Universität der Bundeswehr München
Institut für Technische Produktentwicklung

Optimierung von komplexen Produktentwicklungsprozessen durch Identifikation von Daten- und Informationsflüssen

– Handlungsrichtlinien zur Verwendung von Graphen- und Netzwerktheorie im Systems und Requirements Engineering

M.Sc. Abdo Chahin

Vollständiger Abdruck der von der Fakultät für Luft- und Raumfahrttechnik der Universität der Bundeswehr München genehmigten Dissertation zur Erlangung des akademischen Grades eines

Doktor-Ingenieur (Dr.-Ing.)

Vorsitzender:	Univ.-Prof. Dr. Mattias Gerdts
Gutachter der Dissertation:	1. Univ.-Prof. Dr.-Ing. Kristin Paetzold
	2. Univ.-Prof. Dr.-Ing. Roger Förstner

Die Dissertation wurde am 10.06.2020 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Luft- und Raumfahrttechnik am 18.12.2020 angenommen.
Die mündliche Prüfung fand am 18.12.2020 statt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AM	Anforderungsmanagement
ASIL	Automotive Safety Integrity Level
ASPICE	Automotive Software Process Improvement and Capability Determination
C	Characterstics (Merkmale)
CAD	Computer-aided design
CPM	Characteristics-Properties Modelling
D&I	Daten und Informationen
DIF	Daten- und Informations-Flüsse
DNG	Doors Next Generation
DSM	Design Structure Matrizen
G	Graph
HiL	Hardware in the Loop
ID	Identifikation
iPeM	Integriertes Produktentstehungsmodell
KM	Konfigurationsmanagement
KZ	Kennzahl
LiDAR	laser detection and ranging
LoC	Lines of Code
MA	Mitarbeiter
MBSE	Model-based Systems Engineering
MDM	Multiple Domain Matrizen
MiL	Model in the Loop
NFA	Nicht-Funktionale Anforderungen
NW	Netzwerk
NWT	Netzwerktheorie
OEM	Originalausrüstungshersteller
P	Properties (Eigenschaften)
PDD	Property-Driven Development/Design
PDM	Produktdatenmanagement

PE	Produktentwicklung
PEP	Produktentwicklungsprozess
PLZ	Produktlebenszyklus
PR	Required Properties (erforderliche Eigenschaft)
QM	Qualitätsmanagement
R	Relation
RE	Requirements Engineering
RM	Requirements Management
RiM	Risiko-Management
SE	Systems Engineering
SiL	Software in the Loop
SOP	Start of Production
SysML	Systems Modeling Language
UML	Unified Modeling Language

Vorwort

Diese Arbeit entstand nach meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Technische Produktentwicklung der Universität der Bundeswehr München.

Frau Prof. Dr.-Ing. Kristin Paetzold hat in ihrer Zeit als Institutsleiterin die Arbeitsumgebung geschaffen, die es mir ermöglichte eigenverantwortlich ein Promotionsthema zu finden und neue Ideen umzusetzen. Für ihren Rat und für die Betreuung meiner Dissertation danke ich ihr sehr. Prof. Dr.-Ing. Roger Förstner danke ich für die Übernahme des Zweitgutachtens und seinem Interesse an meiner Arbeit. Prof. Dr. Matthias Gerdts danke ich für die Übernahme des Prüfungsvorsitzes.

Außerdem danke ich allen Mitarbeitern des Instituts für die gute und kollegiale Zusammenarbeit. Ein Dank auch an meine jetzigen und ehemaligen Kollegen – insbesondere Imad Sanduka - für die sehr guten Impulse, Diskussionen und Ideen, die mir halfen, meine Arbeit kritisch zu betrachten und weiterzuentwickeln. Vielen Dank auch an die Kollegen aus dem Gebiet der Produktenwicklung für ihre Bereitschaft und den Einsatz gemeinsam zu Forschen und für die gemeinsamen Publikationen.

Ein besonderer Dank gilt meinen Eltern, für ihren ständigen Zuspruch, Antrieb und ihre Aufopferung. Ein besonderer Dank gilt Julia Fink für Ihre Bereitschaft meine Arbeit sprachlich und kritisch zu beäugen. Zum Schluss möchte ich allen Freunden und meinen Geschwistern danken und dafür, dass sie mich immer mal wieder daran erinnerten, dass es auch Zeiten im Leben ohne Arbeit geben muss.

München, den 10.06.2020 Abdo Chahin

Abstract in English

More and more products are experiencing recall campaigns, are being delivered late (Berlin airport BER) or are being delivered faulty – as the latest diesel scandal of the German automobile manufacturer's shows. At the same time, customer requirements continue to rise, forcing manufacturers to integrate more and more functionalities into one product. The result is a system of elements which depend on each other. The development of such systems is becoming increasingly complicated and the associated data and information flows (DIF) of the product development processes (PDP) has become too complex - for developers and companies. However, since the currently available instruments are not sufficient to cope with the complexity, new or additional approaches are required. In order to change this situation, existing methods must be adapted, combined and expanded to make the DIF visible, thus usable. The goal is to reduce the complexity to a manageable level and to exploit the Data and Information (D&I). This enables not only the understanding of the DIF holistically, but also allows us to control the PDP in a targeted manner and to optimize with regard to desired factors such as development time.

For this purpose, this thesis applies the systems engineering approach, which provides a good basis for the technical-physical consideration and the lighting of the management processes. That allows the identification and tracking of the DIF along with the life cycle during the development. Thus helps to understand how the DIF are created, developed and which decisions are made on the basis thereof. It is important to keep track of how changes of the DIF affect decisions (design, management, etc.). In order to finally identify and track the DIF, methods from requirements engineering - more precisely traceability - are used to formalize the identification and analysis of D&I. Since the DIF are network-dependent and the analysis of the flows is necessary, the network and graph theory is used to translate the D&I into a neutral language. That is how DIF becomes analysable and useable for process optimization. The mathematical analyses also increase the chances of measuring and evaluating the dependencies, making DIF visible and understandable for the individual developer as well as the company. The complexity can thus be reduced to a manageable level.

This thesis shows the necessary steps to be able to use the potential of graph and network theory. For this purpose, each mathematical element is introduced individually and placed in the context. The following work acts as a guideline to understand what the respective element can be used for, in order to apply it to a company's own conditions. Using the example of autonomous driving, a plausibility check is made to show which mathematical analysis are possible and how these can be interpreted in order to manage the increasing complexity. Suggestions are given as to how the mathematical key factors help the developer to derive or justify actions using the example of property validation.

Abstrakt auf Deutsch

Immer mehr Produkte werden zurückgerufen, verspätet (Berliner Flughafen BER) oder fehlerhaft ausgeliefert - wie der jüngste Dieselskandal der deutschen Automobilhersteller zeigt. Gleichzeitig steigen die Kundenanforderungen stetig an und zwingen die Hersteller immer mehr Funktionen in ein Produkt zu integrieren. Die Entwicklung solcher Systeme wird immer komplizierter und die damit verbundenen Daten- und Informationsflüsse (DIF) der Produktentwicklungs-prozesse (PEP) werden zu komplex - für Entwickler und Unternehmen. Da die derzeit verfügbaren Mittel jedoch nicht ausreichen, um die Komplexität zu bewältigen, sind neue oder zusätzliche Ansätze erforderlich. Um diese Situation zu ändern, müssen vorhandene Methoden angepasst, kombiniert und erweitert werden, wodurch die DIF sichtbar und damit nutzbar werden. Ziel ist es, die Komplexität auf ein überschaubares Maß zu reduzieren und die Daten und Informationen (D & I) zu nutzen. Dies ermöglicht nicht nur das ganzheitliche Verständnis der DIF, sondern ermöglicht es uns auch, den PEP gezielt zu steuern und hinsichtlich gewünschter Faktoren wie der Entwicklungszeit zu optimieren.

