der Taxonomie nach Geiger et al. werden in Kapitel 6.3.1 wiederaufgenommen und zur Charakterisierung der Methode TraceEvaluation herangezogen. Darüber hinaus wird die Darstellung aus Abbildung 64 genutzt, um das TraceEvaluation-Crowdsourcing-System zu beschreiben.

6.2.2 TREFFERQUOTE UND PRÄZISION

Trefferquote (engl.: Recall) und *Präzision* (engl.: Precision) sind die beiden am weitest verbreiteten Messgrößen, die genutzt werden, um die Effektivität eines IR-Systems zu messen [Manning et al. 2009, S. 5 & 155]. Da Methoden zur Pflege von Tracelink-Modellen entweder auf IR-Methoden basieren oder ähnliche In- und Output-Parameter haben, werden diese Messgrößen auch häufig zur Bewertung von Methoden im Kontext der durchgängigen Nachverfolgbarkeit angewendet.

Die Präzision einer Methode definiert sich über den Anteil generierter relevanter Vorschläge im Verhältnis zur gesamten Anzahl an Vorschlägen. In Bezug auf Traceability entspricht dies der Anzahl korrekt identifizierter Tracelinks bezogen auf die Anzahl der insgesamt gemachten Vorschläge.

$$\text{Präzision} = \frac{\text{Anzahl relevanter Vorschläge}}{\text{Anzahl Vorschläge}} \quad (15)$$

Die Trefferquote einer Methode berechnet sich über den Anteil generierter relevanter gegenüber der Anzahl maximal möglicher relevanter Vorschläge. In Bezug auf Traceability entspricht dies der Anzahl korrekt identifizierter Tracelinks bezogen auf die Anzahl existierender Abhängigkeiten zwischen den Elementen der Artefakte.

$$\text{Trefferquote} = \frac{\text{Anzahl relevanter Vorschläge}}{\text{Anzahl existierender Abhängigkeiten}} \quad (16)$$

Dabei ist zu berücksichtigen, dass sich Präzision und Trefferquote häufig gegenläufig zueinander verhalten. So ist es möglich eine Trefferquote von 100 % zu erreichen indem bei allen bislang Tracelink-freien Elementekombinationen ein Vorschlag generiert wird. In diesem Fall wäre jedoch die Präzision sehr gering, da viele falsch positive Tracelinks vorgeschlagen werden [Manning et al. 2009, S. 156].

Auch Hayes et al. nutzen diese Messgrößen, um ihren Prototyp RETRO (siehe Kapitel 6.1.2) hinsichtlich seiner Effektivität zu bewerten. Zu diesem Zweck geben sie an, welche Werte für die beiden Größen „Präzision“ und „Trefferquote“ „akzeptabel“, „gut“ und „exzellent“ sind [Hayes et al. 2006, S. 11]. Bei diesen Werten handelt es sich um Erfahrungswerte, welche in industriellen Projekten durch die Analyse zahlreicher (automatisch und manuell generierter) Abhängigkeitsmatrizen gewonnen wurden. Während bei einer exzellenten Präzision und Trefferquote kaum noch manuelle Korrekturen notwendig sind, bedeutet ein akzeptabler Wert, dass der Anwender ohne Unterstützung ähnlich effektiv sein würde. Die Autoren betonen dabei, dass es sich bei diesen Werten um erste Abschätzungen handelt und Feedback willkommen sei [Hayes et al. 2006, S. 11]. Die Werte sind in Tabelle 21 zusammengestellt.

	Akzeptabel	Gut	Exzellent
Präzision	20 % - 29 %	30 % - 49 %	50 % - 100 %
Trefferquote	60 % - 69 %	70 % - 79 %	80 % - 100 %

Tabelle 21: Bewertung der Größen Präzision (Precision) und Trefferquote (Recall) [Hayes et al. 2006, S. 11]

Da dem Autor dieser Arbeit keine alternativen Werte aus der Traceability-Literatur bekannt sind oder aus eigenen Projekten zur Verfügung stehen, wird sich auch im Rahmen dieser Arbeit auf diese Werte berufen. Der Vorteil ist, dass eine Vergleichbarkeit mit den Ergebnissen von Hayes et al. erreicht wird, auch wenn die Spannen der Werte (bspw. exzellente Präzision = 50 % - 100 %) nicht verifiziert werden können.

6.3 TRACEEVALUATION – METHODE ZUR VERTEILTEN BEWERTUNG VON TRACELINKS

Die Methode TraceEvaluation dient der Bewertung bestehender Tracelinks, um falsch positive Tracelinks zu identifizieren und somit die Qualität des Tracelink-Modells entwicklungsbegleitend zu verbessern. Sie basiert auf den in Kapitel 6.2.1 beschriebenen Grundlagen des Crowdsourcings und adressiert die Hypothese 2, die postuliert, dass Nutzer bei der Interaktion mit dem Traceability-Modell falsch modellierte Tracelinks bemerken. Damit ist ein ergänzender Einsatz von TraceEvaluation immer möglich, wenn Tracelinks für die Unterstützung der Entwicklungsprozesse verwendet werden (siehe auch Kapitel 3.4.2). Eine erste Veröffentlichung des zugrundeliegenden Konzepts erfolgte in [Stark et al. 2010].

Ein in [Beier et al. 2012] dokumentierter Ansatz stellt den Anwendern bspw. das Abhängigkeitsnetz in Form des Visualisierungswerkzeugs Ariadne's Eye zur Verfügung, um damit unterschiedliche Aufgaben im Kontext der Systementwicklung durchzuführen (siehe Kapitel 3.6.4.3). Durch die Visualisierung der Abhängigkeiten zwischen den Artefakten eines Systems kann so bspw. die Schaffung eines Systemverständnisses unterstützt werden. Darüber hinaus verfügt Ariadne's Eye über einen Assistenten zur Durchführung von Auswirkungsanalysen zur Vorbereitung von Änderungen. Ariadne's Eye soll durch eine Vielzahl unterschiedlicher Entwickler genutzt werden.

Im Kontext dieser Arbeit dient Ariadne's Eye als beispielhafter Ansatz zur Verwendung des Tracelink-Modells auf den die Methode TraceEvaluation aufsetzt. Während der zuvor erwähnten Routineaufgaben kann TraceEvaluation somit nebenbei angewendet werden, um kontinuierlich die Qualität des Abhängigkeitsnetzes zu verbessern. Anwender sind dabei unterschiedliche Experten der an der Entwicklung des mechatronischen Systems beteiligten Disziplinen.

6.3.1 FUNKTIONSPRINZIP DER METHODE TRACEEVALUATION

Die Methode TraceEvaluation zielt auf die entwicklungsbegleitende Verbesserung der Qualität des Tracelink-Modells durch die Nutzer der Tracelinks. Während an der Erfassung der Tracelinks nur eine sehr begrenzte Anzahl an Personen beteiligt ist, kann TraceEvaluation von einer großen Anzahl an Entwicklern verwendet werden. Diese können dabei ihr Expertenwissen für diesen parallelen Qualitätssicherungsprozess ohne großen Aufwand im Sinne eines Crowdsourcing-Ansatzes zur Verfügung stellen, indem sie Verbesserungsvorschläge für die von Ihnen betrachteten Bereiche des Tracelink-Modells generieren.

Konkret sieht die TraceEvaluation vor, im Kontext der unterschiedlichen Methoden zur Verwendung von Tracelinks die technische Möglichkeit zur Bewertung bereits modellierter Tracelinks bereitzustellen. Dies ist in Abbildung 65 für die Nutzer 1 und n exemp-

larisch für eine beliebig große Anzahl von n Nutzern dargestellt (Zeilen „Nutzer 1“ und „Nutzer n“).

Nutzer 1 führt bspw. eine Auswirkungsanalyse durch und verwendet zu diesem Zweck das Traceability-Modell. Ausgehend von einem zu ändernden Element (bspw. einem Bauteil) verfolgt er somit die Tracelinks zu Anforderungen, Funktionen und anderen Bauteilen, um zu bewerten, ob diese von der Änderung betroffen sind. Bei dieser Tätigkeit, bei der die Abhängigkeit von jeweils zwei verknüpften Elementen zur Zeit bewertet wird, erkennt der Nutzer leicht Tracelinks, deren Korrektheit er bezweifelt. Für diesen Zweck wird ihm durch TraceEvaluation eine Funktionalität zum Melden dieses Zweifels in seinem Autorenwerkzeug zur Verfügung gestellt. Bei Meldung eines Tracelinks wird das Feedback zusammen mit dem entsprechenden Tracelink als Attribut gespeichert.

Gleiches gilt für den Nutzer n, der bei der Durchführung einer FMEA die Tracelinks verwendet, um für eine gegebene Funktion die zugehörigen Fehler aus einem Fehlerbaum sowie Entdeckungsmaßnahmen und Vermeidungsmaßnahmen zusammenzustellen und somit einen FMEA-Vordruck zu füllen (vgl. [Beier et al. 2011]). Fälschlicherweise verknüpfte Elemente fallen auf und können mittels integrierter TraceEvaluation-Funktionalität gemeldet werden.

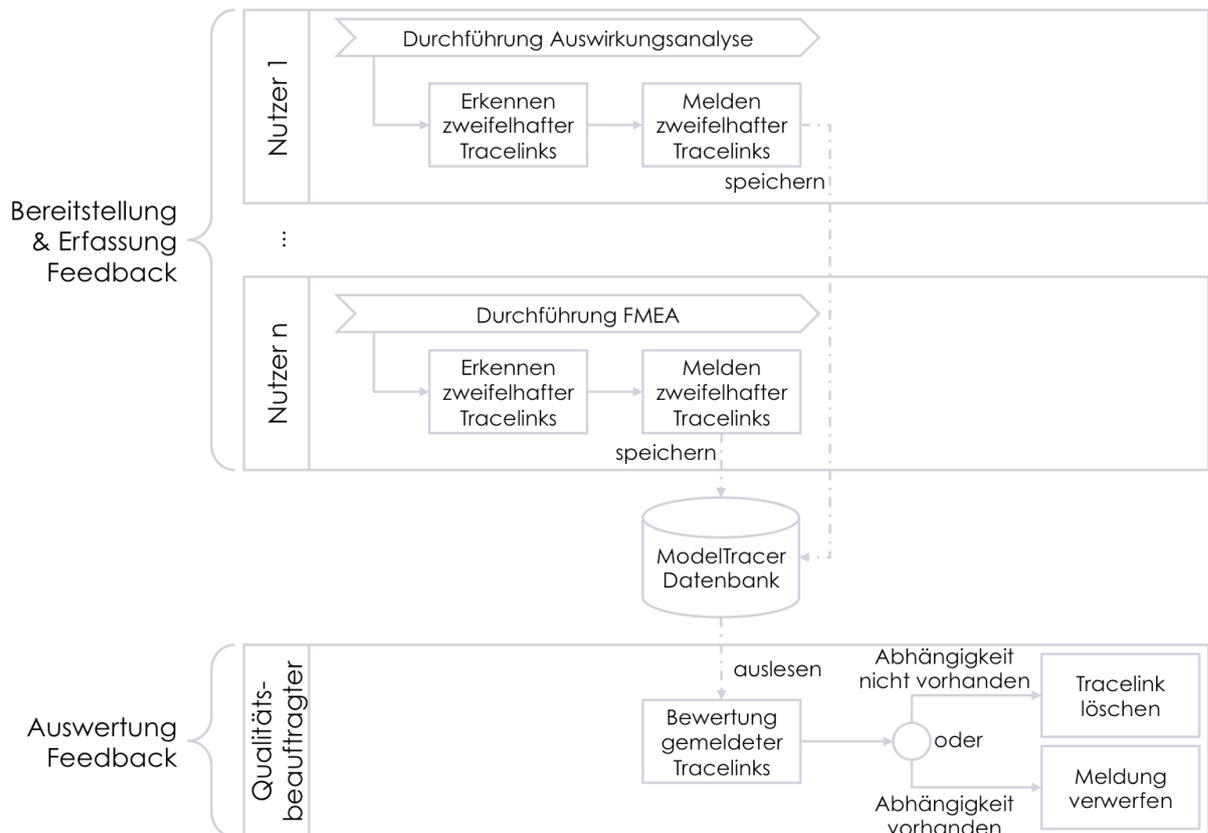


Abbildung 65: Ablauf der Methode TraceEvaluation

Dabei ist diese Bewertung nicht die hauptsächliche Aufgabe der Nutzer, sondern vielmehr können Zweifel über die Korrektheit der Tracelinks, die sich während der Erledigung ihrer eigentlichen Aufgabenstellung ergeben, quasi im Vorbeigehen dokumentiert werden.

Die Auswertung der angezweifelten Tracelinks erfolgt zeitlich unabhängig von deren Erfassung (siehe Abbildung 65, Zeile „Qualitätsbeauftragter“). Zu einem beliebigen Zeitpunkt kann der Qualitätsbeauftragte die Auswertung des Feedbacks durchführen. Dazu werden die in der Datenbank des ModelTracers gespeicherten Attribute ausgelesen und die angezweifelten Tracelinks in aggregierter Form dargestellt. Diese werden nacheinander überprüft und entschieden, ob tatsächlich fälschlicherweise ein Tracelink modelliert wurde. Ggf. ist es dafür notwendig sich mit den jeweiligen Experten, die an der Erstellung der betroffenen Artefakte beteiligt waren, abzustimmen. Wird dabei erkannt, dass keine Abhängigkeit zwischen den verknüpften Elementen besteht, wird der Tracelink gelöscht oder andernfalls, wenn sich der Nutzer in seiner Bewertung geirrt hatte, die Meldung verworfen.

Die Methode TraceEvaluation wurde dabei unter Berücksichtigung der Taxonomie von Geiger et al. entwickelt (siehe Abbildung 64) [Geiger et al. 2011, S. 2] und wird im Folgenden anhand der vier dort aufgegriffenen Kategorien charakterisiert.

Vorauswahl der Mitwirkenden. Die Vorauswahl der Mitwirkenden wird sowohl kontextspezifisch als auch qualifikationsbasiert vorgenommen. Der Kontext ergibt sich daraus, dass nur Mitarbeiter der eigenen Firma bzw. berechnete Zulieferer Zugriff auf die Tracelinks haben. Allerdings wird dieser Pool an Mitarbeitern noch weiter hinsichtlich ihrer Qualifikation eingeschränkt. Diese Einschränkung ergibt sich dadurch, dass nur Mitarbeiter, die sich im Tagesgeschäft mit Artefakten und den zugehörigen Tracelinks befassen, auch zu deren Bewertung herangezogen werden.

Zugänglichkeit der einzelnen Teilergebnisse. Die Zugänglichkeit der Teilergebnisse der Mitarbeiter untereinander ist im vorliegenden Konzept von TraceEvaluation nicht gegeben. Dies ist insofern auch nicht notwendig, da jeweils nur die Einschätzung der einzelnen Personen gefragt ist und diese nicht durch die Einschätzung anderer beeinflusst werden soll.

Aggregation der Teilergebnisse. Bei der Aggregation der Teilergebnisse wird ein integrativer Ansatz gewählt, bei dem alle Inputs der Mitarbeiter berücksichtigt und hinsichtlich ihrer Korrektheit bewertet werden.

Entlohnung für Teilergebnisse. Eine Entlohnung für die Bereitstellung von Feedback bzgl. potenziell falscher Tracelinks ist in der aktuellen Umsetzung des Konzepts nicht vorgesehen, da zunächst dessen Validierung im Vordergrund steht. Da die Entlohnung für die Erbringung von Teilergebnissen jedoch einen wichtigen Aspekt hinsicht-

lich der Motivation bei der Mitwirkung darstellt, ist eine Erweiterung des Konzepts in weiteren Ausbaustufen denkbar. So könnten bspw. Anreize für jeden vorgeschlagenen Tracelink, der sich nach Überprüfung tatsächlich als falsch herausstellt, geschaffen werden. Was diese Anreize genau sind, wäre dann allerdings im jeweiligen Firmenkontext festzulegen.

In Anlehnung an Abbildung 63 wurde in Abbildung 66 die Methode TraceEvaluation als Crowdsourcing-System dargestellt.

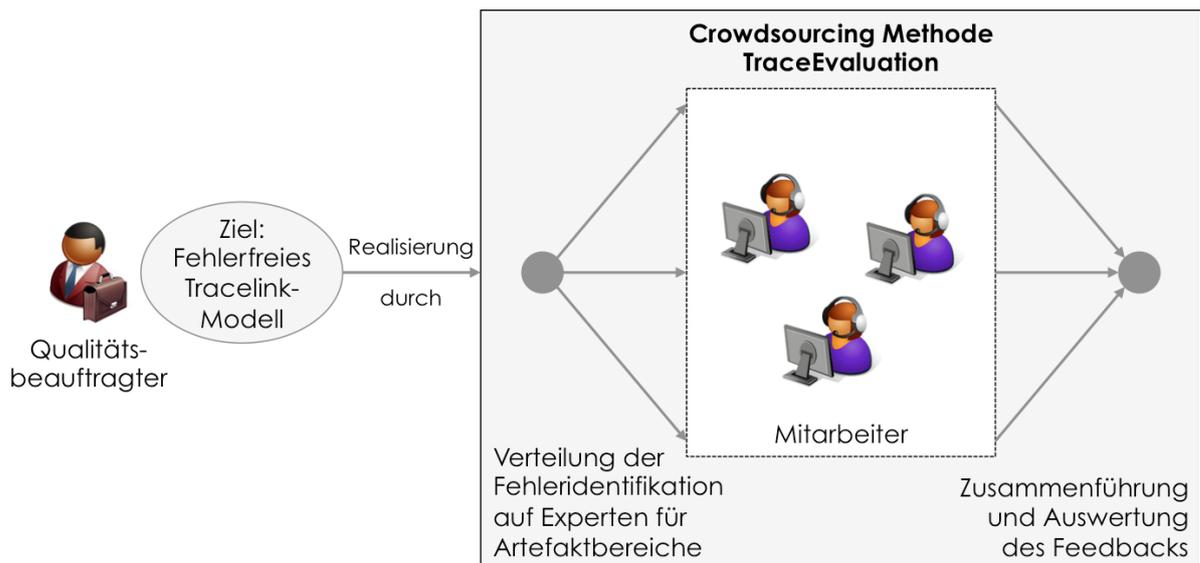


Abbildung 66: TraceEvaluation als Crowdsourcing-System in Anlehnung an [Geiger et al. 2011, S. 2]

Wie zuvor bereits in Abbildung 65 dargestellt besteht das Crowdsourcing-System TraceEvaluation aus zwei Teilen, die in der Implementierung berücksichtigt werden müssen: der Erfassung und der Auswertung des Feedbacks bzgl. der Korrektheit der Tracelinks. Im Folgenden werden diese beiden Teile im Zusammenhang mit deren prototypischer Implementierung erläutert.

6.3.2 PROTOTYPISCHE IMPLEMENTIERUNG DER METHODE TRACEEVALUATION

Um die Wirksamkeit der Methode TraceEvaluation im Rahmen einer Studie und damit den Wahrheitsgehalt der zugehörigen Hypothese 2 überprüfen zu können, wurde TraceEvaluation prototypisch implementiert. Der Prototyp besteht dabei, entsprechend der zweiteiligen Methode, aus zwei Modulen: dem Anwendermodul, mit dem das Feedback der Nutzer erfasst wird, und dem Auswertungsmodul, welches als Plug-In im ModelTracer umgesetzt wurde und der Auswertung des Feedbacks dient.

Das Anwendermodul wurde im Prototyp exemplarisch auf den Traceability-Viewer Ariadne's Eye aufgesetzt. Dessen Konzept sieht vor, die Schnittstelle zwischen Entwickler und Traceability-Modell bereitzustellen. Ariadne's Eye zielt somit auf eine gro-

ße Anwenderzahl, was für die Wirksamkeit der TraceEvaluation einen wichtigen Aspekt darstellt. Insbesondere die Auswirkungsanalysen eignen sich dabei für die Identifikation falscher Tracelinks, da bei der Änderung eines Elements die Tracelinks zu möglicherweise betroffenen Elementen nachverfolgt werden. Somit werden die Tracelinks nicht einfach nur als gegeben hingenommen, sondern deren Korrektheit im Kontext der betrachteten Änderung und damit unterbewusst auch im Allgemeinen bewertet.

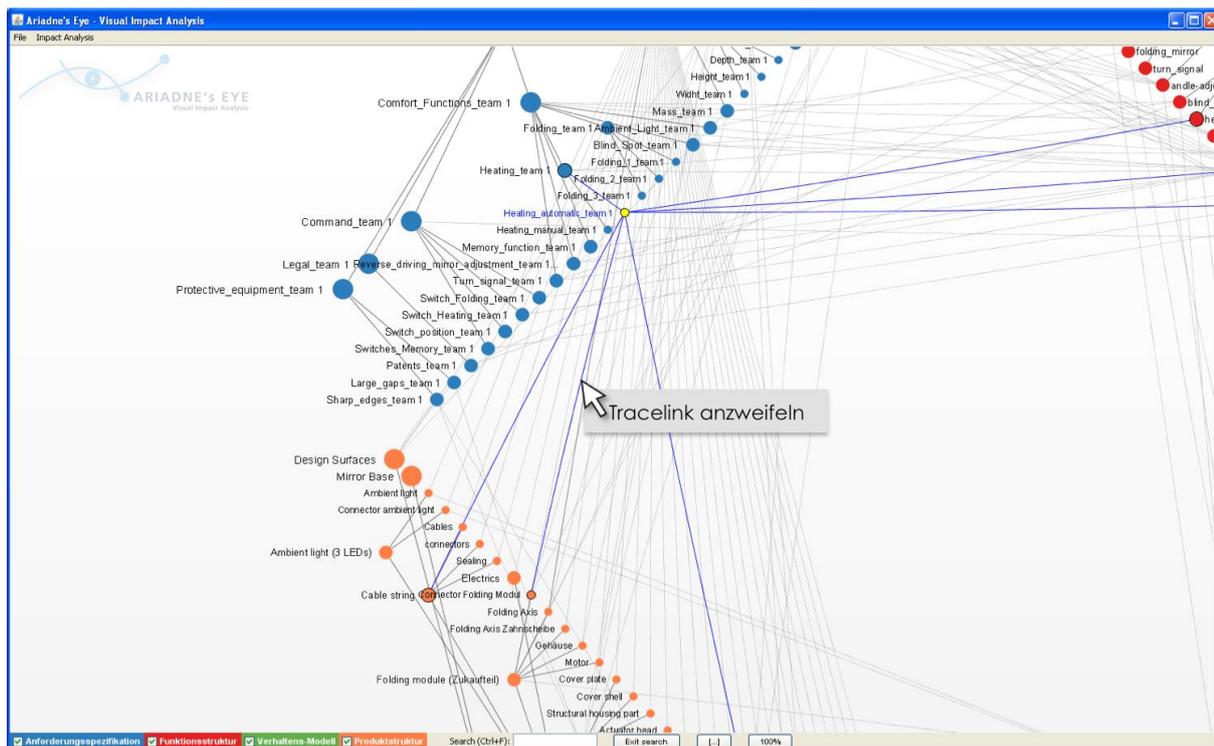


Abbildung 67: Melden eines falschen Tracelinks in Ariadne's Eye

In der prototypischen Implementierung von Ariadne's Eye können die Nutzer Tracelinks, die auf Basis der eigenen Expertise als falsch bewertet werden, auswählen, mit einem Rechtsklick ein Kontextmenü aufrufen und ihren Zweifel an deren Korrektheit melden (siehe Abbildung 67). Dies kann neben der Bearbeitung der eigentlichen Aufgabenstellung (z. B. der Durchführung einer Auswirkungsanalyse), ohne den eigentlichen Arbeitsprozess signifikant zu stören, durchgeführt werden.

Die erfassten Bewertungen werden in der ModelTracer-Datenbank gemeinsam mit den Tracelinks in Form des Attributs „doubtcounter“ gespeichert. Bei dem Attribut handelt es sich um einen Zähler, der initial auf „0“ steht und bei jeder Meldung durch einen Anwender inkrementell erhöht wird.

Der für die durchgängige Nachverfolgbarkeit Qualitätsbeauftragte kann zu jeder Zeit das Auswertungsmodul im ModelTracer aufrufen und eine Qualitätsüberprüfung des

Tracelink-Modells starten, um sich die Ergebnisse der TraceEvaluation anzeigen zu lassen.

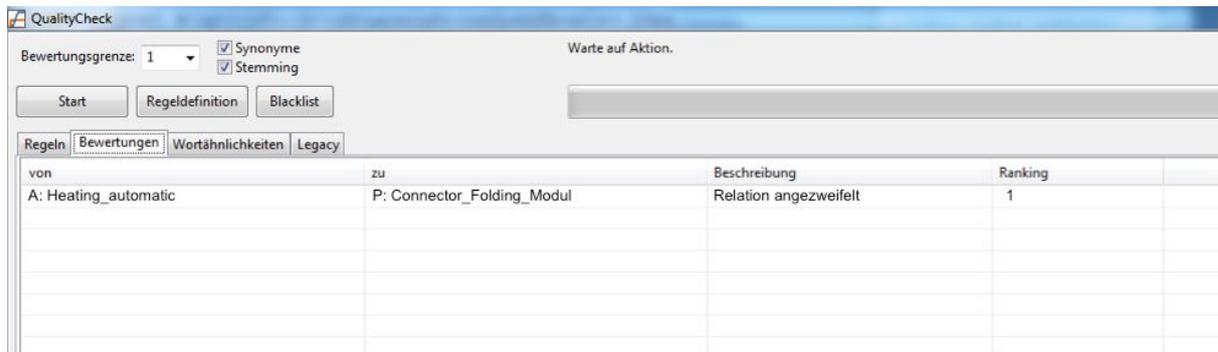


Abbildung 68: Auswertungsmodul zur Analyse der gemeldeten falschen Tracelinks

Dabei besteht die Möglichkeit, einen Schwellwert für das Attribut „doubtcounter“ zu definieren, ab dem angezweifelte Tracelinks in der Zusammenstellung des Feedbacks angezeigt werden. Ziel dieses Schwellwerts ist es, die Signifikanz der Meldung zu erhöhen, indem bspw. mindestens zwei Meldungen eines Tracelinks verlangt werden, bevor dieser auf seine Richtigkeit hin überprüft wird. Da sich der Wert dabei nicht allgemeingültig festlegen lässt, sondern bspw. von der Unternehmensgröße und der Anzahl der Traceability-Anwender abhängt, kann er individuell im Rahmen der Qualitätssicherung angepasst und z. B. schrittweise verringert werden.

Eine denkbare Erweiterung dieses Konzepts wäre es, die Meldungen rollenbezogen auszuwerten und somit zu erfassen, ob die meldende Person für einen bestimmten Artefaktbereich besonderer Kompetenzträger ist. Diese Meldung könnte ein höheres Gewicht erhalten und unabhängig vom festgesetzten Schwellwert immer angezeigt werden. Eine Umsetzung dieser Erweiterung wurde jedoch in der prototypischen Implementierung nicht durchgeführt.

Zur Erstellung der Übersicht gemeldeter Tracelinks werden die Tracelinks nacheinander vom Auswertungsmodul aus der Datenbank ausgelesen und ihr Attribut „doubtcounter“ überprüft. Ist keine negative Bewertung vorhanden und das Attribut „doubtcounter“ somit gleich „0“, wird zur Überprüfung des nächsten Tracelinks übergegangen. Sobald das Attribut ungleich „0“ ist, erfolgt ein Abgleich mit dem im vorigen Absatz beschriebenen Schwellwert. Je nachdem, ob dieser unter- oder überschritten ist, wird entweder zum nächsten ungeprüften Tracelink übergegangen oder aber der Tracelink dem Qualitätsbeauftragten im Auswertungsmodul unter Angabe der Anzahl negativer Bewertungen zur Überprüfung vorgeschlagen (siehe Abbildung 69).

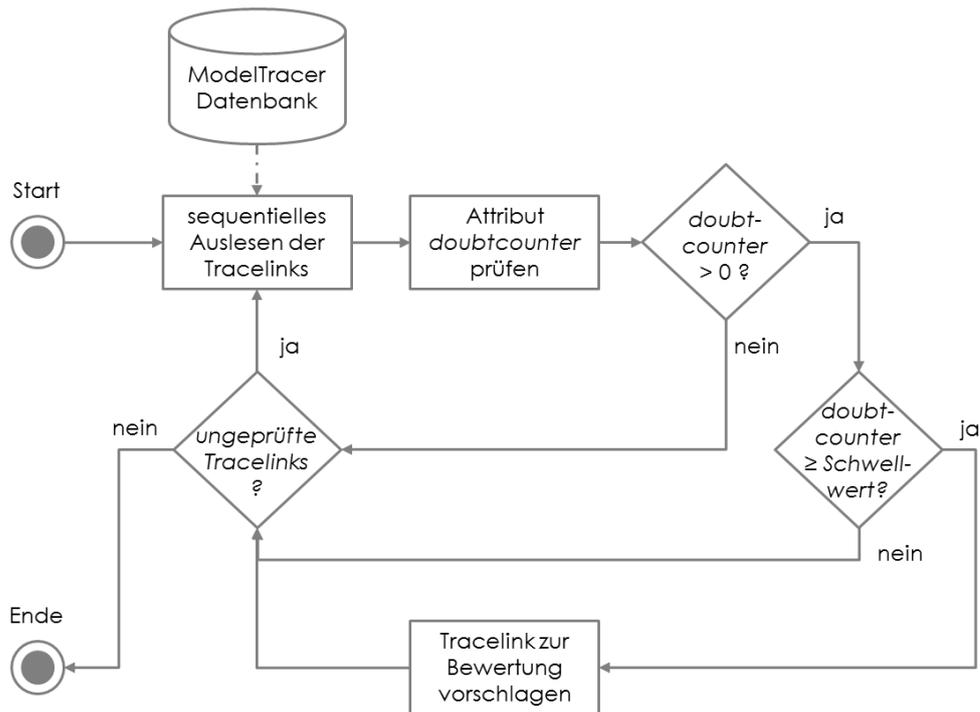


Abbildung 69: Algorithmus zur Zusammenstellung angezweifelter Tracelinks

6.3.3 EVALUATION DER METHODE TRACEEVALUATION

In Hypothese 2 wurde postuliert, dass Nutzer während der Interaktion mit dem Traceability-Modell falsch modellierte Tracelinks erkennen, und dass darüber hinaus die Anzahl der Nutzer einen Einfluss auf die Vollständigkeit der Ergebnisse hat. Ob diese Hypothese durch die Methode TraceEvaluation bestätigt werden kann, wurde mit Hilfe einer Studie evaluiert. Der Aufbau und die Durchführung der Studie werden im folgenden Kapitel 6.3.3.1 beschrieben und die Ergebnisse in Kapitel 6.3.3.2 ausgewertet. Die Diskussion erfolgt dann im Rahmen der Zusammenfassung in Kapitel 6.3.4.

6.3.3.1 AUFBAU UND DURCHFÜHRUNG DER STUDIE

Um die Hypothese 2 überprüfen zu können ist es notwendig, Probanden eine von der Qualitätssicherung losgelöste Aufgabenstellung anhand eines fehlerhaften Tracelink-Modells bearbeiten zu lassen und ihnen dabei die Möglichkeit zur Verfügung zu stellen, falsche Tracelinks zu melden.

Der Aufbau der Studie sah daher vor, dass die Probanden drei Auswirkungsanalysen für sich ändernde Anforderungen bzw. Bauteilen mit Hilfe des Viewers Ariadne's Eye durchführen. Konkret handelte es sich um die folgenden Anforderungen und Bauteile die sich laut Aufgabenbeschreibung geändert hatten und für die potenziell betroffene Elemente identifiziert werden sollten (siehe Anhang B, Aufgabe 3):

1. A: Übersetzung mittels Kettenschaltung,
2. A: Muss mit Handkraft betrieben werden können und
3. A: Hinterradantrieb mittels Fußkurbel und Kette,
4. P: Hydraulische Scheibenbremsen
5. P: Hydraulikleitungen
6. P: Bremshebel
7. P: Bremsscheiben

Zur Bearbeitung der Aufgabenstellungen erhielten die Probanden die Anforderungs-, Funktions- und Produktstrukturartefakte eines Fahrrads sowie ein vollständiges Tracelink-Modell, welches im Vorfeld der Studie während eines Workshops erstellt wurde. Zusätzlich wurden in dieses Tracelink-Modell vorab neun fehlerhafte Tracelinks integriert, die in Tabelle 22 als Kombination ihrer Start- und Zielelemente zusammengestellt sind.

Nr.	Quellelement	Zielelement
1	A1.1.2 Übersetzung mittels Kettenschaltung	P1.6.2 Sattel
2	A1.1.2 Übersetzung mittels Kettenschaltung	P1.7.3 Beleuchtung
3	A1.1.1 Hinterradantrieb mittels Fußkurbel und Kette	P1.2.1 hydraulische Federgabel
4	A1.1.1 Hinterradantrieb mittels Fußkurbel und Kette	P1.4.2 Schaltwerk
5	P1.5.3 Hydraulikleitungen	F1.6 Federung/Dämpfung
6	P1.5.2 Bremshebel	A1.4.1 Unebenheiten im Gelände und Fahrbelag ausgleichen
7	P1.5.2 Bremshebel	F1.5.2 Licht reflektieren
8	P1.5.4 Bremsscheiben	A1.1.1 Hinterradantrieb mittels Fußkurbel und Kette
9	P1.5.4 Bremsscheiben	F1.3.1 Steuersignal aufnehmen

Tabelle 22: Elementepaare, zwischen denen im Vorfeld der Studie falsche Tracelinks modelliert wurden

Die Durchführung der Evaluation wurde in die im Kapitel 5.4.5 beschriebene Studie integriert. Dabei wurde mit einer Einführung in die Interaktionsfunktionalitäten der unterschiedlichen genutzten Prototypen begonnen. Für Ariadne's Eye war dies u. a. die zuvor beschriebene Funktionalität zum Anzweifeln von Tracelinks. In diesem Zusammenhang wurden die Probanden darauf hingewiesen, dass sie wenn möglich zur Qualitätsverbesserung des Tracelink-Modells beitragen sollten. Anschließend wurde zunächst die Modellierung von Tracelinks zwischen den Artefakten mit Hilfe der Methode EcoTracing durchgeführt. Dies hatte den Vorteil, dass eine grundsätzliche Kenntnis der drei Artefakte (Anforderungsmodell mit 10 Anforderungen und 3 sortierende Überschriften, Funktionsliste mit 20 Funktionen und Produktstruktur mit 29 Bauteilen und Baugruppen) vorausgesetzt werden konnte (siehe Abbildung 58). Anschließend sollte die zuvor erwähnte Auswirkungsanalyse durchgeführt werden. Es

wurde hier nur am Rande auf die Funktionalität zur Meldung falscher Tracelinks hingewiesen.

In Tabelle 23 sind die Eigenschaften der Studie in Tabellenform als Steckbrief zusammengefasst.

Analyseeinheit	Meldungen falscher Tracelinks durch die Probanden
Analysemethode	Häufigkeit und Vollständigkeit der Meldung falscher Tracelinks
Methode zur Datenerfassung	Verwendung des Anwendermoduls der prototypischen Implementierung der Methode TraceEvaluation zur Erfassung der Meldungen der Probanden und Analyse im Auswertungsmodul
Aufgabenstellung	Durchführung von Auswirkungsanalysen bei Änderungen vorgegebener Elemente; Meldung falscher Tracelinks nur nebenbei erwähnt; realistische Aufgabenstellung, die an industrielle Aufgabenstellungen angelehnt ist, sich jedoch in Ausprägung der Artefakte hinsichtlich Umfang und Komplexität unterscheidet
Anzahl der Teilnehmer	19
Charakterisierung der Teilnehmer	Studentische Hilfskräfte und wissenschaftliche Mitarbeiter aus dem Bereich Virtuelle Produktentstehung des Fraunhofer IPK und dem Fachgebiet Industrielle Informationstechnik der Technischen Universität Berlin; Grundsätzliche Kenntnisse über Artefakte, deren Entwicklung und Abhängigkeiten zwischen diesen vorhanden
Untersuchungsobjekt	Drei Artefakte eines Fahrrads: <ul style="list-style-type: none"> - Anforderungsmodell mit 10 Anforderungen und 3 sortierenden Überschriften mit zwei Hierarchieebenen - Funktionsliste mit 20 Funktionen auf zwei Hierarchieebenen - Produktstruktur mit 29 Bauteilen und Baugruppen auf zwei Hierarchieebenen Vollständiges Tracelink-Modell, welches durch neun fehlerhafte Tracelinks ergänzt wurde

Tabelle 23: Eigenschaften der Studie zur Überprüfung der Hypothese 2

6.3.3.2 AUSWERTUNG DER STUDIE

Von 19 Probanden nutzten 13 während der Durchführung der Auswirkungsanalysen zusätzlich die Möglichkeit falsche Tracelinks zu melden. Insgesamt wurden 74 mal Tracelinks zur erneuten Prüfung vorgeschlagen wovon 65 Meldungen (88 %) mit den bei der Vorbereitung der Studie hinzugefügten falschen Tracelinks übereinstimmten. Neun Tracelinks (12 %) wurden im Vergleich zu den im Workshop erarbeiteten Tracelinks fälschlicherweise gemeldet (siehe Tabelle 24). Alle Tracelinks müssten in der Realität einer genaueren Prüfung unterzogen werden, es sei denn, der eingestellte Schwellwert würde dies vermeiden.

	Korrekte Meldung falscher Tracelinks	Fälschliche Meldung korrekter Tracelinks	Gesamte Anzahl Meldungen	Präzision	Trefferquote
Proband 1	6	0	6	100 %	67 %
Proband 2	3	1	4	75 %	33 %
Proband 3	5	0	5	100 %	56 %
Proband 4	0	0	0	-	-
Proband 5	7	2	9	78 %	78 %
Proband 6	7	1	8	88 %	78 %
Proband 7	6	0	6	100 %	67 %
Proband 8	7	0	7	100 %	78 %
Proband 9	7	1	8	88 %	78 %
Proband 10	5	0	5	100 %	56 %
Proband 11	3	1	4	75 %	33 %
Proband 12	0	0	0	-	-
Proband 13	0	0	0	-	-
Proband 14	1	1	2	50 %	11 %
Proband 15	0	0	0	-	-
Proband 16	3	1	4	75 %	33 %
Proband 17	0	0	0	-	-
Proband 18	5	1	6	83 %	56 %
Proband 19	0	0	0	-	-
Gesamt	65	9	74	88 %	100 %
Relativ	88 %	12 %	100 %	-	-

Tabelle 24: Meldung von Tracelinks pro Proband

Die insgesamt 74 gemeldeten Tracelinks verteilen sich auf insgesamt 15 Elementekombinationen (siehe Tabelle 25), wovon neun Elementekombinationen korrekter und sechs fälschlicherweise gemeldet wurden. Bei der Betrachtung der Häufigkeit der Nennung dieser 15 Tracelinks fällt auf, dass die bewusst integrierten falschen Tracelinks mit einem Durchschnittswert von 7,2 mal durch die Probanden genannt wurden (zwischen 2 und 10 mal), während die fälschlicherweise gemeldete Tracelinks durchschnittlich lediglich 1,5 mal gemeldet wurden (zwischen 1 und 2 mal). Die Verwendung eines Schwellwerts für den Bewertungszähler würde in diesem Zusammenhang also Sinn machen. Gesetzt den Fall, er würde auf mind. zwei Nennungen definiert werden, könnten immerhin drei der sechs fälschlicherweise gemeldeten Tracelinks aus der Analyseübersicht herausgefiltert werden. Die Ergebnisse zeigen jedoch auch, dass in diesem Zusammenhang Vorsicht geboten ist, da die Differenz der Häufigkeit der Nennung zwischen korrekten und falschen Tracelinks nicht so groß ist, dass die Tracelinks eindeutig bzw. automatisch bewertet werden können. Würde bspw. der Schwellwert auf drei gesetzt werden, würden zwar alle fälschlicherweise gemeldeten Tracelinks herausgefiltert, jedoch auch ein falscher Tracelink, der eigentlich zu untersuchen wäre. Der Wert lässt sich, wie erwähnt, nicht allgemeingültig fest-

legen, sondern hängt bspw. von der Unternehmensgröße ab und muss somit auf Basis der Erfahrung individuell festgelegt werden.

#	Quellelement	Zielelement	Häufigkeit Nennung
Korrekte Meldung falscher Tracelinks			
1	A1.1.1 Hinterradantrieb mittels Fußkurbel und Kette	P1.2.1 hydraulische Federgabel	7
2	A1.1.1 Hinterradantrieb mittels Fußkurbel und Kette	P1.5.4 2 Brems Scheiben	10
3	A1.1.1 Hinterradantrieb mittels Fußkurbel und Kette	P1.4.2 Schaltwerk (Übersetzung hinten)	2
4	A1.1.2 Übersetzung mittels Ketten-schaltung	P1.6.2 Sattel	9
5	A1.1.2 Übersetzung mittels Ketten-schaltung	P1.7.3 Beleuchtung	9
6	A1.4.1 Unebenheiten im Gelände und im Fahrbelag ausgleichen	P1.5.2 2 Bremshebel	7
7	F1.3.1 Steuersignal aufnehmen	P1.5.4 2 Brems Scheiben	7
8	F1.5.2 Licht reflektieren	P1.5.2 2 Bremshebel	10
9	F1.6 Federung/Dämpfung	P1.5.3 2 Hydraulikleitungen	4
Fälschliche Meldung korrekter Tracelinks			
10	A1.1.1 Hinterradantrieb mittels Fußkurbel und Kette	P1.3.3 Hinterrad	1
11	A1.2 Bremsen	P1.5.3 2 Hydraulikleitungen	2
12	A1.2.1 Verschmutzungsrisiko minimieren	P1.5.1 2 hydraulische Scheibenbremsen	2
13	A1.2.1 Verschmutzungsrisiko minimieren	P1.5.4 2 Brems Scheiben	1
14	A1.2.3 muss mit Handkraft betrieben werden können	P1.5.1 2 hydraulische Scheibenbremsen	2
15	F1.2.2 Kraft leiten	P1.5.4 2 Brems Scheiben	1

Tabelle 25: Übersicht der Elementepaare, die von den Probanden als falsch gemeldet wurden

Beim Abgleich von Tabelle 22 mit Tabelle 25 fällt auf, dass alle falschen Tracelinks, die vorsätzlich in das Tracelink-Modell integriert wurden, durch die Probanden identifiziert wurden, obwohl deren eigentliche Aufgabenstellung eine Auswirkungsanalyse vorsah. Die Trefferquote (bezogen auf die Anzahl falscher Tracelinks) für die aggregierten Werte beträgt somit 100 % und die Präzision 88 %. Der erste Teil der Hypothese 2 wurde somit durch die Studie bestätigt: während der Interaktion mit dem Traceability-Modell fallen Nutzern falsch modellierte Tracelinks auf. Dass, wie durch den zweiten Teil der Hypothese 2 postuliert, die Anzahl der Probanden einen Einfluss auf die Vollständigkeit der Ergebnisse hat, lässt sich Tabelle 22 entnehmen. Keiner der Pro-

banden hat allein alle neun falsch modellierten Tracelinks identifizieren können. Durch die Kombination der Ergebnisse aller 19 Probanden konnten jedoch alle falschen Tracelinks identifiziert werden. Dieses Ergebnis lässt den Schluss zu, dass die Methode TraceEvaluation in der industriellen Anwendung einer Vielzahl an Nutzern zur Verfügung gestellt werden sollte, um möglichst viele fehlerhafte Tracelinks zu identifizieren.

6.3.4 ZUSAMMENFASSUNG UND DISKUSSION

Das Kapitel 6.3 adressierte die Forschungsfrage, ob Crowdsourcing-Methoden genutzt werden können, um die Qualität des Tracelink-Modells zu verbessern (siehe Kapitel 4.2). Die in diesem Zusammenhang formulierte Hypothese postuliert, dass Fehler im Traceability-Modell quasi nebenbei bei der Verwendung von Traceability-Modellen identifiziert und gemeldet werden können, und dass die Anzahl der Nutzer eine Auswirkung auf die Vollständigkeit der Ergebnisse hat.

Auf dieser Hypothese aufbauend wurde die Crowdsourcing-Methode TraceEvaluation entwickelt, die den Anwendern von Tracelinks erstmals im Kontext der Traceability bei beliebigen Aufgabenstellungen eine Art Add-on zur Verfügung stellt, mit dessen Hilfe falsche Tracelinks einfach gemeldet werden können. Ergänzt wird dieses Add-on durch ein Modul zur Auswertung, in dem die Meldungen potenziell falscher Tracelinks aggregiert und ausgewertet werden können.

Um die Wirksamkeit der Methode und damit die Hypothese 2 evaluieren zu können, wurde eine prototypische Implementierung vorgenommen, die auf dem Viewer Ariadne's Eye aufsetzt. Darüber wird den Anwendern in ihrem alltäglichen Arbeitsumfeld die Möglichkeit zur Verfügung gestellt, Bewertungen abzugeben. Auf Basis der prototypischen Implementierung wurde eine Studie durchgeführt, bei der die Probanden im Rahmen von Auswirkungsanalysen von Änderungen betroffene Bauteile identifizieren sollten. Um den industriellen Anwendungsfall zu simulieren war die Qualitätssicherung lediglich eine am Rande erwähnte Zusatzoption, die jedoch von der Mehrzahl der Probanden wahrgenommen wurde.

In der Studie gelang es, die Hypothese 2 zu bestätigen, denn alle vorsätzlich im Beispiel platzierten falschen Tracelinks wurden durch die Probanden identifiziert, obwohl eigentlich eine andere Aufgabenstellung bearbeitet werden sollte. Die so erreichte Trefferquote von 100 % sowie die Präzision von 88 % sind als exzellent zu bewerten. Andererseits gelang es keinem einzelnen Probanden, alle falschen Tracelinks zu identifizieren – die Anzahl der Probanden hat somit eine Auswirkung auf die Vollständigkeit der Analyse.

Im Rahmen der Studie wurden durch die Probanden auch korrekte Tracelinks als potenziell falsch eingeordnet, was bei der Auswertung zusätzliche Aufwände bei der Bewertung der Meldungen bedeutet. Jedoch waren diese Meldungen mit einem Anteil von 12 % an der Gesamtanzahl gemeldeter Tracelinks in der Minderheit. Es zeigte sich in diesem Zusammenhang zudem, dass die Anzahl an Meldungen bei tatsächlich falschen Tracelinks mit durchschnittlich 7,5 Nennungen deutlich höher als bei fälschlicherweise gemeldeten Tracelinks (durchschnittlich 1,5 Nennungen) war. Ein Schwellwert würde an dieser Stelle helfen, die Anzahl fälschlicherweise gemeldeter Tracelinks und somit den Aufwand für deren Bewertung zu reduzieren. Die Höhe des Schwellwerts ist jedoch nicht verallgemeinerbar und birgt zudem die Gefahr Meldungen tatsächlich falscher Tracelinks auszublenden.

Wie auch schon in der Diskussion in Kapitel 5.5 angemerkt, gilt im Übrigen auch im Zusammenhang mit der Methode TraceEvaluation, dass die Entscheidung für oder gegen einen Tracelink sehr stark durch die subjektive Interpretation der inhaltlichen Bedeutung der Elemente und des Konzepts „Tracelink“ durch den Anwender beeinflusst ist. Eine eindeutige Kategorisierung der Tracelinks in „richtig“ und „falsch“ ist somit nicht einfach möglich. Jedoch gelingt es mit der Methode TraceEvaluation, potenziell falsche Tracelinks, die durch viele Benutzer gemeldet werden, an einer zentralen Stelle aggregiert zu analysieren. Der für die durchgängige Nachverfolgbarkeit zuständige Entwickler kann nach Auswertung der Meldungen in Diskussionen mit den Fachexperten versuchen, eine einheitliche Sicht auf das Konzept „Tracelink“ einzubringen und somit die Qualität des Tracelink-Modells langfristig steigern.

6.4 TRACELEGACY – METHODE ZUR IDENTIFIKATION FEHLENDER TRACELINKS

Wie in Kapitel 6.1.3 diskutiert, basieren die meisten Methoden des Stands der Technik darauf, Vorschläge für Tracelinks durch einen Vergleich der Bezeichnungen bzw. der detaillierten Beschreibungen der Elemente der Artefakte, zwischen denen eine durchgängige Nachverfolgbarkeit hergestellt werden soll, zu generieren. Für diesen Vergleich kommen unterschiedlich komplexe Algorithmen zum Einsatz, um auf sprachlicher bzw. inhaltlicher Ebene eine Ähnlichkeit der Elemente zu identifizieren. Problematisch ist dabei, dass ähnliche Begriffe verwendet werden müssen, um eine Abhängigkeit zu identifizieren. Selbst wenn eine Synonymbildung angewendet wird, um eventuell genutzte Ausdrücke mit gleicher oder ähnlicher Bedeutung zu erkennen, ist es notwendig, dass die Elemente in den betrachteten Artefakten mit einem ähnlichen Vokabular beschrieben werden. Während dies zwischen einer Anforderungsspezifikation und einer Funktionsstruktur u. U. funktioniert, da bspw. die gewünschten Funktionen explizit in den Anforderungen benannt werden, stoßen diese Ansätze bei der Analyse von Funktionsstrukturen in Kombination mit Produktstrukturen

u. U. an ihre Grenzen, da die Bauteile häufig nicht nach ihren Funktionen benannt werden.

Vor diesem Hintergrund stellt sich die Forschungsfrage, ob Wissen, welches mithilfe von Tracelinks in vergangenen Projekten dokumentiert wurde, geeignet ist, um in aktuellen Projekten zur Steigerung der Qualität des Tracelink-Modells beizutragen. Durch den Autor dieser Arbeit wurde die Hypothese aufgestellt, dass auf dieser Wissensbasis Vorschläge für fehlende Tracelinks generiert werden können. Um diese Hypothese zu evaluieren, wurde die Methode *TraceLegacy* entwickelt und prototypisch evaluiert. Dabei stand nicht die Optimierung der Methode, sondern nur die grundsätzliche Überprüfung der Machbarkeit im Fokus. Während das grundlegende Funktionsprinzip der Methode im folgenden Kapitel 6.4.1 beschrieben wird, erfolgt die Beschreibung des entwickelten Algorithmus in Kapitel 6.4.2 und dessen prototypische Implementierung in Kapitel 6.4.3. Die Evaluierung der Methode wird in Kapitel 6.4.4 erläutert und die Ergebnisse der Untersuchung sowie Potenziale der Weiterentwicklung in Kapitel 6.4.5 diskutiert.

6.4.1 FUNKTIONSPRINZIP DER METHODE TRACELEGACY

Wie beschrieben, greift die Methode *TraceLegacy* auf Entwicklungswissen aus vergangenen, ähnlichen Projekten zurück, um Vorschläge für fehlende Tracelinks zu generieren⁴¹. Dabei werden nicht die aktuellen Artefakte untereinander, sondern mit Artefakten aus Vorgängerprojekten (engl.: *Legacy Projects*) verglichen, die zugehörigen Tracelinks aus Vorgängerprojekten analysiert und so Vorschläge für neue Tracelinks generiert. Das Funktionsprinzip der Methode ist in Abbildung 70 dargestellt. Tracelinks, die in einem Projekt zwischen Artefakten modelliert werden, werden in einer zentralen Datenbank gespeichert (Abbildung 70, A), um zu einem späteren Zeitpunkt wieder darauf zurückgreifen zu können. Werden in einem neuen Projekt dieselben Artefakt-Typen auf Abhängigkeiten zueinander untersucht, wird eine Anfrage an die Datenbank gesendet, um zu ermitteln, ob eine ähnliche Elementekombination bereits zu einem früheren Zeitpunkt schon einmal per Tracelink verknüpft war (Abbildung 70, B). Ist dies der Fall wird ein Vorschlag für einen zu modellierenden Tracelink im aktuellen Projekt ausgegeben (Abbildung 70, C).

Das Funktionsprinzip sieht somit nicht die linguistische Untersuchung der Elemente von Artefakten unterschiedlichen, sondern gleichen Typs vor. In dem in Abbildung 70 dargestellten Beispiel werden somit die Anforderungen des aktuellen mit denen des Vorgängerprojekts verglichen. Das ist insofern vorteilhaft, da die Bezeichner von Ele-

⁴¹ Sie lässt sich somit nutzen, um falsch negative Tracelinks zu identifizieren.

menten gleichen Typs tendenziell eher mit gleichem Vokabular beschrieben werden und somit die Ähnlichkeit der Elemente leichter festgestellt werden kann.

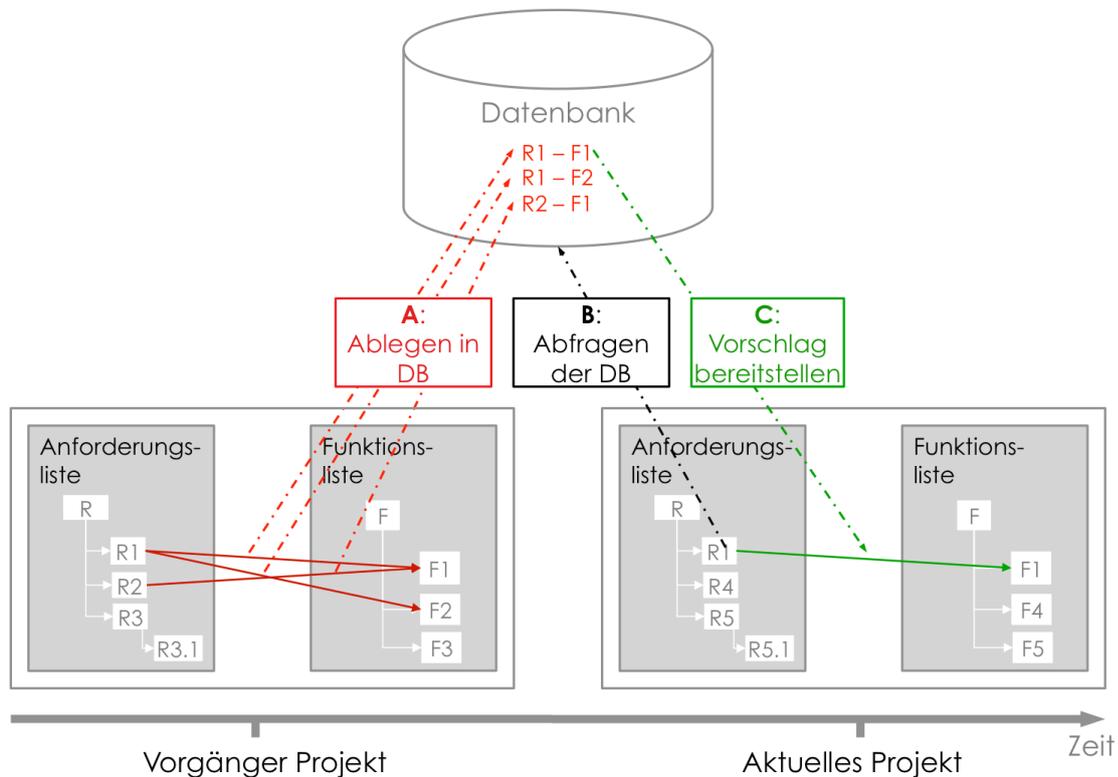


Abbildung 70: Funktionsprinzip der Methode TraceLegacy

Die potenzielle Abhängigkeit zwischen den Elementen unterschiedlichen Typs im aktuellen Projekt (R1 und F1 in Abbildung 70) wird somit nicht auf Basis der aktuellen Artefakte sondern auf Basis eines Tracelinks in einem vergangenen Projekt identifiziert. Dieser Tracelink kann im vergangenen Projekt bspw. in einem manuellen Prozess durch einen Entwickler modelliert worden sein. Das Wissen des Entwicklers, welches in Form eines Tracelinks dokumentiert wurde, wird durch Anwendung der Methode TraceLegacy somit zur Verbesserung der Qualität des aktuellen Tracelink-Modells wiederverwendet.

6.4.2 ALGORITHMUS DER METHODE TRACELEGACY

Der Algorithmus der Methode TraceLegacy lässt sich in die zwei Arbeitsschritte „Vorverarbeitung“ und „Abhängigkeitsanalyse“ unterteilen.

In der Vorverarbeitung werden zunächst die Ähnlichkeiten aktueller Elemente zu denen vergangener Projekte bestimmt. Dabei wird vorbereitend die sog. Stammformreduktion durchgeführt, bei der die Worte der Elementbezeichner des aktuellen Projekts auf ihren Wortstamm reduziert werden. Ziel ist es, verschiedene morphologische Varianten eines Wortes erkennen zu können. Zunächst werden für die Suche irrele-

vante Bindewörter und Artikel (wie bspw. „und“, „oder“, „der“, „die“, „das“) aus den Bezeichnern entfernt. Zusätzlich können beliebige Wörter auf einer sog. Negativliste (engl.: Blacklist) definiert werden, die ebenfalls entfernt und somit im Folgenden ignoriert werden. Anschließend werden die verbliebenen Wörter des Bezeichners mit Hilfe des Porter-Stemmer-Algorithmus auf ihren Wortstamm zurückgeführt [Porter 2006]. Dieser Algorithmus verfügt über mehrere Verkürzungsregeln, die nacheinander auf die Wörter angewendet werden, bis eine minimale Anzahl von Silben übrig bleibt, die möglichst in allen grammatikalischen Formen der Wörter auftaucht⁴². Für nähere Informationen zur Funktionsweise dieses Algorithmus, der auf [Porter 2013b] zum Download angeboten wird, empfiehlt sich die Lektüre von [Porter 2006] oder [Porter 2013a]. Eine beispielhafte Übersicht zurückgeführter deutscher Verben ist auf [Porter 2013a] zu finden.

Übrig bleibt für jedes Element der beiden betrachteten aktuellen Artefakte ein reduzierter Elementbezeichner, welcher aus mehreren auf den Wortstamm reduzierten Worten besteht. Dieser dient im Folgenden als Suchanfrage, um die Ähnlichkeit zu Elementen vergangener Projekte über einen einfachen Stringvergleich zu berechnen. Der Stringvergleich ermöglicht in der prototypischen Implementierung der Methode die Unterscheidung in „ähnlich“, wenn mindestens ein Wort der reduzierten Bezeichner übereinstimmt, und „nicht ähnlich“, wenn kein Wort übereinstimmt.

Zum Zweck der Verwaltung der so ermittelten Ähnlichkeitswerte „1“ (ähnlich) und „0“ (nicht ähnlich) existiert für jeden Artefakt-Typ (also Anforderungsspezifikation, Funktionshierarchie, Produktstruktur usw.) eine projektübergreifende Datenbank, in der die Werte mit Bezug zur Elementkombination dokumentiert werden. Eine schematische Darstellung dieses Verwaltungsprinzips ist in Abbildung 71 in Form einer Matrix dargestellt. Die Elemente aller Projekte werden auf die Zeilen und Spalten verteilt und die Ähnlichkeitswerte in die gemeinsamen Zellen eingetragen.

Somit entsteht für jeden Artefakt-Typ jeweils eine quadratische Matrix, in der die Ähnlichkeit der textuellen Bezeichner der Elemente vergangener und aktueller Projekte untereinander bewertet wird. Da die Ähnlichkeit nicht gerichtet ist, wird nur eine Hälfte der Matrix zur Bewertung der Ähnlichkeiten genutzt. Die Werte auf der Hauptdiagonalen sind immer „1“, da die reduzierten Bezeichner mit sich selbst verglichen werden. Diese vergleichsweise einfache Form der Umsetzung wurde gewählt, da zunächst im Vordergrund stand, die Methode grundsätzlich abzusichern. In einer Weiterentwicklung der Methode TraceLegacy wäre es an dieser Stelle sinnvoll, weitere Methoden (siehe bspw. Kapitel 6.1.2) zu evaluieren, um eine genauere Bewertung

⁴² So ist das Ergebnis der Anwendung des Porter-Stemming-Algorithmus für jedes der Wörter „aufeinanderfolge“, „aufeinanderfolgen“, „aufeinanderfolgend“, „aufeinanderfolgende“, „aufeinanderfolgenden“, „aufeinanderfolgender“ der Wortstamm „aufeinanderfolg“.

der Ähnlichkeit der textuellen Beschreibungen der Elemente vornehmen zu können und eine höhere Präzision zu erreichen.

<u>A</u>		Projekt 1		Projekt 2	
		a ₁	a ₂	a ₃	a ₄
Projekt 1	a ₁	-			
	a ₂	0	-		
Projekt 2	a ₃	1	0	-	
	a ₄	1	1	0	-

<u>B</u>		Projekt 1		Projekt 2	
		b ₁	b ₂	b ₃	b ₄
Projekt 1	b ₁	-			
	b ₂	0	-		
Projekt 2	b ₃	0	1	-	
	b ₄	1	0	0	-

Abbildung 71: Schematische Darstellung der Verwaltung der Ähnlichkeitswerte in Form von Matrizen

Bei der an die Vorverarbeitung (Erstellung der Ähnlichkeitsmatrizen) anschließenden Abhängigkeitsanalyse werden alle Elementekombinationen eines aktuellen Projekts, die bisher noch nicht mit einem Tracelink miteinander verknüpft wurden, hinsichtlich einer möglichen Abhängigkeit zueinander untersucht. Dabei werden für jede Elementekombination der Artefakte A und B die folgenden Schritte durchlaufen (siehe auch Abbildung 72):

1. Für das aktuell betrachtete Element a werden ähnliche Elemente a_i vergangener Projekte aus der Übereinstimmungsmatrix A ermittelt. Gleiches gilt für das Element b , für das ähnliche Elemente b_j aus der Übereinstimmungsmatrix B zusammengestellt werden.
2. Die so ermittelten Elemente a_i und b_j werden miteinander kombiniert und überprüft, ob zwischen ihnen in einem vergangenen Projekt eine Abhängigkeit mit einem Tracelink dokumentiert wurde.
3. Ist dies der Fall, wird eine Empfehlung für die Modellierung eines Tracelinks im aktuellen Projekt ausgegeben.

Um den Algorithmus näher zu erläutern, wird er im Folgenden anhand des generischen Beispiels aus Abbildung 72 beschrieben. Im aktuellen Projekt werden zwei Artefakte A und B betrachtet, deren Elemente a_1 und b_1 noch keinen Tracelink zueinander aufweisen. Dem zuvor beschriebenen TraceLegacy-Algorithmus folgend, werden nun zu a_1 und b_1 ähnliche Elemente aus den Übereinstimmungsmatrizen A und B ermittelt, die alle bis zu diesem Zeitpunkt erstellten Elemente a_i und b_j beinhalten. Die Abfrage ergibt für das Element a_1 , dass a_3 und a_4 aus einem vergangenen Projekt eine Ähnlichkeit aufweisen. Für b_1 wird das Element b_4 zurückgegeben. Diese drei Elemente (a_3 , a_4 und b_4) werden miteinander kombiniert und in der ModelTracer Datenbank überprüft, ob in einem früheren Projekt Tracelinks zwischen ihnen modelliert worden sind. Dies ist für die Kombination a_3 / b_4 der Fall, weshalb für die aktuelle

Kombination a_1 / b_1 die Empfehlung ausgegeben wird, diese auf eine Abhängigkeit zueinander zu überprüfen und ggf. einen Tracelink zu modellieren. Diese Vorgehensweise wird für alle Elementekombinationen der beiden Artefakte A und B, die noch nicht miteinander verknüpft sind, wiederholt.

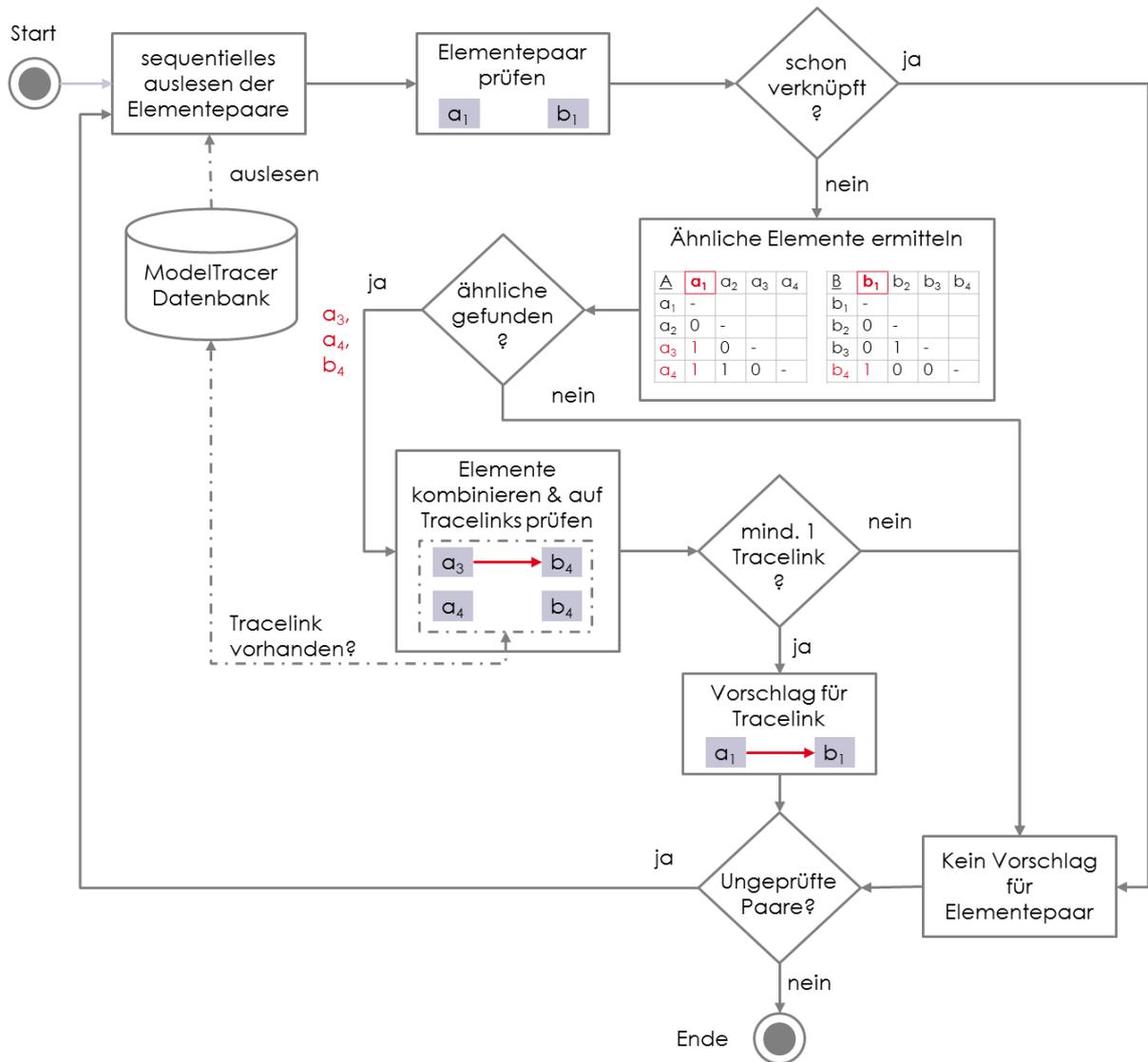


Abbildung 72: Algorithmus der Abhängigkeitsanalyse mit TraceLegacy

6.4.3 PROTOTYPISCHE IMPLEMENTIERUNG DER METHODE TRACELEGACY

Die Methode TraceLegacy wurde auf Basis des in Kapitel 3.6.4.3 vorgestellten ModelTracers prototypisch implementiert. Sie wird, wie die TraceEvaluation über das Menü „Quality Check“ gestartet. Die GUI ist in Abbildung 73 dargestellt und wird im Folgenden erläutert. Bevor die Analyse gestartet wird, kann mit der Auswahlbox (1) definiert werden, ob eine Stammformreduktion durchgeführt werden soll. Zusätzlich kann die Negativliste mit Wörtern, die bei der Analyse ignoriert werden sollen, über den Blacklist-Button (2) aufgerufen werden. Über den Start-Button (3) wird die Vor-

verarbeitung (Erstellung der Übereinstimmungsmatrizen der unterschiedlichen Artefakte) und Abhängigkeitsanalyse gestartet und deren Fortschritt in dem Fortschrittbalken (4) dargestellt.

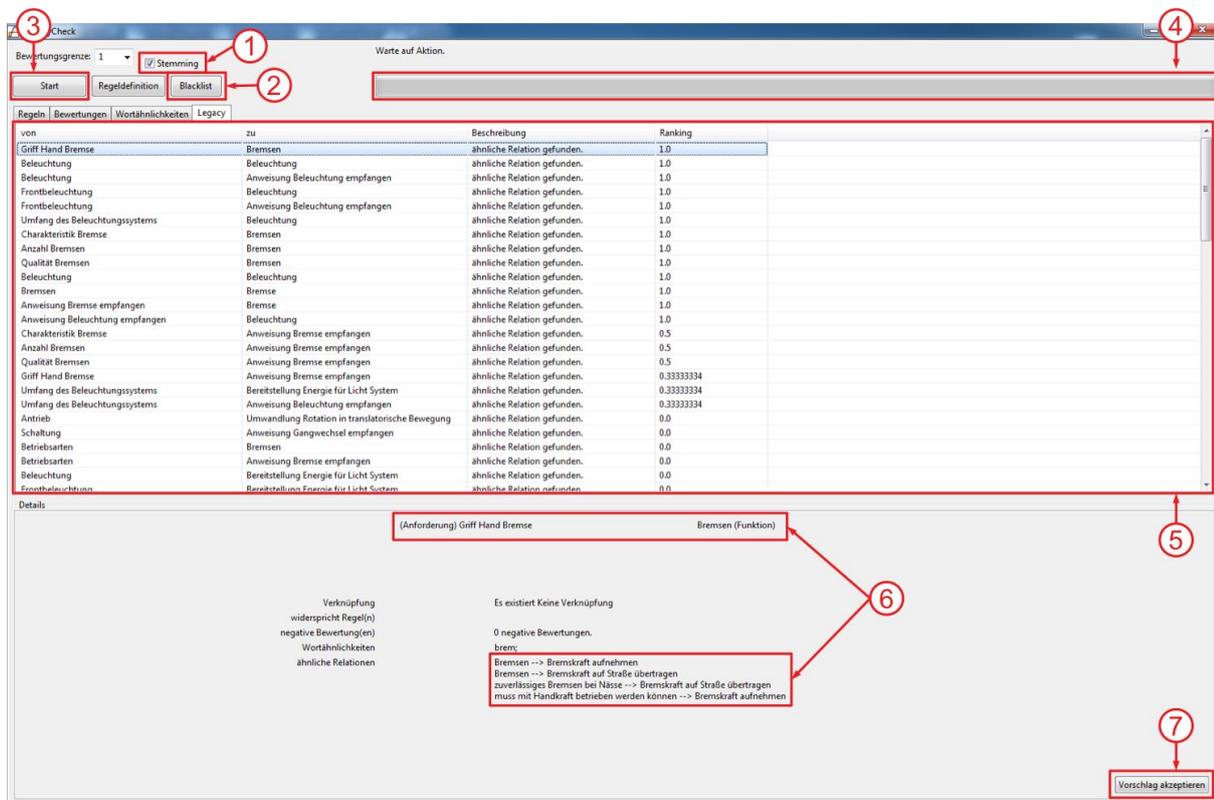


Abbildung 73: GUI des Prototyps der Methode TraceLegacy

Nach Abschluss der Analyse werden die Tracelink-Vorschläge im Hauptfenster (5) dargestellt. Für die Methode TraceLegacy sind dabei die drei ersten Spalten relevant: Start- und Endelement des aktuellen Projekts für die ein Tracelink vorgeschlagen wird sowie die Beschreibung, weshalb ein Tracelink zur Überprüfung vorgeschlagen wird. Wird einer der Tracelink-Vorschläge im Hauptfenster ausgewählt, werden zusätzliche Informationen im unteren Bereich des Fensters angezeigt (6). Zum einen werden die Bezeichner der Elemente, zwischen denen ein Tracelink vorgeschlagen wird, angezeigt und zusätzlich in Klammern dargestellt, welchem Artefakt-Typ sie angehören. Zum anderen werden die Tracelinks, die zwischen Elementen aus Artefakten vergangener Projekte gefunden wurden und aufgrund derer der Vorschlag generiert wurde, angezeigt. Der Vorschlag kann, wenn er als korrekt bewertet wird und demnach im aktuellen Projekt eine Abhängigkeit zwischen den betrachteten Elementen existiert, die bislang noch nicht berücksichtigt wurden, per Klick auf den „Vorschlag akzeptieren“-Button (7) angenommen und somit ein Tracelink modelliert werden.

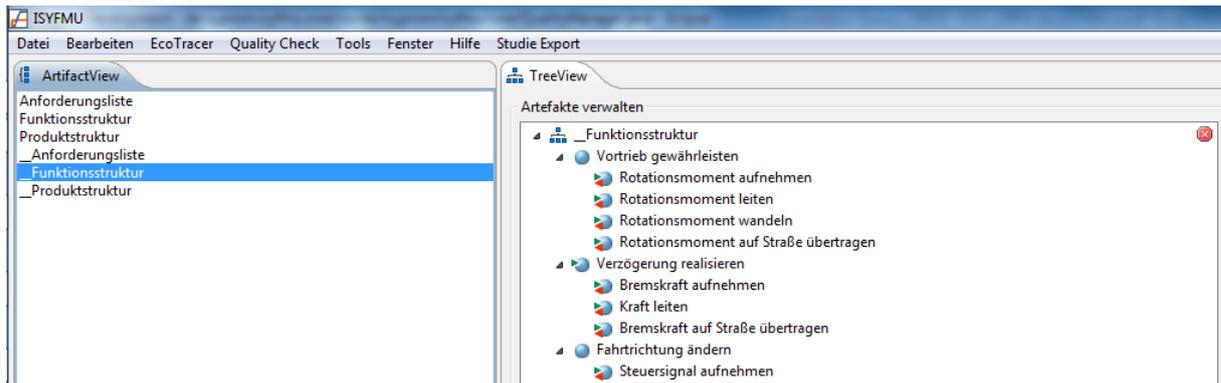


Abbildung 74: Verwaltung der beiden Projekte im ModelTracer (Artefakte des Vorgängerprojekts „Fahrrad“ wurden mit dem Präfix „_“ versehen).

Im Anschluss wurden zwei Analysen durchgeführt. Bei der Ersten wurde die Stammformreduktion, bei der die in den Bezeichnern enthaltenen Worte auf ihren Wortstamm heruntergebrochen werden, deaktiviert. Bei der zweiten war die Stammformreduktion aktiviert. Die Darstellung der Ergebnisse, deren Interpretation und Diskussion erfolgt im anschließenden Kapitel 6.4.4.2.

6.4.4.2 AUSWERTUNG DER EVALUATION

Die beiden Analysen ergaben jeweils eine Anzahl von Vorschlägen für das aktuelle Projekt „Pedelec“, die durch den Algorithmus auf Basis des Vorgänger-Projekts „Fahrrad“ ermittelt wurden. In Abbildung 75 sind die Vorschläge für die Analyse, bei der die Stammformreduktion aktiviert war, dargestellt. Diese Vorschläge inkl. der Informationen bzgl. der ähnlichen Tracelinks im Vorgänger-Projekt wurden im Anschluss manuell in Excel-Tabellen übertragen und hinsichtlich ihrer Korrektheit bewertet.

In Tabelle 28 sind einige exemplarische Vorschläge, die durch die Methode TraceLegacy ohne die Verwendung der Stammformrückführung generiert wurden, zusammengestellt. Die Vorschläge sind zeilenweise angegeben und durchnummeriert⁴⁵. Darüber hinaus sind jeweils die Start- und Endelemente, für die ein Tracelink-Vorschlag erstellt wurde (linke Spalte „Aktuelles Pedelec-Projekt“), und die Start- und Endelemente, auf deren Basis die Empfehlung abgeleitet wurde (rechte Spalte „Vergangenes Fahrrad-Projekt“), angegeben. In der zuletzt genannten Spalte können dabei auch mehrere Start- und Endelemente pro Zeile erfasst sein, wenn die Empfehlung aufgrund mehrerer Übereinstimmungen im vergangenen Projekt generiert wurde. Die Übereinstimmungen der Bezeichner, die den Grund für die Empfehlung darstellen, wurden in den einzelnen Zellen fett markiert. Falls ein Vorschlag als

⁴⁵ Die Nummerierung entspricht dabei der im Anhang, weshalb in den Tabellen keine durchgängige Nummerierung gewährleistet werden kann.

nicht korrekt bewertet wurde, wurde in der zweiten Spalte ein „x“ vermerkt. Die vollständigen Auswertungsergebnisse sind in Anhang C zu finden.

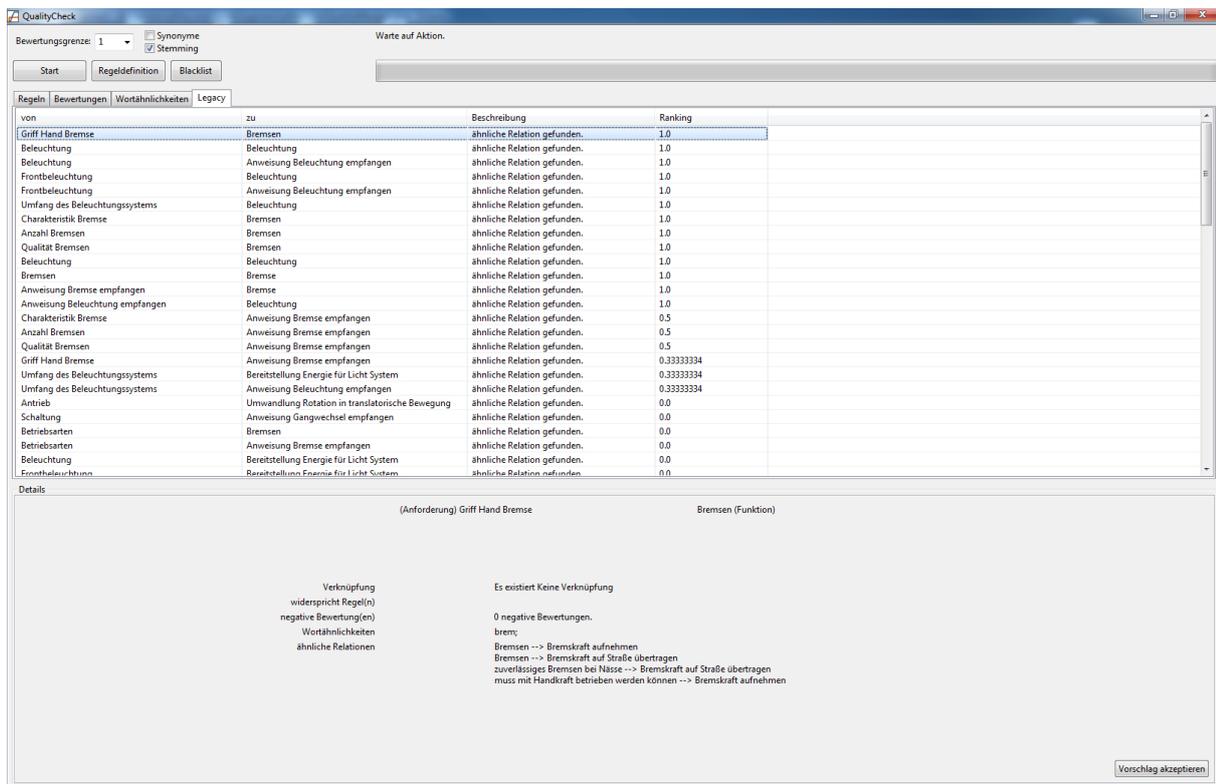


Abbildung 75: Ergebnisliste der prototypischen Implementierung der Methode TraceLegacy anhand der Beispiele „Fahrrad“ und „Pedelec“

#	x	Aktuelles Pedelec-Projekt		Vergangenes Fahrrad-Projekt	
		von	ZU	von	ZU
1		Beleuchtung	Bereitstellung Energie für Licht System	Beleuchtung vorsehen	Licht reflektieren
3		Umwandlung Rotation in translatorische Bewegung	Hinterrad	Rotationsmoment auf Straße übertragen	Hinterrad
5	x	Pedal Signal empfangen	Vorderrad	Signal in Richtungsänderung umsetzen	Vorderrad
7		Griff Handbremse	Bremse	Bremsen zuverlässiges Bremsen bei Nässe muss mit Handkraft betrieben werden können	Bremseinheit Bremseinheit Bremseinheit

Tabelle 28: Evaluationsergebnis ohne Stammformrückführung (Auszug)

Vorschlag Nr. 1: Aufgrund des Tracelinks zwischen der Fahrrad-Anforderung „Beleuchtung vorsehen“ und der Fahrrad-Funktion „Licht reflektieren“ wurde der Vorschlag generiert, die Elementkombination von Pedelec-Anforderung „Beleuchtung“ und der Pedelec-Funktion „Bereitstellung Energie für Licht System“ auf eine

Abhängigkeit zu überprüfen. In der Bewertung wurde dem Vorschlag zugestimmt, da die Pedelec-Funktion durch die Pedelec-Anforderung notwendig wird.

Vorschlag Nr. 3: Aufgrund des Tracelinks zwischen der Fahrrad-Funktion „Rotationsmoment auf Straße übertragen“ und dem Fahrrad-Bauteil „Hinterrad“ wurde der Vorschlag generiert, die Elementekombination von Pedelec-Funktion „Umwandlung Rotation in translatorische Bewegung“ und dem Pedelec-Bauteil „Hinterrad“ auf Abhängigkeiten zu überprüfen. In der Bewertung wurde dem Vorschlag zugestimmt, da das Pedelec-Hinterrad an der Realisierung der Funktion beteiligt ist. Bei diesem Vorschlag wird deutlich, dass die in Kapitel 6.1 vorgestellten Methoden wahrscheinlich nicht zu einem Vorschlag geführt hätten, da sich der inhaltliche Zusammenhang nicht direkt aus einem Wortvergleich oder dem Verständnis der Bezeichner ergibt. Mit Hilfe der Methode TraceLegacy wird jedoch auf Wissen über diesen Zusammenhang zurückgegriffen, das in dieser Form durch menschliche Entscheidungen in der Vergangenheit generiert wurde.