Aus diesem Grund verwendet diese Arbeit den systemtechnischen Ansatz, der eine gute Grundlage für die technisch-physikalische Betrachtung und die Beleuchtung der Managementprozesse bietet. Somit wird es möglich die DIF entlang des Lebenszyklus der Entwicklung zu identifizieren und analysieren. Dies hilft zu ermitteln, wie die DIF erstellt, entwickelt und welche Entscheidungen auf deren Grundlage getroffen werden. Um die DIF endgültig zu differenzieren und zu verfolgen, werden Methoden aus dem Requirements Engineering - genauer gesagt der Traceability – verwendet. Da die DIF netzwerkartig in Abhängigkeit stehen und die Analyse der Flüsse erforderlich ist, wird die Netzwerk- und Graphentheorie verwendet, um die D & I in eine neutrale Sprache zu übersetzen. Auf diese Weise werden DIF aufgedeckt und können für die Prozessoptimierung verwendet werden. Die mathematischen Auswertungen erhöhen die Chancen, die Abhängigkeiten zu messen, zu bewerten und machen DIF sowohl für den einzelnen Entwickler als auch für das Unternehmen sichtbar und verständlich. Die Komplexität kann somit auf ein überschaubares Maß reduziert werden.

Diese Arbeit zeigt die notwendigen Schritte auf, um das Potenzial der Graph- und Netzwerktheorie nutzen zu können. Zu diesem Zweck wird jedes mathematische Element einzeln eingeführt und in den Kontext gestellt. Die folgende Arbeit dient als Richtlinie, wodurch verständlich wird, wofür das jeweilige Element verwendet werden kann. Am Beispiel des autonomen Fahrens wird eine Plausibilitätsprüfung durchgeführt, wodurch gezeigt wird, welche mathematischen Analysen möglich sind und wie diese interpretiert werden können, um die zunehmende Komplexität zu bewältigen. Es werden Vorschläge gemacht, wie die mathematischen Schlüsselfaktoren dem Entwickler helfen, Aktionen am Beispiel der Eigenschaftsvalidierung abzuleiten oder zu rechtfertigen.

Die Algorithmen werden nicht aufbegehren und uns versklaven. Vielmehr werden sie Entscheidungen für uns so gut treffen, dass wir verrückt wären, ihrem Rat nicht zu folgen.

Yuval Noah Harari

Einführung in die Thematik

1.1 Gegenstand der Untersuchung: Unterscheidung von kompliziert und komplex

Der Anspruch, dass ein und dasselbe Produkt immer mehr Funktionen ermöglicht, stellt die Produktentwicklung vor neue Herausforderungen. Ein Beispiel hierfür ist das Mobiltelefon, das zu Beginn dem Nutzer lediglich mobiles Telefonieren ermöglichte und heutzutage weit mehr Funktionen aus singulären Produkten wie dem Wecker, dem Navigationsgerät oder dem Fotoapparat integriert. Da die Produkte die gewünschten Funktionen nicht mehr rein mechanisch ermöglichen, sondern aus einer Summe von physischen Bauteilen, mechatronischen Systemen und Software bestehen, kann man anhand der Anzahl der Elemente verfolgen, wie komplex ein Produkt geworden ist. Exemplarisch kann die Anzahl an „Lines of Code“ (LoC) – also die Anzahl an Zeilen der verbauten Software – eine Messgröße darstellen. Beispielsweise weist das aktuelle Smartphone-Betriebssystem Android ca. 1,2 Mio. LoC auf.

Ein weiteres Beispiel – hier aus der der Luftfahrtentwicklung - zeigt, wie aus „einfachen“ Fluggeräten autonome Drohnen entstehen konnten und mit welcher Komplexität die Entwicklung solcher Flugzeuge einhergeht. Die Funktion der ersten Fluggeräte beschränkte sich auf die Fähigkeit, für eine kurze Zeit in der Luft zu verweilen. Diese Erfindung wurde weiterentwickelt und reicht heute von kommerziellen Flugzeugen, über Hochleistungs-Kampfjets, bis hin zu unbemannten Drohenden aller Art. Zieht man auch hier die LoC zum Vergleich heran, weißt ein F-35 Kampfjet aus dem Jahr 2013 bereits 24 Mio. LoC auf (David McCandless, 2020).

Ähnlich verhält es sich zum Beispiel bei einem A-380, der aus ca. 4 Mio. Einzelteilen besteht, die global gefertigt werden. Dies führt zu einer hohen Komplexität in der Entwicklung, die es zu bewältigen gilt und die die Entwickler somit vor eine noch nie dagewesene Herausforderung stellt. Natürlich ist es nicht bei der bemannten Flugzeugentwicklung geblieben. Die Erfahrungen aus der Luftfahrtentwicklung wurden genutzt und erweitert und ermöglichten die Fahrt ins All. Die Rahmenbedingungen sowie die physikalischen Gegebenheiten führten erneut zu einer Steigerung der Komplexität, die ebenso nur durch neue Methoden und Konzepte für die Zusammenarbeit, Entwicklung, Fertigung, usw. bewältigt werden konnten. Schließlich wurde es möglich, Funktionen zu automatisieren und Fluggeräte völlig autonom zu betreiben.

Vergleicht man nun die Entwicklung der Luftfahrt mit der des Automobils, lassen sich Parallelen erkennen. Begonnen hat die Entwicklung einer motorisierten „Kutsche“ Ende des 18. Jahrhunderts. Die Fülle an Funktionen, die ein Auto heutzutage erfüllt, steigt unaufhörlich weiter. Ähnlich wie im Falle des Flugzeuges wurde aus dem Auto mehr als

nur ein Fortbewegungsmittel. Abgesehen von „Standartfunktionen“, wie dem Navigationsgerät, dem Radio, der Klimaanlage, übernimmt heute das Auto Funktionen, wie autonomes Bremsen, Spurhalten, Müdigkeitserkennung usw. Doch die aktuelle Verfügbarkeit von IOT, Big-Data, Block-Chain, etc. verkürzen im direkten Vergleich zur früheren Entwicklung der Luftfahrtbranche die zeitlichen Intervalle für Innovationen und den Produktlebenszyklus drastisch. Es ist zu erwarten, dass auch Autos autonom fahren werden, ohne dabei Abstriche an der Anzahl und Qualität der derzeit möglichen Funktionen in Kauf nehmen zu müssen. Das Auto wird nicht mehr als ein „abgeschlossenes“ Produkt vom Anwender innerhalb einer bekannten Umgebung gesteuert, sondern steht in ständiger Kommunikation und Interaktion mit seiner Umwelt. Die Höhe des LoC im Fahrzeug von über 100 Mio. bestätigt den Vergleich zum Flugzeug und deutet auf eine Steigerung der Komplexität hin.

Die Realität zeigt jedoch, dass die Zahl an Rückrufaktionen bei den Automobilherstellern steigt. Auch das aktuelle Problem der deutschen Autoindustrie im Hinblick auf falsche Abgaswerte – Stichwort Dieselskandal – zeigt, dass die Entwicklung nicht vollständig erfasst ist.

Verzögerungen erfolgten jedoch auch bei bekannten Großprojekten wie dem A400M, dem Berliner Flughafen BER, der Einführung von Toll Collect etc. Der Grund hierfür ist die Abhängigkeit der Elemente bzw. Systeme voneinander, welche durch die hohe Anzahl an Funktionen verursacht wird, was ebenfalls den Entwicklungsprozess erschwert. Dennoch gilt es, die gesteckten Ziele – Zeit, Kosten Qualität - zu erreichen.

Eine Aussage von Ulrich gibt bereits einen ersten Hinweis darauf, was unter Komplexität zu verstehen ist: „Unter Komplexität wird im Allgemeinen die Möglichkeit der geistigen Erfassung und Beherrschung eines Systems verstanden. Sie beruht auf dem Reichtum der Beziehungen zwischen den Elementen und seiner Umwelt und äußert sich bei dynamischen Systemen in einer sehr hohen Anzahl möglicher Zustände, die das System annehmen kann.“ (S.31 (Ulrich, 1970)).

Generell sind jedoch kompliziert und komplex keine Synonyme. Ein komplizierter Sachverhalt kann mit dem geeigneten Aufwand verständlich sein. Hingegen ist ein komplexer nicht ohne weitere (noch unbekannte) Hilfsmittel zu begreifen.

Mit den beiden Variablen – der Zeit und der Anzahl der Elemente – lässt sich der Unterschied zwischen einem komplizierten und einem komplexen Sachverhalt nachvollziehen und erkennen, ab wann eine Situation nicht mehr zu bewältigen – zu komplex – ist.

Geht man von einem Element aus, so kann man entweder die Anzahl steigern oder sich das Element in der Zeit verändern – vergleiche Abbildung 1 links unten im Quadranten III. Folgt man nun der horizontalen Entwicklung in den Quadranten IV, steigt die Anzahl der Elemente. Dies ist vergleichbar mit Varianten eines bestimmten Produktes zum Beispiel einem Navigationsgerät für PKW, LKW, Motorrad, etc. die jedoch im Grunde „nur“

für die jeweilige Anwendung angepasst werden. Somit stellt die gestiegene Anzahl der Elemente lediglich eine komplizierte Situation dar.

Im Gegensatz dazu kann ein Produkt durch Innovationen mit der Zeit verändert werden. Ein sehr präsentes Beispiel ist wieder das Smartphone, das aus einem „einfachen“ Mobilfunkgerät entstanden ist. Diese dynamische Kontinuität in Verbindung mit einer hohen Anzahl an Funktionen führt schließlich zur Komplexität. Folglich wird die Situation ohne weitere Hilfsmittel nicht zu bewältigen sein, da die Anzahl der Elemente zu hoch ist und sich gleichzeitig die Sachlage dynamisch verändert.

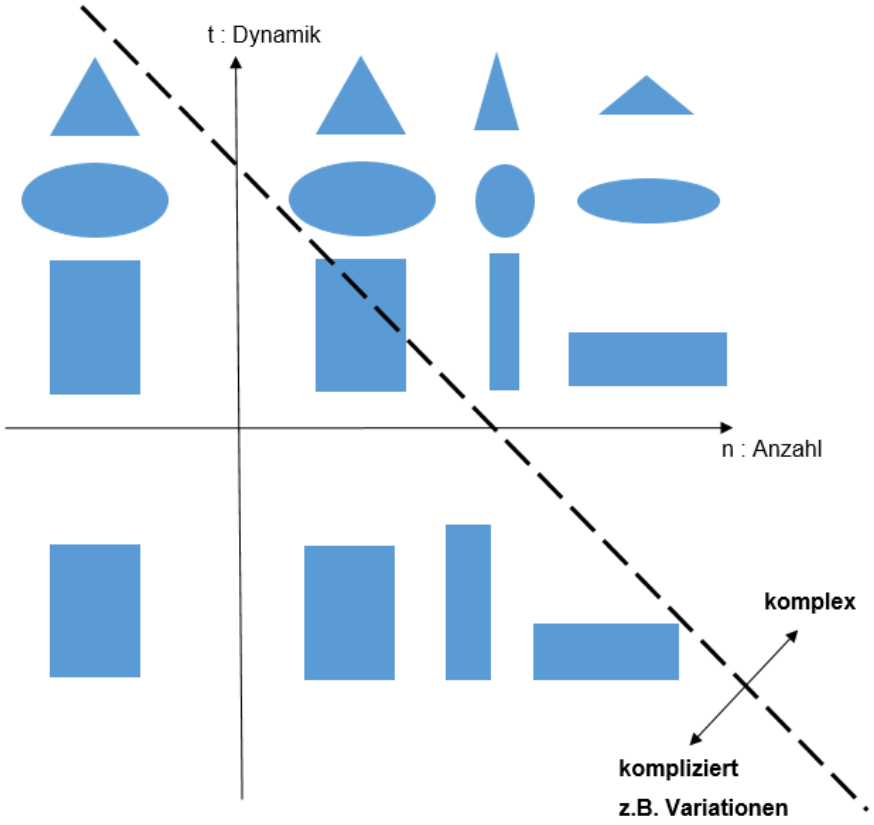


Abbildung 1: Von kompliziert bis komplex

Analog verhält es sich mit Informationen und Abhängigkeiten, da Funktionen durch die Verarbeitung von Daten und Informationen und entsprechender Reaktion ermöglicht werden. Tauscht man nun die Anzahl der Elemente durch Informationen aus, folgt die gleiche Erkenntnis. Ist der Informationsgehalt zeitlich variabel – also nicht stabil – spricht man dann von einem komplexen Sachverhalt. Ergo herrschen in einem komplexen System Unbekannte, die nicht erfasst bzw. verstanden werden. Diese können die Elemente oder ihre Verknüpfung sein. Das heißt das Verhalten eines komplexen Systems ist nicht bekannt, da die Abhängigkeiten unter seinen Elementen noch nicht erfasst sind. Ein Beispiel können die Bauteile eines modernen Fahrzeugs sein. In einer endlichen Zeit kann ein

Ingenieur nachvollziehen, welche Aufgaben die einzelnen Teile haben und wie ihr Verhalten ist. Tauscht man jedoch Elemente aus während der Ingenieur in der Informationsverarbeitung steckt und fügt neue Bauteile ein, die die Funktion verändern, kann er nicht ohne weitere Hilfsmittel das Gesamtverhalten des Systems erkennen. Im letzteren Fall spricht man nicht mehr von einem komplizierten, sondern von einem komplexen System.

Aufgrund der heutigen Kundenanforderungen, ist man gezwungen viele Funktionen durch ein System abzudecken. Das Ergebnis ist meist ein vielschichtiges Produkt, das eine hohe Anzahl an Aufgaben erfüllen muss. Dies hat zur Folge, dass sich die einzelnen Systeme untereinander beeinflussen und in Abhängigkeit zueinander stehen. Somit besteht letztendlich das Erzeugnis aus Elementen, dessen Abhängigkeiten noch nicht alle erfasst sind.

Diese Abhängigkeit zeigt sich wiederum in den Informations- bzw. Datenflüssen. Dabei kann es sich sowohl um Produktdaten (zu Funktion, Struktur oder Verhalten des Produktes), als auch um Informationen handeln, welche in einem bestimmten Entwicklungsstand vorhanden sind oder auch daraus entstehen (Daten zu Rollen, Verantwortung der Mitarbeiter, Workflows, Tasks, etc.).

Die Folgen dieser Entwicklung sind unter anderem, dass die Daten und Informationsmengen unüberschaubar hoch sind. Somit werden vorhandene Daten und Informationen (D&I) nicht vollständig während der Produktentwicklung (PE) ausgeschöpft. Die Nachvollziehbarkeit der Daten und Informations-Flüsse (DIF) ist mit den aktuell zu Verfügung stehenden Mitteln nicht gegeben und folglich die Komplexität nicht handhabbar – weder für den einzelnen Entwickler, noch für die beteiligten Unternehmen.

In Summe kann der PEP in der aktuellen Lage nicht ganzheitlich überblickt und folglich auch nicht im Hinblick gewünschter Faktoren, wie der Entwicklungszeit optimiert werden.

Um diese Situation zu ändern, müssen vorhanden Methoden abgestimmt, kombiniert und erweitert werden, um die DIF für den einzelnen Entwickler und Unternehmen sichtbar und verständlich und damit nutzbar zu machen. Nur dadurch kann die Komplexität auf ein bewältigbares Niveau reduziert werden.

1.2 Aufbau der Arbeit

Die Vorgehensweise für die Erstellung dieser Arbeit richtet sich nach der Design Research Methodology nach Chakrabarti und Blessing, welche methodische Vorschläge erteilt, wie eine wissenschaftliche Arbeit erarbeitet und ausgestellt werden soll (Blessing und Chakrabarti, 2009).