Vorschlag Nr. 5: Aufgrund des Tracelinks zwischen der Fahrrad-Funktion „Signal in Richtungsänderung umsetzen“ und dem Fahrrad-Bauteil „Vorderrad“ wurde der Vorschlag generiert, die Elementekombination von Pedelec-Funktion „Pedal Signal empfangen“ und dem Pedelec-Bauteil „Vorderrad“ auf Abhängigkeiten zu überprüfen. Dieser Vorschlag wurde als falsch eingestuft, da im Pedelec-Beispiel zwischen dem Pedal-Signal und dem Vorderrad kein Zusammenhang besteht. Er wurde generiert, da das Signal im Fahrrad-Beispiel nicht näher spezifiziert wurde.

Vorschlag Nr. 7: Bei Vorschlag Nr. 7 führten mehrere Tracelinks aus dem Fahrrad-Projekt dazu, dass für das Pedelec-Beispiel zwischen der Anforderung „Griff Handbremse“ und dem Bauteil „Bremse“ ein Tracelink-Vorschlag generiert wurde. Dieser Tracelink wurde als korrekt bewertet, da die Anforderung nach einem Griff für die Handbremse sich an das Bremssystem und damit auch an die Bremse richtet. Dieser Vorschlag wäre auch durch einen einfachen Wortvergleich der Elemente des Pedelec-Projekts zu generieren gewesen.

Insgesamt wurden bei der Analyse ohne Stammformrückführung 20 Vorschläge für zu modellierende Tracelinks durch die Methode TraceLegacy gemacht. Von diesen Vorschlägen wurden 18 als korrekt und 2 als nicht korrekt bewertet. Das entspricht einer Präzision von 90,0 % und einer Trefferquote von 4,2 %.

In Tabelle 29 ist ein Auszug des Ergebnisses der Analyse mit aktivierter Stammformrückführung dargestellt. Anhand der fett markierten Suchworte wird deutlich, dass nicht mehr ganze Worte als Suchanfragen genutzt wurden, sondern nur Wortteile, die nach Anwendung der Stammformrückführung übrig geblieben sind. Das führt dazu,

dass mehr Vorschläge zurückgegeben werden, die ohne die Rückführung nicht gefunden werden können.

#	x	Aktuelles Pedelec-Projekt		Vergangenes Fahrrad-Projekt	
		von	zu	von	zu
4		Front beleuchtung	Beleuchtung	Beleuchtung vorsehen	unmittelbare Umgebung beleuchten
20		Antrieb	Umwandlung Rotation in translatorische Bewegung	Hinter antrieb mittels Fußkurbel und Kette Hinter antrieb mittels Fußkurbel und Kette Hinter antrieb mittels Fußkurbel und Kette Hinter antrieb mittels Fußkurbel und Kette	Rotationsmoment aufnehmen Rotationsmoment leiten Rotationsmoment wandeln Rotationsmoment auf die Straße übertragen
21		Schaltung	Anweisung Gang wechsel empfangen	Übersetzung mittels Kettens schaltung	Übersetzung wechseln

Tabelle 29: Evaluationsergebnis mit Stammformrückführung (Auszug)

Vorschlag Nr. 4: Aufgrund des Tracelinks zwischen der Fahrrad-Anforderung „Beleuchtung vorsehen“ und der Fahrrad-Funktion „unmittelbare Umgebung beleuchten“ wurde der Vorschlag generiert, die Elementekombination von Pedelec-Anforderung „Frontbeleuchtung“ und Pedelec-Funktion „Beleuchtung“ auf eine Abhängigkeit zu überprüfen. In der Bewertung wurde dem Vorschlag zugestimmt, da die Pedelec-Funktion durch die Pedelec-Anforderung notwendig wird.

Vorschlag Nr. 20: Bei Vorschlag Nr. 20 führten mehrere Tracelinks aus dem Fahrrad-Projekt dazu, dass für das Pedelec-Beispiel zwischen der Anforderung „Antrieb“ und der Funktion „Umwandlung Rotation in translatorische Bewegung“ ein Tracelink-Vorschlag generiert wurde. Dieser Tracelink wurde als korrekt bewertet, da diese Anforderungskategorie Anforderungen bzgl. der Pedale, dem E-Motor, der Schaltung und der Höchstgeschwindigkeit zusammenfasst und somit direkt mit der Funktion in Abhängigkeit steht. Auch bei diesem Vorschlag wird deutlich, dass die in Kapitel 6.1 vorgestellten Methoden wahrscheinlich nicht zu einer Empfehlung geführt hätten, da sich der inhaltliche Zusammenhang nicht direkt aus einem Wortvergleich oder dem Verständnis der Bezeichner ergibt.

Vorschlag Nr. 21: Aufgrund des Tracelinks zwischen der Fahrrad-Anforderung „Übersetzung mittels Kettenschaltung“ und der Fahrrad-Funktion „Übersetzung wechseln“ wurde der Vorschlag generiert, die Elementekombination von Pedelec-Anforderung „Schaltung“ und der Pedelec-Funktion „Anweisung Gangwechsel empfangen“ auf Abhängigkeiten zu überprüfen. In der Bewertung wurde dem Vorschlag zugestimmt, da die Forderung nach der Integration einer Schaltung direkt mit deren Steuerung in der Funktionsstruktur zusammenhängt. Auch dieser Vorschlag wäre mit den Metho-

den aus Kapitel 6.1 nicht generiert worden, da der Zusammenhang zwischen schalten und wechseln nicht identifiziert worden wäre.

Insgesamt wurden bei der Analyse mit Stammformrückführung 82 Tracelink-Vorschläge durch die Methode TraceLegacy generiert. Von diesen Vorschlägen wurden 50 als korrekt und 32 als nicht korrekt bewertet. Anzumerken ist jedoch, dass ein Vorschlag auf Grundlage unterschiedlicher Tracelinks im Vorgänger-Projekt doppelt ausgegeben wurde. Dieser wird somit nur einmal gezählt. Daraus ergibt sich eine korrigierte Präzision von 60,9 % und eine Trefferquote von 11,5 %.

6.4.5 ZUSAMMENFASSUNG UND DISKUSSION

Das Kapitel 6.4 beschäftigte sich mit der Forschungsfrage, ob Wissen, welches mithilfe von Tracelinks in vergangenen Projekten dokumentiert wurde, geeignet ist, um in aktuellen Projekten zur Steigerung der Qualität des Tracelink-Modells beizutragen. Durch den Autor dieser Arbeit wurde die Hypothese aufgestellt, dass auf dieser Wissensbasis Vorschläge für fehlende Tracelinks generiert werden können.

Die vorgestellte Methode *TraceLegacy* zielt daher darauf ab, Entwicklungswissen vergangener Projekte zu nutzen, um Vorschläge für fehlende Tracelinks in aktuellen Projekten zu generieren. Sie ist somit dazu geeignet, falsch negative Tracelinks zu identifizieren. Dabei werden im Gegensatz zu den Methoden im Stand der Technik nicht die aktuellen Artefakte miteinander, sondern diese mit Artefakten gleichen Typs aus Vorgängerprojekten verglichen, die zugehörigen Tracelinks analysiert und so Vorschläge für neue Tracelinks generiert. Der Vorteil dieses Funktionsprinzips ist, dass immer Artefakte gleichen Typs (bspw. ein aktuelles Anforderungsartefakt mit einem Anforderungsartefakt aus einem vergangenen Projekt) miteinander verglichen werden, in denen die Elemente tendenziell mit ähnlichem Vokabular beschrieben werden. Die Vorschläge für Tracelinks zwischen den Elementen zweier aktueller Artefakte unterschiedlichen Typs (bspw. zwischen einem Anforderungs- und einem Funktionsartefakt), bei denen sich die Bezeichner hinsichtlich der verwendeten Vokabeln unterscheiden können, werden dann auf Basis der zuvor berechneten Ähnlichkeiten sowie der Tracelinks des Vorgängerprojekts berechnet. Das Wissen aus dem Vorgängerprojekt wird durch Anwendung der Methode *TraceLegacy* somit zur Verbesserung der Qualität des aktuellen Tracelink-Modells wiederverwendet.

In der prototypischen Implementierung des *TraceLegacy*-Algorithmus kommt ein Wortvergleich mit optionaler Stammformrückführung zum Einsatz, um die Übereinstimmungsmatrizen zu erstellen, die benötigt werden, um die Ähnlichkeit der Elemente des aktuellen Projekts zu denen aus vergangenen Projekten zu ermitteln. Auf dieser Basis werden Elemente vergangener Projekte identifiziert, hinsichtlich vorhande-

ner Tracelinks beurteilt und Vorschläge für Tracelinks im aktuellen Projekt ausgegeben.

Die Evaluation der Methode TraceLegacy wurde, mit dem Ziel die grundsätzliche Funktionsfähigkeit der Methode nachzuweisen und damit die Hypothese 3 zu bestätigen, anhand zweier Beispiele durchgeführt, die unabhängig voneinander entwickelt wurden. Dieses Szenario bildet somit den Anwendungsfall ab, dass bei der Entwicklung des aktuellen Projekts „auf der grünen Wiese“ begonnen wurde, ohne Artefakte aus dem Vorgängerprojekt wiederzuverwenden.

Auf Basis der Artefakte der beiden Projekte wurde die Analyse einmal ohne und einmal mit Stammformrückführung durchgeführt. Bei ersterer wurden 20 Tracelinks durch die Methode TraceLegacy vorgeschlagen. Von diesen Vorschlägen wurden 18 als korrekt und 2 als nicht korrekt bewertet. Das entspricht einer Präzision von 90 % und einer Trefferquote von ca. 4 %. Bei der Analyse mit Stammformrückführung wurden 82 einzelne Vorschläge für zu modellierende Tracelinks generiert. Von diesen Vorschlägen wurden 50 als korrekt und 32 als nicht korrekt bewertet. Die Präzision entspricht somit ca. 60 % und die Trefferquote ca. 11 %.

Die grundsätzliche Funktionsfähigkeit der Methode wurde nachgewiesen und damit die Hypothese 3 bestätigt, indem relevante Vorschläge für Tracelinks auf Basis eines ähnlichen Vorgängerprojekts berechnet werden konnten. Unter diesen Vorschlägen waren auch solche, die mit den beschriebenen Ansätzen des Stands der Technik nicht zu identifizieren gewesen wären, da die Bezeichner der Elemente der Artefakte des aktuellen Projekts keine identischen oder synonymen Worte aufwiesen.

Es wurde bei der Evaluation ebenfalls deutlich, dass der Algorithmus zur Berechnung der Element-Ähnlichkeit einen großen Einfluss auf die charakteristischen Messwerte Trefferquote und Präzision hat. So ist die Präzision von TraceLegacy ohne Stammformrückführung deutlich höher als mit, da nur bei vollständiger Übereinstimmung ein Vorschlag generiert wird. Jedoch werden dafür insgesamt weniger zu modellierende Tracelinks vorgeschlagen, weshalb die Trefferquote geringer ist. Für beide Varianten gilt, dass die ermittelten Präzisionswerte nach der Skala von Hayes et al. als exzellent und die Trefferquoten als schlecht zu bewerten sind [Hayes et al. 2006, S. 11].

Im Rahmen dieser Arbeit stand jedoch nicht die Optimierung der Methode TraceLegacy, sondern nur die grundsätzliche Überprüfung der Machbarkeit im Fokus. Da diese nachgewiesen wurde, sollte aufbauend auf dem grundsätzlichen Funktionsprinzip die Betrachtung möglicher Kombinationen der Methode TraceLegacy mit anderen Algorithmen zur Bewertung der Ähnlichkeit (siehe bspw. Kapitel 6.1) Gegenstand weiterführender Forschung sein.

Zusätzlich wird auch durch die Methode *TraceLegacy* bestätigt, was in Kapitel 3.4.1 beschrieben wurde: nach Meinung des Autors eignen sich die bisher vorhandenen Methoden zur automatisierten Erfassung von Tracelinks nicht dazu, bei der Erfassung der Tracelinks in der Mechatronik zum Einsatz zu kommen. Das liegt einerseits an der Präzision, die bei keiner der betrachteten Methoden 100 % ist, und zum anderen an den üblicherweise geringen Trefferquoten, die besagen, dass viele Tracelinks nicht automatisiert identifiziert werden können. Während die Erfassung der Tracelinks, entweder direkt bei der Entwicklung der Artefakte oder im Anschluss, durch die Entwickler erfolgen sollte, können die automatisierten Methoden geeignet kombiniert werden und bei der Sicherung der Qualität des Tracelink-Modells unterstützen.

6.5 ZUSAMMENFASSUNG UND DISKUSSION

In Kapitel 6 wurden Methoden zur Pflege von Tracelink-Modellen thematisiert. Dazu wurden zunächst Methoden des aktuellen Stands der Technik vorgestellt (siehe Kapitel 6.1), anhand ihrer Eigenschaften kategorisiert und so Forschungspotenziale aufgedeckt. Einerseits wurde deutlich, dass sich die dargestellten Methoden in ihrer beschriebenen Umsetzung nicht für die Ermittlung falsch negativer Tracelinks eignen. Andererseits zeigte sich, dass die Methoden zur Analyse lediglich Informationen aus aktuellen Projekten berücksichtigen. Das hat insbesondere bei der Analyse von Artefakten unterschiedlichen Typs (bspw. bei einer Funktionsstruktur in Kombination mit einer Produktstruktur) den Nachteil, dass deren Elemente überlappende Inhalte in Form von Wörtern, Synonymen oder Beschreibungen aufweisen müssen, damit Abhängigkeiten erkannt werden können. An diesen vom Stand der Technik noch nicht adressierten Aspekten orientierten sich die im Folgenden beschriebenen Methoden *TraceEvaluation* und *TraceLegacy*, die ein weiteres „Puzzlestück“ zur Sicherung der Qualität des Tracelink-Modells darstellen.

Die Crowdsourcing-Methode *TraceEvaluation* adressiert Hypothese 2. Diese besagt, dass Fehler im Traceability-Modell quasi nebenbei bei der Nutzung von Tracelinks identifiziert und gemeldet werden können. *TraceEvaluation* ermöglicht es den Anwendern, im Tagesgeschäft falsche Tracelinks einfach zu melden. Diese Meldungen laufen in einem Auswertungsmodul zusammen und können durch den Qualitätssicherungsbeauftragten analysiert werden. Die Wirksamkeit der Methode wurde in einer Studie evaluiert, bei der die Probanden im Rahmen von Auswirkungsanalysen von Änderungen betroffene Bauteile identifizieren sollten. Den Probanden gelang es dabei, alle vorsätzlich im Beispiel platzierten falschen Tracelinks zu identifizieren. Die so erreichte Trefferquote von 100 % sowie die Präzision von 88 % sind als exzellent zu bewerten. Darüber hinaus gelang es jedoch keinem der Probanden allein, alle fal-

schen Tracelinks zu identifizieren – die Anzahl der Probanden hat somit eine Auswirkung auf die Vollständigkeit der Analyse.

Die Methode *TraceLegacy* adressiert die Hypothese 3. Diese postuliert, dass auf der Basis von Tracelink-Modellen vergangener Projekte Vorschläge für fehlende Tracelinks in aktuellen Projekten generiert werden können. *TraceLegacy* berechnet dementsprechend für alle Elemente, die in Artefakten des aktuellen Projekts enthalten sind, die Ähnlichkeit zu Elementen desselben Typs aus vergangenen Projekten. Unter Berücksichtigung des Tracelink-Modells des vergangenen Projekts, können auf diese Weise Vorschläge für Tracelinks im aktuellen Projekt generiert werden. Die Evaluation der Methode *TraceLegacy* wurde mit dem Ziel, deren grundsätzliche Machbarkeit zu überprüfen und damit die Hypothese 3 zu bestätigen, anhand zweier Beispiele durchgeführt, die unabhängig voneinander entwickelt wurden. Je nach gewähltem Vergleichsalgorithmus (ohne bzw. mit Stammformrückführung) konnten so exzellente Präzisionswerte von ca. 90 % bzw. 61 % erzielt werden. Für die Trefferquote liefert *TraceLegacy* mit ca. 4 % bzw. 11 % vergleichsweise schlechte Ergebnisse. Die Methode lieferte somit in jedem Fall relevante Vorschläge, wodurch die Hypothese 3 bestätigt wurde. Zur Optimierung der Methode, die im Rahmen dieser Arbeit nicht im Vordergrund stand, sollten zukünftig andere Algorithmen zur Bewertung der Ähnlichkeit, als der eingesetzte Stringvergleich, evaluiert werden.

Die im Kontext der Qualitätssicherung durchgeführten Studien und Evaluationen bestätigten im Übrigen, was schon in der Diskussion in Kapitel 5.5 angemerkt wurde: die Entscheidung für oder gegen einen Tracelink ist sehr stark durch die subjektive Interpretation der inhaltlichen Bedeutung der Elemente und des Konzepts „Tracelink“ durch den Anwender geprägt. Eine eindeutige Kategorisierung der Tracelinks in „richtig“ und „falsch“ ist somit nicht immer einfach möglich. Jedoch gelingt es mit der Methode *TraceEvaluation*, potenziell falsche Tracelinks, die durch viele Benutzer gemeldet werden, an einer zentralen Stelle zusammenlaufen zu lassen. Der für die durchgängige Nachverfolgbarkeit zuständige Entwickler kann nach Auswertung der Meldungen in Diskussionen mit den Fachexperten versuchen, eine einheitliche Sicht auf das Konzept „Tracelink“ zu generieren und somit die Qualität des Tracelink-Modells langfristig steigern. Ähnlich wirkt auch die Methode *TraceLegacy*. Basierend auf dem Wissen vergangener Projekte (und damit potenziell anderer Entwickler), werden Vorschläge für Tracelinks generiert, die dann in Diskussionen auf ihre Richtigkeit überprüft werden können. Das anvisierte Ziel, mit den beiden Methoden zwei weitere Puzzlestücke zur Sicherung der Qualität des Tracelink-Modells bereitzustellen, konnte somit erreicht werden.

7 ANWENDUNG DER METHODEN ZUR ENTWICKLUNG EINES TECHNISCHEN SYSTEMS

In diesem Kapitel wird die Anwendung der in den Kapiteln 5.4, 6.3 und 6.4 beschriebenen Methoden *EcoTracing*, *TraceEvaluation* und *TraceLegacy* im Entwicklungsprozess vorgestellt. Zu diesem Zweck werden die genannten Methoden zunächst in Kapitel 7.1 im V-Modell des Fraunhofer IPK verortet, um darzustellen wann sie während der Entwicklung eingesetzt werden können. Anschließend werden die Methoden in Kapitel 7.2 im Kontext der Entwicklung des mechatronischen Außenspiegels, die anhand ihrer Prozessphasen beschrieben wird, demonstriert.

Ziel des Kapitels ist es, den Einsatz der Methoden unter realistischen (wenngleich hinsichtlich Umfang und Komplexität vereinfachten) Bedingungen darzustellen und Vor- und Nachteile, die sich durch den Einsatz für die Anwender ergeben, aufzuzeigen. Dabei wird nicht auf alle Aspekte der Entwicklung eingegangen, sondern der Fokus auf die Erstellung und Änderung der wichtigsten Artefakte gelegt und darüber hinaus von einem sequenziellen Entwicklungsverlauf ohne Iterationen ausgegangen. Aus diesem Entwicklungsverlauf werden ausgewählte Entwicklungsschritte näher erläutert, die Einsatzmöglichkeiten der zuvor erwähnten Methoden am Beispiel dargestellt und dabei vergleichend auf die Vorgehensweise mit und ohne jeweilige Methode eingegangen.

Abschließend werden die Einsatzmöglichkeiten und die Anwendung der Methoden in Kapitel 7.3 zusammengefasst und diskutiert.

7.1 VERORTUNG DER METHODEN IM V-MODELL

Die in den vorhergehenden Kapiteln 5.4, 6.3 und 6.4 beschriebenen Methoden EcoTracing, TraceEvaluation und TraceLegacy dienen der Unterstützung der Erfassungs- und Pflegeaktivitäten von Tracelinks und Tracelink-Modellen. Da diese Aktivitäten kontinuierlich während der Entwicklung technischer Systeme durchgeführt werden müssen, bieten sich unterschiedliche Einsatzzeitpunkte für die genannten Methoden an. In Abbildung 76 sind diese anhand des V-Modells des Fraunhofer IPK schematisch dargestellt.

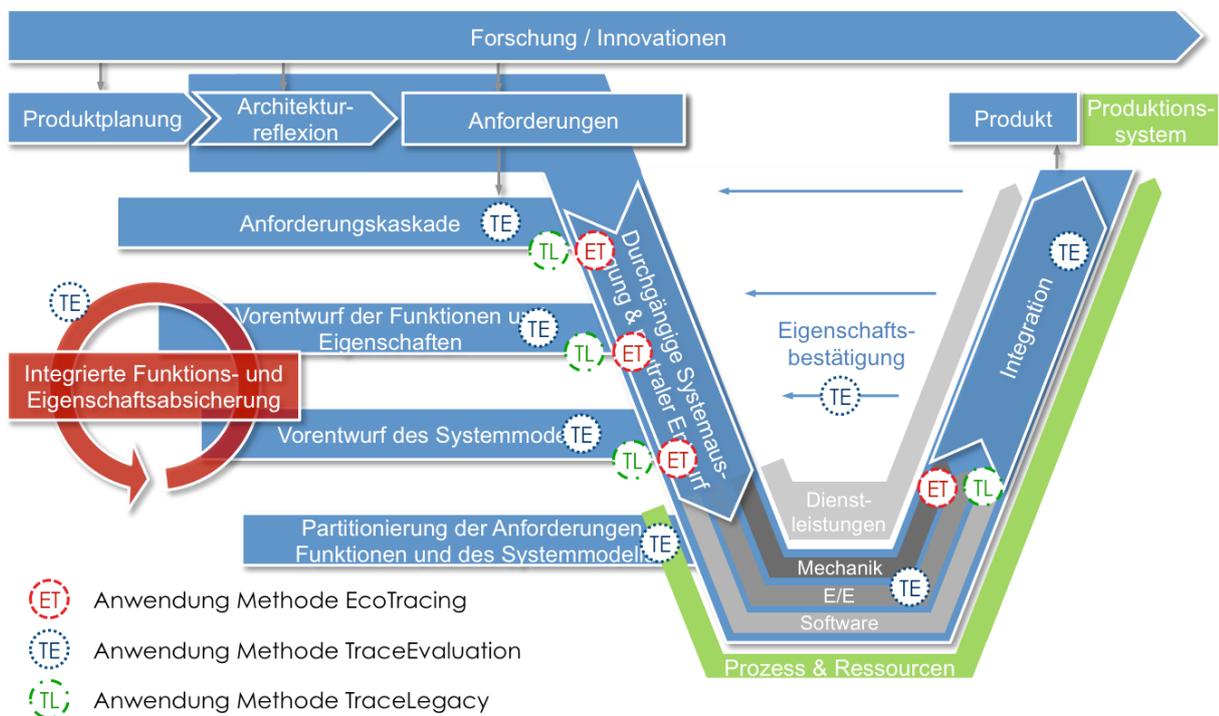


Abbildung 76: Verortung der Methoden im V-Modell des Fraunhofer IPK

Die Einsatzmöglichkeiten der Methode EcoTracing (ET) sind jeweils den Prozessschritten, in denen Artefakte modelliert werden, nachgelagert bzw. in ihnen enthalten. Abhängig ist der Einsatzzeitpunkt von der Tracelink-Modellierungsstrategie im Rahmen des Entwicklungsprojekts. Werden die Tracelinks erst nach Fertigstellung der Artefakte modelliert, wird EcoTracing im Anschluss an die jeweiligen Prozessschritte durchgeführt. Sollen hingegen bereits Zwischenstände der Artefakte hinsichtlich vorhandener Abhängigkeiten zu anderen Artefakten analysiert werden, so findet die Anwendung der Methode während des jeweiligen Prozessschritts statt.

Gleiches gilt für die Methode TraceLegacy (TL), deren Einsatz in Abbildung 76 jeweils zum gleichen Zeitpunkt wie EcoTracing dargestellt wird. Hintergrund ist, dass sich die Durchführung der Qualitätssicherung des Tracelink-Modells mit TraceLegacy im Anschluss an die Modellierung der Tracelinks anbietet, um zu prüfen, ob Abhängigkeiten bei der Analyse übersehen wurden. Grundsätzlich lässt sich die Methode jedoch zu jedem Zeitpunkt im Entwicklungsprozess⁴⁶ durchführen.

Die Methode TraceEvaluation (TE) wird im Gegensatz dazu überwiegend während der Prozessschritte angewendet. Da TraceEvaluation auf einem Crowdsourcing-Ansatz basiert, mit dem Hinweise bzgl. möglicher falscher Tracelinks von den Entwicklern erfasst werden, ist der Einsatz der Methode immer sinnvoll, wenn eine Interaktion mit den Artefakten (die in den Prozessschritten erstellt, bearbeitet und verwendet werden) stattfindet. Dies ist insbesondere bei der integrierten Funktions- und Eigenschaftsabsicherung sowie bei der Eigenschaftsbestätigung der Fall, da bei diesen Aktivitäten die Tracelinks verwendet werden, um aktuelle IST- mit SOLL-Parametern bspw. der Anforderungen abzugleichen, wobei falsche Tracelinks besonders auffallen.

7.2 DEMONSTRATION DER METHODEN AM BEISPIEL DER ENTWICKLUNG EINES MECHATRONISCHEN AUßENSPIEGELS

Die Vorgehensweise während der Entwicklung des mechatronischen Außenspiegels im Rahmen der Lehrveranstaltung „Virtual Engineering in Industry“ orientierte sich an dem V-Modell des Fraunhofer IPK (siehe auch Kapitel 2.1.5). Aufgrund begrenzter Ressourcen wurden dabei nicht alle Prozessschritte des V-Modells durchlaufen sondern insb. auf die Phase der durchgängige Systemauslegung und des neutralen Entwurf fokussiert. In Abbildung 77 sind die durchgeführten Prozessschritte mittels roter Umrandung dargestellt.

Insgesamt wurden die folgenden sechs Prozessschritte durchlaufen:

1. Anforderungen: die Studenten erhielten eine Spezifikation mit vorgegebenen Anforderungen
2. Anforderungskaskade: die vorgegebenen Anforderungen wurden ergänzt, strukturiert und in die PLM-Software ENOVIA übertragen
3. Vorentwurf der Funktionen und Eigenschaften: aus den Anforderungen wurden zu realisierende Funktionen identifiziert und mittels CATIA V6 in einer Funktionsstruktur modelliert

⁴⁶ Einzige Voraussetzungen sind, dass mehr als ein Artefakt vorhanden ist und in der Vergangenheit bereits ähnliche Entwicklungsprojekte durchgeführt wurden.

5. Vorentwurf des Systemmodells: Wirkprinzipien für die modellierten Funktionen wurden gewählt und das Verhalten abgebildet
7. Integrierte Funktions- und Eigenschaftsabsicherung: durch Simulation des Verhaltens des Außenspiegels wurde die Realisierung von Funktionen und Eigenschaften überprüft und mit den Anforderungen abgeglichen
8. Mechanik: mechanische Komponenten wurden in CATIA modelliert

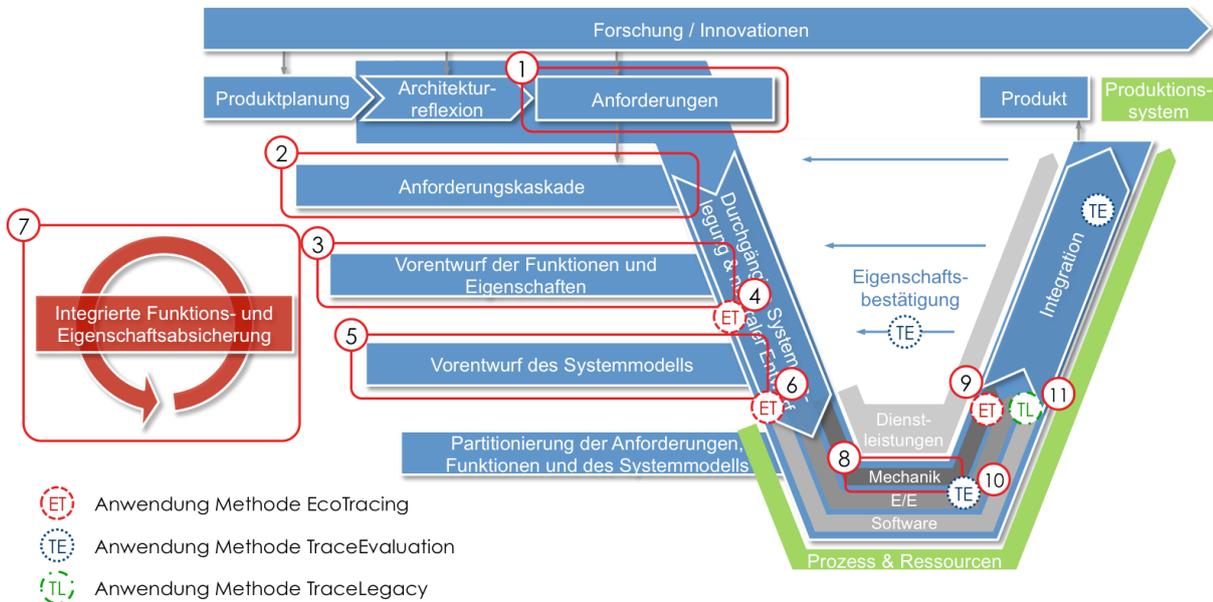


Abbildung 77: Darstellung der durchgeführten Prozessschritte während der Entwicklung des mechatronischen Außenspiegels

Diese Prozessschritte wurden durch unterstützende Aktivitäten begleitet, von denen im Kontext dieser Arbeit insb. die Erfassung von Tracelinks von Interesse ist. In Abbildung 77 ist dies durch die Aktivitäten (4), (6) und (9) in Form der Anwendung der Methode EcoTracing visualisiert⁴⁷.

Die so durchnummerierten Prozessschritte und Aktivitäten dienen im Folgenden als eine Art Drehbuch, anhand dessen die Entwicklung des mechatronischen Außenspiegels beschrieben wird. Um die Zuordnung zu erleichtern wurden diese Nummern in Kapitel 7.2 als Unterüberschriften 7.2.x wieder aufgenommen.

Zusätzlich wird der Einsatz der Methoden TraceEvaluation (Aktivität 10) und TraceLegacy (Aktivität 11) zur Pflege des Tracelink-Modells beschrieben. In Kapitel 7.2 findet sich dies in den Kapiteln „7.2.10 Durchführung einer Auswirkungsanalyse“ und „7.2.11 Durchführung einer Qualitätskontrolle“ wieder.

⁴⁷ Tatsächlich wurde durch die Studenten eine manuelle Tracelink-Modellierung durchgeführt. Nähere Informationen sind den zugehörigen Kapiteln 7.2.4, 7.2.6 und 7.2.9 zu entnehmen.

7.2.1 ANFORDERUNGEN

Zu Beginn des Projektes erhielt die Studentengruppe von den Betreuern eine Anforderungsspezifikation, in der grundlegende Randbedingungen der Entwicklung sowie des Außenspiegels spezifiziert wurden. Insgesamt enthielt die Spezifikation elf Anforderungen, die bspw. das Maximalgewicht, Kosten oder spezifische Funktionen (wie die Verstellung des Spiegels bei Rückwärtsfahrt) vorgaben. In Tabelle 30 sind die Eigenschaften der Anforderungsspezifikation zusammengefasst.

Artefaktname	Customer Requirments Specification
Strukturierungsart	Liste
Anzahl Elemente	11
Anzahl Hierarchieebenen	1

Tabelle 30: Eigenschaften der initialen Anforderungsspezifikation

7.2.2 ANFORDERUNGSKASKADE

Da diese Anforderungen nicht ausreichten, um sich ein umfassendes Bild von dem zu entwickelnden System zu machen, führten die Studenten ein Brainstorming sowie eine Machbarkeitsprüfung durch, um die Anforderungsspezifikation zu ergänzen. Die so gesammelten 27 Anforderungen wurden in ENOVIA textuell beschrieben (Eigenschaften siehe Tabelle 31). Zur Strukturierung des Anforderungsartefakts wurde eine Projektion mit sechs Kategorien auf höchster Ebene verwendet (siehe auch Kapitel 5.2.1.2). Im Folgenden werden diese Kategorien kurz vorgestellt:

- Basic Functions: Unter der Kategorie „Basic Functions“ wurden alle Anforderungen zusammengefasst, welche die Grundfunktionen eines Außenspiegel-systems betreffen. Dazu zählen bspw. die manuelle und automatische Verstellung des Spiegelglases.
- Boundary conditions: Randbedingungen, die überwiegend durch die Aufgabenstellung definiert wurden, wie maximale Kosten, Gewicht und Dimensionen des Außenspiegels;
- Command: Enthält Anforderungen, die Bedienung des Außenspiegelsystems durch den Benutzer betreffend;
- Comfort Functions: Kategorie für Anforderungen, die erweiterte Komfortfunktionen des Außenspiegelsystems fordern;
- Protective Equipment: Zusammenstellung von Anforderungen, die Sicherheitsaspekte (insbesondere bzgl. der Interaktion mit dem Außenspiegelsystem) betreffen;
- Legal: Anforderungen, die sich aus der Patent- und Rechtssituation ergeben.

Ergänzend zur textuellen Beschreibung der Anforderungen wurden ihre Wichtigkeit, die Schwierigkeit ihrer Umsetzung sowie die geplante Validierungsmethode festgelegt und ebenfalls in ENOVIA dokumentiert.

Artefaktnamen	Side Mirror Requirements Specification
Strukturierungsart	Hierarchie (Strukturierung mittels Projektion)
Anzahl Elemente	36
Anzahl Hierarchieebenen	3

Tabelle 31: Eigenschaften der Anforderungsspezifikation des mechatronischen Außenspiegels

In Abbildung 78 ist ein Ausschnitt der beschriebenen Anforderungshierarchie (links) sowie die textuelle Beschreibung der Anforderung „Maximum Angle“ (rechts) dargestellt, welche den maximalen Ausschlagwinkel des Spiegelglases vorgibt. Es wird deutlich, dass die Anforderungen über eine textuelle Beschreibung verfügen (im Feld „Beschreibung“ in der rechten Abbildung) und Parameter nicht explizit sondern nur als Bestandteil der Beschreibung angegeben wurden.

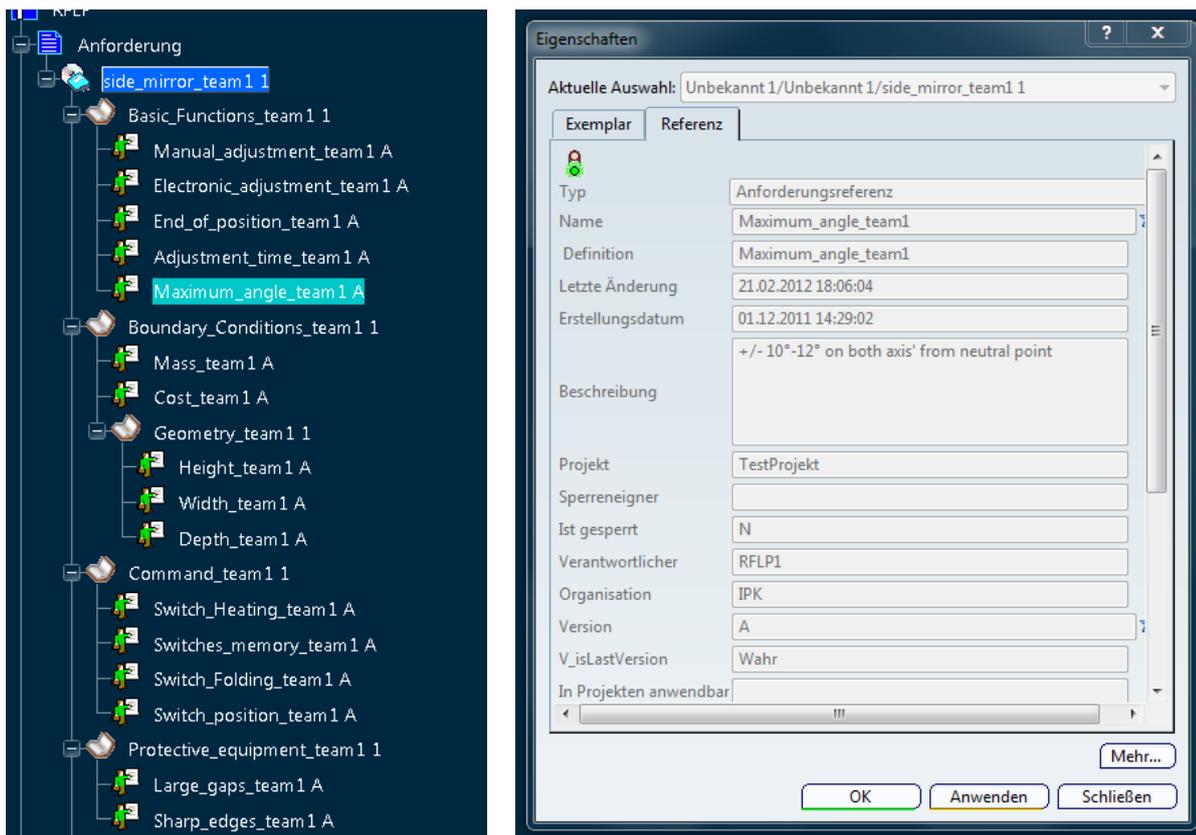


Abbildung 78: Darstellung eines Ausschnitts der Anforderungshierarchie (links) sowie Eigenschaften der Anforderung „Maximum Angle“ (rechts) in CATIA V6 (vgl. [Amin et al. 2012])

7.2.3 VORENTWURF DER FUNKTIONEN UND EIGENSCHAFTEN

Im Anschluss an die Dokumentation der Anforderungen wurden alle für deren Erfüllung notwendigen Funktionen des mechatronischen Außenspiegels identifiziert und im Funktionsmodul von CATIA V6 in einer Funktionsstruktur mit sog. „Building Blocks“ modelliert. Die Funktionsstruktur bestand aus insgesamt 20 Elementen. Eine Auflistung der wichtigsten Funktionen wird im Folgenden kurz dargestellt:

- Hauptfunktionen (Spiegelglaseinstellung)
- Sicherheitsfunktionen (Toter Winkel Warnung, Spiegelglasbeheizung)
- Komfortfunktionen (Spiegel einklappen, Fahrtrichtung anzeigen, Außenbeleuchtung)
- Außenspiegel kontrollieren (Regelung der unterschiedlichen Funktionen des Außenspiegels)
- Nutzereingaben erfassen (Erfassung der Nutzereingaben zur Steuerung unterschiedlicher Funktionen)
- Umgebungszustand erfassen

Nach Modellierung der Funktionen wurden diese mit Informationsflüssen verknüpft, um interne Abhängigkeiten abzubilden. Die so modellierte Funktionsstruktur ist in Abbildung 79 dargestellt.

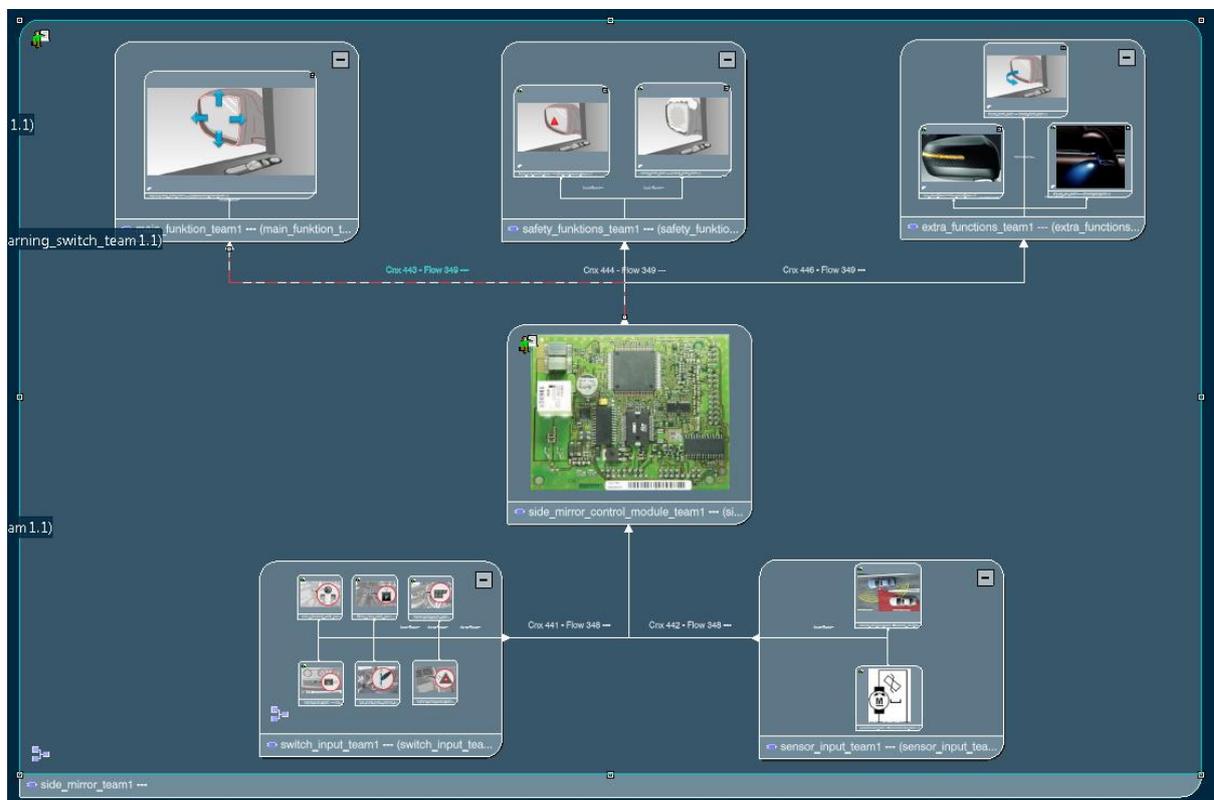


Abbildung 79: Funktionsstruktur des mechatronischen Außenspiegels [Amin et al. 2012]

Ziel der Studenten bei der Modellierung der Funktionsstruktur (Eigenschaften siehe Tabelle 32) war es, einen Schritt von den abstrakten Anforderungen hin zur tatsächlichen späteren Umsetzung zu nehmen. Deshalb wurden die Funktionen nicht lösungsneutral formuliert, sondern unter Berücksichtigung der tatsächlichen Lösungselemente. Diese Vorgehensweise erwies sich später als sinnvoll, da sie im weiteren Verlauf des Projekts aufgrund abgegrenzter Aufgaben und klar definierter Schnittstellen paralleles Arbeiten ermöglichte.