Für das Vorgehen einer wissenschaftlichen Arbeit werden vier Stufen vorgeschlagen, welche hier auch Verwendung finden, vergleiche Abbildung 2 (Blessing und Chakrabarti, 2009):

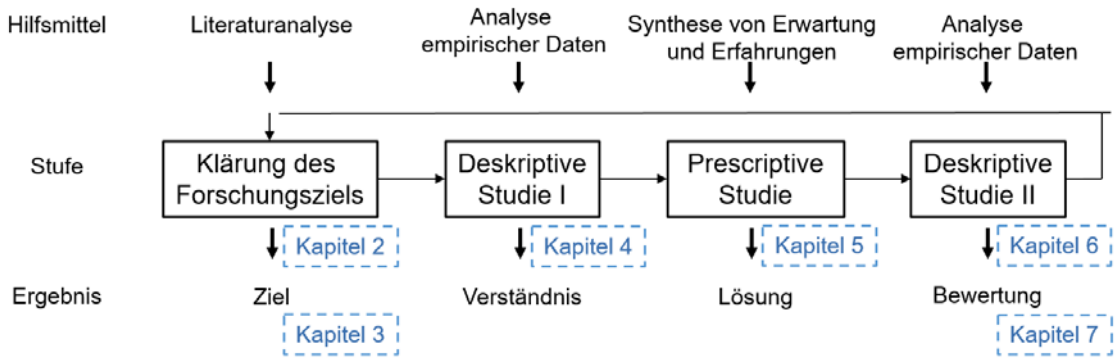


Abbildung 2: Phasen der Arbeit angelehnt an Chakrabarti und Blessing

Entsprechend zur Darstellung setzt sich der erste Teil der Arbeit mit den derzeit zur Verfügung stehenden Methoden auseinander, mit deren Hilfe Daten- und Informationsflüsse eines Produktentwicklungsprozesses aufgezeigt und die steigende Komplexität bewältigt werden kann. Es wird zwar die technisch physikalische Betrachtung des Produktes berücksichtigt, allerdings wird bereits hier der Fokus auf die prozessuale Betrachtung des PEP gelegt. Hier gilt es Lücken zu identifizieren, aber auch Ansätze zu finden, die helfen könnten, das gesteckte Ziel dieser Arbeit zu erreichen. Die Erkenntnisse aus Kapitel 2 führen zur Formulierung der Forschungsfragen in Kapitel 3.

In der zweiten Phase wird die Möglichkeit untersucht, den Daten- und Informationsfluss am Beispiel des Absicherungsmanagements mit Hilfe eines auf der Netzwerktheorie basierten Modells zu unterstützen – vergleiche Kapitel 4. Dafür wird die Nutzung von netzwerkartigen Modellen untersucht und es werden Möglichkeiten erarbeitet, um den notwendigen DIF beschreiben und analysieren zu können.

In der dritten Phase (Kapitel 5) wird die methodische Erweiterung beschrieben. Es wird erörtert, wie die Daten für das Absicherungsmanagement in ein Modell überführt und ausgewertet werden. Eine Guideline erklärt, wie das Framework aus den verschiedenen zuvor analysierten Werkzeugen aufgebaut werden muss, um die Mittel der Graphen- und Netzwerktheorie für die Problemstellung nutzen zu können.

Die vierte und letzte Phase dieser Arbeit bewertet die Umsetzung, evaluiert das Modell und beschreibt somit seine Grenzen – vergleiche Kapitel 6. Dabei werden repräsentativ einzelne Analysen erläutert, wodurch die Daten- und Informationsflüsse anhand eines theoretischen Beispiels sichtbar und somit analysierbar werden. Dem Anwender wird somit erläutert, wie das Framework anzuwenden ist und an welcher Stelle Anpassungen für

die eigene Situation notwendig sind. Es werden somit keine Fallbeispiele untersucht, sondern exemplarisch plausibilisiert, welche neuen Möglichkeiten durch das Framework entstehen. Der Ausblick – in Kapitel 7 -weist auf mögliche Erweiterungen hin.

2 Stand der Technik

Dieses Kapitel behandelt die Methodik, welche für Unterstützung bei der Komplexitätsbewältigung im Erkennen und Analysieren von DIF sorgen kann. Ziel ist es, zu untersuchen, welche Ansätze dabei hilfreich sind, die DIF aufzudecken, zu analysieren und für die genannten Aspekte zu verwenden.

Angelehnt an die aktuelle Norm Automotive SPICE (ASPICE) aus der Automobilindustrie, welche verschiedene Level für eine Prozessgüte vorschreibt, kann abgeleitet werden, welches Verständnis über die DIF gefordert ist. Die „ASPICE-Norm“ schreibt dabei fünf Level vor, die erreicht werden können. Dabei startet Level 0 bei nicht vorhandener Prozessdokumentation und geht über einen dokumentierten Prozess in einen mess-, manage- und schließlich steuerbaren Prozess über (Automotive, 2010). Das Ziel dieser Arbeit ist es die DIF vergleichbar zu den ASPICE-Leveln nicht nur zu erkennen, sondern diese zu messen, um den Prozess gezielt steuern zu können – vergleiche Abbildung 3:

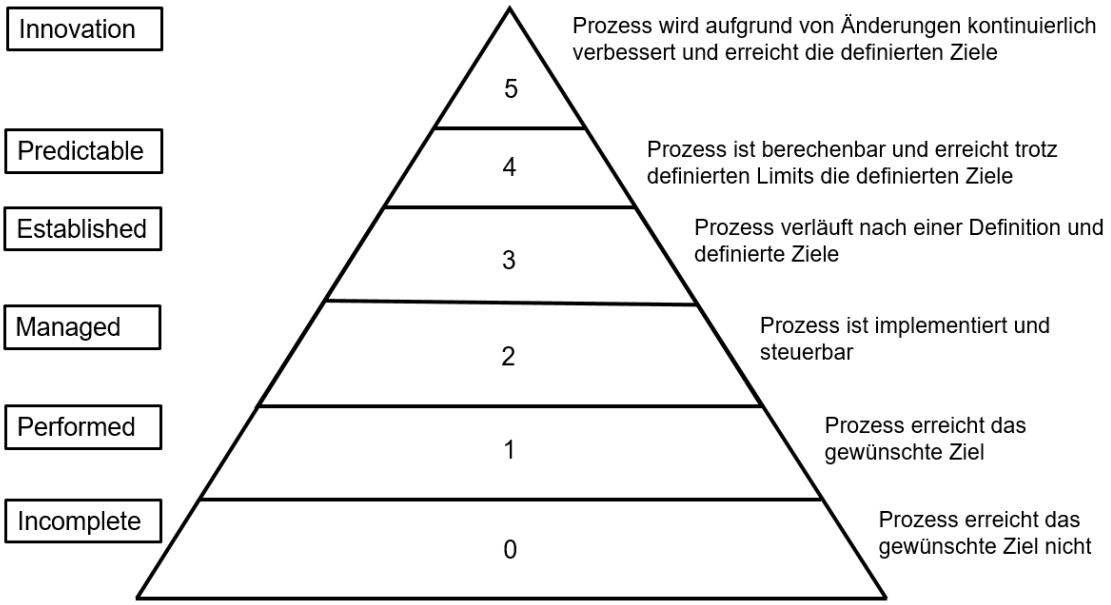


Abbildung 3: Sechs Level der Prozessgüte nach ASPICE (Automotive, 2010)

Das heißt die DIF der Produktentwicklung müssen in erster Linie nachvollzogen werden – vergleiche ASPICE Level 1. Um das zu erreichen, gilt es die DIF zu identifizieren. Erst dann können die DIF analysiert werden und ermöglichen – analog zu Level 2 - eine gezielte Steuerung des PEP. Nur dann kann der PEP, wie in Level 3, nach definierten Zielen ablaufen und sogar nach bestimmten Faktoren, wie am Beispiel der Entwicklungszeit, optimiert werden – vergleiche Level 4. In Summe muss der PEP auf der Grundlage des Verständnisses über DIF stetig verbessert werden – vergleiche Level 5.

Die folgenden Unterkapitel behandeln deshalb zuerst die bekannten Ansätze aus der methodischen Produktentwicklung und dann in Folgendem speziell das Anforderungsmanagement, um aus einem Grundverständnis diejenigen DIF aufzudecken, die notwendig sind, um das Ziel dieser Arbeit zu erreichen.

2.1 Vorgehenslogik in der Produktentwicklung

2.1.1 Systemischer Ansatz zur Bewältigung von Komplexität

2.1.1.1 Definition und Bedeutung des Systems

Wie bereits eingeführt, werden in diesem Kapitel Methoden behandelt, um die wachsende Komplexität zu bewältigen. Dazu wird die Definition nach Ulrich wieder herangezogen, in welcher er das „**System**“ betrachtet, das in einer komplexen Umgebung definiert ist.

Ziel des systemischen Ansatzes ist es also, die Beziehungen seiner Elemente untereinander und zu einer definierten Umwelt zu erfassen. Somit können die Struktur und das Verhalten des Systems aufgezeigt und in Folge gesteuert werden. Dafür ist es jedoch notwendig, nicht nur Teile des Systems zu erfassen, sondern dieses und dessen Umwelt ganzheitlich zu betrachten. Natürlich gilt die Voraussetzung, dass das System für einen bestimmten Zweck entwickelt wird.

Da fast jedes System Bestandteil einer Anwendungslandschaft ist, folgt daraus die Notwendigkeit, das System abzugrenzen. Das heißt, es muss definiert werden, welche Bestandteile zum System gehören und welche außerhalb des Systems stehen und damit der „Umwelt“ angehören. Es hilft zu erfassen, welche Teile gestaltbar sind. Bei Teilen auf die das System zurückgreifen muss, die jedoch außerhalb der eigenen Einflussnahme stehen, spricht man von einem Nachbarsystem bzw. – Systemen (Haberfellner, R., de Weck, O., Fricke, E., & Vössner, 2012). Durch diese Abgrenzung lassen sich interne Abhängigkeiten, der Austausch zu Nachbarsystemen und der Einfluss der Umwelt erkennen (Taket, Checkland und Holwell, 1999). Die Umwelt beinhaltet alle Elemente, die das System beeinflussen und gleichzeitig außerhalb der Reichweite der Veränderung liegen.

Zusammenfassend ist nach Patzack ein System als eine „Menge von Elementen, welche Eigenschaften bzw. Funktionen besitzen, und welche durch Beziehungen (Relationen) miteinander so verknüpft sind, dass ein gesetztes Ziel verfolgt werden kann“ definiert (S.19 (Patzak, 1982)).

Eine weitere Definition beleuchtet die Elemente näher. Demnach besteht das System “aus Elementen, die durch strukturelle, hierarchische und funktionale Beziehungen verbunden

sind. Das zu entwickelnde System wird durch eine willkürlich gesetzte Grenze (Systemgrenze) festgeleg“ (S.66 (Gausemeier *u. a.*, 2013)). In Summe besteht folglich ein System aus Elementen, die innerhalb einer festgelegten Grenze sind. Diese können je nach Blickwinkel unter verschiedenen Gesichtspunkten betrachtet, differenziert und definiert werden. Das Systemdenken „ermöglicht es, komplexe Systeme besser verstehen und gestalten zu können. Im Mittelpunkt des Systemdenkens steht die modellhafte Abbildung. Dadurch können komplexe Systeme und Zusammenhänge veranschaulicht werden“ (S.67 (Gausemeier *u. a.*, 2013)).

Zwischenfazit:

Um die Komplexität zu bewältigen, zeigte sich der systemische Ansatz als zielführend, da er die Elemente erfasst, die innerhalb eines zu definierten Systems liegen und versucht die Abhängigkeit intern und extern zu beschreiben.

Allerdings bleiben Fragen in Bezug auf die Definition der Systemgrenze und dem Austausch des Systems mit der Umwelt offen. Ebenfalls sind verschiedene Sichtweisen auf das „System“ möglich, die es zu erörterten gilt.

2.1.1.2 Der ZOPH-Ansatz

Die verschiedenen Betrachtungsweisen und damit die erste Differenzierung des „Systems“ werden durch den ZOPH-Ansatz nach Negele initiiert. Der Ansatz nach Negele beschreibt eine übergeordnete Systemmodellierung (Rudolph *u. a.*, 1999). Dabei wird das Systemmodell inhaltlich gruppiert und strukturiert, wodurch die Vernetzungen innerhalb des Systemmodells verdeutlicht werden. Der Prozess wird als Lösung einer Problemstellung verstanden. Die Realisierung erfolgt schrittweise, bis hin zum realen Objekt.

Im Detail werden die äußeren Einflüsse, das heißt die Systemumgebung oder Systemumwelt, die nicht zum zu entwickelnden System gehört, abgegrenzt. Das System wird in vier Teilbereichen gruppiert (Negele, Fricke und Igenbergs, 1997):

- **Z**ielsystem (goal system): Probleme, Wünsche, Ziele, Anforderungen, Pflichten, Normen, Gesetze. Handlungsergebnisse, die zu realisieren sind.
- **O**bjektsystem (product system): Produkt und alle Zwischenergebnisse – egal in welcher Form. Dokumente, Modelle, Prototypen.
- **P**rozesssystem (process system): Handlungen, die notwendig sind, um das Ziel zu erreichen. Auch Verknüpfungen untereinander, wie Informationsflüsse und Abhängigkeiten zwischen den Handlungen.
- **H**andlungssystem (agent system): In erster Linie betroffene Personen, die handeln. Aber auch Ressourcen, Software, Knowhow.

Die Abhängigkeiten der vier Systeme untereinander werden in Abbildung 4 (Negele, Fricke und Igenbergs, 1997) verdeutlicht. Demzufolge sind Einflussfaktoren, welche außerhalb des zu entwickelnden (eigenen) Systems liegen, als „Systemumwelt“ dargestellt.

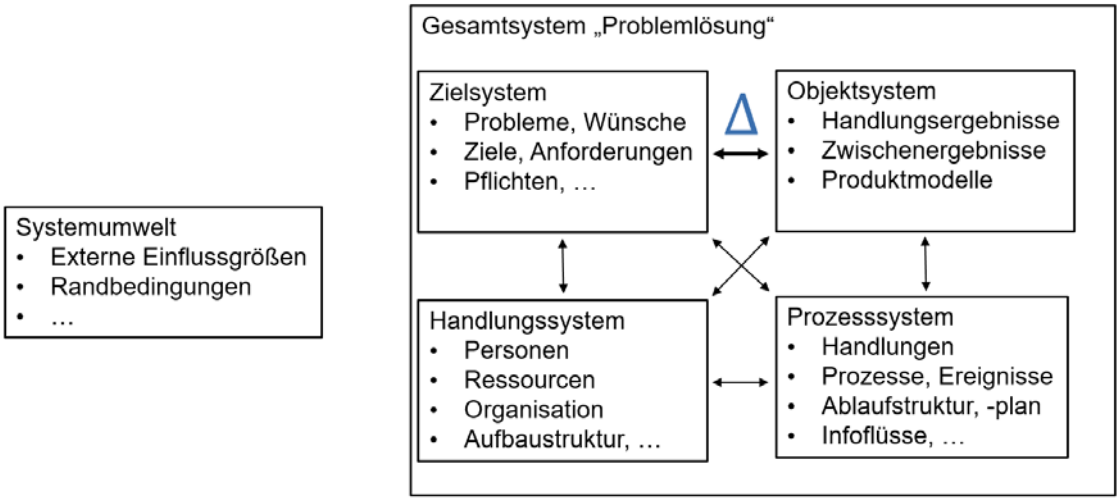


Abbildung 4: ZOPH-Modell nach Negele (Negele, Fricke und Igenbergs, 1997)

Ein Vorteil des ZOPH-Ansatzes gegenüber der bisher erwähnten Betrachtung ist die Fokussierung auf die Rahmenbedingungen - mit Blick auf die Anforderungen, die nun durch die situationsbezogene Adaption in den eigenen Kontext gesetzt werden können – vergleiche Zielsystem. Es werden Teilsysteme miteinander vernetzt. Diese stehen mit der Systemumwelt im gegenseitigen Austausch. Je nach Fokus kann nun das System differenziert werden. Die Informationsflut kann somit strukturiert werden, wodurch die Komplexität handhabbar wird.

Zwischenfazit:

Der Entwickler kann durch die Aufteilung und einem Delta (Δ), das zwischen Zielsystem und Objektsystem besteht, die Diskrepanz zwischen Anforderung und tatsächlichem Produkt erkennen. Es ist jedoch nicht ersichtlich, wie mit Unsicherheiten umzugehen ist, da dieses Modell den Einfluss durch die Randbedingungen nicht näher analysiert, sondern lediglich den Problemlösungszyklus verständlicher macht. Ebenfalls ist die Differenzierung nach den vier Kategorien keine exakte Trennung. Die Grenzen der vier Systembetrachtungen sind fließend und bilden nur eine grobe Einteilung.

2.1.1.3 Systems Engineering

Das Systems Engineering (SE) widmet sich der Betrachtung des PEP unter dem systemischen Ansatz. Dieses versteht sich als einen ganzheitlichen Ansatz zur Entwicklung komplexer Systeme (Booton und Ramo, 1984). Dabei wird das System ganzheitlich über den Entwicklungsprozess, angefangen bei der Definition der Anforderungen über den Systementwurf bis zur Validierung, betrachtet.