Artefaktnamen	Side Mirror Functional
Strukturierungsart	Hierarchie
Anzahl Elemente	20
Anzahl Hierarchieebenen	2

Tabelle 32: Eigenschaften der Funktionsstruktur des mechatronischen Außenspiegels

7.2.4 ERFASSUNG VON TRACELINKS ZWISCHEN ANFORDERUNGEN UND FUNKTIONEN

Im Anschluss an die Fertigstellung der Funktionsstruktur galt es die Nachverfolgbarkeit zwischen den Anforderungen und der Funktionsstruktur herzustellen, um zu dokumentieren, welche Funktionen zur Erfüllung welcher Anforderungen beitragen. Die Studenten wendeten zu diesem Zweck die in CATIA vorgesehene manuelle Modellierung von Tracelinks an, die im Folgenden kurz beschrieben wird:

Manuelle Modellierung von Tracelinks: Zum Zweck der Verknüpfung von Elementen verfügt CATIA V6 über die sog. Implementierungsbeziehungen, die von den Studenten verwendet wurden, um die Anforderungen und Funktionen miteinander zu verknüpfen. Diese Beziehungen sind typisierte und gerichtete Tracelinks, die ausdrücken, dass eine Anforderung durch eine Funktion implementiert wird.

Ausgehend von der Anforderungsspezifikation wird dabei in CATIA der Strukturbaum abgelaufen und nacheinander für jedes der 36 Anforderungs-Elemente überprüft, durch welche der 20 Funktionen die Anforderung realisiert wird. Sollte eine Abhängigkeit bestehen wird durch Auswahl der Anforderung und Öffnen des Kontextmenüs die Modellierung einer Implementierungsbeziehung gestartet (siehe Abbildung 80). Anschließend wird durch Navigation in der Funktionsstruktur die betroffene Funktion ausgewählt und die Implementierungsbeziehung bestätigt.

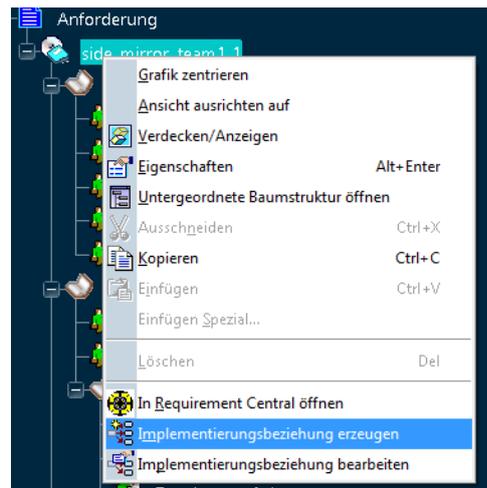


Abbildung 80: Anlegen einer Implementierungsbeziehung zwischen einer Anforderung und einer Funktion (vgl. [Amin et al. 2012])

Diese Vorgehensweise wurde teilweise online während der Modellierung der Artefakte durch die Studenten verwendet, um einen Tracelink zu modellieren. Um bei dieser Analyse jedoch keine Abhängigkeit zu übersehen, wurden die Elemente zusätzlich nach Fertigstellung der Artefakte (offline) auf Abhängigkeiten überprüft. Dabei wurden alle Anforderungen nacheinander betrachtet und hinsichtlich Abhängigkeiten zu den Funktionen der Funktionsstruktur analysiert. Dieses Vorgehen erforderte insgesamt 720 (teilweise implizite) Entscheidungen der Studenten über die Existenz einer Abhängigkeit zwischen jeweils zwei Elementen. Wie in Kapitel 3.7.2 beschrieben, sind diese Entscheidungen nicht immer leicht zu treffen. So ist die Entscheidung, einen Tracelink zwischen der Anforderung, in der die maximale Heizdauer von 360 s spezifiziert wird, und der Funktion „Außenspiegel beheizen“ zu modellieren, vergleichsweise einfach. Auch die Entscheidungen, dass die genannte Anforderung keine Abhängigkeit zur Funktion „Toter Winkel Warnung“ aufweist, ist leicht zu treffen. Welche der modellierten Funktionen an der Umsetzung der Anforderung „elektronische Verstellbarkeit des Außenspiegels vorsehen“ beteiligt sind, ist hingegen weniger leicht zu bestimmen, da mehrere Teilfunktionen die Verstellbarkeit realisieren.

Modellierung von Tracelinks mit EcoTracing: Zur strukturierten Durchführung dieser Abhängigkeitsanalyse, bei der keine Kombinationen übersehen und trotzdem die notwendigen Aufwände minimiert werden, lässt sich die Methode EcoTracing einsetzen. Durch eine Integration der Methode in V6 lassen sich die bereits modellierten Artefakte zu Beginn der Analyse auswählen. In Abbildung 81 ist dies für die Anforderungsspezifikation und die Funktionsstruktur dargestellt⁴⁸.

⁴⁸ Da zum aktuellen Zeitpunkt noch keine Integration des Prototypen mit der V6-Suite von Dassault Systèmes existiert, wurden die Artefakte des Außenspiegels nachmodelliert und in den ModelTracer importiert.

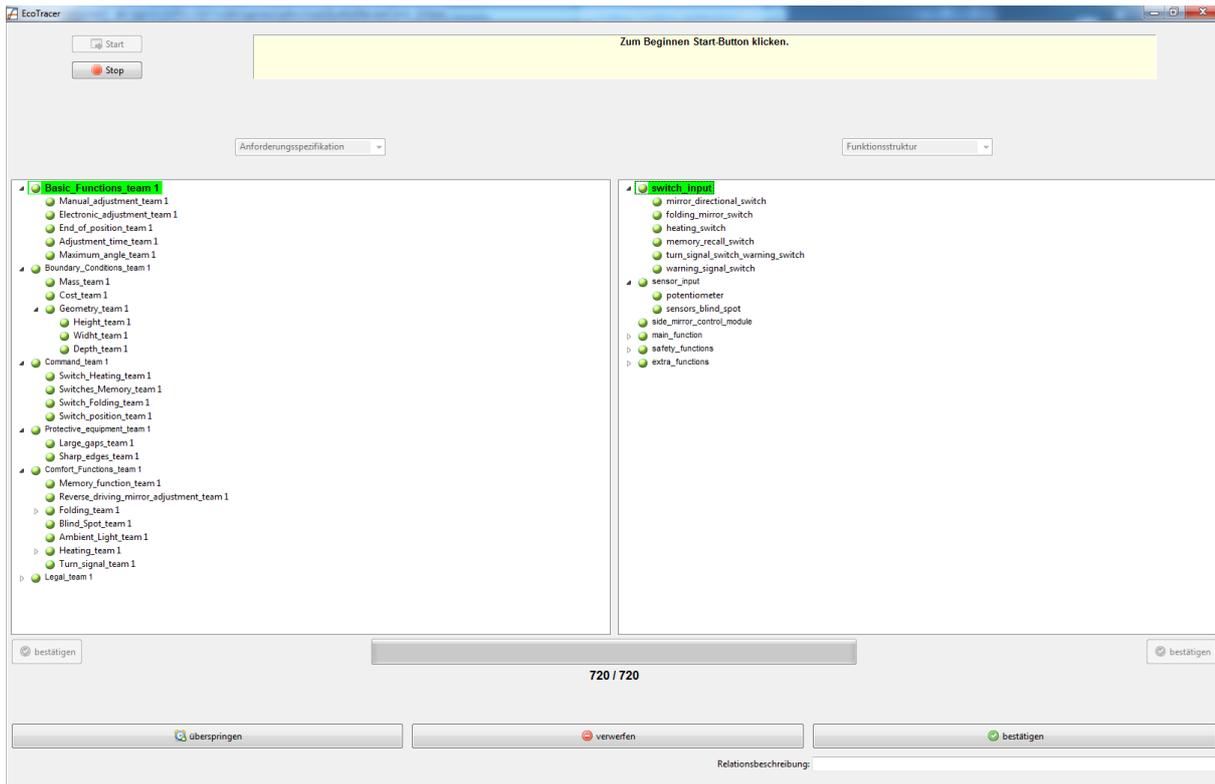


Abbildung 81: Anwendung der Methode EcoTracing zur Abhängigkeitsanalyse zwischen Anforderungen und Funktionen des Außenspiegels

Anschließend wird der Analysevorgang gestartet. Der EcoTracing-Algorithmus hebt dabei nacheinander jeweils ein Elementepaar zur Bewertung grün hervor. Der Entwicklungsingenieur, der diese Bewertung durchführt, entscheidet durch Auswahl des jeweiligen Buttons, ob eine Abhängigkeit zwischen Anforderung und Funktion besteht. Im Anschluss an die Entscheidung springt der Algorithmus zum nächsten Elementepaar. Wie bei der manuellen Modellierung von Tracelinks sind insgesamt maximal 720 Kombinationen von Anforderungen und Funktionen zu untersuchen (dargestellt als Fortschrittsbalken).

Vergleich der beiden Vorgehensweisen: Für den Nutzer, der die Abhängigkeitsanalyse mit EcoTracing durchführt, ergeben sich im Vergleich zur Standardvorgehensweise in CATIA V6 einige Vorteile. So wird das Übersehen einer Elementkombination und damit eines möglichen Tracelinks durch die schrittweise Abarbeitung des Algorithmus (siehe Abbildung 53) ausgeschlossen. Zwar können dem Anwender teilweise schwierige Entscheidungen hinsichtlich der Abhängigkeit von Elementen nicht gänzlich erspart werden, jedoch ermöglicht der Algorithmus die Verringerung der Anzahl an Entscheidungen um ca. 59 %, da nicht tatsächlich 720 Elementkombinationen, wie bei der manuellen Vorgehensweise, sondern nur 295 Kombinationen untersucht werden müssen. Zur Modellierung eines Tracelinks müssen dabei aufgrund der Umsetzung der Methode als Wizard im Gegensatz zur manuellen Vorgehensweise keine Kontextmenüs geöffnet und Auswahlen durchgeführt, sondern nur auf den Button

„bestätigen“ geklickt werden. Diese Art der Tracelink-Modellierung kann natürlich auch nachteilig sein, wenn während der Modellierung der Artefakte schnell ein Tracelink erstellt werden soll⁴⁹. Allerdings ist diese unstrukturierte Vorgehensweise ohne nachgelagerte Vollständigkeitsüberprüfung ohnehin nicht zu empfehlen, da nicht davon ausgegangen werden kann, dass tatsächlich alle relevanten Elemente in Kombination untersucht wurden. Das Ergebnis der EcoTracing-Anwendung ist hingegen eine vom Nutzer durchgeführte Abhängigkeitsanalyse, deren Vollständigkeit garantiert werden kann, ohne dass alle Elementekombinationen tatsächlich untersucht werden müssen.

7.2.5 VORENTWURF DES SYSTEMMODELLS

Im Anschluss an die Abhängigkeitsanalyse wurden für die zu realisierenden Funktionen Wirkprinzipien und Lösungselemente gewählt und als Verhaltensmodell in dem in CATIA integrierten DYMOLA-Editor modelliert. Insgesamt wurden sieben Module als Blöcke definiert: Spiegelglasverstellung, Einklappmechanismus, Speichermodul, Blinker-Modul, Toter-Winkel-Warner, Außenbeleuchtung sowie Spiegelglasbeheizung.

Diese Module wurden aufgrund klar definierter Schnittstellen durch mehrere Studenten gleichzeitig in Form von Modelica-Modellen entwickelt. Die in diesen Modellen enthaltenen Systemelemente wurden dabei mit Elementen aus der Modelica-Bibliothek abgebildet. In Abbildung 82 ist dies für die Spiegelglasbeheizung dargestellt. In diesem Modell steuern ein „timer“, mit dem die maximale Dauer der Beheizung definiert wird sowie ein Schalter, der mit „booleanExpression2“ abgebildet wurde, die Beheizung des Außenspiegels. Sind beide aktiv, wird die Heizspirale, repräsentiert durch einen Heizwiderstand, aktiviert. Das Spiegelglas wird durch die Kombination eines „heatCapacitor“ und eines „thermalConductors“ abgebildet.

Wie in Abbildung 82 zu erkennen, werden die verwendeten Objekte dabei auf einer Zeichenfläche durch Piktogramme dargestellt und mit Hilfe von Signal- und Energieflüssen miteinander verknüpft. Neben den genannten wurden zur Modellierung der anderen Module zahlreiche weitere Objekte verwendet, die hier jedoch nicht im Einzelnen vorgestellt werden. Für nähere Details, den Aufbau der Modelica-Modelle betreffend, sei an dieser Stelle auf [Amin et al. 2012, S. 37-46] verwiesen. Die Eigenschaften des Verhaltensmodells sind in Tabelle 33 zusammengefasst.

⁴⁹ Im Fall der Methode EcoTracing müsste dafür dann der Wizard gestartet, die entsprechenden Artefakte ausgewählt und das Element gesucht werden.

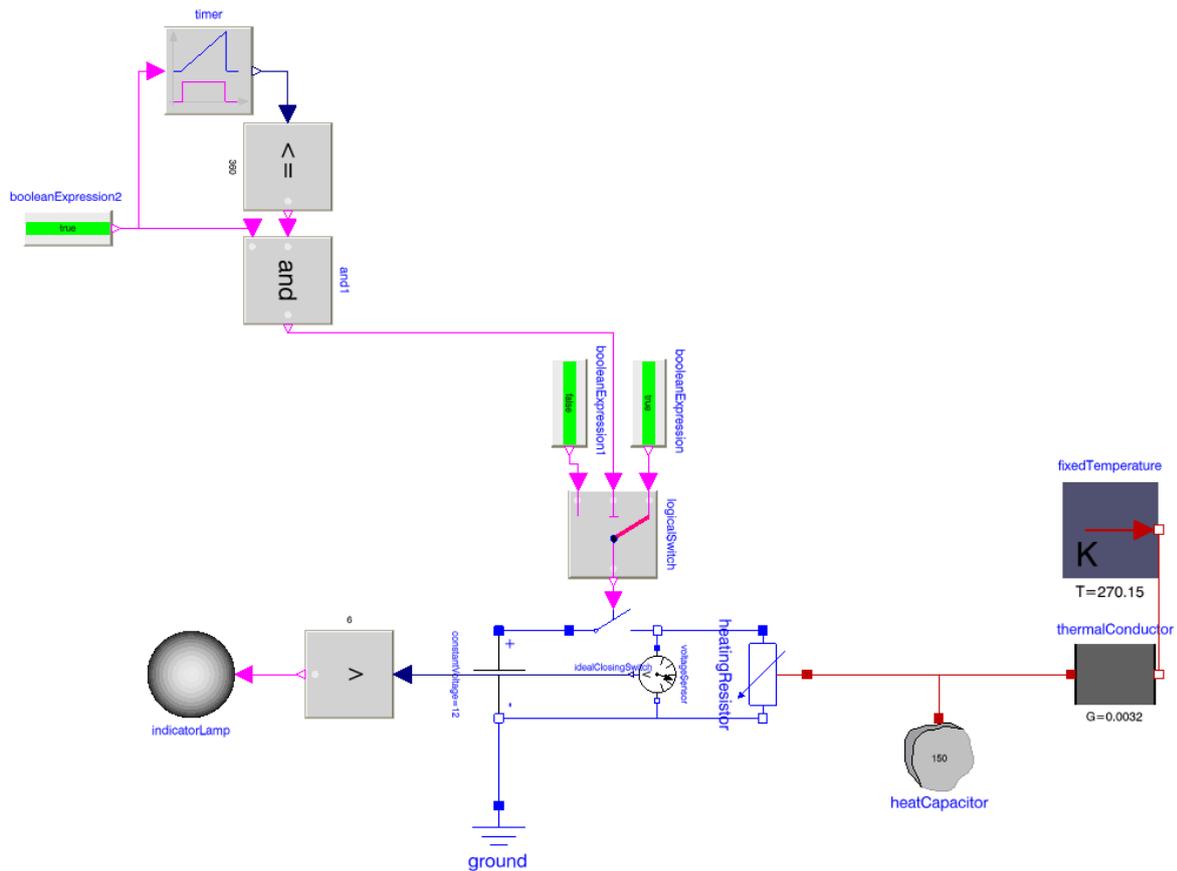


Abbildung 82: Modelica-Modell zur Simulation des Verhaltens der Spiegelglasbeheizung (vgl. [Amin et al. 2012])

Artefaktnamen	Side Mirror Logical
Strukturierungsart	Hierarchie
Anzahl Elemente	120
Anzahl Hierarchieebenen	2

Tabelle 33: Eigenschaften des Verhaltensmodells des mechatronischen Außenspiegels

7.2.6 ERFASSUNG VON TRACELINKS ZWISCHEN FUNKTIONEN UND VERHALTENSMODELL

Nachdem das Verhaltensmodell fertig modelliert war, wurde die durchgängige Nachverfolgbarkeit zu der Funktionsstruktur hergestellt, um abzubilden, welche Module an der Realisierung der Funktionen beteiligt sind. Aufgrund des geringen Detaillierungsgrades der Funktionsstruktur war es dabei ausreichend, die höchste Ebene des Verhaltensmodells in Kombination mit der Funktionsstruktur zu betrachten. Eine detailliertere Analyse wäre zwar möglich gewesen, allerdings hätten diese Tracelinks zwischen abstrakten Funktionen und detaillierten Systemelementen im Verhaltensmodell, wie durch Mäder et al. beschrieben [Mäder et al. 2009a, S. 4] und in Kapitel 5.2.1 diskutiert, keine höhere Aussagekraft als solche auf abstrakter Ebene.

Manuelle Modellierung von Tracelinks: Die Vorgehensweise der Studenten war bei der Abhängigkeitsanalyse die gleiche, wie sie bereits zwischen Anforderungs- und Funktionsartefakt angewendet wurde: durch sequenzielle Betrachtung aller Funktionen in Kombination mit den logischen Modulen und dem Anlegen von Implementierungsbeziehungen. Insgesamt mussten somit bei der Analyse 140 Entscheidungen durch die Studenten getroffen werden. U. a. wurde so ein Tracelink zwischen der Funktion 5.2 „Außenspiegel beheizen“ und dem Modul „Spiegelglasbeheizung“ modelliert.

Modellierung von Tracelinks mit EcoTracing: Wie schon bei der Analyse der Anforderungen in Kombination mit den Funktionen wäre auch in diesem Fall der Einsatz der Methode EcoTracing möglich gewesen, um die Vollständigkeit der Analyse zu garantieren. Die Anwendung wäre dabei analog zum zuvor beschriebenen Einsatz durchzuführen: die Funktionsstruktur und das Verhaltensmodell werden geladen und die Analyse begonnen. Nacheinander werden Elementepaare aus Funktionen und Modulen durch den Algorithmus hervorgehoben und müssen hinsichtlich ihrer Realisierungsbeziehung zueinander bewertet werden. Sobald die gewünschte Granularität der Tracelinks erreicht ist, wird die Analyse beendet. Insgesamt müssten 104 Entscheidungen getroffen werden.

Vergleich der beiden Vorgehensweisen: Aufgrund der flachen Hierarchie des Verhaltensartefakts, bei dem abgesehen von der höchsten hierarchischen Ebene alle weiteren Elemente bei der Analyse nicht berücksichtigt werden, lassen sich nicht dieselben Einsparungen erreichen wie bei der in Kapitel 7.2.4 beschriebenen Analyse. Dennoch ermöglicht die Modellierung der Tracelinks mit EcoTracing die schrittweise, geführte Abarbeitung aller relevanten Elementkombinationen der beiden Artefakte. Dabei ist die Verringerung der Anzahl notwendiger Entscheidungen um ca. 26 % möglich, weshalb die Anwendung der Methode im Fall der Analyse von Funktions- und Verhaltensartefakten sinnvoll ist.

7.2.7 INTEGRIERTE ABSICHERUNG

Die integrierte Absicherung wurde durch die Studenten auf Basis des Verhaltensmodells durchgeführt, um für jedes der Module möglichst frühzeitig dessen Eigenschaften abschätzen und mit den Anforderungen abgleichen zu können. So wurde bspw. für die in Kapitel 7.2.5 beschriebene Spiegelglasbeheizung das Aufheizverhalten des Spiegels simuliert. Die Simulation kam zu dem Ergebnis, dass eine Spiegeloberflächentemperatur von 15 °C nach 12 Sekunden erreicht war. Mit Hilfe der zuvor modellierten Tracelinks konnten anschließend leicht die relevanten Anforderungen bestimmt werden. Zu diesem Zweck wurde ausgehend vom Modul „Spiegelglasheizung“ die vorhandene Implementierungsbeziehung verfolgt und so die Funktion 5.2 „Außen-

spiegel beheizen“ identifiziert. Wiederum ausgehend von dieser Funktion wurden die Anforderungen „heating“, „heating_manual“ und „heating_automatic“ bestimmt und hinsichtlich der Spezifikation des Aufheizverhaltens überprüft. Dies wurde in der Anforderung „heating“ mit einer zu erreichenden Temperatur von 15 °C nach 20 Sekunden beschrieben, womit das modellierte Verhalten das geforderte übertraf.

Diese Absicherung wurde für die unterschiedlichen Module iterativ wiederholt, wenn enthaltene Komponenten geändert wurden und somit eine Änderung der Eigenschaften eines Moduls zu erwarten war.

7.2.8 ENTWICKLUNG MECHANISCHER KOMPONENTEN

Parallel zur Entwicklung des Systemmodells bzw. leicht zeitversetzt begann die Detailkonstruktion der Bauteile und Baugruppen in CATIA. Dazu wurde zunächst in ENOVIA eine Produktstruktur erstellt (Eigenschaften siehe Tabelle 34), in der die wichtigsten Baugruppen abgebildet wurden. In dieser Produktstruktur wurden kontinuierlich die Bauteile eingepflegt. Aus Sicht der durchgängigen Nachverfolgbarkeit ist vor allen Dingen diese Produktstruktur von Interesse, weshalb an dieser Stelle darauf verzichtet wird die Modellierung der Bauteile näher zu beschreiben. Für detaillierte Informationen, die Konstruktion der einzelnen Bauteile betreffend, sei auf den Projektbericht von Amin et al. verwiesen [Amin et al. 2012, S. 47-67].

Artefaktnamen	Side_Mirror_Physical
Strukturierungsart	Hierarchie
Anzahl Elemente	40
Anzahl Hierarchieebenen	3

Tabelle 34: Eigenschaften der Produktstruktur des mechatronischen Außenspiegels

7.2.9 ERFASSUNG VON TRACELINKS ZWISCHEN PRODUKTSTRUKTUR UND VERHALTENSMODELL SOWIE ANFORDERUNGEN

Im Anschluss an die Definition der Produktstruktur war es erneut notwendig, die durchgängige Nachverfolgbarkeit herzustellen. Im Gegensatz zu den in den Kapiteln 7.2.4 und 7.2.6 beschriebenen Analysen waren dabei zwei unterschiedliche Artefakte in Kombination mit der Produktstruktur auf Abhängigkeiten zueinander zu untersuchen. Zum einen wurden sog. „Ports“ verwendet, um das Verhaltensmodell und die 3D-Geometrie in der Produktstruktur zu verknüpfen. Hintergrund dieser Verknüpfungen war die Zuordnung von modellierten Systemelementen im Verhaltensmodell zu deren geometrischer Ausprägung, um die Geometrie auf Basis der Simulationen im Verhaltensmodell animieren zu können. So war es bspw. möglich, das Einklappen des Außenspiegels oder das Verstellen des Spiegelglases zu visualisieren und ggf.

Überschneidungen der Bauteile festzustellen. Ein Einsatz der Methode EcoTracing ist zur Erstellung dieser Art von Verknüpfungen nicht vorgesehen.

Manuelle Modellierung von Tracelinks zwischen Produktstruktur und Anforderungen:

Zum anderen wurden Tracelinks zwischen der Produktstruktur und den Anforderungen modelliert, da eine indirekte Verknüpfung über die Artefakte Funktionsstruktur und Verhalten bei nicht-funktionalen Anforderungen nicht immer möglich ist. Während bspw. die nicht-funktionale Anforderung, in der die maximale Heizdauer von 360 s spezifiziert wird, der Funktion „Außenspiegel beheizen“ zugeordnet werden kann, ist es nicht möglich, die Anforderung, die die maximale Masse des Außenspiegels vorgibt, mit einer Funktion (abgesehen von der Gesamtfunktion) zu verknüpfen. Bei der Analyse wurden somit nur nicht-funktionale Anforderungen berücksichtigt und auf Abhängigkeiten zu den Bauteilen und Baugruppen in der Produktstruktur untersucht. Zu diesem Zweck wurden auch hier die Verknüpfungen durch die Studenten mit Hilfe der Implementierungsbeziehung modelliert. Die Vorgehensweise war dabei analog zu den Analysen zuvor: Ausgehend von den Anforderungen, die nacheinander durchlaufen werden, wurde überprüft, welche der Bauteile an der Realisierung der jeweiligen Anforderung beteiligt sind. Lag eine Abhängigkeit vor, wurde die Modellierung über das Kontextmenü einer Implementierungsbeziehung gestartet, das Bauteil in der Produktstruktur ausgewählt und die Beziehung bestätigt. Insgesamt waren 480 Entscheidungen zu treffen.

Modellierung von Tracelinks zwischen Produktstruktur und Anforderungen mit EcoTracing:

Zur Modellierung der Tracelinks zwischen nicht-funktionalen Anforderungen und Bauteilen mit EcoTracing wurden zunächst die beiden Artefakte ausgewählt, beim Anforderungsartefakt die Vorauswahl auf nicht-funktionale Anforderungen eingeschränkt und anschließend die Analyse begonnen. Nacheinander wurden Elementepaare aus Anforderungen und Bauteilen hervorgehoben und es musste bewertet werden, ob eine Abhängigkeit besteht. Insgesamt müssten 300 Entscheidungen getroffen werden.

Vergleich der beiden Vorgehensweisen:

Auch bei der Modellierung von Tracelinks zwischen Anforderungen und Bauteilen in der Produktstruktur ist der Einsatz von EcoTracing sinnvoll, da durch die schrittweise Abarbeitung des Algorithmus ein Übersehen von Elementkombinationen ausgeschlossen wird. Zusätzlich können 37,5 % der Entscheidungen eingespart werden, da insgesamt 180 Entscheidungen weniger getroffen werden müssen.

7.2.10 DURCHFÜHRUNG EINER AUSWIRKUNGSANALYSE AUFGRUND EINER GEÄNDERTEN ANFORDERUNG

Zu jeder Zeit während des Entwicklungsprozesses können die bis dahin modellierten Tracelinks genutzt werden, um bspw. Auswirkungsanalysen durchzuführen. Dabei werden im Fall von Änderungen die entsprechenden Tracelinks nachverfolgt und es wird bewertet, welche der verknüpften Elemente von der Änderung betroffen sind. Im vorliegenden Fall wird diese Analyse mit dem Traceability-Viewer Ariadne's Eye durchgeführt. Im Folgenden wird beispielhaft eine solche Auswirkungsanalyse beschrieben und dabei zusätzlich auf die Anwendung der Methode TraceEvaluation eingegangen:

Nachdem die Funktionsstruktur und das Verhaltensmodell bereits fertiggestellt wurden und die Produktstruktur in ihren Grundzügen festgelegt ist, ergibt sich eine Änderung einer Anforderung. Erfahrungen des Kunden, für den der Außenspiegel entwickelt wird, belegen, dass die Zeit, nach der die Heizung im Spiegel automatisch deaktiviert wird, zu kurz gewählt wurde. Diese Zeit soll auf seinen Wunsch hin von 360 s auf 420 s erhöht werden, um das einwandfreie Abtauen von Schnee und Eis zu garantieren. Diese Änderung wird durch den Anforderungsmanager in den Eigenschaften der Anforderung „Heating_automatic“ vorgenommen.

Da dem Anforderungsmanager nicht klar ist, welche Funktionen und Bauteile von dieser Änderung betroffen sind, entschließt er sich, eine Auswirkungsanalyse durchzuführen. Zu diesem Zweck startet er Ariadne's Eye und verfolgt, ausgehend von der geänderten Anforderung, die Tracelinks zu den Funktionen und Bauteilen. Dabei identifiziert er folgende, potenziell betroffene Funktionen und Bauteile:

- Funktion: Nutzereingaben Erfassen (heating_switch),
- Funktion: Außenspiegel kontrollieren (side_mirror_control_module),
- Funktion: Spiegelglasbeheizung (heating_mirror),
- Bauteil: Heizspirale (heating).

Außerdem fällt ihm auf, dass die betrachtete Anforderung, neben den zuvor genannten Verknüpfungen, auch einen Tracelink zu dem Bauteil „Connector Folding Modul“ aufweist, den er nicht erwartet hätte. Er wählt diesen Link in Ariadne's Eye aus, ruft das Kontextmenü auf und markiert ihn für eine spätere Überprüfung (siehe Abbildung 83). Im Anschluss fährt er mit seiner Auswirkungsanalyse fort und setzt sich mit seinen Kollegen aus der Detailkonstruktion sowie mit dem, für die Funktionsstruktur verantwortlichen, Systemarchitekten in Verbindung.

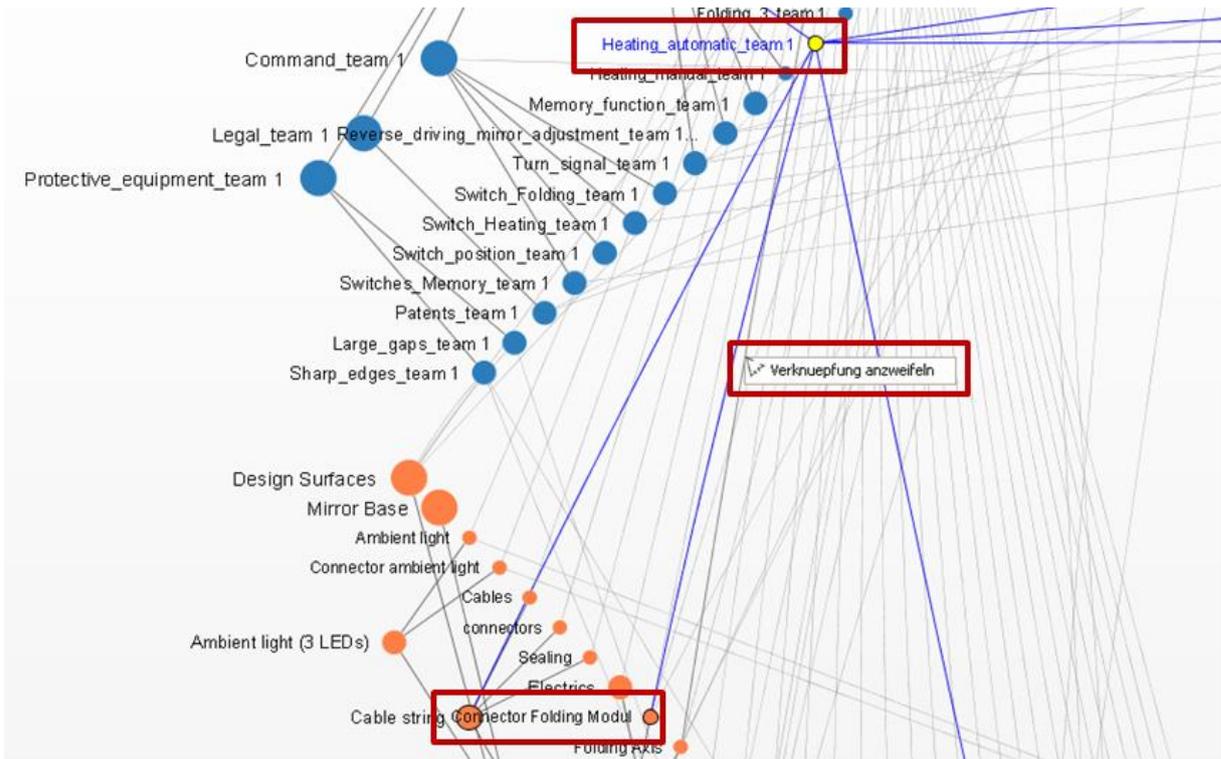


Abbildung 83: Impact Analyse in Ariadne's Eye & Anwendung der Methode TraceEvaluation

7.2.11 DURCHFÜHRUNG EINER QUALITÄTSKONTROLLE DES TRACELINK-MODELLS

Zu einem späteren Zeitpunkt führt der Traceability-Verantwortliche eine Qualitätskontrolle für das aktuelle Außenspiegel-Entwicklungsprojekt durch. Dazu startet er das Auswertungsmodul im ModelTracer und wählt zunächst den Reiter „Bewertungen“ aus (siehe Abbildung 84). In diesem laufen alle Meldungen über potenziell falsche Tracelinks zusammen, so dass sie nacheinander abgearbeitet werden können. Im vorliegenden Fall hat seit der letzten Kontrolle nur der Anforderungsmanager einen potenziell falschen Tracelink gemeldet (siehe Kapitel 7.2.10). Der Traceability-Verantwortliche erhält die Informationen über die Start- und Zielelemente sowie die Anzahl der Meldungen, die gleichzeitig als Ranking genutzt werden. Nach kurzer Überprüfung der Baugruppe in CATIA V6 entscheidet er, dass der Tracelink fälschlicherweise modelliert wurde und löscht ihn aus der Datenbank des ModelTracers.

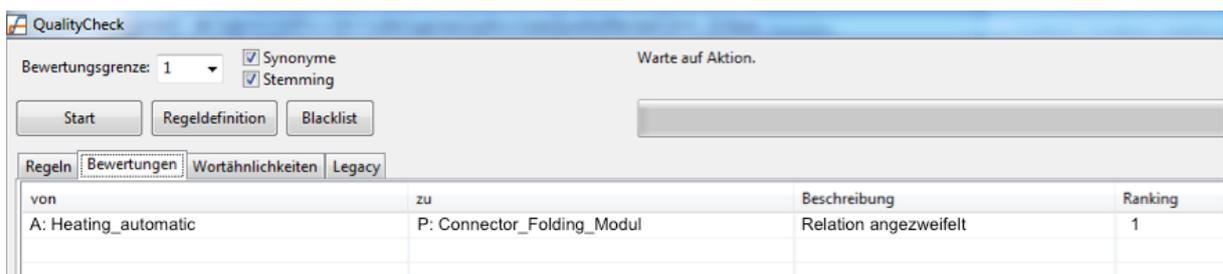


Abbildung 84: Auswertungsmodul der Methode TraceEvaluation

Im Anschluss an die Auswertung der Methode TraceEvaluation entschließt sich der Traceability-Verantwortliche eine weitere Maßnahme durchzuführen. Er wählt zu diesem Zweck im Qualitätssicherungsmodul des ModelTracers den Reiter Legacy aus und konfiguriert die Einstellungen der Methode TraceLegacy. Zunächst editiert er durch Klick auf den Button „Blacklist“ die Negativliste, in der Begriffe dokumentiert sind, die während der Analyse ignoriert werden. Da alle Elemente zur leichteren Identifikation im PDM-System die Endung des Entwicklungsteams „Team 1“ tragen, fügt er dieser Liste den Begriff „team 1“ hinzu, da sonst u. U. irrelevante Ähnlichkeiten identifiziert werden würden. Bevor er die Analyse startet, aktiviert er die Stammformrückführungsoption, mit deren Hilfe die Worte der Elementbezeichner auf ihren Wortstamm zurückgeführt werden. Das Ergebnis der Analyse ist in Abbildung 85 dargestellt.

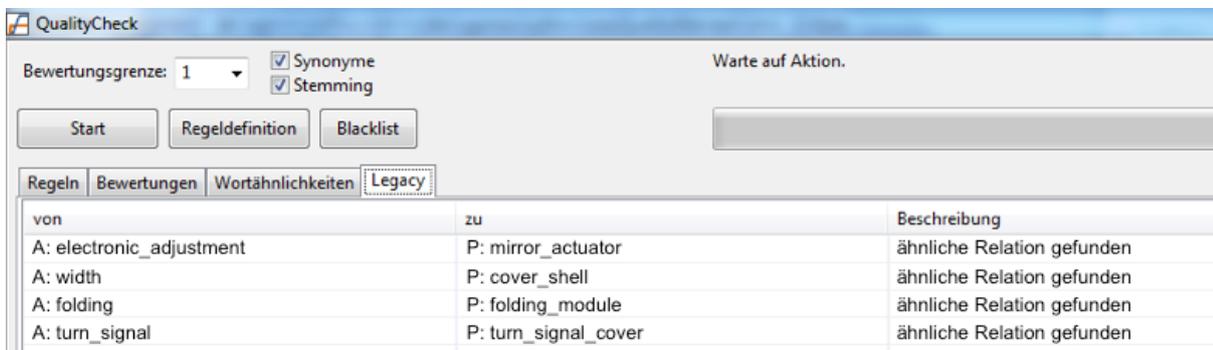


Abbildung 85: Ergebnis der Analyse mittels TraceLegacy

Da es sich nicht um die erste Außenspiegel-Entwicklung seiner Firma handelt, sondern bereits mehrere Entwicklungen im selben Umfeld durchgeführt wurden, greift das System auf einen großen Wissensschatz, der mittels Tracelinks dokumentiert wurde, zurück. Insgesamt werden ihm durch das System sieben Vorschläge für fehlende Tracelinks angeboten. Diese Vorschläge wurden aufgrund ähnlich benannter Anforderungen bzw. Bauteile in vergangenen Projekten ausgegeben. Nach Überprüfung stellt der Traceability-Verantwortliche fest, dass die folgenden vier Vorschläge korrekt sind:

- Anforderung: electronic_adjustment | Baugruppe: mirror_actuator,
- Anforderung: width | Bauteil: cover_shell,
- Anforderung: folding | Baugruppe: folding_module,
- Anforderung: turn_signal | Bauteil: turn_signal_cover.

Er wählt die Vorschläge nacheinander aus, bestätigt die Abhängigkeiten zwischen den jeweiligen Elementen und modelliert somit direkt vier Tracelinks, die im Tracelink-Modell bisher gefehlt haben.

Mit Hilfe der beiden Methoden TraceEvaluation und TraceLegacy hat der Traceability-Verantwortliche somit einen falschen Tracelink und vier Abhängigkeiten, bei denen vergessen wurde, einen Tracelink zu modellieren, identifizieren können. Während die Beseitigung aller Fehler wichtig ist, tragen insbesondere die durch ihn hinzugefügten vier Tracelinks zur Verbesserung des Tracelink-Modells bei, da somit bspw. bei zukünftigen Auswirkungsanalysen diese Abhängigkeiten berücksichtigt werden können.

7.3 ZUSAMMENFASSUNG

In Kapitel 7 wurde die Anwendung der in dieser Arbeit entwickelten Methoden EcoTracing, TraceEvaluation und TraceLegacy während der Entwicklung eines technischen Systems beschrieben. Dazu wurde zunächst in Kapitel 7.1 anhand des Fraunhofer IPK V-Modells aufgezeigt, wann die Methoden im Entwicklungsprozess eingesetzt werden können. Anschließend wurden die Methoden am Beispiel der Entwicklung eines mechatronischen Außenspiegels demonstriert. Ziel war es, den Einsatz der Methoden durch Entwickler und die sich daraus ergebenden Vorteile zu beschreiben. So lässt sich EcoTracing zur strukturierten Abhängigkeitsanalyse der meisten entstehenden Artefakte einsetzen und durch die Anwendung der Methode die Berücksichtigung aller Elemente garantieren. Darüber hinaus konnte im dargestellten Beispiel die Anzahl teilweise schwer zu treffender Entscheidungen durch den EcoTracing-Algorithmus um 26 % - 59 % reduziert werden (siehe Abbildung 86).

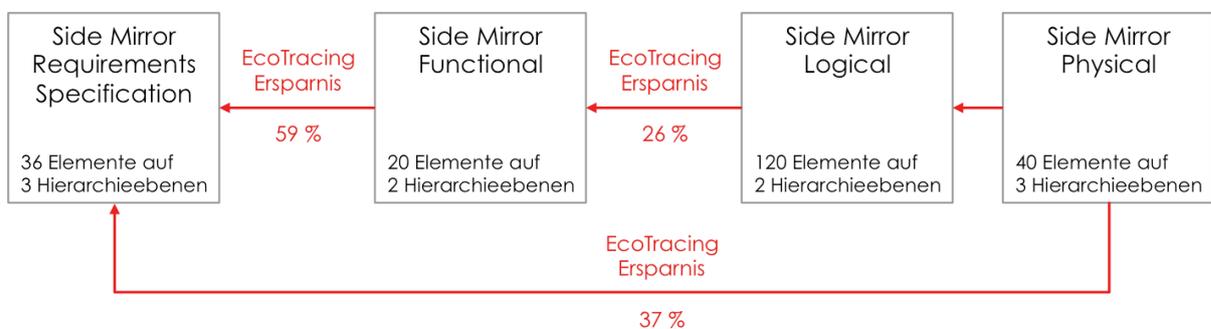


Abbildung 86: Übersicht der erstellten Artefakte, modellierter Tracelinks (rote Pfeile) sowie der erreichten Ersparnis durch EcoTracing (wenn angewendet)

Hinsichtlich der intuitiven Erfassung von Tracelinks kann der Methode EcoTracing ihre Konzeption als Offline-Methode negativ ausgelegt werden. Dabei werden Tracelinks nicht bei Erkennen einer Abhängigkeit während der Erstellung der Artefakte, sondern in einem dedizierten Arbeitsschritt erfasst. Während eine Funktionalität zur schnellen Online-Modellierung von Tracelinks aus Sicht des Autors dringend erforderlich ist, da die Artefakte aufgrund bestimmter Abhängigkeiten modelliert werden (bspw. wurde die Funktion „Spiegelglasbeheizung“ nur modelliert, weil es eine entsprechende Anforderung in der Anforderungsspezifikation gab), die direkt dokumentiert werden sollten, kann aufgrund dieser unstrukturierten Vorgehensweise nicht davon ausgegan-

gen werden, dass tatsächlich alle relevanten Elementepaare untersucht wurden. Genau hier liegen die Vorteile der Offline-Erfassung mit EcoTracing, bei der strukturiert alle Elementekombinationen überprüft werden und eine Vollständigkeit der Analyse garantiert werden kann. Die Kombination einer Online- mit einer nachgelagerten Offline-Erfassung ist somit empfehlenswert.

Der Einsatz der Methode TraceEvaluation wurde exemplarisch anhand einer Auswirkungsanalyse beschrieben, die aufgrund einer geänderten Anforderung durchgeführt wurde (siehe Kapitel 7.2.10). Dabei wurde ein falsch positiver Tracelink durch einen Anforderungsmanager bei Durchführung seiner Analyse identifiziert, gemeldet und später vom Traceability-Verantwortlichen gelöscht (siehe Kapitel 7.2.11). Es wurde deutlich, dass die Methode TraceEvaluation in Methoden, die aktuell verwendet werden, integriert werden muss, um eine große Reichweite und damit Wirksamkeit zu erreichen.

Als weitere Qualitätssicherungsmaßnahme wurde die Methode TraceLegacy auf die Beispieldaten angewendet. Dabei wurden vier Abhängigkeiten von Elementen des aktuellen Projekts auf Basis modellierter Tracelinks aus einem vergangenen Projekt identifiziert und anschließend durch Tracelinks dokumentiert. Die Identifikation dieser falsch negativen Tracelinks ist insofern wichtig, da deren Fehlen eine fehlerhafte Auswirkungsanalyse und damit die Nichtberücksichtigung von Auswirkungen einer Änderung zur Folge haben kann.

8 ZUSAMMENFASSUNG UND AUSBLICK

Die durchgängige Nachverfolgbarkeit ist eine aus der Informatik stammende Methode, bei der Abhängigkeiten zwischen unterschiedlichen Artefakten der Entwicklung mit sog. Tracelinks dokumentiert werden. Die vorliegende Arbeit widmet sich dem Thema „durchgängige Nachverfolgbarkeit“ aus Sicht der Entwicklung mechatronischer Systeme. Um diesen Kontext zu beschreiben, wurden in Kapitel 2 fünf Vorgehensmodelle sowie die für die Analyse- und Syntheseschritte verwendeten Modelle und Dokumente analysiert und diskutiert. Vor diesem Hintergrund wurden in Kapitel 3 die Grundlagen durchgängiger Nachverfolgbarkeit vorgestellt, die Vor- und Nachteile beschrieben und herausgearbeitet, dass sich nicht alle aus der Softwareentwicklung bekannten Prinzipien auf den erweiterten Kontext der Entwicklung mechatronischer Systeme anwenden lassen. Insbesondere zur Erfassung und Pflege von Tracelinks existieren bislang nur wenige Methoden, die eine Unterstützung für diese essentiellen Phasen bieten, was massiv zum schlechten (wahrgenommenen) Aufwand/Nutzen-Verhältnis der durchgängigen Nachverfolgbarkeit beiträgt und daher eine große Hürde für deren industrielle Nutzung darstellt. Aus diesem Grund beschränkt sich die Anwendung durchgängiger Nachverfolgbarkeit aktuell hauptsächlich auf Bereiche, in denen sie explizit durch die Gesetzgebung gefordert wird.

Um eine größere industrielle Verbreitung der durchgängigen Nachverfolgbarkeit und der damit verbundenen Vorteile für die Entwicklung mechatronischer Systeme zu erreichen, ist es notwendig, unterstützende Methoden für eben diese Phasen der Erstel-

lung sowie Pflege des Tracelink-Modells zur Verfügung zu stellen. Die Forderung nach einer Reduktion des für die durchgängige Nachverfolgbarkeit notwendigen Aufwands stellt den Ausgangspunkt für die drei, in dieser Arbeit behandelten, Forschungsfragen dar.

Zur Beantwortung dieser Forschungsfragen wurden insgesamt drei Methoden entwickelt und evaluiert. Die erarbeiteten Ergebnisse werden im folgenden Kapitel 8.1 zusammengefasst und ein Ausblick auf notwendige Weiterentwicklungen für einen industriellen Einsatz der Methoden gegeben. Im anschließenden Kapitel 8.2 werden Einsatzmöglichkeiten aufgezeigt und die damit verbundenen Auswirkungen auf die Integration der Methoden in die IT-Architektur eines Unternehmens diskutiert. Abschließend werden in Kapitel 8.3 unterschiedliche, unbeantwortete Forschungsaspekte aus dem Themengebiet der durchgängigen Nachverfolgbarkeit zusammengefasst, die während der Bearbeitung der vorliegenden Arbeit identifiziert wurden, die weitere Forschungsrichtungen erschließen.

8.1 ZUSAMMENFASSUNG DER ERGEBNISSE DIESER ARBEIT

Im Rahmen der vorliegenden Arbeit wurden die drei Methoden EcoTracing, TraceEvaluation und TraceLegacy entwickelt, um Forschungsfragen aus den Bereichen der Erstellung und Pflege von Tracelink-Modellen zu beantworten. In den folgenden drei Kapiteln 8.1.1 bis 8.1.3 werden die Entwicklung der Methoden sowie ihre Evaluation zusammenfassend beschrieben und die zugehörigen Forschungsfragen beantwortet. Darüber hinaus wird ein Ausblick auf Weiterentwicklungen gegeben, die für die Industrialisierung der Methoden notwendig wären.

8.1.1 ECOTRACING

In Kapitel 5 wurde zunächst die Modellierung von Tracelinks betrachtet und die Forschungsfrage untersucht, ob eine methodische Unterstützung des Entwicklers helfen kann, den Aufwand bei der Modellierung von Tracelinks zu verringern. Während vergleichbare Methoden des Stands der Technik zusätzliche Bearbeitungsschritte benötigen, nutzt der Algorithmus, der in diesem Zusammenhang entwickelten Methode *EcoTracing*, den hierarchischen Aufbau der systembeschreibenden Artefakte, um die Anwender strukturiert durch Abhängigkeitsanalysen zu führen und dabei Aufwände einzusparen. Bei der, zum Zweck der Evaluation der Leistungsfähigkeit der Methode *EcoTracing*, durchgeführten Studie konnte im arithmetischen Mittel eine Aufwandsersparnis von ca. 60 % erreicht werden. Die Forschungsfrage kann somit positiv beantwortet werden: der Modellierungsaufwand lässt sich durch methodische Unterstützung, welche die Eigenschaften kompositioneller Hierarchien ausnutzt, redu-

zieren. Allerdings wurde als Einschränkung festgestellt, dass es für den Einsatz der Methode erforderlich ist, dass der Anwender über eine umfassende Kenntnis der zu analysierenden Artefakte und deren Aufbau verfügt, da er sonst nicht in der Lage ist, die notwendigen Entscheidungen auf hoher Abstraktionsebene zu fällen.

Um die Methode EcoTracing im industriellen Umfeld einsetzen zu können, sind noch weiterführende Implementierungs- und Evaluationsaufgaben im Anschluss an diese Arbeit zu leisten. Zunächst sollte eine Umsetzung in Rational DOORS verfolgt werden (siehe Kapitel 3.6.1.1), da es sich hierbei um eine weit verbreitete Software zur Verwaltung von Anforderungen handelt, zwischen denen die durchgängige Nachverfolgbarkeit bereits heute vielfach manuell etabliert wird. Darüber hinaus wird DOORS während der Entwicklung von Embedded Systems teilweise zweckentfremdet, um neben Anforderungsspezifikationen auch Funktionshierarchien zu verwalten (siehe dazu auch Kapitel 5.1 und 5.2.3). Es wäre somit möglich, zwei unterschiedliche, bereits heute existierende Anwendungsfälle (prozessphasen-interne und -externe Tracelinks) abzudecken. Diese Umsetzung sollte anschließend genutzt werden, um eine Evaluation der Methode mit Datenumfängen industriellen Ausmaßes (hinsichtlich der Anzahl an Elementen und Hierarchieebenen) durchzuführen. Dabei ist zu berücksichtigen, dass möglichst die Entwickler selbst die Abhängigkeitsanalyse durchführen, um die zuvor erwähnte notwendige Kenntnis der Artefakte voraussetzen zu können. Sollte diese Evaluation positiv verlaufen und Interesse seitens der Industrie vorliegen, wäre im nächsten Schritt die Implementierung der Methode in einem PLM-System (bspw. Teamcenter) durchzuführen, um möglichst alle Artefakte, die während der Entwicklung erstellt werden, mit EcoTracing analysieren zu können. In Abhängigkeit der geplanten Verwendung der Tracelinks und des verwendeten PLM-Systems wäre es dabei sinnvoll, die Methode EcoTracing um die Funktionalität, typisierte Tracelinks modellieren zu können, zu erweitern.

8.1.2 TRACEEVALUATION

Einmal erstellt, müssen die Tracelink-Modelle kontinuierlich gepflegt werden, um ihre Qualität und damit ihre Einsetzbarkeit bspw. für Auswirkungsanalysen sicherzustellen. Da die Aufwände auch hierfür als erheblich einzuschätzen sind, wurden unterschiedliche Methoden im Bereich der Unterstützung der Qualitätssicherung von Tracelink-Modellen erforscht. Zunächst wurde untersucht, ob sich Crowdsourcing (deutsch: Schwarmauslagerung), als relativ neue Strategie einer Vielzahl von Diensten im Internet, für die Qualitätssicherung von Tracelink-Modellen eignet. Denn obwohl das Crowdsourcing eine immer größere Verbreitung findet, wurde dessen Einsatz im Kontext der durchgängigen Nachverfolgbarkeit bisher weder erforscht noch kommerziell realisiert. Die zur Überprüfung dieser Frage entwickelte Methode *TraceEvaluation*

stellt Experten während der Bearbeitung ihrer Aufgaben im Tagesgeschäft die Möglichkeit zur Verfügung, fehlerhafte Tracelinks für eine spätere Überprüfung zu melden. Durch dieses Verfahren lässt sich das Feedback vieler Nutzer schnell sammeln und zentral leicht auswerten. Auch diese Methode wurde im Rahmen einer Studie evaluiert. Die dabei erreichte Trefferquote von 100 % sowie die Präzision von 88 % sind als exzellent zu bewerten. Hinsichtlich der gestellten Forschungsfrage lässt sich somit aussagen, dass sich Methoden des Crowdsourcings sehr gut zur Steigerung Tracelink-Modell-Qualität eignen. Es zeigte sich jedoch auch, dass keiner der Probanden allein in der Lage war, alle falschen Tracelinks zu identifizieren. Mehrere hingegen schon, weshalb die Methode in industrieller Anwendung einer großen Anzahl an Nutzern zur Verfügung gestellt werden sollte.

Für die Industrialisierung der Methode TraceEvaluation muss zunächst ein Unternehmen gefunden werden, das bereits heute zumindest die durchgängige Nachverfolgbarkeit der Anforderungen etabliert. Bei der Wahl sollte darauf geachtet werden, dass dies möglichst nicht nur zu Dokumentationszwecken geschieht, sondern die Tracelinks darüber hinaus für weitere Verwendungszwecke (bspw. Auswirkungenanalysen) genutzt werden. Hintergrund dieser Forderung ist, dass erst bei der tatsächlichen Nutzung falsch modellierte Tracelinks auffallen und somit auch erst dann gemeldet werden können. Bei diesem Anwender sollte evaluiert werden, welche Software-Werkzeuge die Tracelinks verwenden und wo diese verwaltet werden. Diese Erfassung stellt dann die Basis für die Integration des Anwender- und des Auswertungsmoduls entsprechend der Beschreibung in Kapitel 6.3.2 dar. Abschließend ist es notwendig, die Entwickler aufzufordern, zur Sicherung der Qualität des Tracelink-Modells beizutragen und ggf. ein entsprechendes Entlohnungssystem (siehe Kapitel 6.3.1) zu entwickeln.

8.1.3 TRACELEGACY

Ebenfalls aus dem Bereich der Unterstützung der Qualitätssicherung von Tracelink-Modellen stammt die dritte betrachtete Forschungsfrage. Sie hinterfragt, ob sich Tracelinks aus vergangenen Entwicklungsprojekten als Wissensquelle für die Sicherung der Tracelink-Modellqualität aktueller Projekte nutzen lassen. Zur Untersuchung dieser Frage wurde die Methode *TraceLegacy* entwickelt, die Entwicklungswissen vergangener Projekte nutzt, um Vorschläge für fehlende Tracelinks zu generieren. Die Evaluation, die anhand zweier Studentenprojekte durchgeführt wurde, ergab Trefferquoten von 4 % bis 11 % sowie Präzisions-Werte von 61 % bis 90 %. Die grundsätzliche Funktionsfähigkeit der Methode wurde nachgewiesen und damit die Forschungsfrage positiv beantwortet. Aus den Tracelinks vergangener Projekte können zur Steigerung der Tracelink-Modell-Qualität sinnvolle Vorschläge für Tracelinks aktuel-

ler Projekte berechnet werden. Es konnten zudem Vorschläge für Tracelinks generiert werden, die mit den Ansätzen aus dem Stand der Technik nicht zu identifizieren gewesen wären. Anzumerken bleibt, dass insbesondere hinsichtlich der erreichbaren Präzision ein großes Potenzial für Verbesserung besteht. Da im Rahmen dieser Arbeit die Überprüfung der Machbarkeit und nicht die Optimierung der Methode im Vordergrund stand, wurde ein vergleichsweise einfacher Algorithmus zur Bestimmung der Ähnlichkeit der Elemente verwendet (Stringvergleich stammrückgeführter Elementbezeichner), auf den die schlechten Präzisions-Werte zurückzuführen sind.

Erster Schritt auf dem Weg zur Industrialisierung der Methode TraceLegacy muss es somit sein, den Algorithmus gegen mächtigere aus dem Bereich der Informationsrückführung auszutauschen und somit die Trefferquote der Methode zu steigern. Denkbar wäre es in diesem Zusammenhang, ähnlich der Realisierung in der Software ConWeaver (siehe Kapitel 3.6.4.2), mehrere unterschiedliche Algorithmen zu implementieren. Aus diesen könnten, je nach Bedürfnissen der Anwender bzw. in Abhängigkeit der Art und des Aufbaus der zu analysierenden Artefakte, passende Algorithmen ausgewählt und die Methode TraceLegacy somit individuell konfiguriert werden. Anschließend sollte die Integration in die Software, die für die Verwaltung der Tracelinks verwendet wird, erfolgen.

8.1.4 FAZIT

Im Rahmen der vorliegenden Arbeit wurden die drei Methoden EcoTracing, TraceEvaluation und TraceLegacy konzipiert, prototypisch implementiert und evaluiert, um Forschungsfragen aus den Bereichen der Erfassung von Tracelinks sowie der Pflege von Tracelink-Modellen beantworten zu können. Übergeordnetes Ziel war es in diesem Zusammenhang, Entwickler bei der oft manuellen Modellierung von Tracelinks und dem Suchen nach Fehlern im Tracelink-Modell zu unterstützen und, wenn möglich, die Aufwände für diese Tätigkeiten im Vergleich zur gelebten Praxis zu reduzieren. Dieses Ziel wurde erreicht, da mit EcoTracing eine Reduzierung des notwendigen Modellierungsaufwands erreicht und mit TraceEvaluation sowie TraceLegacy falsche bzw. fehlende Tracelinks identifiziert werden können. Dabei wurden, wie in den vorangegangenen Kapiteln 8.1.1 bis 8.1.3 beschrieben, unterschiedliche Stadien der Anwendungsreife erreicht. Während TraceEvaluation, abgesehen von der optionalen Entwicklung eines Entlohnungssystems, ohne Anpassungen in kommerzielle Softwareprodukte integriert werden könnte, sind sowohl bei EcoTracing als auch TraceLegacy noch weiterführende Evaluations- und Implementierungsschritte durchzuführen.

Für alle drei Methoden gilt dabei gleichermaßen, dass sie lediglich Teilaspekte der in Kapitel 3.7 beschriebenen Herausforderungen adressieren und damit drei Puzzletei-

len entsprechen, die einen Beitrag zur Steigerung der Effizienz und Qualität durchgängiger Nachverfolgbarkeit leisten können [Aizenbud-Reshef et al. 2006, S. 523]. Um jedoch eine umfassende Lösung bereitzustellen, ist es notwendig, EcoTracing, TraceEvaluation und TraceLegacy zielgerichtet mit anderen Methoden zu kombinieren. Neben Methoden aus dem Bereich der Erstellung und Pflege von Tracelink-Modellen sollten dabei insbesondere Methoden zur Verwendung der modellierten Tracelinks berücksichtigt werden, um die vielfältigen Anwendungsmöglichkeiten der durchgängigen Nachverfolgbarkeit auszunutzen und somit das Aufwand/Nutzen-Verhältnis positiv zu beeinflussen.

8.2 EINSATZMÖGLICHKEITEN DER METHODEN UND DEREN AUSWIRKUNGEN AUF DIE IT-ARCHITEKTUR

Für die im Rahmen dieser Arbeit entwickelten Methoden ergeben sich während des Entwicklungsprozesses mehrere Einsatzmöglichkeiten (siehe auch Kapitel 7.1 und Abbildung 76).

So kann die Methode EcoTracing in Abhängigkeit der Tracelink-Modellierungsstrategie während bzw. im Anschluss an die Modellierung von Artefakten (je nachdem, ob Zwischenstände oder finale Artefakte analysiert werden) eingesetzt werden. Die Wahl der Modellierungsstrategie kann dabei auch Auswirkungen auf das IT-Architekturkonzept haben (siehe Abbildung 87). Werden die Tracelinks während der Erstellung der Artefakte durch die Entwickler modelliert, so ist die Integration auf Ebene der Autoren-Systeme sinnvoll, um die Methode direkt in der gewohnten Entwicklungsumgebung bereitzustellen. Ausgehend von dem jeweiligen Artefakt könnten die Entwickler den EcoTracing-Wizard starten, ein weiteres Artefakt, welches über den PLM-Backbone bereitgestellt wird, auswählen und mit der Verknüpfung der beiden Artefakte beginnen. Findet die Modellierung der Tracelinks hingegen nach Fertigstellung der Artefakte bspw. durch den Traceability-Verantwortlichen oder Systemarchitekten statt, so kann EcoTracing auch direkt in das PLM-System integriert werden (siehe Abbildung 87, PLM-Backbone).

TraceLegacy wurde hingegen als qualitätssichernde Methode konzipiert, die nicht von allen Entwicklern, sondern dem Traceability-Verantwortlichen genutzt wird, um fehlende Tracelinks auf Basis vergangener Projekte zu identifizieren. Damit ist eine Integration der Methode in unterschiedliche Autoren-Systeme nicht notwendig, sondern im Tracelink-verwaltenden System ausreichend. In Abbildung 87 wird dies durch den PLM-Backbone realisiert, der die Daten disziplinübergreifend verwaltet⁵⁰. Hin-

⁵⁰ Alternativ wäre auf gleicher Ebene eine PLM-unabhängige Umsetzung bspw. mit Hilfe des ModelTracers (siehe Kapitel 3.6.4.3) denkbar.

sichtlich der zeitlichen Einordnung der Einsatzmöglichkeiten ist eine Kopplung an die Modellierung von Tracelinks (und damit an den Einsatz der Methode EcoTracing) sinnvoll, da so direkt überprüft werden kann, ob Abhängigkeiten, die in vergangenen Projekten bereits mit einem Tracelink bestätigt wurden, übersehen worden sind. Grundsätzlich lässt sich die Methode jedoch zu jedem Zeitpunkt im Entwicklungsprozess anwenden.

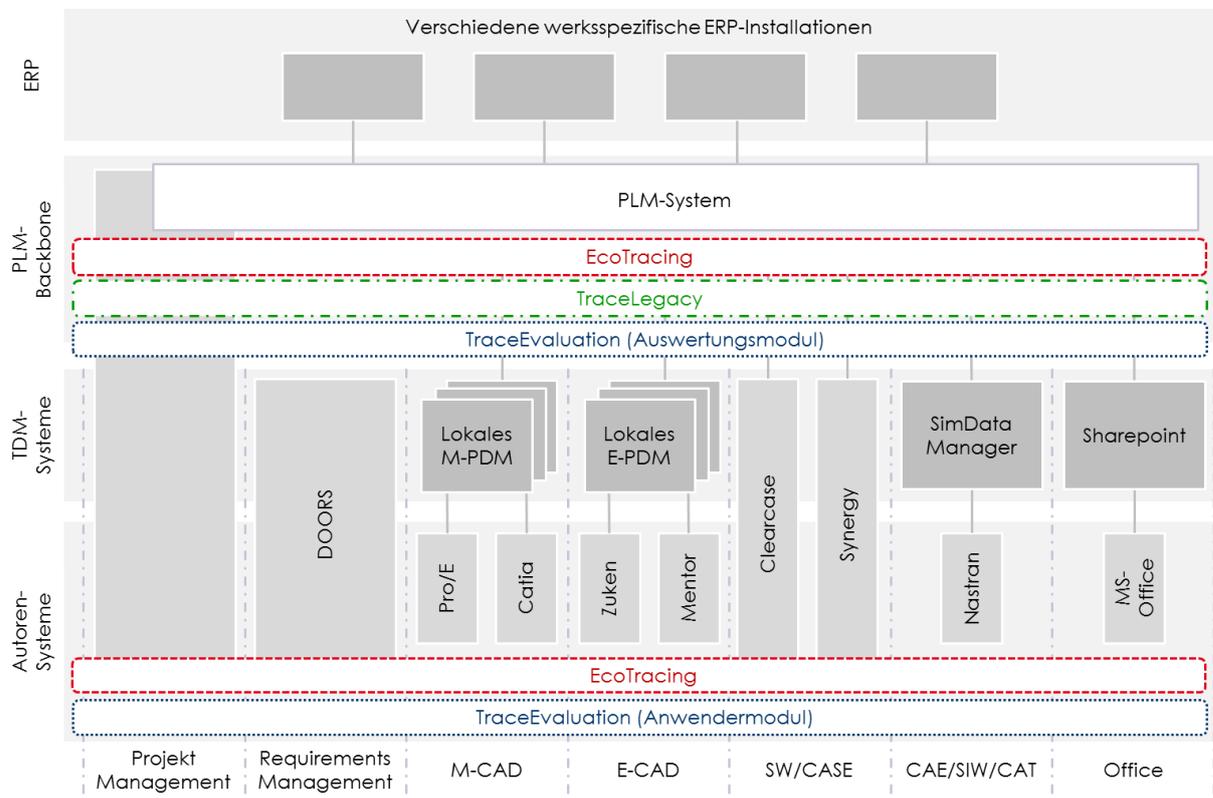


Abbildung 87: Einordnung der Methoden EcoTracing, TraceEvaluation und TraceLegacy in einem beispielhaften vierstufigen IT-Architekturkonzept (in Anlehnung an [Eigner und Stelzer 2009, S. 43])

Die Methode TraceEvaluation wird im Gegensatz dazu überwiegend während der Modellierung der Artefakte und damit in den Prozessschritten angewendet, um das Feedback der Entwickler zu erfassen (siehe Abbildung 76). Damit muss das Anwendermodul der zweiteiligen Methode auf Autoren-System-Ebene in solche Werkzeuge integriert werden, in denen Tracelinks verwendet werden (siehe Abbildung 87). Dies ist insbesondere bei der integrierten Funktions- und Eigenschaftsabsicherung sowie bei der Eigenschaftsbestätigung der Fall, da bei diesen Aktivitäten die Tracelinks verwendet werden, um aktuelle IST- mit SOLL-Parametern bspw. der Anforderungen abzugleichen, wobei falsche Tracelinks besonders auffallen. Die Implementierung sollte dabei vorzugsweise so umgesetzt werden, dass lediglich ein Rechtsklick auf den entsprechenden Tracelink und eine Bestätigung zu erfolgen haben, um den Aufwand für den Anwender so gering wie möglich zu halten. Das Auswertungsmodul, in dem das Feedback gesammelt und bewertet wird, ist hingegen nur für die Ver-

wendung durch den Traceability-Verantwortlichen vorgesehen. Es kann somit auf PLM-Backbone-Ebene implementiert und zu jeder Zeit im Entwicklungsprozess genutzt werden, um zu überprüfen, ob potenziell falsche Tracelinks gemeldet wurden.

8.3 AUSBLICK

Neben den in Kapitel 8.1 beschriebenen konkreten Weiterentwicklungsansätzen, die verfolgt werden sollten, um eine industrielle Anwendbarkeit der Methoden zu erreichen, gilt es, einige grundsätzliche Forschungsaspekte zu klären, um die Verbreitung durchgängiger Nachverfolgbarkeit zu steigern. Diese wurden insb. bei der Durchführung der Studien und damit bei der Interaktion mit potenziellen Anwendern identifiziert und werden im Folgenden kurz zusammengefasst.

Mittelfristig erscheint die vollständig automatisierte Erfassung von Tracelinks mit Hilfe konfigurierbarer Regeln und Methoden der Informationsrückführung aus Sicht des Autors als unmöglich. Aus diesem Grund muss die Entscheidung über die Modellierung eines Tracelinks letztlich immer beim Anwender liegen, da die Vollständigkeit einer Analyse sonst nicht vorausgesetzt werden kann. Allerdings sind auch Anwender natürlich nicht unfehlbar, weshalb sich die Traceability-Forschung u. a. auf deren Unterstützung fokussieren sollte.

So ist die Entscheidung, ob eine Abhängigkeit zwischen zwei Elementen besteht und somit ein Tracelink modelliert wird, eine sehr subjektive, die von der Erfahrung des Modellierenden, der Strukturierung der Artefakte sowie der Beschreibung der Elemente abhängig und häufig nicht eindeutig ist. Die Frage ist in diesem Zusammenhang, wie Entwickler als Anwender unterschiedlicher Methoden zur Erfassung von Tracelinks bei dem Treffen der einzelnen Entscheidung besser unterstützt werden können.

Als ein Ansatz könnte bspw. die Strukturierung von Anforderungen nach funktionalen Aspekten bei der Abhängigkeitsanalyse von Anforderungs- und Funktionsartefakten eine große Hilfe darstellen. Die detailliertere Untersuchung unterschiedlicher Formen der Elementbeschreibung könnte ebenfalls vielversprechend hinsichtlich einer optimierten Abhängigkeitsanalyse sein, um bei den Anwendern eine klarere Vorstellung von deren Inhalt hervorzurufen und somit die eindeutigere Ableitung von Tracelinks zu ermöglichen. Weiter wäre es möglich, Anwendern bei der Modellierung von Tracelinks Hintergrundinformationen zum jeweils aktuell betrachteten Elementepaar zur Verfügung zu stellen. Auch die algorithmische Bewertung⁵¹ der Wahrscheinlich-

⁵¹ Zur Bewertung der Wahrscheinlichkeit könnten dieselben Algorithmen, die bei der Methode TraceLegacy zur Identifikation der Ähnlichkeit zweier Elemente genutzt werden (siehe Kapitel 8.1.3), zum Einsatz kommen.

keit, mit der eine Abhängigkeit zwischen zwei Elementen besteht, könnte bei der Modellierung von Tracelinks hilfreich sein. In diesem Fall wäre allerdings zu untersuchen, ob Anwender Entscheidungen noch selber treffen oder ob sie immer der automatischen Bewertung Glauben schenken.

Auch die Einführung eines Konfidenzniveaus für Tracelinks sollte zukünftig näher betrachtet werden. Mit diesem Niveau könnten bei der Erfassung von Tracelinks Unsicherheiten bzgl. der Existenz eines Tracelinks ausgedrückt werden, was den Erfassungsprozess für den Anwender vereinfacht, da keine schwarz/weiß-Entscheidungen mehr getroffen werden müssen. Tracelinks mit einem hohen Konfidenzniveau wären demnach eindeutig vorhanden. Bei Tracelinks mit einem geringen Konfidenzniveau wäre die Abhängigkeit zweier Elemente als weniger eindeutig einzuschätzen, was sowohl bei deren Verwendung (bspw. bei Auswirkungsanalysen) als auch bei der Pflege des Tracelink-Modells Berücksichtigung finden könnte. So könnten Tracelinks mit einem geringen Konfidenzniveau bspw. öfter oder durch mehr Anwender hinsichtlich ihrer Korrektheit bewertet werden. Interessant wäre es in diesem Zusammenhang zu untersuchen welche Auswirkungen die Einführung dieses Konfidenzniveaus auf die modellierenden Anwender hätte. So wäre es denkbar, dass Anwender sehr häufig ein niedriges Niveau wählen, um keine eindeutigen, auf sie zurückführbaren Entscheidungen mehr treffen zu müssen.

Langfristig sollte jedoch bei der Erfassung von Tracelinks und der Pflege von Tracelink-Modellen eine möglichst hohe (wenn nicht vollständige) Automatisierung anvisiert werden, um die manuellen Aufwände auf ein Minimum zu reduzieren und somit eine weite Verbreitung der durchgängigen Nachverfolgbarkeit zu erreichen.

LITERATURVERZEICHNIS

- ABMA, B. J. M. (2009): *Evaluation of requirements management tools with support for traceability-based change impact analysis*. Master Thesis, University of Twente.
- AIZENBUD-RESHEF, N.; NOLAN, B. T.; RUBIN, J. UND SHAHAM-GAFNI, Y. (2006): Model traceability. In: *IBM Systems Journal* 45 (3), S. 515-526.
- ALLEN, T.F. (2011): *A Summary of the Principles of Hierarchy Theory*. <http://www.iss.org/hierarchy.htm>. Zuletzt geprüft am 17. April 2011.
- ALMEFELT, L.; BERGLUND, F.; NILSSON, P. UND MALMQVIST, J. (2006): Requirements management in practice: findings from an empirical study in the automotive industry. In: *Research in Engineering Design*, S. 113-134.
- ALTHEIDE, F.; DÖRR, H. UND SCHÜRR, A. (2002): Requirements to a Framework for Sustainable Integration of System Development Tools. In: *Proceedings of the 3rd European Systems Engineering Conference (EuSEC'02)*, H. STOEWER UND L. GARNIER (Hrsg.), Toulouse, Frankreich, S. 53-57.
- Amazon.com (2013): *Amazon Mechanical Turk. Artificial Intelligence*. <https://www.mturk.com/mturk/welcome>. Zuletzt geprüft am 13. Februar 2013.
- AMIN, A.; SAID AL-DAHABI, M.; FRIEDRICH, F. UND THIELE, J. (2012): *Virtual Engineering in Industry: Development of a Mechatronic Side Mirror*. Projekt Report, Technische Universität Berlin.
- ANTONIOL, G.; CANFORA, G.; LUCIA, A. DE UND CASAZZA, G. (2000): Information Retrieval Models for Recovering Traceability Links between Code and Documentation. In: *Proceedings of the International Conference on Software Maintenance (ICSM 2000)*, L. BRIAND (Hrsg.). San Jose, Kalifornien, USA, 11 - 14 Oktober, S. 40-49.
- ARKLEY, P. UND RIDDLE, S. (2005): Overcoming the traceability benefit problem. In: *Proceedings of the 13th IEEE International Conference on Requirements Engineering*. Paris, Frankreich, 29. August - 2. September, S. 385-389.
- BALLUCHI, A.; FERRARI, A.; SANGIOVANNI-VINCENTELLI, A.; FLORA, R.; GAVIANI, G.; NESCI, W. UND SERRA, G. (2002): Functional And Architectural Specification For Power-Train Control System Design. In: *Proceedings of the 2nd IFAC Conference on Mechatronic Systems*. Berkeley, Kalifornien, USA, 9.-11. Dezember, S. 319-324.
- BEIER, G.; FIGGE, A.; LEHNER, T. UND METIN, A. (2011): Durchgängige Nachverfolgbarkeit in der Systementwicklung. Datendurchgängigkeit für die Entwicklung von Fahrzeugfunktionen. In: *ZwF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 106 (6), S. 462-465.
- BEIER, G.; FIGGE, A.; MÜLLER, R.; ROTHENBURG, U. UND STARK, R. (2013): Supporting Product Development through Cross-Discipline Dependency-Modeling – Novel Approaches for Traceability-Usage. In: *Lecture Notes on Information Theory (LNIT)* 1 (1), S. 21-28.
- BEIER, G.; ROTHENBURG, U.; WOLL, R. UND STARK, R. (2012): Durchgängige Entwicklung mit erlebbaren Prototypen. Modellbasiertes Systems Engineering. In: *Digital Engineering* 11 (3).
- BIANCHI, A.; FASOLINO, A. R. UND VISAGGIO, G. (2000): An Exploratory Case Study of the Maintenance Effectiveness of Traceability Models. In: *Proceedings of the*

- 8th International Workshop on Program Comprehension (IWPC 2000)*. Limerick, Irland, 10. - 11. Juni, S. 149-158.
- BIEDERMANN, W.; KREIMEYER, M. UND LINDEMANN, U. (2009): Measurement System to Improve Data Acquisition Workshops. In: *Proceedings of the 11th International DSM Conference (DSM'09)*. Greenville, South Carolina, USA, 12.-13. Oktober, S. 119-130.
- BIEDERMANN, W.; STRELKOW, B.; KARL, F.; LINDEMANN, U. UND ZAEH, M. F. (2010): Reducing Data Acquisition Effort by Hierarchical System Modelling. In: *Proceedings of the 12th International DSM Conference – Managing Complexity by Modelling Dependencies*, D. WYNN; M. KREIMEYER; K. EBEN; M. MAURER; U. LINDEMANN UND P. CLARKSON (Hrsg.). Cambridge, UK, 22.-23. Juli, S. 309-318.
- BLESSING, L.T.M. UND CHAKRABARTI, A. (2009): *DRM, a Design Research Methodology*. Springer-Verlag, London.
- BORNAT, P. UND MSADEK, N. (2011): Variant Modeling Approach of Automotive Electric/Electronic (E/E) Architectures and Bridge to AUTOSAR infrastructure. In: *Proceedings of Complex Systems Design & Management (CSD&M)*. Paris, Frankreich, 7.-9. Dezember.
- BORNATH, M. (2011a): *Untersuchung und prototypische Implementierung einer Methode zur effizienten Modellierung von Abhängigkeiten zwischen Partialmodellen*. Studienarbeit, Technische Universität Berlin.
- BORNATH, M. (2011b): *Untersuchung und prototypische Implementierung unterschiedlicher Methoden zur Pflege des Verknüpfungsmodells zwischen Partialmodellen*. Diplomarbeit, Technische Universität Berlin.
- British Standards Institution (2008). *ISO 12207:2008: Systems and software engineering*. ISO/IEC-IEEE, Geneva.
- CERBAH, F. UND EUZENAT, J. (2001): Traceability between models and texts through terminology. In: *Data & Knowledge Engineering* 38 (1), S. 31-43.
- CHAKRABARTI, A. UND BLIGH, T. P. (2001): A scheme for functional reasoning in conceptual design. In: *Design Studies* 22 (6), S. 493-517.
- CLELAND-HUANG, J.; CHANG, C. UND CHRISTENSEN, M. (2003): Event-based traceability for managing evolutionary change. In: *IEEE Transactions on Software Engineering* 29 (9), S. 796-810.
- CONRAD, M.; FEY, I.; GROCHTMANN, M. UND KLEIN, T. (2005): Modellbasierte Entwicklung eingebetteter Fahrzeugsoftware bei DaimlerChrysler. In: *Informatik - Forschung und Entwicklung* 20 (1-2) (2005), S. 3-10.
- CORNEY, J. R.; TORRES-SÁNCHEZ, C. UND JAGADEESAN, A. P. (2010): Outsourcing labour to the cloud. In: *International Journal of Innovation and Sustainable Development* S. 294-313.
- DANNENBERG, J. UND BURGARD, J. (2007): *2015 car innovation. Innovationsmanagement in der Automobilindustrie*. Oliver Wyman.
- Dassault Systèmes (2013): *3D Design & Engineering Best in Class Software Products. CAD, Modeling, Finite Element Simulation and more: From Designer to Consumer, Creating Brand User Experiences*. <http://www.3ds.com/products/>. Zuletzt geprüft am 17. Februar 2013.
- Department of Defense (1994). *Military Standard MIL-STD-498: Software Development and Documentation*.

- Department of Defense (2001): *Systems Engineering Fundamentals*. Defense Acquisition University Press, Fort Belvoir, Virginia.
- DIN Deutsches Institut für Normung e. V. (2005). DIN EN ISO 9000: *Qualitätsmanagementsysteme - Grundlagen und Begriffe*. Beuth Verlag GmbH.
- DIN Deutsches Institut für Normung e. V. (2006). ISO/IEC 15504-5: *Informationstechnik - Prozess-Assessment - Teil 5: Beispiel für ein Prozess-Assessmentmodell*. Beuth Verlag GmbH.
- DIN Deutsches Institut für Normung e. V. (2013). DIN EN 9115: *Qualitätsmanagementsysteme - Anforderungen an Organisationen der Luftfahrt, Raumfahrt und Verteidigung - Mitgelieferte Software*. Beuth Verlag GmbH.
- DIRSCH-WEIGAND, A.; SCHMIDT, I.; REIN, B.; STENZEL, R. UND KAMPS, T. (2006): ConWeaver - Automatisierte Wissensnetze für die semantische Suche. In: *Proceedings der 28. Online-Tagung der DGI und 58. Jahrestagung der DGI*, M. OCKENFELD (Hrsg.). Frankfurt am Main, 4.-6. Oktober.
- DOAN, A.; RAMAKRISHNAN, R. UND HALEVY, A. Y. (2011): Crowdsourcing systems on the World-Wide Web. In: *Communications of the ACM* 54 (4), S. 86-96.
- DÖMGES, R. UND POHL, K. (1998): Adapting traceability environments to project-specific needs. In: *Communications of the ACM* 41 (12) (1998), S. 54-62.
- DROGIES, S. (2006): Objektorientierte Modellbildung des fahrdynamischen Verhaltens mit MODELICA. In: *Fahrdynamik-Regelung*, R. ISERMANN (Hrsg.). Vieweg, Wiesbaden, S. 71-91.
- DUBBEL, H., BEITZ, W. UND GROTE, K.-H. (2011): *Dubbel. Taschenbuch für den Maschinenbau*. Springer-Verlag, Berlin.
- EBEN, K.; DANIILIDIS, C. UND LINDEMANN, U. (2010): Interrelating and Prioritising Requirements on multiple Hierarchy Levels. In: *Proceedings of International Design Conference - Design 2010*. Dubrovnik, Kroatien, 17.-20. Mai, S. 1055-1064.
- EDWARDS, M. UND HOWELL, S. (1992): *A Methodology for Requirements Specification and Traceability for Large Real-Time Complex Systems*. Naval Surface Warfare Center.
- EGYED, A. (2000): *Heterogeneous view integration and its automation*. Dissertation, University of Southern California.
- EGYED, A. (2003): A scenario-driven approach to trace dependency analysis. In: *IEEE Transactions on Software Engineering* 29 (2), S. 116-132.
- EGYED, A. UND GRÜNBACHER, P. (2002): Automating Requirements Traceability: Beyond the Record & Replay Paradigm. In: *Proceedings of the 17th IEEE International Conference on Automated Software Engineering (ASE)*. Edinburg, Schottland, 23.-27. September, S. 163-171.
- EGYED, A.; GRÜNBACHER, P.; Heindlm M. und BIFFL, S. (2009): Value-Based Requirements Traceability: Lessons Learned. In: *Design Requirements Engineering: A Ten-Year Perspective*, K. LYYTINEN; P. LOUCOPOULOS; J. MYLOPOULOS UND B. ROBINSON (Hrsg.). Springer-Verlag, Berlin, S. 240-257.
- EHRENSPIEL, K. (2007): *Integrierte Produktentwicklung. Denkabläufe, Methodeneinsatz, Zusammenarbeit*. Carl Hanser Verlag, München.
- EIGNER, M. UND STELZER, R. (2009): *Product Lifecycle Management. Ein Leitfaden für Product Development und Life Cycle Management*. Springer-Verlag, Berlin, Heidelberg.