Das integrierte Denken in Zusammenhängen erlaubt es, die Wechselwirkungen zu erfassen und zu verstehen. Dies erfolgt durch die Einordnung in ein größeres Ganzes und den

interdisziplinären Ansatz zur Unterstützung unterschiedlicher Sichtweisen. Dabei kommen einheitliche Begriffe und Modelle zum Einsatz, die die Basis des ganzheitlichen Ansatzes darstellen (Winzer, 2016).

Der Ansatz des SE erfasst das Problem als Ganzes und integriert über die technischen Prozesse die Betrachtung anhängender Managementprozesse, wie der Projektplanung, Qualitätssicherung, Konfigurationsmanagement, etc. Deshalb können diese Erkenntnisse helfen, die bereits aufgeführte Komplexität durch die entsprechende Strukturierung und Differenzierung handhabbar zu machen. Die Chance die DIF somit zu erkennen, steigt, da gezielt nach relevanten Informationen gesucht wird und somit DIF sichtbar und nachvollziehbar werden.

Um relevante Methoden aus dem SE für die Komplexitätsbewältigung zu extrahieren, ist es notwendig den Ansatz des SE genau zu definieren:

Ramo beschreibt: „Systems Engineering ist [...] die Gestaltung und Anwendung von etwas Ganzem (System) […]. Es betrachtet ein Problem allumfassend, indem es alle Aspekte und Einflussgrößen berücksichtigt und soziale und technische Belange in Relation zueinander bringt.“ (S.306 (Booton und Ramo, 1984)).

Nach Oliver Alt ist SE: „eine Gesamtheit von Elementen, die aufeinander bezogen sind […] und sich in dieser Hinsicht gegenüber der sie umgebenden Umwelt abgrenzen.” Es ist eine „Menge von Elementen […] die gemäß einem Entwurf in einer Beziehung zueinanderstehen.” „Ein Systemelement kann auch selbst wieder ein System sein.” (S.7 (Alt, 2012)).

Nach INCOSE stellt: „Systems Engineering (SE) einen interdisziplinären Ansatz zur Unterstützung der Realisierung von erfolgreichen Systemen dar. SE fokussiert darauf, die Kundenbedarfe und die geforderte Funktionalität möglichst früh im Entwicklungsprozess zu definieren, die Anforderungen zu dokumentieren […]. Systems Engineering betrachtet sowohl die wirtschaftlichen als auch die technischen Bedarfe aller Kunden, mit dem Ziel, ein qualitativ hochwertiges Produkt zu schaffen, das den Bedarfen des Kunden gerecht wird“ (S.11 (INCOSE, 2007)).

In Summe beschreiben die Definitionen einen ganzheitlichen Top-down-Ansatz, welcher alle relevanten Elemente erfasst, die im Produktlebenszyklus zu berücksichtigen sind. Diese enthält alle Prozesse der Systemlebensdauer und umfasst somit sowohl die technischen- als auch die Managementprozesse. Als Indikator gilt dabei, die Anforderungen so früh wie möglich zu erkennen und entsprechend ihrer Erfüllung nachzuweisen.

Da es keine Komplexitätsreduzierung wäre, alle Elemente und ihre Abhängigkeiten zu erfassen, gibt die Philosophie des SE vor, für den Problem-Lösungs-Zyklus zwischen einem technischen System und dem Vorgehensmodell zu differenzieren - vergleiche Abbildung 5.

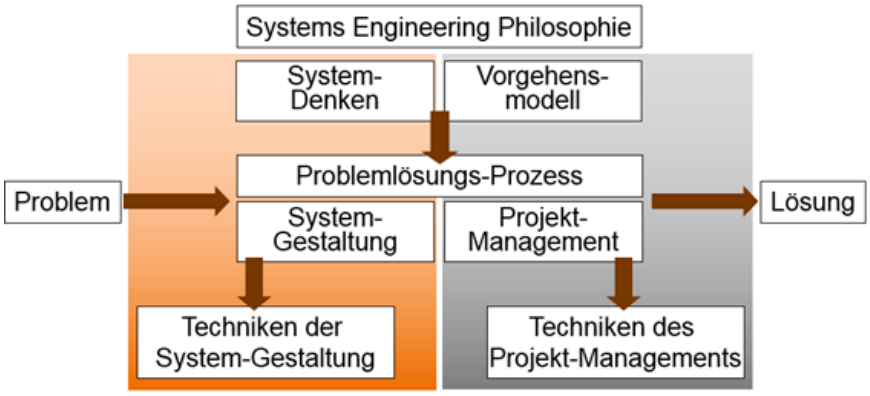


Abbildung 5: Dualität der Systems Engineering Philosophie nach Haberfellner (Haberfellner, R., de Weck, O., Fricke, E., & Vössner, 2012)

Haberfellner unterscheidet entsprechend die System-Gestaltung und die Techniken des allgemeinen Projekt-Managements, welche einerseits die Systemspezifikation und andererseits die Koordination der Entwicklungsaktivitäten betreffen (Paetzold, 2017)

Betrachtet man nur den **technisch-physikalischen** Gesichtspunkt, eignet sich die Unterscheidung von Oliver Alt. Ihm zufolge besteht das SE hauptsächlich aus den drei Bausteinen – vergleiche Abbildung 6:

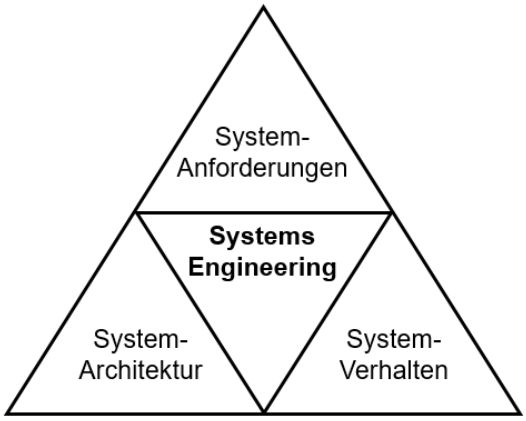


Abbildung 6: Das Dreieck des Systems Engineerings nach Alt (Alt, 2012)

Hier steht das System als Produkt im Vordergrund – angefangen bei den Anforderungen, über die Architektur, bis hin zum Verhalten. Dieser Fokus hilft die Komplexität zu reduzieren, da es das System jeweils unter einen bestimmten Aspekt stellt. Die Anforderungen sind dabei der Trigger für den PEP und Indikator. Die Architektur wird bei der Herstellung und dem Design eine wichtige Rolle spielen. Die Summe von allem ist das Verhalten, das die Funktionalität des Produktes schließlich ergibt.

Nach dem SE-Systemdenken kann das Produkt in all seinen Ebenen wiedergegeben und vom Groben zum Detail betrachtet werden. Umgekehrt können Subsysteme zu Systemen zusammengefasst werden. Dies erlaubt es, das Produkt beliebig zu detaillieren.

Eine ähnliche Aufteilung ist aus der Automobilbranche bekannt. Dort werden zugekaufte Systeme, Komponenten oder Teile nach der Zuliefererpyramide nach verschiedenen Ebenen/ Leveln - englisch „Tier“ – unterteilt – vergleiche Abbildung 7:

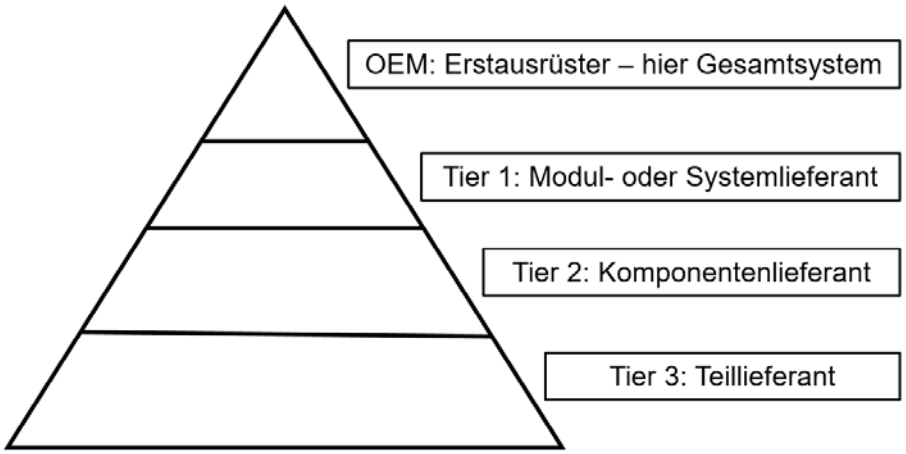


Abbildung 7: Pyramide der Hersteller und Zulieferer nach Reh (Daniel Reh, 2009)

Dabei steht der Hersteller bzw. Originalausrüstungshersteller (OEM) an der Spitze.