- ERDEN, M. S.; KOMOTO, H.; VAN BEEK, T. J.; D'AMELIO, V.; ECHAVARRIA, E. UND TOMIYAMA, T. (2008): A review of function modeling: Approaches and applications. In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 22 (02), S. 147-169.
- ESTELLES-AROLAS, E. UND GONZALEZ-LADRON-DE-GUEVARA, F. (2012): Towards an integrated crowdsourcing definition. In: *Journal of Information Science* 38 (2), S. 189-200.
- FEI, G.; GAO, J.; OWODUNNI, O. UND TANG, X. (2011): A method for engineering design change analysis using system modelling and knowledge management techniques. In: *International Journal of Computer Integrated Manufacturing* 6 (24), S. 535-551.
- FELLNER, D. W.; KAMPS, T.; KOHLHAMMER, J. UND STRICKER, A. (2008): Vorsprung durch Wissen. In: *ZwF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 103 (4), S. 205-208.
- FIELITZ, S., OSTERLINK, M., GÖZE, D., KAWANO, K., KIM, S., LENGERT, D., VINOT, M., ELSNER, M. UND KONTAR, S. (2013): *Product Life Cycle and Systems Engineering*. Pedelec Team 2 - MobSys VE12/13. http://prezi.com/f8diw0st8-qa/product-life-cycle-and-systems-engineering/?auth_key=a13457a2737e1f05b99460045a07a8f5c04cf645. Zuletzt geprüft am 4. Februar 2013.
- Geensoft (2010): *Reqtify*. <http://www.geensoft.com/en/article/reqtify>. Zuletzt geprüft am 5. März 2012.
- GEIGER, D.; SCHULZE, T.; SEEDORF, S.; NICKERSON, R. UND SCHADER, M. (2011): Managing the Crowd: Towards a Taxonomy of Crowdsourcing Processes. In: *Proceedings of the 17th Americas Conference on Information Systems*. Detroit, Michigan, USA, 4.-7. August.
- GOTEL, O. UND FINKELSTEIN, A. (1994): An Analysis of the Requirements Traceability Problem. In: *1st International Conference on Requirements Engineering*. Colorado Springs, CO, U.S.A, S. 94-101.
- GOTEL, O. UND FINKELSTEIN, A. (1996): Revisiting requirements production. In: *Software Engineering Journal* 11 (3), S. 166-182.
- GROBSTEIN, C. (1973): Hierarchical Order and Neogenesis. In: *Hierarchy Theory. The challenge of complex systems*, H. H. PATTEE (Hrsg.). G. Braziller, New York, S. 31-47.
- HASKINS, C. (2006): *Systems Engineering Handbook. A guide for system life cycle processes and activities*. INCOSE - International Council on Systems Engineering.
- HAYES, J.; DEKHTYAR, A. UND SUNDARAM, S. (2006): Advancing candidate link generation for requirements tracing: the study of methods. In: *IEEE Transactions on Software Engineering* 32 (1), S. 4-19.
- HOOD, C. UND WIEBEL, R. (2005): *Optimieren von Requirements Management & Engineering. Mit dem HOOD Capability Model*. Springer-Verlag, Berlin, New York.
- HORSTKÖTTER, J.; METZ, P.; NTIMA, A. UND SEIM, W. (2010): *Funktionale Sicherheit und ISO 26262 - Entmystifiziert*. SynSpace und Fleckner & Simon. <http://www.synspace.com/view-document-details/60-funktionale-sicherheit-und-iso-26262-entmystifiziert.html>. Zuletzt geprüft am 07. März 2012.
- HYBERTSON, D.W. (2009): *Model-oriented systems engineering science. A unifying framework for traditional and complex systems*. Complex and enterprise systems engineering. CRC Press, Boca Raton.

- IBM Corporation (2008): *The Telelogic DOORS application. Requirements management for complex systems and software.*
- ID-Systems GmbH (2013): *METUS - The System Design Method.* http://www.id-consult.com/fileadmin/user_upload/PDFs/IDS_METUS_Raute_neu_090405.pdf. Zuletzt geprüft am 22. April 2013.
- IEEE Standards Board (1990) IEEE 610.12-1990: *IEEE standard glossary of software engineering terminology.* Institute of Electrical and Electronics Engineers, New York, N.Y., USA.
- IEEE-SA Standards Board (1998). IEEE 1219: *IEEE standard for software maintenance.* Institute of Electrical and Electronics Engineers, New York, N.Y., USA.
- International Organization for Standardization (2011). ISO 26262-1: *Road vehicles - Functional Safety - Part 1: Vocabulary.*
- International Organization for Standardization (2011). ISO 26262-2: *Road vehicles - Functional Safety - Part 2: Management of functional safety.*
- International Organization for Standardization (2011). ISO 26262-3: *Road vehicles - Functional Safety - Part 3: Concept phase.*
- International Organization for Standardization (2011). ISO 26262-4: *Road vehicles - Functional Safety - Part 4: Product development at the system level.*
- International Organization for Standardization (2011). ISO 26262-5: *Road vehicles - Functional Safety - Part 5: Product development at the hardware level.*
- International Organization for Standardization (2011). ISO 26262-6: *Road vehicles - Functional Safety - Part 6: Product development at the software level.*
- International Organization for Standardization (2011). ISO 26262-7: *Road vehicles - Functional Safety - Part 7: Production and operation.*
- International Organization for Standardization (2011). ISO 26262-8: *Road vehicles - Functional Safety - Part 8: Supporting processes.*
- International Organization for Standardization (2011). ISO 26262-9: *Road vehicles - Functional Safety - Part 9: Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses.*
- JANSCHKE, K. (2010): *Systementwurf mechatronischer Systeme. Methoden, Modelle, Konzepte.* Springer-Verlag, Berlin, Heidelberg.
- JONES, G. J. F. (2013): An Introduction to Crowdsourcing for Language and Multimedia Technology Research. In: *Information Retrieval Meets Information Visualization*, D. HUTCHISON; T. KANADE; J. KITTLER; J. M. KLEINBERG; F. MATTERN; J. C. MITCHELL; M. NAOR; O. NIERSTRASZ; C. PANDU RANGAN; B. STEFFEN; M. SUDAN; D. TERZOPOULOS; D. TYGAR; M. Y. VARDI; G. WEIKUM; M. AGOSTI; N. FERRO; P. FORNER; H. MÜLLER UND G. SANTUCCI (Hrsg.). Springer-Verlag, Berlin, Heidelberg, S. 132-154.
- KAINDL, H.; KRAMER, S. UND DIALLO, P. S. N. (1999): Semiautomatic generation of glossary links: a practical solution. In: *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia: Returning to our Diverse Roots.* Darmstadt, Deutschland, 21.-25. Februar, S. 3-12.
- KÖNIGS, S. (2012): *Konzeption und Realisierung einer Methode zur templategestützten Systementwicklung.* Dissertation, Technische Universität Berlin.
- KÖNIGS, S. F.; BEIER, G.; FIGGE, A. UND STARK, R. (2012): Traceability in Systems Engineering. Review of industrial practices, state-of-the-art technologies and new research solutions. In: *Advanced Engineering Informatics* 26 (4), S. 924-940.

- KOTONYA, G. UND SOMMERVILLE, I. (1998): *Requirements engineering. Processes and techniques*. Worldwide series in computer science. Wiley, Chichester.
- KROLL, E. (2012): Design theory and conceptual design: contrasting functional decomposition and morphology with parameter analysis. In: *Research in Engineering Design* 24 (2), S. 165-183.
- LAGO, P.; MUCCINI, H. UND VAN VLIET, H. (2009): A scoped approach to traceability management. In: *Journal of Systems and Software* 82 (1), S. 168-182.
- Lancaster University - Centre for Computer Corpus Research on Language (2013): *CLAWS part-of-speech tagger for English*. <http://ucrel.lancs.ac.uk/claws/>. Zuletzt geprüft am 25. Januar 2013.
- LANE, D. (2006): Hierarchy, Complexity, Society. In: *Hierarchy in Natural and Social Sciences*, D. PUMAIN (Hrsg.). Springer-Verlag, Berlin, Heidelberg, S. 81-119.
- LEEMHUIS, H. (2005): *Funktionsgetriebene Konstruktion als Grundlage verbesserter Produktentwicklung*. Dissertation, Technische Universität Berlin.
- LEENDERS, R. T. A. J.; VAN ENGELEN, J. M. L. UND KRATZER, J. (2007): Systematic Design Methods and the Creative Performance of New Product Teams: Do They Contradict or Complement Each Other? In: *Journal of Product Innovation Management* 24 (2), S. 166-179.
- LINDEMANN, U., MAURER, M. UND BRAUN, T. (2009): *Structural complexity management. An approach for the field of product design*. Springer-Verlag, Berlin.
- Linux.org (2013). <http://www.linux.org/>. Zuletzt geprüft am 13. Februar 2013.
- LUCIA, A. DE; FASANO, F.; OLIVETO, R. UND TORTORA, G. (2007): Recovering traceability links in software artifact management systems using information retrieval methods. In: *ACM Transactions on Software Engineering and Methodology* 16 (4) (2007), S. 13:1–13:50.
- MÄDER, P.; GOTEL, O. UND PHILIPPOW, I. (2008): Enabling automated traceability maintenance by recognizing development activities applied to models. In: *Proceedings on 23rd International Conference on Automated Software Engineering (ASE2008)*. L'Aquila, Italien, 17.-19. September.
- MÄDER, P.; GOTEL, O. UND PHILIPPOW, I. (2009a): Getting Back to Basics: Promoting the Use of a Traceability Information Model in Practice. In: *Proceedings of the 5th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE2009)*. Vancouver, Canada, 18. Mai.
- MÄDER, P.; GOTEL, O. UND PHILIPPOW, I. (2009b): Motivation Matters in the Traceability Trenches. In: *Proceedings of the 17th International Requirements Engineering Conference (RE'09)*. Atlanta, Georgia, USA, 31. August - 4. September.
- MÄDER, P.; GOTEL, O. UND PHILIPPOW, I. (2009c): traceMAINTAINER: A Tool for the Semi-automated Maintenance of Model Traceability. In: *Proceedings of the 5th ECMDA Traceability Workshop (ECMDA-TW 2009)*. Enschede, Netherlands, 23. Juni, S. 7-16.
- MANNING, C.D., RAGHAVAN, P. UND SCHUTZE, H. (2009): *An Introduction to Information Retrieval*. Cambridge University Press, Cambridge, England.
- MAURER, M.S. (2007): *Structural awareness in complex product design*. Verlag Dr. Hut, München.
- Modelica Association (2012): *Modelica - A Unified Object-Oriented Language for Systems Modeling. Language Specification*. Version 3.3.

- Modelica Association (2013): *Modelica Tools*. <https://www.modelica.org/tools>.
Zuletzt geprüft am 22. April 2013.
- MÜLLER, P.; PASCH, F.; DREWINSKI, R. UND HAYKA, H. (2013): *Kollaborative Produktentwicklung und digitale Werkzeuge. Defizite heute - Potenziale morgen*. Fraunhofer Institut für Produktionsanlagen und Konstruktionstechnik, Berlin.
- NASA STI (2007): *Systems engineering handbook*. National Aeronautics and Space Administration, Washington, DC, USA.
- NEJMEH, B.; DICKEY, T. UND WARTIK, S. (1989): Traceability Technology at the Software Productivity Consortium. In: *Proceedings of the 11th IFIP Congress*, 28. August - 1. September, S. 981-984.
- OTTER, M. (2011): *Modelica Overview*.
<https://modelica.org/education/educational-material/lecture-material/english/ModelicaOverview.pdf/view>. Zuletzt geprüft am 1. August 2013.
- PAHL, G., BEITZ, W., FELDHUSEN, J. UND GROTE, K.-H. (2007): *Konstruktionslehre. Grundlagen erfolgreicher Produktentwicklung Methoden und Anwendung*. Springer-Verlag, New York, Berlin, Heidelberg.
- PAIGE, R.; OLSEN, G. K.; KOLOVOS, D.; ZSCHALER, S. UND POWER, C. (2008): Building model-driven engineering traceability classifications. In: *Proceedings of the ECMDA Traceability Workshop (ECMDA-TW) 2008*, J. OLDEVIK; K. GØRAN; T. NEPLE; R. PAIGE UND J. OLDEVIK (Hrsg.). Berlin, Deutschland, 12. Juni, S. 49-58.
- PALMER, J. D. (2000): Traceability. In: *Software requirements engineering*, R. H. THAYER; M. DUNCAN UND A. M. DAVIS (Hrsg.). IEEE Computer Society Press, Los Alamitos, S. 412-422.
- PFEIFER, M.; SEILER, S.; BÖTTJER, T. UND LIEBERUM, M. (2012): *Virtual Engineering in Industry: Development of a Mechatronic Side Mirror*. Projekt Report, Technische Universität Berlin.
- PFLEEGER, S. UND BOHNER, S. (1990): A framework for software maintenance metrics. In: *Proceedings of the Conference on Software Maintenance*. IEEE Computer Society Press, Washington, DC, S. 320-327.
- PIERCE, R. A. (1978): A Requirements Tracing Tool. In: *ACM SIGMETRICS Performance Evaluation Review* 7 (3-4), S. 53-60.
- PINHEIRO, F. A. C. (2003): Requirements Traceability. In: *Perspectives on software requirements*, J. C. SAMPAIO DO PRADO LEITE UND J. H. DOORN (Hrsg.). Kluwer Academic Publishers, Boston, S. 91-113.
- PLETTE, A. (2009): *Ganzheitliches Lastenheftmanagement*. IBM Rational Roadshow. 2009.
- POHL, M. UND SCHEEL, C. (2004): *Toolvorstellung Reqtify*. Technische Universität Berlin. <http://swt.cs.tu-berlin.de/lehre/eks/ws0304/SlidesPaper/ausarbeitung%20reqtify.pdf>. Zuletzt geprüft am 5. März 2012.
- PONN, J. UND LINDEMANN, U. (2008): *Konzeptentwicklung und Gestaltung technischer Produkte. Optimierte Produkte - systematisch von Anforderungen zu Konzepten*. Springer-Verlag, Berlin, Heidelberg.
- PORTER, M. (2006): An algorithm for suffix stripping. In: *Program: electronic library and information systems* 40 (3), S. 211-218.
- PORTER, M. (2013a): *German stemming algorithm*. <http://snowball.tartarus.org/algorithms/german/stemmer.html>. Zuletzt geprüft am 2. Februar 2013.

- PORTER, M. (2013b): *Snowball*. <http://snowball.tartarus.org/index.php>. Zuletzt geprüft am 2. Februar 2013.
- PROBST, G.J.B. UND ULRICH, H. (1988): *Anleitung zum ganzheitlichen Denken und Handeln. Ein Brevier für Führungskräfte*. Haupt Verlag, Bern, Stuttgart.
- PUMAIN, D., Ed. (2006): *Hierarchy in Natural and Social Sciences*. Springer-Verlag, Berlin, Heidelberg.
- RAIA, F. (2005): Students' Understanding of Complex Dynamic Systems. In: *Journal of Geoscience Education*, S. 297-308.
- RAMESH, B. UND EDWARDS, M. (1993): Issues in the development of a requirements traceability model. In: *Proceedings on the IEEE International Symposium on Requirements Engineering (RE '93)*. San Diego, CA USA, 4.-6. Januar, S. 256-259.
- RAMESH, B. UND JARKE, M. (2001): Toward reference Models for Requirements Traceability. In: *IEEE Transactions on Software Engineering* 27 S. 1-54.
- RAMESH, B.; STUBBS, C.; POWERS, T. UND EDWARDS, M. (1997): Requirements traceability: Theory and practice. In: *Annals of Software Engineering*. Kluwer Academic Publishers, S. 397-415.
- ROSENAUER, G. B. (2008): *A Standards-Based Approach to Dynamic Tool Integration Using Java Business Integration. A Redesign of the ToolNet Framework built on Enterprise Integration Standards*. Diplomarbeit, ISIS Vienna University of Technology.
- ROUSE, A. (2010): A preliminary taxonomy of crowdsourcing. In: *Proceedings of the 21st Australasian Conference on Information Systems (ACIS 2010)*. Brisbane, Australia, 1.-3. Dezember, S. 76-87.
- SCHÖNSLEBEN, P. (2007): *Integrales Logistikmanagement. Operations and Supply Chain Management in umfassenden Wertschöpfungsnetzwerken*. Springer-Verlag, Berlin.
- SCHUH, G.; LENDERS, M.; NUBBAUM, C. UND RUDOLF, S. (2012): Produktarchitekturgestaltung. In: *Handbuch Produktion und Management 3. Innovationsmanagement*, G. SCHUH (Hrsg.). Springer-Verlag, Berlin, S. 115-160.
- Siemens PLM Software (2010): *Mechatronics Concept Designer. Ein funktionsorientierter Ansatz für den Maschinen- und Anlagenbau*. www.siemens.de/plm/mcd. Zuletzt geprüft am 17. Februar 2013.
- Siemens PLM Software (2011): *White Paper: Managing the development of complex automotive products through a systems-driven process*. http://m.plm.automation.siemens.com/en_us/Images/Siemens-PLM-Systems-Driven-Product-Development-wp_tcm1224-121850.pdf. Zuletzt geprüft am 27. August 2013.
- SIMON, H. A. (1973): The Organization of Complex Systems. In: *Hierarchy Theory. The challenge of complex systems*, H. H. PATTEE (Hrsg.). G. Braziller, New York, S. 3-27.
- SPANOUKAKIS, G.; ZISMAN, A.; PEREZ-MINANA, E. UND KRAUSE, P. (2004): Rule-based generation of requirements traceability relations. In: *The Journal of Systems and Software* 72 S. 105-127.
- STARK, R. (2011): *Challenges in modern Product Creation Processes. PEP2015*. Technical Steering Committee. 29. Juni 2011, Darmstadt.
- STARK, R.; BEIER, G.; FIGGE, A. UND WÖHLER, T. (2010): Cross-Domain Dependency Modelling - How to achieve consistent System Models with Tool Support. In:

- Proceedings EuSEC 2010 - European Systems Engineering Conference*. Stockholm, Sweden, 23.-26. Mai.
- STARK, R. UND FIGGE, A. (2011): Eco Tracing - A Systems Engineering Method for Efficient Tracelink Modelling. In: *Proceedings of the 18th International Conference on Engineering Design (ICED11)*, Vol. 4. *Product and Systems Design*, S. J. CULLEY; B. J. HICKS; T. C. MCALOONE; T. J. HOWARD UND U. LINDEMANN (Hrsg.). Kopenhagen, Dänemark, S. 21-32.
- SUGDEN, R. UND STRENS, M. (1996): Strategies, tactics and methods for handling change. In: *Proceedings: IEEE Symposium and Workshop on Engineering of Computer-Based Systems*. Friedrichshafen, Germany, 11.-15. März, S. 457-463.
- SUTINEN, K.; ALMEFELT, L. UND MALMQVIST, J. (2000): Implementation of requirements traceability in systems engineering tools. In: *Proceedings of the Product Models Conference*. Linköping, Schweden, 7.-8. November, S. 313-330.
- SUTINEN, K.; ALMEFELT, L. UND MALMQVIST, J. (2002): Supporting Concept Development Using Quantitative Requirements Traceability. In: *Proceedings of the 12th Annual International Symposium INCOSE 2002*. Las Vegas, Nevada, USA, 28. Juli - 1. August.
- SUTINEN, K.; GUSTAFSSON, G. UND MALMQVIST, J. (2004): Computer Support for Requirements Management in an international Product Development Project. In: *Proceedings of the ASME 2004 Design Engineering and Technical Conferences and Computers and Information in Engineering Conference (DETC'04)*. Salt Lake City, Utah, USA, 28. September - 2. Oktober.
- Teseon GmbH: *LOOME0 complexity software*. <http://teseon.de/loomeo>. Zuletzt geprüft am 18. August 2012.
- Teseon GmbH (2012): *LOOME0 Highlights*. <http://teseon.de/component/photodownload/category/2-loomeo-public?download=6:loomeo-highlights-de>. Zuletzt geprüft am 18. August 2012.
- TILSTRA, A. H.; SEEPERSAD, C. C. UND WOOD, K. (2009): Distributed Modeling of Component DSM. In: *Proceedings of the 11th International DSM Conference (DSM'09)*. Greenville, South Carolina, USA, 12.-13. Oktober, S. 243-258.
- TOMIZUKA, M. (2002): Mechatronics: from the 20th to 21st century. In: *Control Engineering Practice* 10 (8), S. 877-886.
- TRETOW, G.; GÖPFERT, J. UND HEESE, C. (2008): In sieben Schritten systematisch entwickeln. In: *CAD-CAM Report* (8), S. 36-39.
- TUDORACHE, T. (2006): *Employing Ontologies for an Improved Development Process in Collaborative Engineering*. Dissertation, Technische Universität Berlin.
- UMEDA, Y. UND TOMIYAMA, T. (1995): FBS modeling: modeling scheme of function for conceptual design. In: *Proceedings of the 9th International Workshop on Qualitative Reasoning*. Amsterdam, Niederlande, 16.-19. Mai, S. 271-278.
- VAJNA, S. (2009): *CAX für Ingenieure. Eine praxisbezogene Einführung*. Springer-Verlag, Berlin, Heidelberg.
- VAN BEEK, T. J. UND TOMIYAMA, T. (2008): Requirements for Complex Systems Modeling. In: *Proceedings of the CIRP Design Conference: Design Synthesis*. Enschede, Niederlande, 7.-9. April.
- VAN DEN HAMER, P. UND LEPOETER, K. (1996): Managing design data: the five dimensions of CAD frameworks, configuration management, and product data management. In: *Proceedings of the IEEE* 84 (1), S. 42-56.

- VAN GORP, P.; ALTHEIDE, F. UND JANSSENS, D. (2006): Traceability and Fine-Grained Constraints in Interactive Inconsistency Management. In: *Second ECMDA Traceability Workshop (ECMDA-TR 2006)*, T. NEPLE; J. OLDEVIK UND J. AAGEDAL (Hrsg.). Bilbao, Spanien, 10. Juli.
- Verein Deutscher Ingenieure (2004). VDI-Richtlinie 2206: *Entwicklungsmethodik für mechatronische Systeme*. Beuth Verlag GmbH, Berlin.
- Verein Deutscher Ingenieure (2012). VDI 2219: *Informationsverarbeitung in der Produktentwicklung - Einführung und Wirtschaftlichkeit von EDM/PDM-Systemen*. Beuth Verlag GmbH, Berlin.
- VERMAAS, P. E. (2009): The flexible Meaning of Function in Engineering. In: *Proceedings of the 17th International Conference on Engineering Design (ICED'09)*, Vol. 2, M. NORELL BERGENDAHL; M. GRIMHEDEN; L. LEIFER; P. SKOGSTAD UND U. LINDEMANN (Hrsg.). Stanford University, California, USA, 24.-27. August, S. 113-124.
- WEBER, C.; FRANKE, H.-J. UND STELZER, R. (2007): Wissensmanagement. In: *Innovationspotenziale in der Produktentwicklung*, F.-L. KRAUSE; H.-J. FRANKE UND J. GAUSEMEIER (Hrsg.). Carl Hanser Verlag, München, Wien, S. 141-149.
- WEBER, J. (2009): *Automotive development processes. Processes for successful customer oriented vehicle development*. Springer-Verlag, Dordrecht, New York.
- WEILKIENS, T. (2006): *Systems engineering mit SysML/UML. Modellierung, Analyse, Design*. Dpunkt-Verlag, Heidelberg.
- WIERINGA, R. (1995): *An introduction to requirements traceability. Technical Report IR-389*. Faculty of Mathematics and Computer Science, University of Vrije, Amsterdam.
- Wikimedia Foundation Inc (2013): *Wikipedia. The Free Encyclopedia*. <http://en.wikipedia.org/>. Zuletzt geprüft am 13. Februar 2013.
- WINKLER, S. UND PILGRIM, J. (2010): A survey of traceability in requirements engineering and model-driven development. In: *Software & Systems Modeling* 9 (4), S. 529-565.
- WINTER, R. (2003): Modelle, Techniken und Werkzeuge im Business Engineering. In: *Business engineering. Auf dem Weg zum Unternehmen des Informationszeitalters*, H. ÖSTERLE (Hrsg.). Springer-Verlag, Berlin [u.a.], S. 87-118.
- WRASSE, K. R. (2011): *Modellübergreifende Abhängigkeiten in der Entwicklung mechatronischer Systeme. Erarbeitung und Aufbereitung eines industriennahen Modells zu Evaluationszwecken*. Studienarbeit, Technische Universität Berlin.
- Yahoo! Inc. (2013): *Yahoo! Answers*. <http://answers.yahoo.com/>. Zuletzt geprüft am 13. Februar 2013.
- YEH, R. UND ZAVE, P. (1980): Specifying software requirements. In: *Proceedings of the IEEE* 68 (9), S. 1077-1085.
- ZISMAN, A.; SPANOUDAKIS, G.; PÉREZ-MIÑANA, E. UND KRAUSE, P. (2003): Tracing Software Requirements Artefacts. In: *Proceedings of the 2003 International Conference on Software Engineering Research and Practice (SERP'03)*. Las Vegas, USA, 23.-26. Juni, S. 448-455.

GLOSSAR

Begriff	Beschreibung	Synonym
Abhängigkeit	Die Abhängigkeitsbeziehung (engl. dependency) ist eine Beziehung zwischen zwei Elementen, die beschreibt, dass ein Element ein anderes Element für seine Spezifikation oder Implementierung benötigt [Weilkiens 2006, S. 233].	
Algorithmus	Endliches Set von definierten Regeln für die Lösung eines Problems mit einer endlichen Anzahl von Schritten [IEEE 610.12-1990, S. 9].	
Allokation	Prozess der Verteilung von Anforderungen, Ressourcen oder anderen Elementen auf die Komponenten eines Systems (in Anlehnung an [IEEE 610.12-1990, S. 10] und [ISO 26262-1, S. 1]).	
Anforderung	Dokumentierte Beschreibung eines Zustands oder einer Fähigkeit, die von einem Nutzer benötigt wird, um ein Problem zu lösen oder ein Ziel zu erreichen (in Anlehnung an [IEEE 610.12-1990, S. 62]).	
Architektur	Repräsentation der Systems, die die Identifikation der Systemelemente, Systemgrenzen und Schnittstellen erlaubt [ISO 26262-1, S. 2]).	
Artefakt	Beschreibt jegliche Art von während der Produktentwicklung erzeugten produktbeschreibenden Informationen (Spezifikationen, Funktionsstrukturen, Produktstrukturen etc.), unabhängig von ihrer informationstechnischen Ausführbarkeit [Cleland-Huang et al. 2003, S. 797].	Partialmodell
artefaktinterner Tracelink	Beschreibt die Abhängigkeit zwischen zwei Elementen aus dem gleichen Artefakt [Königs et al. 2012, S. 930].	
artefaktübergreifender Tracelink	Beschreibt die Abhängigkeit zwischen zwei Elementen aus unterschiedlichen Artefakten [Königs et al. 2012, S. 930].	
Auswirkungsanalyse	Traceability-basierte Methode, bei welcher alle Elemente der Produkt-beschreibenden Artefakte eines Systems identifiziert werden, die von einer vorgeschlagenen Änderung direkt oder indirekt potentiell betroffen sein können.	Impact Analyse

Begriff	Beschreibung	Synonym
CATIA	CAD-Programm der Firma Dassault Systèmes (Computer Aided Three-Dimensional Interactive Application).	
Crowdsourcing	Beim Crowdsourcing wird eine von einem Prozesseigner definierte Aufgaben- oder Problemstellungen durch eine größere Anzahl an Menschen bearbeitet [Doan et al. 2011, S. 87].	
Daten	„Daten sind Symbole und Zahlenwerte ohne Bedeutung“ [Weber et al. 2007, S. 141]	
Design Research Methodology	Die Design Research Methodology (DRM) nach Blessing und Chakrabarti ist eine Methodik zur Durchführung von Forschungsarbeiten im Bereich der Konstruktion und Entwicklung [Blessing und Chakrabarti 2009].	
Design Structure Matrix	Die Design Structure Matrix (DSM) ist eine Matrix, in der Abhängigkeiten zwischen Elementen eines Artefakts dokumentiert und ausgewertet werden können [Maurer 2007, S. 54].	
Disziplin	Abgrenzung der klassischen Fachgebiete wie Mechanik, Elektrik & Elektronik und Informationstechnik [VDI-Richtlinie 2206, S. 14]	Domäne
Domain Mapping Matrix	Die Domain Mapping Matrix (DMM) ist eine Matrix, in der Abhängigkeiten zwischen Elementen unterschiedlicher Artefakte dokumentiert und ausgewertet werden können [Maurer 2007, S. 58].	
Durchgängige Nachverfolgbarkeit		siehe. Traceability
Dymola	Auf Modelica basierende Simulationsumgebung der Firma Dassault Systèmes.	
Element	Elemente stellen die Bestandteile der Artefakte dar. Sie können innerhalb eines Artefakts hierarchisch strukturiert und darüber hinaus über Parameter konkretisiert sein.	
Elternelement	Als Elternelemente werden Elemente bezeichnet, die einem anderen Element (sog. Kinderelemente) in einer Hierarchie übergeordnet sind.	
ENOVIA	PLM-System der Firma Dassault Systèmes.	
Falsch negativer Tracelink	Fälschlicherweise fehlender Tracelink zwischen einem Elementpaar, zwischen dem eine Abhängigkeit besteht.	

Begriff	Beschreibung	Synonym
Falsch positiver Tracelink	Fälschlicherweise vorhandener Tracelink zwischen einem Elementpaar, zwischen dem keine Abhängigkeit besteht [Hayes et al. 2006, S. 4].	
Funktion	Funktionen stellen im Kontext der Systementwicklung „eine am Zweck orientierte, lösungsneutrale, als Operation beschriebene Beziehung zwischen Eingangs- und Ausgangsgrößen eines Systems“ dar [Ponn und Lindemann 2008, S. 56].	
Funktionsstruktur	Verknüpfung von Teilfunktionen zu einer Gesamtfunktion [Pahl et al. 2007, S. 243].	
Geschwisterelement	Als Geschwisterelemente werden Elemente bezeichnet, die auf einer hierarchischen Ebene einer Struktur liegen.	
Granularität	Bezeichnet im Kontext der durchgängigen Nachverfolgbarkeit den Detaillierungsgrad auf dem zwischen unterschiedlichen Artefakten Tracelinks modelliert werden [Egyed et al. 2009, S. 2]. Während eine hohe oder feine Granularität bedeutet, dass sehr detailliert modelliert wird (z. B. zwischen Parametern unterschiedlicher Elemente), bedeutet eine geringe oder grobe Granularität, dass auf einer abstrakten Ebene modelliert wird.	
Hierarchie	Struktur, bei der Komponenten in Ebenen unterschiedlichen Rangs angeordnet sind. Dabei hat jede Komponente null, eine oder mehr Kinder und keine Komponente mehr als eine Elternkomponente (in Anlehnung an [IEEE 610.12-1990, S. 37]).	
Identifizier	Name, Adresse, Label oder kennzeichnender Index eines Elements in einem Computer Programm [IEEE 610.12-1990, S. 38].	ID, Unique ID
Impact Analyse		siehe Auswirkungenanalyse
Inklusionshierarchie	Hierarchisches Konstrukt, bei dem die übergeordneten Elemente abstrakte Container für die untergeordneten Elemente darstellen.	
Informationen	„Informationen sind an einen speziellen Kontext gebundene Daten mit für den Menschen verständlicher Bedeutung“ [Weber et al. 2007, S. 141]	

Begriff	Beschreibung	Synonym
Integrative Software Werkzeuge	Im Kontext der durchgängigen Nachverfolgbarkeit werden Software Werkzeuge als integrativ bezeichnet, wenn mit ihnen Tracelinks zwischen Artefakten, die aus bzw. mit existierenden Auto- renwerkzeugen importiert bzw. synchronisiert werden können.	
Kandidaten-Link	Kandidaten-Links sind Vorschläge für Tracelinks, die mit einer gewissen Wahrscheinlichkeit zwischen zwei Elemente bestehen. Sie werden von Methoden zur Unterstützung der Erfassung und Pflege von Traceability-Modellen ausgegeben und bedürfen einer manuellen Überprüfung.	
Kinderelement	Als Kinderelemente werden Elemente bezeichnet, die einem anderen Element (sog. Elternelemente) in einer Hierarchie untergeordnet sind.	
Komplexität	Die Komplexität bezieht sich auf die Anzahl und Art der Beziehungen zwischen Elementen in einem System [Weilkiens 2006, S. 10].	
Kompliziertheit	Bezieht sich auf die Anzahl der unterschiedlichen Elemente [Weilkiens 2006, S. 10].	
Komponente	Bestandteil aus welchen ein System aufgebaut ist. Komponenten können sowohl Hard- als auch Software sein und weiter in andere Komponenten unterteilt werden [IEEE 610.12-1990, S. 18].	
Kompositionelle Inklusionshierarchie	Die kompositionelle Inklusionshierarchie ist umgangssprachlich auch als „is part of“ oder „is composed of“-Hierarchie bekannt. Sie wird verwendet, um Systeme zu strukturieren [Lane 2006, S. 85-86; Pumain 2006, S. 4]. Dabei wird ein System so lange in seine Subsysteme gegliedert, bis man zu einer Ebene gelangt, die sich nicht weiter unterteilen lässt bzw. nicht mehr sinngemäß ist.	
Konsistenz	Grad der Uniformität, Standardisierung und Freiheit von Widersprüchen zwischen den Artefakten bzw. Teilen eines Systems oder Komponente [IEEE 610.12-1990, S. 21].	
Mechatronik	„Mechatronik ist die synergetische Integration physikalischer Systeme mit Informationstechnologie[...]“ (aus dem Englischen nach [Tomizuka 2002, S. 877] sowie in Anlehnung an [VDI-Richtlinie 2206, S. 14])	

Begriff	Beschreibung	Synonym
Metamodell	Bezeichnet ein übergeordnetes Modell, in dem für ein Traceability-Werkzeug die zur Verknüpfung erlaubten Artefakte, Elemente und Tracelinks beschrieben werden.	Traceability Schema
Methode	Planmäßiges Vorgehen zum Erreichen eines bestimmten Ziels [Pahl et al. 2007, S. 784].	
Modelica	Objektorientierte Beschreibungssprache für physikalische Modelle.	
Modell	Ein dem Zweck entsprechender Repräsentant (Vertreter) eines Originals [Pahl et al. 2007, S. 784].	
Modellbasierte Entwicklung	Verwendet Modelle als zentrale Komponente. Während des Entwicklungsprozesses werden unterschiedliche Arten von Modellen für spezifische Aufgabenstellungen entwickelt, die spezifische Aspekte des Systems beschreiben [Tudorache 2006, S. 23].	
Modellieren	Erstellen und Verändern (Modifizieren) eines Modells im Gesamten oder in Teilen [Pahl et al. 2007, S. 784].	
Multiple-Domain Matrix	Die Multiple-Domain Matrix (MDM) stellt eine Kombination aus mehreren DSMs und DMMs dar [Maurer 2007, S. 60].	
Nicht-integrative Software Werkzeuge	Im Kontext der durchgängigen Nachverfolgbarkeit werden Software Werkzeuge als nicht-integrativ bezeichnet, wenn mit ihnen ausschließlich Tracelinks zwischen Artefakten modelliert werden können, die in demselben Werkzeug erstellt wurden.	
Partialmodell	Dokumente, Modelle, Code usw., die während einer Phase des Entwicklungsprozesses erstellt werden	Artefakt
phaseninterner Tracelink	Beschreibt die Abhängigkeit zwischen zwei Elementen aus Artefakten der gleichen Prozessphase.	
phasenübergreifender Tracelink	Beschreibt die Abhängigkeit zwischen zwei Elementen aus Artefakten unterschiedlicher Prozessphasen.	

Begriff	Beschreibung	Synonym
Präzision	Ein Parameter zur Angabe der Relevanz einer Treffermenge, der sich aus dem Verhältnis der Anzahl der korrekterweise identifizierten Elemente und der Vereinigungsmenge aller identifizierten Elemente ergibt [Manning et al. 2009, S. 155].	Precision
Produktmodell	Gesamtmenge aller, im Rahmen der Entwicklung entstehenden, Produkt-beschreibenden Modelle (wie bspw. Anforderungen oder Geometrie-Modelle)	
Qualität	Grad zu welchem ein System, eine Komponente oder Prozess die spezifizierten Anforderungen erfüllt [IEEE 610.12-1990, S. 60].	
Qualitative Traceability	Lässt nur die Identifikation abhängiger Elemente zu, ohne Aussagen über den Grad der Abhängigkeit treffen zu können.	
Quantitative Traceability	Lässt Aussagen über das Ausmaß der Beeinflussung eines Elements durch die Änderung eines abhängigen Elements zu [Sutinen et al. 2002, S. 1].	
Trefferquote	Ein Parameter zur Angabe der Vollständigkeit einer Treffermenge, der sich aus dem Verhältnis der Anzahl der korrekterweise identifizierten Elemente zu der Vereinigungsmenge der relevanten Elemente ergibt [Manning et al. 2009, S. 155].	Recall
Requirements Traceability	Bezieht sich auf die Fähigkeit, das Leben einer Anforderung zu beschreiben und zu verfolgen: sowohl vorwärts als auch rückwärts [Gotel und Finkelstein 1994, S. 94].	
Subelement	-	siehe Kinder-element
Subsumtive Inklusionshierarchie	Mit Hilfe der subsumtiven Inklusionshierarchie können Objekte von allgemein bis spezifisch klassifiziert werden. Sie wird auch „is a“-Hierarchie genannt und entspricht der Generalisierung im Blockdefinitionsdiagramm der SysML [Weilkiens 2006].	
System	Ein System ist eine Zusammenstellung unterschiedlicher, miteinander interagierender Systemelemente, die ein gemeinsames oder mehrere Ziele verfolgen [Department of Defense 2001, S. 3; Haskins 2006, S. 1.5].	

Begriff	Beschreibung	Synonym
Systems Engineering	Interdisziplinäre Methodik zur erfolgreichen Entwicklung von Systemen [Haskins 2006, S. 1.5].	
Top-Down	Vorgehensweise, bei der ausgehend vom Abstrakten schrittweise konkretisiert wird.	
Traceability	Traceability ist eine Methode zur Unterstützung der Entwicklung mechatronischer Systeme, bei welcher Abhängigkeiten zwischen den Elementen Produkt-beschreibender Artefakte explizit dokumentiert werden. Das Ziel der Methode, der Zustand einer durchgängigen Nachverfolgbarkeit zwischen den produktbeschreibenden Artefakten, wird ebenfalls mit dem Begriff Traceability beschrieben.	Durchgängige Nachverfolgbarkeit
Traceability-Matrix	Matrix zur Dokumentation von Beziehungen zwischen zwei oder mehr Produkten (Artefakten) des Entwicklungsprozesses [IEEE 610.12-1990, S. 78].	
Traceability-Modell	Bezeichnet die Gesamtheit aus betrachteten Artefakten und ihrer, die gegenseitigen Abhängigkeiten beschreibenden, Tracelinks (in Anlehnung an [Pinheiro 2003]).	Integriertes Modell
Traceability-Schema	Bezeichnet ein Schema, in dem für ein Traceability-Werkzeug die zur Verknüpfung erlaubten Artefakte, Elemente und Tracelinks beschrieben werden [Winkler und Pilgrim 2010, S. 536].	Metamodell
Tracelink	Bei Tracelinks handelt es sich um Software-Objekte, die je nach Umsetzung bspw. die IDs der voneinander abhängigen Elemente oder Parameter beinhalten und darüber deren Abhängigkeit ausdrücken [van Gorp et al. 2006, S. 2].	Trace Link, Traceability Link, Verknüpfung, Relation
Tracelink-Modell	Bezeichnet die Gesamtheit der Tracelinks zwischen einer festzulegenden Anzahl von Artefakten. Ist Teilmenge des Traceability-Modells.	Abhängigkeitsmodell
Verknüpfung	-	siehe Tracelink
Wissen	„Wissen ist angewandte Information mit Bezug zu anderen Informationen, d. h. Wissen ist relational“ [Weber et al. 2007, S. 142]	

ANHANG

A. MATERIAL ZUR STUDIE „EVALUATION DER HYPOTHESE 1* IM KONTEXT DER SYSTEMENTWICKLUNG“

STUDIENEINFÜHRUNG UND AUFGABENSTELLUNG

Abhängigkeiten zwischen Funktionen und Bauteilen

Vielen Dank für Ihre Teilnahme an unserer Studie!

In dieser Untersuchung geht es um Abhängigkeiten zwischen den Funktionen eines Geräts und den Bauteilen des Geräts. Die beiden Geräte, die im Rahmen der Studie untersucht werden, sind eine Kaffee-Vollautomaten und ein Fahrrad.

Der Versuchsablauf gliedert sich folgendermaßen:

- 1) Einführung in die Thematik und
- 2) Einführung in die verwendete Versuchs-Software
- 3) Bearbeitung zweier Teilaufgaben jeweils an den beiden Beispielen Kaffee-Vollautomat und Fahrrad:
 1. Bearbeitung der Abhängigkeiten zwischen Gesamtfunktionen und Baugruppen bzw. Systemen
 2. Bearbeitung der Abhängigkeiten zwischen Teilfunktionen und Bauteilen
- 4) Ausfüllen eines Fragebogens
- 5) Nachbesprechung

Für Zwischenfragen steht Ihnen der Versuchsleiter jederzeit gerne zur Verfügung!

Zuerst füllen sie bitte folgende Angaben zu ihrer Person aus:

Alter	
Geschlecht	
Abschluss bzw. Fachgebiet	
Einschätzung der eigenen Kenntnis des Gerätes „Kaffee-Vollautomat“	

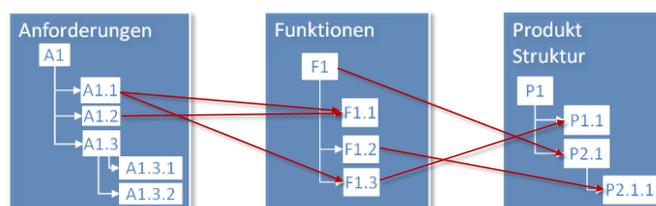
Einschätzung der eigenen Kenntnis des Geräts „Fahrrad“			
	Kenne ich nicht	Kenne die Bauteile und weiß in etwa wie es funktioniert	Habe ich selbst schon repariert

1. Einführung in die Thematik

Im Folgenden möchten wir Ihnen einen kurzen inhaltlichen Einblick in das Themengebiet geben, aus dem die zu lösenden Aufgaben entstammen. Bitte lesen sie ihn aufmerksam, da ein grundsätzliches Verständnis der Beispiele für die Bearbeitung notwendig ist.

1.1 Partialmodelle, Abhängigkeiten und Tracelinks

Im Entwicklungsprozess eines Gerätes entstehen verschiedene *Partialmodelle*, die unterschiedliche Sichtweisen auf das System strukturiert abbilden. Am Anfang des Entwicklungsprozesses stehen zunächst **Anforderungen** an ein zu entwickelndes Produkt. In der sich anschließenden Entwurfsphase entstehen **Funktions**-Modelle, um die funktionelle Realisierung der Anforderungen zu beschreiben, sowie Modelle, die die **Produktstruktur** im Hinblick auf Baugruppen, Systeme und Bauteile abbilden.



Zwischen den Elementen, aus denen diese Modelle aufgebaut sind, bestehen dabei inhaltliche Abhängigkeiten, d.h.: „dieses Bauteil trägt zur Umsetzung dieser Funktion bei“. So kann bspw. die Funktion „Energieversorgung realisieren“ durch das Bauteil „Batterie“ in der Produktstruktur definiert werden. Im Rahmen der folgenden Aufgabenstellungen werden Sie Funktions-Modelle und Produktstrukturen auf genau solche inhaltlichen Abhängigkeiten zueinander untersuchen.

1.2 Matrixdarstellungen zur Erfassung der Abhängigkeiten

Zum Zweck dieser Untersuchungen werden Matrizen in Excel genutzt (siehe Abbildung 1). Dabei werden die Baugruppen und Bauteile in den Zeilen links von der Matrix dargestellt. Die Funktionen sind für eine bessere Lesbarkeit ebenfalls waagrecht

und oberhalb der Matrix angeordnet. Abhängigkeiten zwischen einem Bauteil und einer Funktion werden in deren gemeinsamer Zelle innerhalb der Matrix gekennzeichnet. Es geht dabei um **direkte Abhängigkeiten** im Sinne von „**Funktion wird durch Bauteil (teilweise) realisiert**“. Um die Bauteile und Funktionen eindeutig identifizieren zu können, wenn die gemeinsame Zelle ausgewählt ist, werden sie gelb markiert.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2														
3		START												
4														
5														
6														
7														
8														
9														
10														
11														

Abbildung 1: Matrix mit Funktionen (oben) und Bauteilen (links)

Ihre Aufgabe ist es im Folgenden alle Zellen der Matrix mit den Pfeiltasten durchzugehen und die zugehörigen gelb markierten Bauteil-Funktion-Kombinationen auf inhaltliche Abhängigkeiten zueinander zu untersuchen. Sollte Ihrer Meinung nach eine Abhängigkeit zwischen Bauteil und Funktion bestehen, dokumentieren Sie dies mit einer „1“ (eine Abhängigkeit ist vorhanden). Sind sie hingegen der Meinung, dass ein Bauteil keinen Beitrag zur Erfüllung einer Funktion leistet, dokumentieren Sie dies mit einer „0“ (es ist keine Abhängigkeit vorhanden).

Wenn Sie bereit sind die Studie zu beginnen, klicken Sie bitte auf den Start Button. Im Anschluss wählen sie die Zelle links oben in der Matrix aus und fangen an Ihre Entscheidungen mit „0“ und „1“ einzutragen. Die Navigation von einer Zelle zur nächsten innerhalb der Matrix erfolgt mit den Pfeiltasten: „Pfeil nach rechts“ → und „Pfeil nach links“ ←. Sollte der rechte Rand der Matrix erreicht werden springt die ausgewählte Zelle automatisch bei Druck der Taste → in die erste Zelle der darauffolgenden Zeile. Sollten Sie eine Entscheidung nachträglich ändern wollen, können Sie die zugehörige Zelle entweder mit den Tasten oder der Maus erreichen.

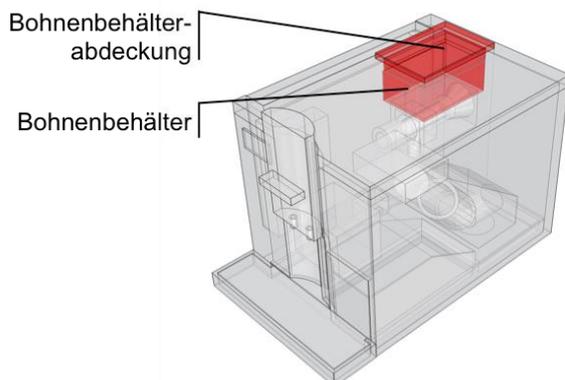
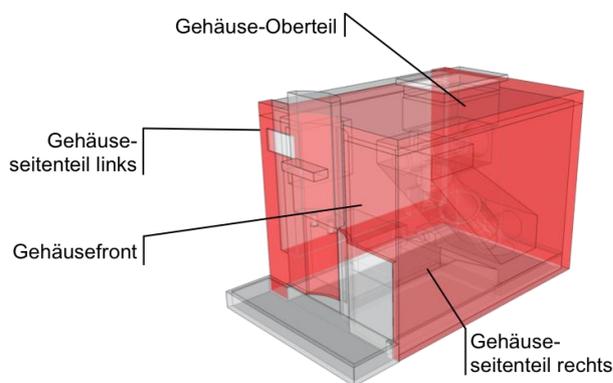
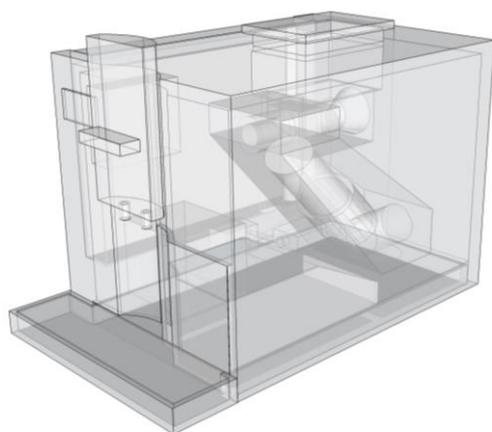
1.3 Einführungsbeispiel

Um den Umgang mit Abhängigkeiten zwischen Funktionen und Bauteilen und deren Dokumentation in der Excel Tabelle zu üben, bearbeiten Sie zunächst ein einfaches und kleines Beispiel. In diesem Einführungsbeispiel handelt es sich um ein Wohnhaus mit unterschiedlichen Räumen und Einrichtungsgegenständen und deren Funktionen. Das Beispiel ist genau wie die später folgenden zwei geteilt: zunächst werden

Sie Abhängigkeiten zwischen Gesamtfunktionen und Baugruppen und im Anschluss auf Teilfunktionen und Bauteile untersuchen. Bitte zögern Sie nicht, Fragen zu äußern, die während der Bearbeitung des Einführungsbeispiels aufkommen!

2. Aufgabe „Kaffeevollautomat“

Damit Sie, auch wenn Sie Ihre Kenntnis des Beispiels „Kaffeevollautomat“ als gering eingestuft haben, die Aufgabenstellung lösen können, werden Ihnen im Folgenden die Systeme, Baugruppen und Bauteile sowie die grundlegende Funktionsweise des Kaffeevollautomaten erläutert. Dabei liegt der Fokus auf den wichtigsten Baugruppen und Funktionen.



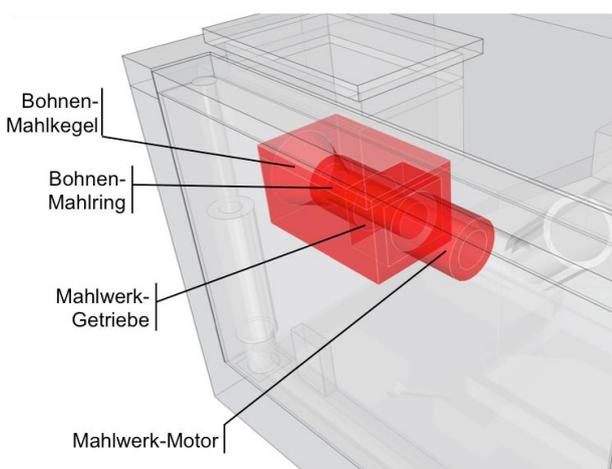
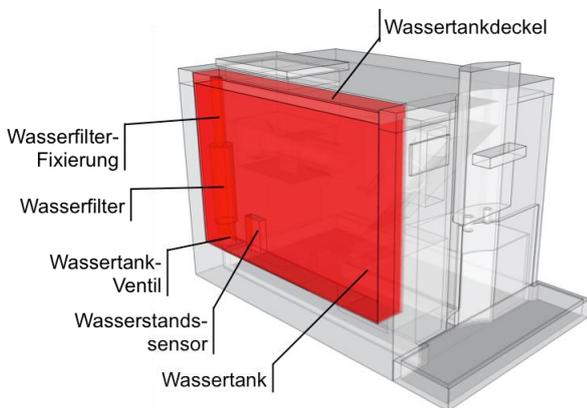
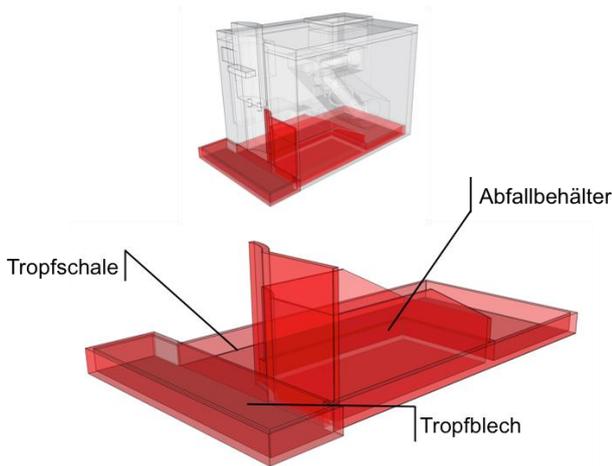
Als Modell des Kaffeeautomaten wurde die am IPK vorhandene Variante gewählt, so dass die Wahrscheinlichkeit groß ist, dass sie den Automaten schon einmal gesehen haben. In der Abbildung auf der linken Seite ist der Kaffeevollautomat schematisch dargestellt. Anhand dieser Abbildung werden folgend die Baugruppen, Systeme und Bauteile des Kaffeevollautomaten erläutert.

Gehäuse:

Das Gehäuse des Vollautomaten besteht aus insgesamt vier Bauteilen: der **Gehäusefront** und dem **Gehäuse-Oberteil** sowie den beiden **Gehäuse-Seitenteilen**.

Bohnenpeicher:

In dem Bohnenspeicher werden, wie der Name schon sagt, die ungemahlene Bohnen bis zur Verwendung aufbewahrt. Er besteht aus zwei Bauteilen: dem **Bohnenbehälter** sowie einer **Abdeckung**, um Schmutz und Staub von den Bohnen fern zu halten.



Auffangeinheit:

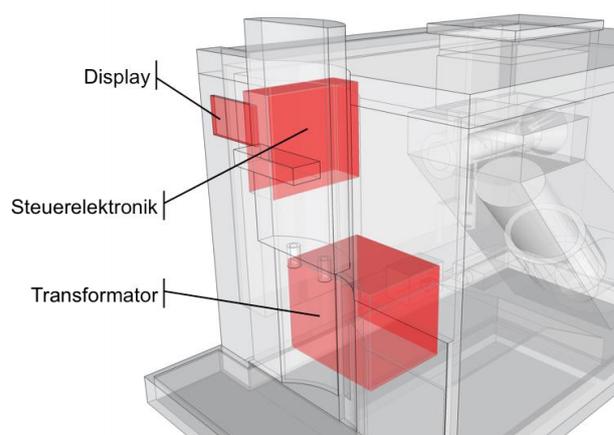
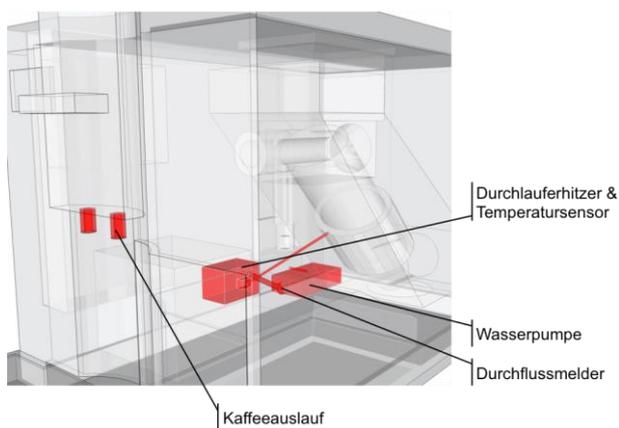
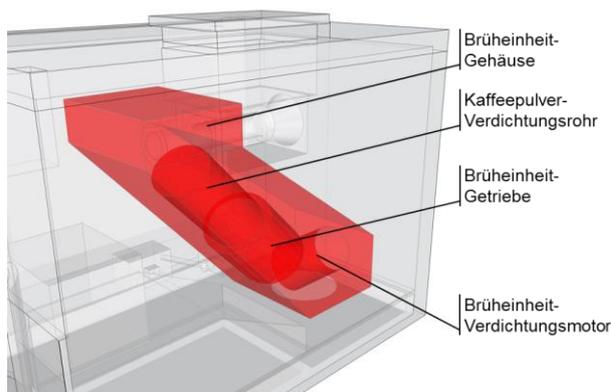
Die Auffangeinheit ist unten im Kaffeevollautomaten angeordnet. In dieser Baugruppe werden alle Abfälle gesammelt. Sie besteht aus der **Tropfschale**, in der das Schmutzwasser gesammelt wird. Im vorderen Bereich wird sie vom **Tropfblech** abgedeckt, auf das die Kaffeetasse gestellt wird. Mittig wird der **Abfallbehälter** auf die Tropfschale gesetzt, in dem der gemahlene und benutzte Kaffee gesammelt wird.

Wasserbehälter:

Die Baugruppe „Wasserbehälter“ umfasst eine vergleichsweise große Anzahl an Bauteilen. Das offensichtlichste ist der **Wassertank**, in dem Frischwasser zur Verfügung gestellt wird. Die Abdeckung gegen Schmutz und Staub wird durch den **Wassertankdeckel** gewährleistet. Damit das Wasser beim Nachfüllen nicht unten aus dem Tank herausläuft gibt es ein **Wassertank-Ventil**, welches nur im eingesetzten Zustand geöffnet ist. Zusätzlich verfügt der Wasserbehälter über einen **Wasserstandssensor**, der der Steuerelektronik meldet, wenn der Tank leer ist. Bei Bezug eines Kaffees wird das Wasser durch einen **Wasserfilter** gezogen, um eine Verkalkung des Vollautomaten zu verhindern. Dieser Wasserfilter wird mit Hilfe der **Wasserfilter-Fixierung** in Position gehalten.

Mahleinheit:

In der Mahleinheit werden die Bohnen aus dem Bohnen-Speicher zu Kaffeepulver gemahlen. Dabei werden die Bohnen zwischen den sich drehenden **Mahlkegel** und **Mahlring** zerkleinert. Der Antrieb erfolgt mit Hilfe des **Mahlwerk-Motors**. Um das für den Mahlvorgang notwendige Drehmoment aufbringen zu können, wird es mit dem **Mahlwerk-Getriebe** übersetzt.



Brüheinheit:

In der Baugruppe „Brüheinheit“ wird das gemahlene Kaffeepulver im **Kaffeepulver-Verdichtungsrohr** verdichtet. Dies wird mit Hilfe des **Brüheinheit-Verdichtungsmotors** und dem **Brüheinheit-Getriebe** durchgeführt. Sobald das Pulver verdichtet wurde wird es mit unter Druck stehendem Frischwasser durchsetzt. Um die Brüheinheit als Ganzes aus dem Vollautomaten entnehmen und reinigen zu können, verfügt sie über ein eigenes **Brüheinheit-Gehäuse**.

Fluidsystem:

Das Fluidsystem dient der Wasserversorgung innerhalb des Kaffeevollautomaten. Es besteht aus einer **Wasserpumpe**, die das Wasser aus dem Tank pumpt und einen hohen Druck herstellt. Dieses Wasser wird im **Durchlauferhitzer** erwärmt und von dort in die Brüheinheit gepumpt. Der fertige Kaffee wird über den **Kaffeeauslauf** in die Tasse geleitet. Zusätzlich verfügt das Fluidsystem noch über einen **Durchflussmelder** und einen **Temperatursensor**, die die Steuerelektronik mit Informationen zur Steuerung der Zubereitung versorgen. Die ebenfalls zugehörigen **Schläuche** verbinden die unterschiedlichen Bauteile und sind in der Abbildung links nicht dargestellt.

Elektrisches System:

Das elektrische System besteht in seiner vereinfachten Form insbesondere aus der **Steuerelektronik**, die die Sensordaten des Fluidsystems und des Wasserbehälters als Input verarbeitet und die Kaffee-Zubereitung steuert. Um Feedback an den Nutzer ausgeben zu können verfügt es über ein **Display**. Ebenfalls dem elektrischen System zugeordnet wird der **Transformator**, der die Spannung von 220 V auf 12 V transformiert. Kabelstränge werden im vorliegenden Beispiel vernachlässigt.

Sie erhalten nun eine Excel Matrix, in der die Funktionen und Bauteile, die Sie eben vorgestellt bekommen haben, enthalten sind. Bitte füllen Sie die Matrix genau so aus, wie Sie es bei dem Einführungsbeispiel getan haben.

3. Aufgabe „Fahrrad“



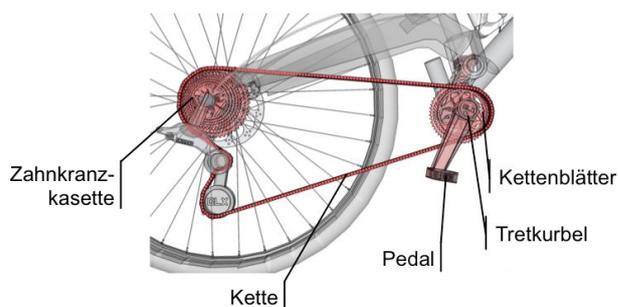
Das Fahrrad-Beispiel sollte jedem vom grundsätzlichen Aufbau her bekannt sein. In der Abbildung auf der linken Seite ist ein Fahrrad schematisch dargestellt. Anhand dieser Abbildung werden folgend die wichtigsten Baugruppen, Systeme und Bauteile des Fahrrads erläutert.

Rahmen:

Der Rahmen ist das Gestell des Fahrrads. Er trägt das Gewicht des Fahrers, das er auf die Räder überträgt und an ihm sind alle Komponenten befestigt, die für weitere Funktionen des Fahrrades (zum Beispiel Lenkung und Antrieb) benötigt werden. Im vorliegenden Beispiel besteht der Rahmen aus zwei Bauteilen: dem **Grundrahmen** und dem **Hinterbau**. Dieser Aufbau ist so gewählt, um eine Federung des Hinterrads zu realisieren, die jedoch im Folgenden vernachlässigt wird.

Lenkung:

Die Lenkung dient der Beeinflussung der Fahrtrichtung des Fahrrads. Sie besteht aus dem **Lenker**, welcher über den **Vorbau** mit der **Federgabel** verbunden ist. Die drehbar gelagerte Verbindung zum Rahmen wird über den **Steuersatz** hergestellt.

**Antriebseinheit:**

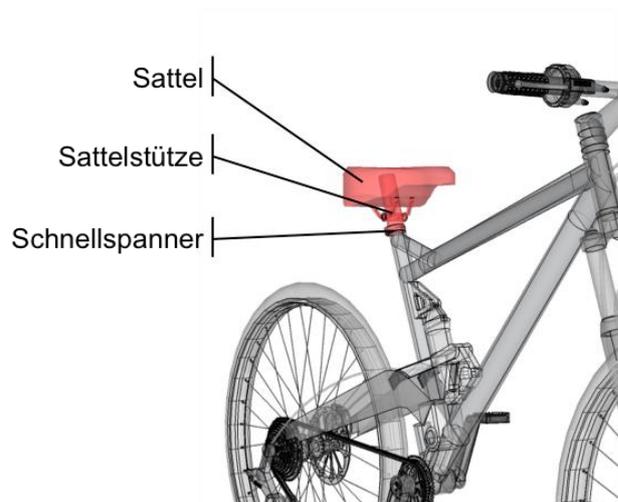
Die Antriebseinheit realisiert den Vortrieb des Fahrrads, indem die Fußkraft des Fahrers auf das Hinterrad übertragen wird. Sie besteht aus den **Pedalen** und **Tretkurbeln**, die mit den **Kettenblättern** (Zahnräder vorne) verbunden sind. Die Fußkraft wird über diese Bauteile auf die **Kette** und von dieser auf die **Zahnkranzkassette** (Zahnräder hinten) übertragen.

**Schaltung:**

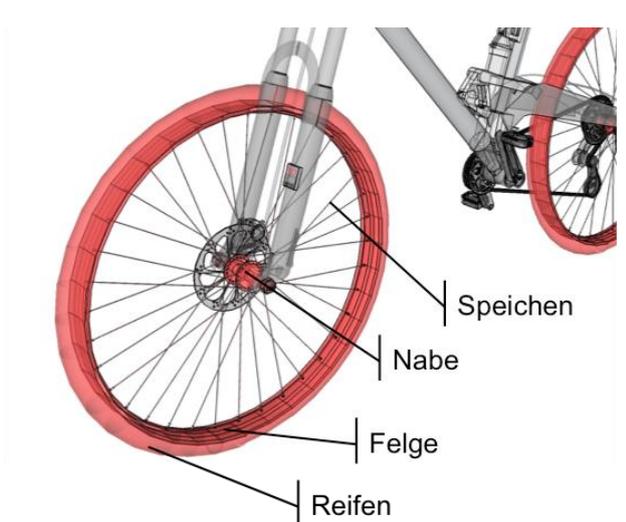
Mit Hilfe der Schaltung kann das Übersetzungsverhältnis der Antriebseinheit verändert werden. Sie besteht aus dem **Schaltwerk**, welches die seitliche Verschiebung der Kette auf der Zahnkranzkassette (Zahnräder hinten) realisiert und dem **Umwerfer**, der die gleiche Funktion für die Kettenblätter (Zahnräder vorne) hat. Die Wahl des Gangs erfolgt im vorliegenden Beispiel mit zwei **Schaltdrehgriffen**, die über **Bowdenzüge** mit den Schaltvorrichtungen verbunden sind.

**Bremssystem:**

An dem Fahrrad im Beispiel wird ein Scheibenbremssystem verwendet. Es besteht aus den beiden **Scheibenbremsen** am Vorder- und Hinterrad, welche von den **Bremssätteln** umschlossen werden. Diese sind über **Hydraulikleitungen** mit den **Bremshebeln** am Lenker verbunden.

**Sitzeinheit:**

Die Sitzeinheit besteht aus dem **Sattel** der fest mit der **Sattelstütze** verbunden ist. Über den **Schnellspanner** sind diese mit dem Rahmen verbunden und können einfach in der Höhe verstellt werden.



Vorder- und Hinterrad:

Da die Zahnkranzkassette der Antriebseinheit und die Bremsscheiben dem Bremssystem zugeordnet werden sind das Vorder- und das Hinterrad in diesem Beispiel baugleich. Sie bestehen aus einer **Nabe**, die am Rahmen bzw. der Federgabel befestigt wird. Die **Felge**, die den **Reifen** aufnimmt wird mit der Nabe über **Speichen** verbunden.

Sie erhalten nun eine Excel Matrix, in der die Funktionen und Bauteile, die Sie eben vorgestellt bekommen haben, enthalten sind. Bitte füllen Sie die Matrix genau so aus, wie Sie es bei dem Einführungsbeispiel getan haben.

ARTEFAKTE DES BEISPIELPRODUKTS „HAUS“

Bauteile abstrakt	Funktionen abstrakt
Wohnzimmer	Lebensmittel verarbeiten
Badezimmer	Freizeit genießen
Schlafzimmer	Körperpflege ermöglichen
Küche	Kleidung lagern
	Schlafgelegenheit bieten

Bauteile detailliert	Funktionen detailliert
Sofa	Essen erwärmen
Fernseher	Lebensmittel kühlen
Dusche	Videos anzeigen
Waschbecken	Körper waschen
Bett	Zahnputzabwasser aufnehmen
Kleiderschrank	Kleidung lagern
Herd	Schlafgelegenheit bieten
Kühlschrank	

ARTEFAKTE DES BEISPIELPRODUKTS „KAFFEEVOLLAUTOMAT“

Bauteile abstrakt	Funktionen abstrakt
Gehäuse	Bohnen aufbewahren
Bohnenspeicher	Umwelteinflüsse fernhalten
Auffangeinheit	Abfall sammeln
Wasserbehälter	Frischwasser bereitstellen
Mahleinheit	Kaffee zubereiten
Brüheinheit	Zubereitung steuern
Fluidsystem	Benutzer warnen
Elektrisches System	

Bauteile detailliert	Funktionen detailliert
Gehäuse-Oberteil	Bohnen aufbewahren
Gehäusefront	Umwelteinflüsse fernhalten
Gehäuseseite links	Schmutzwasser sammeln
Gehäuseseite rechts	Trester sammeln
Bohnenbehälter	Frischwasser speichern
Bohnenbehälterabdeckung	Frischwasser filtern
Tropfschale	Frischwasser vor Umwelt schützen
Tropfblech	Frischwasser zuführen
Abfallbehälter	Bohnen mahlen
Wassertank	Kaffeepulver verdichten
Wassertankdeckel	Wasser erhitzen
Wasserstandssensor	Wasser verdichten
Wassertank-Ventil	Kaffee in Tasse füllen
Wasserfilter	Zubereitung steuern
Wasserfilter-Fixierung	Warnen wenn kein Frischwasser vorhanden
Bohnen-Mahlkegel	Warnen, wenn Schmutzwasser voll
Bohnen-Mahlring	Warnen, wenn Tresterbehälter voll
Mahlwerk-Getriebe	
Mahlwerk-Motor	
Brüheinheit-Gehäuse	
Brüheinheit-Getriebe	
Brüheinheit-Verdichtungsmotor	
Kaffee-Verdichtungsrohr	
Wasserpumpe	
Durchflussmelder	
Durchlauferhitzer	
Temperatursensor	
Kaffeeauslauf	

Bauteile detailliert	Funktionen detailliert
Schläuche	
Display	
Steuerelektronik	
Transformator	

ARTEFAKTE DES BEISPIELPRODUKTS „FAHRRAD“

Bauteile abstrakt	Funktionen abstrakt
Rahmen	Antrieb realisieren
Lenkung	Bremsen realisieren
Antriebseinheit	Lenken ermöglichen
Schaltung	Variable Übersetzung realisieren
Bremssystem	
Sitzeinheit	
Vorderrad	
Hinterrad	

Bauteile detailliert	Funktionen detailliert
Grundrahmen	Fußkraft aufnehmen
Hinterbau	Fußkraft übertragen
Lenker	Fußkraft auf Hinterrad übertragen
Vorbau	Bremskraft aufnehmen
Steuersatz	Bremskraft übertragen
Federgabel	Bremskraft auf Räder übertragen
Kettenblätter	Lenkbewegung aufnehmen
Tretkurbel	Lenkbewegung auf Rad übertragen
Pedal	Schaltkraft aufnehmen
Kette	Schaltkraft übertragen
Zahnkranzkassette	Übersetzung wechseln
Schaltdrehgriffe	
Bowdenzüge	
Umwerfer	
Schaltwerk	
Bremshebel	
Hydraulikleitung	
Bremssattel	
Scheibenbremse	
Sattel	
Sattelstütze	

Bauteile detailliert	Funktionen detailliert
Schnellspanner	
Nabe	
Speichen	
Felge	
Reifen	

B. MATERIAL ZUR STUDIE „EVALUATION DER METHODEN ECOTRACING UND TRACEVALUATION“

STUDIENEINFÜHRUNG UND AUFGABENSTELLUNG

Studie zu Abhängigkeiten zwischen Modellen

Bitte füllen Sie zunächst die untenstehenden Felder aus.

Alter: Geschlecht:

Studiengang/Beruf:

Einführung in das Thema

Im Rahmen der Entwicklung von Produkten werden viele unterschiedliche Modelle erstellt (siehe Abbildung 1). Die bekanntesten Modelle beschreiben die Anforderungen (was soll das Produkt leisten), die Funktionen (wie sollen die Anforderungen realisiert werden) und die Produktstruktur (welche geometrischen Baugruppen und -teile werden benötigt). Die genannten Modelle bestehen aus mehreren Elementen, die häufig hierarchisch strukturiert sind (z.B. Eltern-Kind-Beziehung).

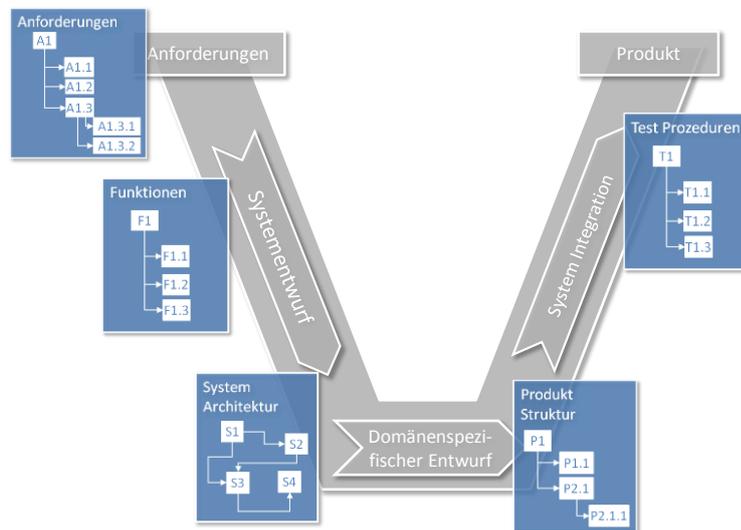


Abbildung 1: Entwicklung unterschiedlicher Modelle entlang des Entwicklungsprozesses

Zwischen den Modellelementen bestehen viele inhaltliche Zusammenhänge. Eine Funktion beispielsweise erfüllt eine Anforderung und wird durch ein oder mehrere Produktstrukturelemente umgesetzt. Zum Beispiel wird die Funktion „Klima regeln“ vom Produktstrukturelement „Klimaanlage“ umgesetzt. „Klima regeln“ oder „Klima-

anlage“ können hierbei noch beliebig viele weitere Detaillierungen (Kindelemente), wie beispielsweise „heizen“, „kühlen“ oder „Luftfilter“ enthalten.

Die inhaltlichen Zusammenhänge zwischen den Elementen werden durch sogenannte Tracelinks gespeichert, um u. a. bei einer Änderung während der Produktentwicklung nachverfolgen zu können, auf welche anderen Modellelemente diese Einfluss haben könnte. Die roten Linien in Abbildung 2 repräsentieren solche Tracelinks. Die Gesamtheit aller für ein Produkt existierenden Tracelinks bildet das **Abhängigkeitsmodell**.

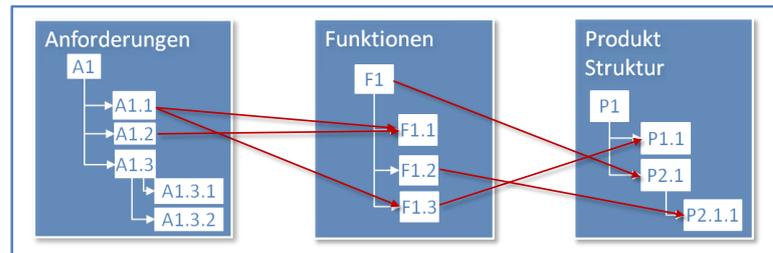


Abbildung 2: Abhängigkeiten zwischen Modellelementen

Zu dieser Studie

In dieser Studie sollen Sie das Anlegen, Überprüfen und Nutzen des Abhängigkeitsmodells durchführen.

Für das relativ einfache und alltägliche Produktbeispiel eines Fahrrads, sollen Sie zunächst mittels eines Softwaretools ein solches Abhängigkeitsmodell erstellen (eine Übersicht einiger eventuell unklarer Fahrradteile können Sie auch der ausgelegten Abbildung entnehmen). Im nächsten Schritt erhalten Sie ein vorgegebenes Abhängigkeitsmodell. In verschiedenen Szenarios sollen Sie die Auswirkungen von Änderungen nachempfinden. Dies geschieht mit Hilfe eines Softwaretools zur Darstellung des Abhängigkeitsmodells (Viewer). Abschließend sollen Sie in einer weiteren Aufgabe erneut entscheiden, welche Elemente des Fahrradbeispiels ihrer Meinung nach durch eine Änderung an einem bestimmten Modellelements beeinflusst werden. Für die letzte Aufgabe steht Ihnen jedoch lediglich eine einfache Listendarstellung der Modellelemente zur Verfügung.

Vor Beginn der Aufgaben erhalten Sie Gelegenheit sich mit der Bedienung der beiden zu nutzenden Softwarewerkzeuge vertraut zu machen.

Lesen Sie bitte bei jeder Aufgabe zunächst die vollständige Aufgabenstellung.

Falls Sie an dieser Stelle offene Fragen haben, zögern Sie bitte nicht, diese jetzt dem Betreuer zu stellen.

Aufgabe 1

In dieser Aufgabe erlernen Sie den Umgang mit dem Softwaretool EcoTracer (Abbildung 3), sowie dem entsprechenden Viewer (Abbildung 4). Mit dem EcoTracer können Sie zwischen den Elementen zweier Teilmodelle (z.B. Anforderungen und Funktionen) Verknüpfungen anlegen. Das Programm schlägt Ihnen Schritt für Schritt Elementepaare für eine Verknüpfung vor. Sie haben in jedem Schritt die Wahl, die vorgeschlagene Paarung als Tracelink zu *bestätigen* (Button unten rechts), sie zu *verwerfen* (Button unten Mitte) oder die Paarung zunächst zu *überspringen* und später zu bearbeiten (Button unten links). Über die notwendigen Vorbereitungen, um den Ablauf zu starten, werden Sie am Platz informiert. Sie haben nun zunächst etwas Zeit um den Ablauf des Programms zu testen und zu verstehen. In dieser Einführung brauchen Sie sich keine Gedanken um Fehler oder falsche Entscheidungen bei der Modellierung zu machen.

Prinzipiell gilt:

- Sind Sie sich sicher, dass zwischen den markierten Elementen **eine Abhängigkeit** besteht, klicken Sie auf **bestätigen**. (Beispiel: Die Anforderung „Sicherheit des Fahrers gewährleisten“ eines Kfz sollte definitiv mit dem Bauteil „Airbag“ verknüpft werden.)
- Sind Sie sich sicher, dass zwischen den markierten Elementen **keine Abhängigkeit** besteht, klicken sie auf **verwerfen**. (Beispiel: Die Anforderung „Sicherheit des Fahrers gewährleisten“ eines Kfz hat keinen direkten Bezug zu der Funktion „Klima regeln“ und sollte daher als Verknüpfung abgelehnt werden.)
- Bei **Unsicherheiten** können Sie das Elementepaar **überspringen**.

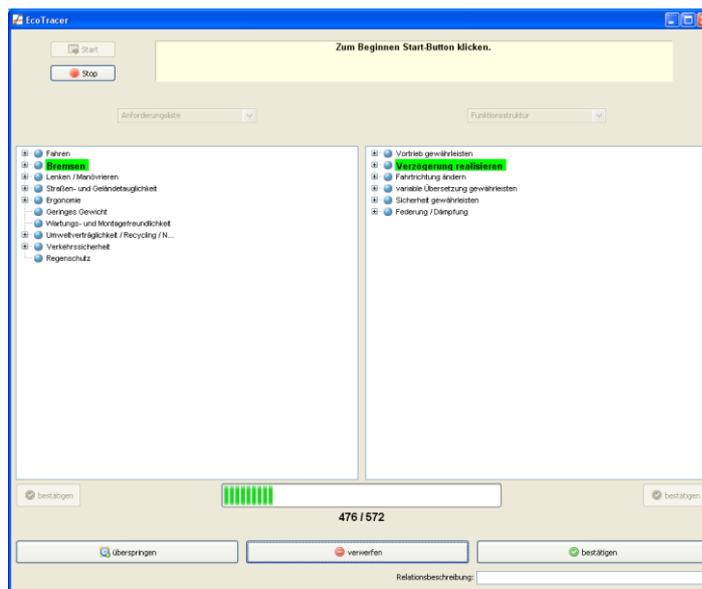


Abbildung 3: grafische Oberfläche des EcoTracers

Sobald Sie sich mit der Bedienung des EcoTracers vertraut gemacht haben, bekommen Sie Gelegenheit, die Funktionalitäten des Viewers (Abbildung 4) kennenzulernen.

Mit diesem Tool haben Sie die Möglichkeit, ein Abhängigkeitsmodell übersichtlich zu betrachten und mit ihm zu interagieren.

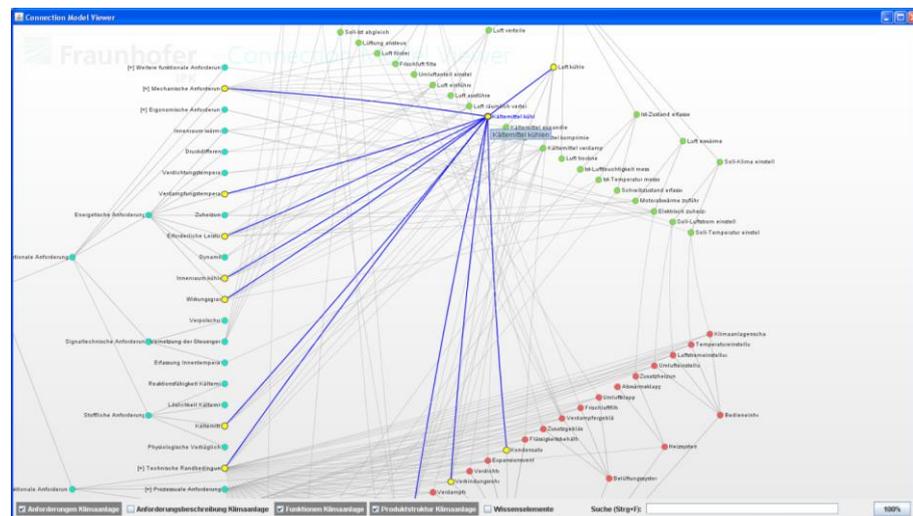


Abbildung 4: Screenshot des Viewers

Für jedes Modell wird die gesamte Modellstruktur ihrer Hierarchie entsprechend dargestellt. Verknüpfungen werden nur auf der jeweils tiefstmöglichen Ebene dargestellt. Die Verknüpfungslinien befinden sich also hauptsächlich im freien Raum zwischen den Modellen. Die hierarchisch höher gelegenen Elemente eines Modells „wachsen“ nach außen. Sie können also stets nachverfolgen, zu welchem Modell ein Element gehört.

Bitte machen Sie sich zumindest mit den folgenden Interaktionsfunktionen vertraut:

- Vergrößern/Verkleinern der Modelle
 - Verschieben der Modelle
 - Anzweifeln einer Verknüpfung / rückgängig machen einer Anzweiflung
 - (De)Selektion eines Elements
 - Verändern der Schriftgröße von Elementnamen
 - Ein- Ausblenden eines Modells
 -
- Eine Übersicht der Interaktionen ist am Platz ausgelegt.