Allerdings sind für die ganzheitliche Erfassung des Systems auch die unterstützenden **Management-Prozesse** relevant, da sie großen Einfluss auf die Entscheidungen und den Verlauf des PEPs haben. Im Detail folgen diese dem Management-Cycle und weisen folgende Bereiche auf:

- Das Risikomanagement
- Die Produkt- und Qualitätssicherung
- Das Absicherungsmanagement
- Anforderungsmanagement & -Engineering
- Das Konfigurationsmanagement/ Änderungswesen

Abbildung 8 verdeutlicht, dass die Managementprozesse nicht klar voneinander abzugrenzen sind und dass sie als Teildisziplinen im SE integriert sind.

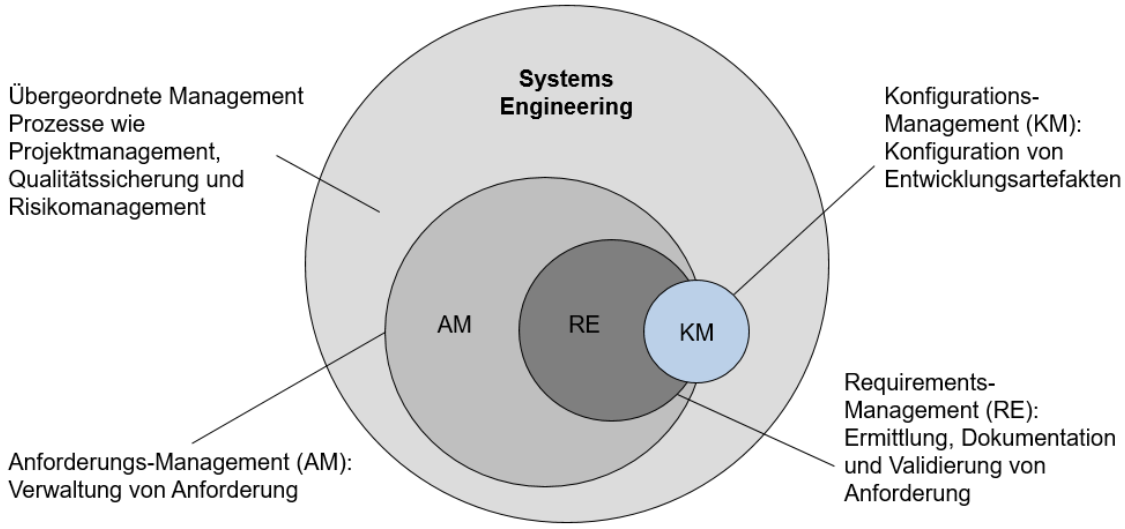


Abbildung 8: Zusammenhang von Management-Prozesse innerhalb von Systems Engineering

Angefangen beim **Risiko-Management** (RiM), das heißt der Nichterfüllung von Anforderungen, verfolgt das RiM nach Lindemann vier Hauptziele:

- Transparenz zu schaffen – bezüglich Existenz und Einfluss von Entwicklungsrisiken
- Risiken hinreichend zu berücksichtigen, um kritische Entscheidungsprozesse zu unterstützen
- Risiken auf ein Minimum zu reduzieren, um Kosten zu senken
- Unbekannte Risiken mittels bestimmter Methoden aufzudecken

Deckt man die Risiken auf und analysiert deren Einfluss auf den PEP, kann das Ursache-Wirkungs-Prinzip aufgezeigt werden. Eine solche Analyse hilft Unwägbarkeiten zu erkennen und die DIF zu vervollständigen. Geleichzeitig ist das Gebiet des RiM ein umfassendes Gebiet mit vielen Fassetten und Möglichkeiten (Lindemann und Lindemann, 2016).

Absichtlich hier allgemein gehalten, zeigt exemplarisch die ISO 31000 die Mindestanforderungen an Risikomanagement und gibt folgenden Prozess an die Hand –vergleiche Abbildung 9.

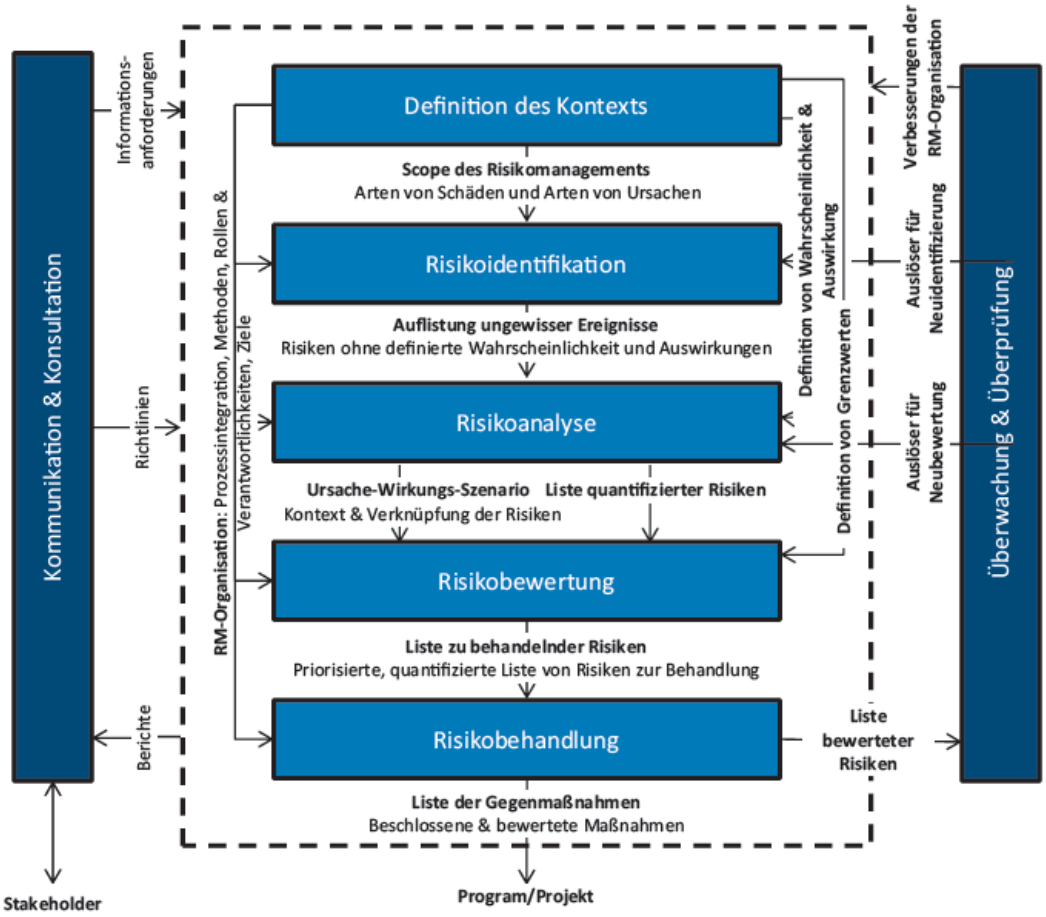


Abbildung 9: Auszug aus ISO 31000 für das Risikomanagement (ISO - The International Organization for Standardization, 2009)

Jedes dieser Felder stellt im RiM ein eigenes Gebiet dar und kann beliebig ausdetailliert werden. Allerdings liegt der Fokus dieser Arbeit auf der Identifizierung, Darstellung und Analyse der DIF in komplexen PEP, die nicht ausspezifiziert werden. Deshalb ist der Prozess nach ISO 31000 für die Intensionen dieser Arbeit völlig ausreichend. Selbst innerhalb dieses Prozesses kann die Definition des Kontextes ausgegrenzt werden, da diese vorausgesetzt werden kann. Der Prozess legt das Augenmerk auf das Risiko, eine Anforderung nicht zu erfüllen (Marc Neumann, 2017), stellt Abhängigkeiten zum Produkt her, beleuchtet weitere Anforderungen und bietet somit eine gute Möglichkeit, die DIF nachzuverfolgen. Folglich ist dieses identifizierte Risiko analysierbar und erlaubt es, geeignete Gegenmaßnahmen zu treffen. Im Umkehrschluss kann die nachträgliche Analyse solcher Entscheidungen helfen, in Zukunft das Risiko die Anforderungen nicht zu erfüllen zu minimieren, indem entsprechende Gegenmaßnahmen proaktiv eingeplant werden.