Aufgabe 2

Nun sollen Sie, wie in Aufgabe 1 getestet, ein vollständiges Abhängigkeitsmodell für das Fahrradbeispiel erstellen. Laden Sie hierzu nacheinander die Modellkombinationen

1. Anforderungsliste-Funktionsstruktur,
2. Funktionsstruktur-Produktstruktur und
3. Anforderungsliste-Produktstruktur

und arbeiten Sie diese mit dem EcoTracer vollständig ab.

Für jede Modellkombination haben Sie 15 Minuten Zeit.

Es gilt nach wie vor:

- Sind Sie sich sicher, dass zwischen den markierten Elementen **eine Abhängigkeit** besteht, klicken Sie auf **bestätigen**.
- Sind Sie sich sicher, dass zwischen den markierten Elementen **keine Abhängigkeit** besteht, klicken sie auf **verwerfen**.
- Bei **Unsicherheiten** können Sie das Elementepaar **überspringen**.

Haben Sie alle Elemente einer Modellkombination bewertet bzw. sind sich bei den übriggebliebenen Verknüpfungen nicht sicher, klicken Sie auf *Stop* und laden die nächste Modellkombination.

Achtung: Es gibt keine Möglichkeit Aktionen nachträglich zu ändern oder zurückzunehmen. Überlegen Sie sich ihre Interaktionen also bitte sorgfältig.

Sobald Sie die Aufgabe abgeschlossen haben, geben Sie bitte Bescheid.

Aufgabe 3

In dieser Aufgabe arbeiten Sie mit dem Viewer und einem Abhängigkeitsmodell, das bereits für Sie modelliert wurde. Das vorgegebene Abhängigkeitsmodell kann Fehler enthalten. Sollten Sie Fehler im Modell entdecken, markieren Sie diese bitte direkt im Viewer (wie geübt), damit wir die Qualität des Modells sukzessive verbessern können. Versetzen Sie sich bitte in die unterschiedlichen Situationen und bearbeiten Sie die beschriebenen Aufgaben.

(3a)

Sie sind Verantwortlicher für die Fertigungsplanung des neuen Fahrradmodells Ihrer Firma. Sie erhalten Informationen über **Anforderungen**, die geändert wurden. Sie müssen nun die Fertigungsabteilung von den Änderungen in Kenntnis setzen. Um un-

nötigen Planungsaufwand zu verhindern, müssen Sie zunächst die von der Änderung betroffenen **Bauteile** identifizieren.

Folgende Anforderungen haben sich geändert.

- 1) Übersetzung mittels Kettenschaltung
- 2) Muss mit Handkraft betrieben werden können
- 3) Hinterradantrieb mittels Fußkurbel und Kette

Identifizieren Sie anhand der Verknüpfungslinien die von den Änderungen betroffenen Bauteile und notieren Sie die relevanten. Es genügt die Nummerierung des betroffenen Elementes zu notieren (z.B: A1.2.3).

(3b)

Sie sind Verantwortlicher der Einkaufsabteilung. Ihre Firma hat alternative Angebote für folgende Bauteile erhalten.

- 1) Hydraulische Scheibenbremsen
- 2) Hydraulikleitungen
- 3) Bremshebel
- 4) Bremsscheiben

Mit den aufgeführten Bauteilalternativen könnten die Kosten für die Fahrradherstellung drastisch reduziert werden. Es muss jedoch zunächst geprüft werden, ob die **Funktionen und Anforderungen** auch mit den neuen Bauteilen weiterhin erfüllt werden können. Hierfür identifizieren und notieren Sie bitte alle von den Bauteiländerungen betroffenen Anforderungen und Funktionen. Es genügt die Nummerierung des betroffenen Elementes zu notieren (z.B: A1.2.3).

Sobald Sie die Aufgabe abgeschlossen haben, geben Sie bitte Bescheid.

Aufgabe 4

Bei der Entwicklung von Produkten kommt es häufig zu Änderungen während des laufenden Prozesses. Ändert sich ein Element, sind davon häufig auch andere Elemente direkt betroffen. Ändert sich bspw. die Anforderung über das „zulässige Maximalgewicht“ eines Fahrrads, muss häufig das Material des Rahmens angepasst werden– bei der nun folgende Analyse müsste also das Element „Rahmen“ der Produktstruktur markiert werden.

Ihre Aufgabe besteht darin, nach Vorgabe einer Änderung an einem Element, zu bestimmen, welche Elemente (in allen drei Modellen) von dieser Änderung betroffen sein könnten.

In dieser letzten Aufgabe erhalten Sie erneut das Modell des Fahrrads ohne Verknüpfungen. Diesmal arbeiten Sie mit einer einfachen Listendarstellung der Anforderungen, Funktionen und Produktstruktur (enthält die Bauteile) des Fahrrads.

Kommentare zur Studie

AGGREGIERTE ANZAHL MODELLIERTER TRACELINKS ZWISCHEN ANFORDERUNGEN UND PRODUKTELEMENTEN

	0 – 1 bzw. 18 – 19 Tracelinks
	2 – 3 bzw. 16 – 17 Tracelinks
	4 – 5 bzw. 14 – 15 Tracelinks
	6 – 7 bzw. 12 – 13 Tracelinks
	8 – 11 Tracelinks

	A1.1 Vortrieb	A1.1.1 Hinterradantrieb mittels Fußku...	A1.1.2 Übersetzung mittels Kettenscha...	A1.2 Bremsen	A1.2.1 Verschmutzungsrisiko minimieren	A1.2.2 zuverlässiges Bremsen bei Nässe	A1.2.3 muss mit Handkraft betrieben w...	A1.2.4 geringer Verschleiß von Kompon...	A1.3 sicheres Lenken / Manövrieren	A1.4 geländetaugliche Federung und Üb...	A1.4.1 Unebenheiten im Gelände und im...	A1.4.2 Verschiedene Übersetzungen	A1.5 Beleuchtung vorsehen
P1.1 Rahmen	7	6	5	7	5	2	2	2	8	12	8	3	9
P1.2 Steuereinheit	5	1	2	4	2	2	3	1	19	17	14	2	2
P1.2.1 hydraulische Federgabel	1	1	1	1	1		1		12	16	14	1	1
P1.2.2 Vorbau	2	1	1	2	1		2		16	3		1	
P1.2.3 Lenkstange	2	1	1	3			2		19	3		1	1
P1.2.4 Steuersatz	2	1	1	1			1		17	4		1	
P1.2.5 Vorderrad	4	1	1	4		2	2		15	10	7	1	
P1.3 Antriebseinheit	19	19	18	5	2	3	2	4	4	12	2	11	1
P1.3.1 Tretkurbel und Kettenblätter	19	19	14	2	1	1	1		1	7		6	
P1.3.2 Kette	19	19	15	2	1	1	1		1	5		4	
P1.3.3 Hinterrad	19	18	9	5		3	1	2	2	9	2	2	1
P1.3.4 Zahnkranzkassette	19	17	17	1		1	1		1	10		10	
P1.4 Schalteinheit	11	5	11						2	17	1	16	1
P1.4.1 Umwerfer (Übersetzung vorne)	10	4	10						1	16	1	14	
P1.4.2 Schaltwerk (Übersetzung hinten)	10	5	10						2	16	1	15	1
P1.4.3.2 Bowdenzüge (für Schaltwerk u...	8	2	7						1	13		12	
P1.4.4.2 Schalthebel (für Schaltwerk ...	7	1	7						1	14		14	1
P1.5 Bremseinheit	1			19	6	18	17	15	7	4	1		
P1.5.1 2 hydraulische Scheibenbremsen...				17	3	16	5	12	6	2	1		
P1.5.2 2 Bremshebel (Vorderrad- und H...	1			18	1	9	16	9	7	3	1		
P1.5.3 2 Hydraulikleitungen				18	2	11	7	10	5	1	1		
P1.5.4 2 Bremsscheiben	1			19	4	16	4	14	7	3	1		
P1.6 Sitzeinheit	1								1	10	9		2
P1.6.1 Patentsattelstütze und Sattels...	1									7	7		2
P1.6.2 Sattel	1								1	10	9		
P1.7 Sonstige Komponenten	2			5	4					2			19
P1.7.1 Reflektoren	1												17
P1.7.2 Schutzbleche	1			4	4					1			2
P1.7.3 Beleuchtung	1												19

AGGREGIERTE ANZAHL MODELLIRTER TRACELINKS ZWISCHEN FUNKTIONEN UND PRODUKTELEMENTEN

- 0 – 1 bzw. 18 – 19 Tracelinks
- 2 – 3 bzw. 16 – 17 Tracelinks
- 4 – 5 bzw. 14 – 15 Tracelinks
- 6 – 7 bzw. 12 – 13 Tracelinks
- 8 – 11 Tracelinks

	F1.1 Vortrieb gewährleisten	F1.1.1 Fußkraft aufnehmen	F1.1.2 Antriebskraft leiten	F1.1.3 Kraft / Moment wandeln	F1.1.4 Rotationsmoment auf Straße übe...	F1.2 Verzögerung realisieren	F1.2.1 Handkraft aufnehmen	F1.2.2 Kraft leiten	F1.2.3 Bremskraft auf Hinterrad / Str...	F1.3 Fahrtrichtung ändern	F1.3.1 Steuersignal aufnehmen	F1.3.2 Richtungsänderung ausführen	F1.4 variable Übersetzung gewährleisten	F1.4.1 Fingerkraft aufnehmen	F1.4.2 Kraft leiten	F1.4.3 Übersetzung wechseln	F1.5 Sichtbarkeit gewährleisten	F1.5.1 unmittelbare Umgebung beleuchten	F1.5.2 Licht reflektieren	F1.6 Federung / Dämpfung
P1.1 Rahmen	5	4	4	2	2	5	2	1	1	7	3	2	4		1		5	2	3	14
P1.2 Steuereinheit	7	2	3	3	3	4	2	2	1	19	18	18	2	1	1	1	1			15
P1.2.1 hydraulische Federgabel						2		1		7	2	4	1	1	1	1	1			15
P1.2.2 Vorbau						2		1		16	12	11	1	1	1	1	1			4
P1.2.3 Lenkstange	1		1	1		1		1		19	17	14	1	1	1	1	1			3
P1.2.4 Steuersatz	1	1	1	1		1		1		18	12	12	1	1	1	1				4
P1.2.5 Vorderrad	7	2	2	3	3	4	1	2	1	15	4	14	1	1	1	1	1			11
P1.3 Antriebseinheit	19	19	19	19	18	3	1	1	1	2		1	13	2	7	11	1			5
P1.3.1 Tretkurbel und Kettenblätter	19	19	18	18	6	2	1	1	1	2		1	12	1	4	7	1			1
P1.3.2 Kette	19	12	19	8	7	1	1		1	2		1	10	1	4	6	1			
P1.3.3 Hinterrad	19	8	14	10	18	3	1	1	1	2		1	6	1	2	4	1			4
P1.3.4 Zahnkranzkassette	19	10	15	14	9	1	1	1		2		1	12	1	5	11	1			
P1.4 Schalteinheit	14	3	5	7	2	2				1		1	19	16	15	17				
P1.4.1 Umwerfer (Übersetzung vorne)	11	1	5	5	1	1				1			18	1	6	16				
P1.4.2 Schaltwerk (Übersetzung hinten)	10	1	3	5		2				1			19	2	6	16				
P1.4.3 2 Bowdenzüge (für Schaltwerk u.	6	1	1	2		2				1			18	6	13	9				

	F1.1 Vortrieb gewährleisten	F1.1.1 Fußkraft aufnehmen	F1.1.2 Antriebskraft leiten	F1.1.3 Kraft / Moment wandeln	F1.1.4 Rotationsmoment auf Straße über...	F1.2 Verzögerung realisieren	F1.2.1 Handkraft aufnehmen	F1.2.2 Kraft leiten	F1.2.3 Bremskraft auf Hinterrad / Str...	F1.3 Fahrtrichtung ändern	F1.3.1 Steuersignal aufnehmen	F1.3.2 Richtungsänderung ausführen	F1.4 variable Übersetzung gewährleisten	F1.4.1 Fingerkraft aufnehmen	F1.4.2 Kraft leiten	F1.4.3 Übersetzung wechseln	F1.5 Sichtbarkeit gewährleisten	F1.5.1 unmittelbare Umgebung beleuchten	F1.5.2 Licht reflektieren	F1.6 Federung / Dämpfung
P1.4.4 2 Schalthebel (für Schaltwerk ...	8		2	3	1	2				1			19	15	8	10				
P1.5 Bremsseinheit						19	17	17	18	1		1					1			1
P1.5.1 2 hydraulische Scheibenbrems.						18	2	6	18	1		1					1			
P1.5.2 2 Bremshebel (Vorderrad- und H.						19	16	12	8	1		1					1			1
P1.5.3 2 Hydraulikleitungen						18	4	17	11	1		1					1			1
P1.5.4 2 Bremscheiben						19	3	7	18	1		1					1			
P1.6 Sitzeinheit	1					1											1	1	1	13
P1.6.1 Patentsattelstütze und Sattels...	1					1											1	1	1	9
P1.6.2 Sattel	1					1														13
P1.7 Sonstige Komponenten													1	1		1	19	17	18	1
P1.7.1 Reflektoren																	19	7	18	
P1.7.2 Schutzbleche																	3		1	1
P1.7.3 Beleuchtung																	19	17	5	

C. MATERIAL ZUR EVALUATION DER METHODE TRACELEGACY

TRACELINKS ZWISCHEN ANFORDERUNGS- UND FUNKTIONSHIERARCHIE DES FAHR-
RADS (VERGANGENES PROJEKT)

	Vortrieb	Hinterradantrieb mittels Fußkurbel un...	Übersetzung mittels Kettenschaltung	Bremsen	Verschmutzungsrisiko minimieren	zuverlässiges Bremsen bei Nässe	muss mit Handkraft betrieben werden k...	geringer Verschleiß von Komponenten	sicheres Lenken / Manövrieren	Geländetauglichkeit	Unebenheiten im Gelände und im Fahrbe...	Verschiedene Übersetzungen	Beleuchtung vorsehen
Vortrieb gewährleisten	1	1											
Rotationsmoment aufnehmen	1	1											
Rotationsmoment leiten	1	1											
Rotationsmoment wandeln	1	1											
Rotationsmoment auf Straße übertragen	1	1											
Verzögerung realisieren				1		1	1	1					
Bremskraft aufnehmen				1			1						
Kraft leiten				1									
Bremskraft auf Straße übertragen				1		1							
Fahrtrichtung ändern									1				
Steuersignal aufnehmen									1				
Signal in Richtungsänderung umsetzen									1				
variable Übersetzung gewährleisten	1	1	1							1		1	
Schaltkraft aufnehmen													
Kraft leiten													
Übersetzung wechseln	1		1							1		1	
Sichtbarkeit gewährleisten													1
unmittelbare Umgebung beleuchten													1
Licht reflektieren													1
Federung / Dämpfung										1	1		

TRACELINKS ZWISCHEN ANFORDERUNGSHIERARCHIE UND PRODUKTSTRUKTUR DES FAHRRADS (VERGANGENES PROJEKT)

	Vortrieb	Hinterradantrieb mittels Fußkurbel un...	Übersetzung mittels Kettenschaltung	Bremsen	Verschmutzungsrisiko minimieren	zuverlässiges Bremsen bei Nässe	muss mit Handkraft betrieben werden k...	geringer Verschleiß von Komponenten	sicheres Lenken / Manövrieren	Geländetauglichkeit	Unebenheiten im Gelände und im Fahrbe...	Verschiedene Übersetzungen	Beleuchtung vorsehen
Rahmen													
Steuereinheit									1	1	1		
hydraulische Federgabel									1	1	1		
Vorbau													
Lenkstange									1				
Steuersatz									1				
Vorderrad													
Antriebseinheit	1	1	1							1		1	
Tretkurbel	1	1											
Kette	1	1											
Hinterrad	1												
Zahnkranzkassette	1	1	1							1		1	
Schalteinheit	1		1							1		1	
Umwerfer (Übersetzung vorne)	1		1							1		1	
Schaltwerk (Übersetzung hinten)	1		1							1		1	
2 Bowdenzüge (für Schaltwerk und Umwe...	1		1							1		1	
2 Schalthebel (für Schaltwerk und Umw...	1		1							1		1	
Bremseinheit				1	1	1	1	1					
2 hydraulische Scheibenbremsen				1	1	1		1					
2 Bremshebel				1			1						
2 Hydraulikleitungen				1									
2 Brems Scheiben				1	1	1		1					
Sitzeinheit													
Patentsattelstütze und Sattelstange													
Sattel													
Sonstige Komponenten													1
Reflektoren													
Schutzbleche													
Beleuchtung													1

TRACELINKS ZWISCHEN FUNKTIONSHIERARCHIE UND PRODUKTSTRUKTUR DES FAHR-
RADS (VERGANGENES PROJEKT)

	Vortrieb gewährleisten	Rotationsmoment aufnehmen	Rotationsmoment leiten	Rotationsmoment wandeln	Rotationsmoment auf Straße übertragen	Verzögerung realisieren	Bremskraft aufnehmen	Kraft leiten	Bremskraft auf Straße übertragen	Fahrtrichtung ändern	Steuersignal aufnehmen	Signal in Richtungsänderung umsetzen	variable Übersetzung gewährleisten	Schaltkraft aufnehmen	Kraft leiten	Übersetzung wechseln	Sichtbarkeit gewährleisten	unmittelbare Umgebung beleuchten	Licht reflektieren	Federung / Dämpfung
Rahmen																				
Steuereinheit										1	1	1								1
hydraulische Federgabel										1		1								1
Vorbau										1		1								
Lenkstange										1	1	1								
Steuersatz										1		1								
Vorderrad										1		1								
Antriebseinheit	1	1	1	1	1								1			1				
Tretkurbel	1	1																		
Kette	1		1																	
Hinterrad	1				1															
Zahnkranzkassette	1			1									1			1				
Schalteinheit													1	1	1	1				
Umwerfer (Übersetzung vorne)													1			1				
Schaltwerk (Übersetzung hinten)													1			1				
2 Bowdenzüge (für Schaltwerk und Umwe...													1		1					
2 Schalthebel (für Schaltwerk und Umw...													1	1						
Bremseinheit						1	1	1	1											
2 hydraulische Scheibenbremsen						1			1											
2 Bremshebel						1	1													
2 Hydraulikleitungen						1		1												
2 Bremsscheiben						1			1											
Sitzeinheit																				
Patentsattelstütze und Sattelstange																				
Sattel																				
Sonstige Komponenten																	1	1	1	

	Vortrieb gewährleisten	Rotationsmoment aufnehmen	Rotationsmoment leiten	Rotationsmoment wandeln	Rotationsmoment auf Straße übertragen	Verzögerung realisieren	Bremskraft aufnehmen	Kraft leiten	Bremskraft auf Straße übertragen	Fahrtrichtung ändern	Steuersignal aufnehmen	Signal in Richtungsänderung umsetzen	variable Übersetzung gewährleisten	Schaltkraft aufnehmen	Kraft leiten	Übersetzung wechseln	Sichtbarkeit gewährleisten	unmittelbare Umgebung beleuchten	Licht reflektieren	Federung / Dämpfung
Reflektoren																	1		1	
Schutzbleche																				
Beleuchtung																	1	1		

ANFORDERUNGSHIERARCHIE DES PEDELECS (AKTUELLES PROJEKT)

- 1 Antrieb
 - 1.1 Pedal
 - 1.2 E Motor
 - 1.3 Schaltung
 - 1.4 Höchstgeschwindigkeit
- 2 Geometrie
 - 2.1 Reifen
 - 2.2 Breite
 - 2.3 Wendekreis
 - 2.4 Höhe
 - 2.5 Länge
- 3 Fahrer
 - 3.1 Größe
 - 3.2 Gewicht
 - 3.3 Führerschein
- 4 Energie und Widerstand
 - 4.1 Energieverbrauch ohne Motor Unterstützung
 - 4.2 Energieverbrauch mit Motor Unterstützung
 - 4.3 Reichweite
 - 4.4 Rekuperation
 - 4.5 Stop & Go
- 5 Ergonomie
 - 5.1 Griff Hand Bremse
 - 5.2 Sitz
 - 5.3 Einstellung Höhe
- 6 Betriebsarten
 - 6.1 Generator im Stehen
 - 6.2 Generator beim Fahren
 - 6.3 Motor beim Fahren
 - 6.4 Muskelkraft beim Fahren
- 7 Szenarien
 - 7.1 Manager

- 7.1.1 Anzug
- 7.1.2 Wetter Schutz Haar
- 7.1.3 Wetter Schutz Körper
- 7.1.4 Erscheinung
- 7.2 Mutter
 - 7.2.1 Transport des Kindes
 - 7.2.2 Transport der Kinderutensilien
 - 7.2.3 Wetter Schutz Haar
 - 7.2.4 Wetter Schutz Körper
 - 7.2.5 Kinder Sitz
- 7.3 Grill Team
 - 7.3.1 Grill
 - 7.3.2 Kühlbox
- 8 Beleuchtung
 - 8.1 DIN 22958 Erfüllung
 - 8.1.1 Rücklicht
 - 8.1.1.1 Standlicht
 - 8.1.2 Frontbeleuchtung
 - 8.1.3 Reflektoren
 - 8.2 Dynamo
 - 8.3 Reflektoren Pedal
 - 8.4 Energie Versorgung
 - 8.5 Umfang des Beleuchtungssystems
- 9 Hupsystem
 - 9.1 DIN 33946 Erfüllung
 - 9.2 Lautsprecher System
- 10 Sicherheit
 - 10.1 Charakteristik Bremse
 - 10.2 Integrierte Kabel
 - 10.3 Federung
 - 10.4 Rückspiegel
 - 10.5 Reifen
 - 10.6 Anzahl Bremsen
 - 10.7 Qualität Bremsen
 - 10.8 Zuladung
- 11 Infotainment
 - 11.1 Fixierung Cockpit
 - 11.2 Fahr Modus
 - 11.3 Energie Parameter
 - 11.3.1 Batterie Status
 - 11.3.2 Ladung
 - 11.4 Kommunikation
- 12 Wartung
 - 12.1 Reguläre Wartung
 - 12.2 Ersatzteile
- 13 Kosten
 - 13.1 Kosten Effizient
- 14 Zeitplan
 - 14.1 Erstes Design Review
 - 14.2 Zweites Design Review
 - 14.3 Finales Design Review
 - 14.4 Finaler Report

FUNKTIONSHIERARCHIE DES PEDELECS (AKTUELLES PROJEKT)

- 1 Bewege Pedelec
 - 1.1 Anpassung der mechanischen Leistung
 - 1.2 Umwandlung Rotation in translatorische Bewegung
 - 1.3 Einkuppeln / Auskuppeln
- 2 Unterstützung Fahren
 - 2.1 Umwandlung mechanische Leistung
 - 2.2 Verwaltung elektrische Leistung
 - 2.3 Bereitstellung Energie für Licht System
- 3 Fahrer schützen
 - 3.1 Reduktion Schwingungen
 - 3.2 Beleuchtung
 - 3.3 Bremsen
- 4 Verwaltung
 - 4.1 System kontrollieren
 - 4.1.1 Geschwindigkeit kontrollieren
 - 4.1.2 Energie Speicher kontrollieren
 - 4.1.3 Fahrmodus kontrollieren
 - 4.2 Verwaltung Pedelec von Informationen
 - 4.3 System Input empfangen
 - 4.3.1 Pedal Signal empfangen
 - 4.3.2 Fahrmodus Anweisung empfangen
 - 4.3.3 Anweisung Bremse empfangen
 - 4.3.4 Anweisung Gangwechsel empfangen
 - 4.3.5 Anweisung Lenkung empfangen
 - 4.3.6 Anweisung Beleuchtung empfangen

PRODUKTSTRUKTUR DES PEDELECS (AKTUELLES PROJEKT)

- 1 Rad
 - 1.1 Hinterrad
 - 1.1.1 Schwalbe Reifen
 - 1.1.2 Schwalbe Schlauch
 - 1.1.3 Alu Hinterrad
 - 1.2 Vorderrad
 - 1.2.1 Schwalbe Schlauch
 - 1.2.2 Pegasus Reifen
 - 1.2.3 Alu Vorderrad
- 2 Rahmen
 - 2.1 Dach
 - 2.1.1 Plexiglas
 - 2.1.2 Vorderes Befestigungsschild
 - 2.2 Grundrahmen
 - 2.2.1 Alu Rohr
 - 2.3 Stauraum
 - 2.3.1 Alu Rohr
- 3 Sitz
 - 3.1 Fahrer Sitz
 - 3.2 Kinder Sitz
- 4 Lenkung
 - 4.1 Lenker
 - 4.1.1 Lenkerbügel
 - 4.2 Griff
 - 4.2.1 Moos-Gummi Griffbezüge

- 4.3 Lenkerbügel
- 4.3.1 Lenkerbügel Tiller
- 5 Bremse
- 5.1 Scheibenbremse
- 5.1.1 Shimano Scheibenbremse
- 6 E Motor
- 7 Batterie / Rekuperation
- 8 Schaltung
- 9 Beleuchtung
- 9.1 Rücklicht
- 9.2 Frontleuchte
- 10 Hupe
- 11 Infotainment
- 12 Steuerung

VERWENDETE NEGATIVLISTE

0	gegen	ein	darf	oder
1	haben	eine	das	sind
2	hat	einem	dass	so
3	im	einen	der	muss
4	in	einer	über	müssen
5	ist	und	die	auf
6	kann	von	dürfen	bei
7	können	werden	sollte	zu
8	mit	wird	sollte	mittels
9				

ERGEBNIS DER ANWENDUNG VON TRACELEGACY OHNE STEMMING

#	x	Aktuelles Pedelec-Projekt		Vergangenes Fahrrad-Projekt	
		von	zu	von	zu
1		Beleuchtung	Bereitstellung Energie für Licht System	Beleuchtung vorsehen	Licht reflektieren
2		Umfang des Beleuchtung ssystems	Bereitstellung Energie für Licht System	Beleuchtung vorsehen	Licht reflektieren
3		Umwandlung Rotation in translatorische Bewegung	Hinterrad	Rotationsmoment auf Straße übertragen	Hinterrad
4		Umwandlung Rotation in translatorische Bewegung	Alu Hinterrad	Rotationsmoment auf Straße übertragen	Hinterrad
5	x	Pedal Signal empfangen	Vorderrad	Signal in Richtungsänderung umsetzen	Vorderrad
6	x	Pedal Signal empfangen	Alu Vorderrad	Signal in Richtungsänderung umsetzen	Vorderrad

#	x	Aktuelles Pedelec-Projekt		Vergangenes Fahrrad-Projekt	
		von	zu	von	zu
7		Griff Hand Bremse	Bremse	Bremsen zuverlässiges Brem- sen bei Nässe muss mit Handkraft betrieben werden können	Bremseinheit Bremseinheit Bremseinheit
8		Beleuchtung	Beleuchtung	Beleuchtung vorse- hen	Beleuchtung
9		Umfang des Be- leuchtung ssystems	Beleuchtung	Beleuchtung vorse- hen	Beleuchtung
10		Charakteristik Brem- se	Bremse	Bremsen zuverlässiges Brem- sen bei Nässe	Bremseinheit Bremseinheit
11		Anzahl Bremsen	Bremse	Bremsen zuverlässiges Brem- sen bei Nässe	Bremseinheit Bremseinheit
12		Qualität Bremsen	Bremse	Bremsen zuverlässiges Brem- sen bei Nässe	Bremseinheit Bremseinheit
13		Griff Hand Bremse	Scheibenbremse	Bremsen zuverlässiges Brem- sen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
14		Griff Hand Bremse	Shimano Scheiben- bremse	Bremsen zuverlässiges Brem- sen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
15		Charakteristik Brem- se	Scheibenbremse	Bremsen zuverlässiges Brem- sen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
16		Charakteristik Brem- se	Shimano Scheiben- bremse	Bremsen zuverlässiges Brem- sen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
17		Anzahl Bremsen	Scheibenbremse	Bremsen zuverlässiges Brem- sen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
18		Anzahl Bremsen	Shimano Scheiben- bremse	Bremsen zuverlässiges Brem- sen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
19		Qualität Bremsen	Scheibenbremse	Bremsen zuverlässiges Brem- sen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
20		Qualität Bremsen	Shimano Scheiben- bremse	Bremsen zuverlässiges Brem- sen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen

ERGEBNIS DER ANWENDUNG VON TRACELEGACY MIT STEMMING

#	x	Aktuelles Projekt		Altes Projekt	
		von	zu	von	zu
1		Griff Hand Bremse	Bremsen	Bremsen Bremsen zuverlässiges Brem- sen bei Nässe muss mit Handkraft betrieben werden können	Bremskraft aufneh- men Bremskraft auf Stra- ße übertragen Bremskraft auf Stra- ße übertragen Bremskraft aufneh- men
2		Beleuchtung	Beleuchtung	Beleuchtung vorse- hen	unmittelbare Um- gebung beleuchten
3		Beleuchtung	Anweisung Be- leuchtung empfan- gen	Beleuchtung vorse- hen	unmittelbare Um- gebung beleuchten
4		Front beleuchtung	Beleuchtung	Beleuchtung vorse- hen	unmittelbare Um- gebung beleuchten
5		Front beleuchtung	Anweisung Be- leuchtung empfan- gen	Beleuchtung vorse- hen	unmittelbare Um- gebung beleuchten
6		Umfang des Be- leuchtungssystem s	Beleuchtung	Beleuchtung vorse- hen	unmittelbare Um- gebung beleuchten
7		Charakteristik Brem- se	Bremsen	Bremsen Bremsen zuverlässiges Brem- sen bei Nässe	Bremskraft aufneh- men Bremskraft auf Stra- ße übertragen Bremskraft auf Stra- ße übertragen
8		Anzahl Bremsen	Bremsen	Bremsen Bremsen zuverlässiges Brem- sen bei Nässe	Bremskraft aufneh- men Bremskraft auf Stra- ße übertragen Bremskraft auf Stra- ße übertragen
9		Qualität Bremsen	Bremsen	Bremsen Bremsen zuverlässiges Brem- sen bei Nässe	Bremskraft aufneh- men Bremskraft auf Stra- ße übertragen Bremskraft auf Stra- ße übertragen
10		Beleuchtung	Beleuchtung	unmittelbare Um- gebung beleuchten	Beleuchtung
11		Bremsen	Bremse	Bremskraft aufneh- men Bremskraft aufneh- men Bremskraft auf Stra- ße übertragen Bremskraft auf Stra- ße übertragen	Bremseinheit 2 Bremshebel Bremseinheit 2 Bremsscheiben

#	x	Aktuelles Projekt		Altes Projekt	
		von	zu	von	zu
12		Anweisung Bremse empfangen	Bremse	Bremskraft aufnehmen Bremskraft aufnehmen Bremskraft auf Straße übertragen Bremskraft auf Straße übertragen	Bremseinheit 2 Bremshebel Bremseinheit 2 Bremsscheiben
13		Anweisung Beleuchtung empfangen	Beleuchtung	unmittelbare Umgebung beleuchten	Beleuchtung
14		Charakteristik Bremse	Anweisung Bremse empfangen	Bremsen Bremsen zuverlässiges Bremsen bei Nässe	Bremskraft aufnehmen Bremskraft auf Straße übertragen Bremskraft auf Straße übertragen
15		Anzahl Bremsen	Anweisung Bremse empfangen	Bremsen Bremsen zuverlässiges Bremsen bei Nässe	Bremskraft aufnehmen Bremskraft auf Straße übertragen Bremskraft auf Straße übertragen
16	x	Qualität Bremsen	Anweisung Bremse empfangen	Bremsen Bremsen zuverlässiges Bremsen bei Nässe	Bremskraft aufnehmen Bremskraft auf Straße übertragen Bremskraft auf Straße übertragen
17		Griff Hand Bremse	Anweisung Bremse empfangen	Bremsen Bremsen zuverlässiges Bremsen bei Nässe muss mit Handkraft betrieben werden können	Bremskraft aufnehmen Bremskraft auf Straße übertragen Bremskraft auf Straße übertragen Bremskraft aufnehmen
18		Umfang des Beleuchtungssystem s	Bereitstellung Energie für Licht System	Beleuchtung vorsehen	Licht reflektieren
19		Umfang des Beleuchtungssystem s	Anweisung Beleuchtung empfangen	Beleuchtung vorsehen	unmittelbare Umgebung beleuchten
20		Antrieb	Umwandlung Rotation in translatorische Bewegung	Hinter antrieb mittels Fußkurbel und Kette Hinter antrieb mittels Fußkurbel und Kette Hinter antrieb mittels Fußkurbel und Kette Hinter antrieb mittels Fußkurbel und Kette	Rotationsmoment aufnehmen Rotationsmoment leiten Rotationsmoment wandeln Rotationsmoment auf die Straße übertragen

#	x	Aktuelles Projekt		Altes Projekt	
		von	zu	von	zu
21		Schaltung	Anweisung Gang- wechsel empfangen	Übersetzung mittels Kettenschaltung	Übersetzung wechseln
22		Betriebsarten	Bremsen	muss mit Handkraft betrieben werden können	Bremskraft aufnehmen
23		Betriebsarten	Anweisung Bremse empfangen	muss mit Handkraft betrieben werden können	Bremskraft aufnehmen
24		Beleuchtung	Bereitstellung Energie für Licht System	Beleuchtung vorsehen	Licht reflektieren
25		Front beleuchtung	Bereitstellung Energie für Licht System	Beleuchtung vorsehen	Licht reflektieren
26	x	Sicherheit	Umwandlung Rotation in translatorische Bewegung	sicheres Lenken / Manövrieren	Fahr richtung ändern
27		Sicherheit	Unterstützung Fahren	sicheres Lenken / Manövrieren	Fahr richtung ändern
28	x	Sicherheit	Umwandlung mechanische Leistung	sicheres Lenken / Manövrieren	Fahr richtung ändern
29		Sicherheit	Fahrer Schützen	sicheres Lenken / Manövrieren	Fahr richtung ändern
30	x	Sicherheit	Pedal Signal empfangen	sicheres Lenken / Manövrieren sicheres Lenken / Manövrieren	Steuers ignal aufnehmen Signal in Richtungsänderung umsetzen
31		Umwandlung Rotation in translatorische Bewegung	Rad	Rotationsmoment auf Straße übertragen Fahr richtung ändern	Hinterr ad Vorderr ad
32		Umwandlung Rotation in translatorische Bewegung	Hinterrad	Rotationsmoment auf Straße übertragen	Hinterrad
33		Umwandlung Rotation in translatorische Bewegung	Alu Hinterrad	Rotationsmoment auf Straße übertragen	Hinterrad
34	x	Umwandlung Rotation in translatorische Bewegung	Vorderrad	Fahr richtung ändern	Vorderrad
35	x	Umwandlung Rotation in translatorische Bewegung	Alu Vorderrad	Fahr richtung ändern	Vorderrad
36	x	Umwandlung Rotation in translatorische Bewegung	Vorderes Befestigungsschild	Fahr richtung ändern	Vorderrad
37	x	Umwandlung Rotation in translatorische Bewegung	Lenker	Fahr richtung ändern	Lenk stange
38	x	Unterstützung Fahren	Rad	Fahr richtung ändern	Vorderr ad
39	x	Unterstützung Fahren	Vorderrad	Fahr richtung ändern	Vorderrad

#	x	Aktuelles Projekt		Altes Projekt	
		von	zu	von	zu
40	x	Unterstützung Fahren	Alu Vorderrad	Fahr richtung ändern	Vorderrad
41	x	Unterstützung Fahren	Vorderes Befestigungsschild	Fahr richtung ändern	Vorderrad
42	x	Unterstützung Fahren	Lenker	Fahr richtung ändern	Lenkstange
43	x	Umwandlung mechanische Leistung	Rad	Fahr richtung ändern	Vorderrad
44	x	Umwandlung mechanische Leistung	Alu Rad	Fahr richtung ändern	Vorderrad
45	x	Umwandlung mechanische Leistung	Vorderes Befestigungsschild	Fahr richtung ändern	Vorderrad
46	x	Umwandlung mechanische Leistung	Lenker	Fahr richtung ändern	Lenkstange
47	x	Fahrer schützen	Rad	Fahr richtung ändern	Vorderrad
48	x	Fahrer schützen	Vorderrad	Fahr richtung ändern	Vorderrad
49	x	Fahrer schützen	Alu Vorderrad	Fahr richtung ändern	Vorderrad
50	x	Fahrer schützen	Vorderes Befestigungsschild	Fahr richtung ändern	Vorderrad
51	x	Fahrer schützen	Lenker	Fahr richtung ändern	Lenkstange
52		Bremsen	Scheibenbremse	Bremskraft auf Straße übertragen	2 hydraulische Scheibenbremsen
53		Bremsen	Shimano Scheibenbremse	Bremskraft auf Straße übertragen	2 hydraulische Scheibenbremsen
54	x	Pedal Signal empfangen	Rad	Signal in Richtungsänderung umsetzen	Vorderrad
55	x	Pedal Signal empfangen	Vorderrad	Signal in Richtungsänderung umsetzen	Vorderrad
56	x	Pedal Signal empfangen	Alu Vorderrad	Signal in Richtungsänderung umsetzen	Vorderrad
57	x	Pedal Signal empfangen	Vorderes Befestigungsschild	Signal in Richtungsänderung umsetzen	Vorderrad
58	x	Pedal Signal empfangen	Lenker	Signal in Richtungsänderung umsetzen Steuer signal aufnehmen	Lenkstange Lenkstange
59		Anweisung Bremsen empfangen	Scheibenbremse	Bremskraft auf Straße übertragen	2 hydraulische Scheibenbremsen
60		Anweisung Bremsen empfangen	Shimano Scheibenbremse	Bremskraft auf Straße übertragen	2 hydraulische Scheibenbremsen
61	x	Anweisung Gang wechsel empfangen	Hinterrad	Übersetzung wechseln	Schaltwerk (Übersetzung hinten)
62	x	Anweisung Gang wechsel empfangen	Alu Hinterrad	Übersetzung wechseln	Schaltwerk (Übersetzung hinten)
63	x	Anweisung Gang wechsel empfangen	Vorderrad	Übersetzung wechseln	Umwerfer (Übersetzung vorne)

#	x	Aktuelles Projekt		Altes Projekt	
		von	zu	von	zu
64	x	Anweisung Gang- wechsel empfangen	Alu Vorderrad	Übersetzung wech-seln	Umwerfer (Übersetzung vorne)
65	x	Anweisung Gang- wechsel empfangen	Vorderes Befestigungsschild	Übersetzung wech-seln	Umwerfer (Übersetzung vorne)
66		Griff Hand Bremse	Bremse	Bremsen Bremsen Bremsen zuverlässiges Brem-sen bei Nässe zuverlässiges Brem-sen bei Nässe muss mit Handkraft betrieben werden können muss mit Handkraft betrieben werden können	Bremseinheit 2 Bremshebel 2 Bremsscheiben Bremseinheit 2 Bremsscheiben Bremseinheit 2 Bremshebel
67		Beleuchtung	Beleuchtung	Beleuchtung vorsehen	Beleuchtung
68		Umfang des Beleuchtungssystems	Beleuchtung	Beleuchtung vorsehen	Beleuchtung
69		Charakteristik Brem-se	Bremse	Bremsen Bremsen Bremsen zuverlässiges Brem-sen bei Nässe zuverlässiges Brem-sen bei Nässe	Bremseinheit 2 Bremshebel 2 Bremsscheiben Bremseinheit 2 Bremsscheiben
70		Anzahl Bremsen	Bremse	Bremsen Bremsen Bremsen zuverlässiges Brem-sen bei Nässe zuverlässiges Brem-sen bei Nässe	Bremseinheit 2 Bremshebel 2 Bremsscheiben Bremseinheit 2 Bremsscheiben
71		Qualität Bremsen	Bremse	Bremsen Bremsen Bremsen zuverlässiges Brem-sen bei Nässe zuverlässiges Brem-sen bei Nässe	Bremseinheit 2 Bremshebel 2 Bremsscheiben Bremseinheit 2 Bremsscheiben
72		Front beleuchtung	Beleuchtung	Beleuchtung vorsehen	Beleuchtung
73		Griff Hand Bremse	Scheibenbremse	Bremsen zuverlässiges Brem-sen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen

#	x	Aktuelles Projekt		Altes Projekt	
		von	zu	von	zu
74		Griff Hand Bremse	Shimano Scheibenbremse	Bremsen zuverlässiges Bremsen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
75		Charakteristik Bremse	Scheibenbremse	Bremsen zuverlässiges Bremsen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
76		Charakteristik Bremse	Shimano Scheibenbremse	Bremsen zuverlässiges Bremsen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
77		Anzahl Bremsen	Scheibenbremse	Bremsen zuverlässiges Bremsen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
78		Anzahl Bremsen	Shimano Scheibenbremse	Bremsen zuverlässiges Bremsen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
79		Qualität Bremsen	Scheibenbremse	Bremsen zuverlässiges Bremsen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
80		Qualität Bremsen	Shimano Scheibenbremse	Bremsen zuverlässiges Bremsen bei Nässe	2 hydraulische Scheibenbremsen 2 hydraulische Scheibenbremsen
81		Betriebsarten	Bremse	muss mit Handkraft betrieben werden können muss mit Handkraft betrieben werden können	2 Bremshebel Bremseinheit
82		Sicherheit	Lenker	sicheres Lenken / Manövrieren	Lenkstange