Ein Grund für das **Qualitätsmanagement** (QM) ist die Reduzierung von Fehlern und damit die Vermeidung einer unnötigen Steigerung der Entwicklungskosten bzw. einer Verlängerungen der Entwicklungsdauer – vergleiche Abbildung 10:

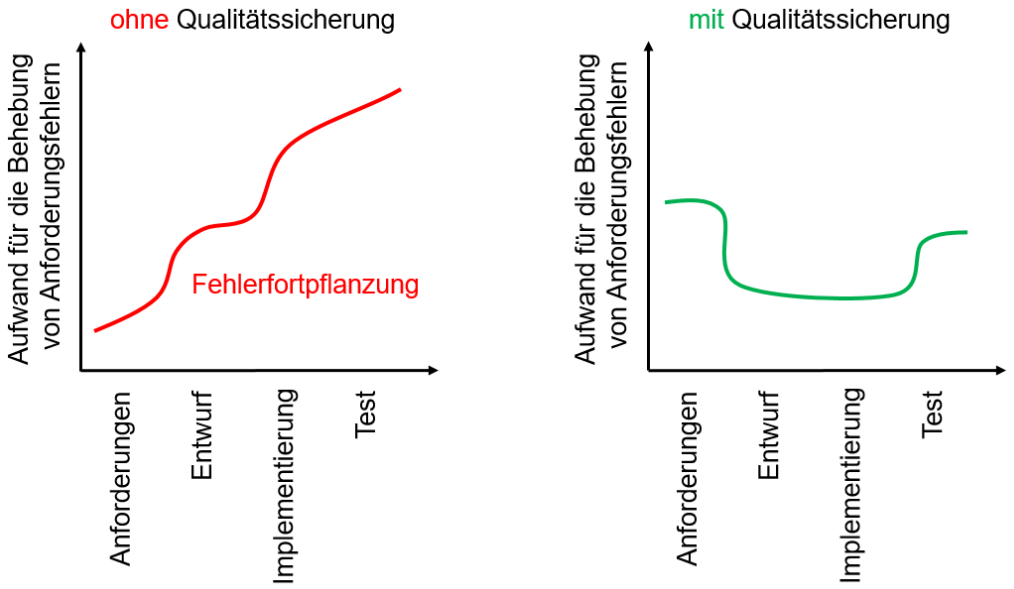


Abbildung 10: Vergleich von Fehlerfortpflanzung mit und ohne Qualitätsmanagement nach Brauns (Christoph Brauns, 2016)

Auf den ersten Blick spielt das QM eine übergeordnete Rolle bei der Entwicklung komplexer Systeme. Allerdings lassen sich Parallelen zwischen der Fehlerfortpflanzung und dem dadurch entstehenden Aufwand einerseits und einer unnötigen Steigerung der Komplexität andererseits ziehen. Daher ist es unerlässlich die Qualitätssicherung in den frühen Phasen mit einzugliedern, um die Komplexität so gut wie möglich zu beherrschen.

Allgemein unterscheidet man zwei Arten des QM:

Konstruktive Qualitätssicherung: Vermeidung von Fehlern während der Erstellung

- Richtlinien
- Vorgehensmodelle
- Methoden

Analytische Qualitätssicherung: Überprüfung auf Fehlern nach der Fertigstellung

- Reviews
- Perspektivenbasiertes Lesen
- Prüfung durch Prototypen
- Einsatz von Checklisten

Die Entscheidung, welche Methode sich in welcher Situation eignet, richtet sich nach den D&I in der jeweiligen Entwicklungsphase. Das heißt das Verständnis über die DIF kann als Basis dienen, um zwischen den Methoden des QM wählen zu können. Man unterscheidet zwischen „vor SOP (Start of Production)“ und „nach SOP“. Im weiteren Verlauf dieser Arbeit liegt der Fokus auf Entwicklungstätigkeiten vor dem SOP. Dies beinhaltet

dennoch Überprüfungen, ob das gewünschte Produkt in seinem Entwicklungsstadium den Anforderungen entspricht oder nicht.

Nimmt man die Anforderungen als Beispiel, wird nach Rupp, etc. eine vordefinierte Qualitätsanforderung an die Definition gestellt, die auch notwendig ist, um die Information zufriedenstellend zu dokumentieren (Klaus Pohl, 2011; Rupp *u. a.*, 2014). Diese Qualität stellt die Grundlage für die Nachverfolgung der DIF dar. Sollten die Qualitätskriterien vernachlässigt werden, minimiert dies die Möglichkeiten, die Abhängigkeiten zu analysieren.

Man kann die Qualität der Anforderungen in Semantik, Syntax und Abgestimmtheit mit den Stakeholdern unterteilen. Bei der Semantik wird überprüft, ob alle relevanten Anforderungen ermittelt und im erforderlichen Detaillierungsgrad erfasst werden. Der Syntax bezieht sich auf die festgelegten Dokumentations- und Spezifikationsvorschriften. Die Abgestimmtheit hingegen prüft, ob alle Stakeholder mit den dokumentierten Anforderungen übereinstimmen bzw. alle Konflikte aufgelöst sind. Beispielkriterien für diese drei Aspekte sind (IEEE-Computer-Society, 1998):

Semantik:

- Vollständig
- Verfolgbar
- Korrekt
- Konsistent
- Überprüfbar

Syntax:

- Konform zu Dokumentationsstruktur
- Verständlich
- Eindeutig

Abgestimmtheit:

- Abgestimmt nach Änderungen
- Konflikte aufgelöst

Abgesehen von Qualitäten, die Anforderungen nachweisen sollten, gibt es Prüftechniken, um zu überprüfen, ob die Anforderungen richtig verstanden wurden und entsprechend erfüllt werden.

Pohl nennt zudem weitere Aspekte, die die Anforderungen erfüllen sollten (Pohl und Rupp, 2015):

- Beteiligung der richtigen Stakeholder

- Trennung von Fehlersuche und Fehlerkorrektur
- Prüfung aus unterschiedlichen Sichten
- Geeigneter Wechsel der Dokumentationsform
- Konstruktion von Entwicklungsartefakten
- Wiederholte Prüfung

Wie aus der Beschreibung ersichtlich wird, steht die Erfüllung der Kundenanforderungen in der Gesamtbetrachtung der Entwicklungssituation als System im Vordergrund.

In der Summe helfen Qualitätskriterien hinsichtlich der Anforderungsbeschreibung und Nachverfolgung Fehler zu minimieren und die Komplexität zu reduzieren. Es zeigt sich darüber hinaus, wie wichtig es für den Erfolg des PE ist, Anforderungen richtig zu formulieren und deren Erfüllung zu überprüfen.

Folglich sind Absicherungsmaßnahmen aus dem **Absicherungsmanagement** im Entwicklungsprozess von essenzieller Bedeutung, da hier fortlaufend überprüft wird, ob das Produkt die Anforderungen erfüllt oder nicht. Auf diese Weise wird einerseits die prozessbegleitende Absicherung unterstützt und andererseits die Rückkopplung zur Entwicklung ermöglicht. Dies impliziert die Integration des Verifikations- und Validierungsmanagements. In Summe kann am Beispiel des fortlaufenden Absicherungsmanagements der DIF verfolgt werden. Dies bietet die Möglichkeit einer ganzheitlichen Betrachtung bzw. Verfolgung.

Dafür ist es notwendig zu unterscheiden, was im PEP unter Verifikation und Validierung verstanden wird (VDI, 2004).

Definition **Validierung** nach VDI 2206:

> *Validierung meint ursprünglich die Gültigkeitsprüfung einer Messmethode in der empirischen Sozialforschung, das heißt, inwieweit die Testresultate tatsächlich das erfassen, was durch den Test bestimmt werden soll. Übertragen auf technische Systeme ist hierunter die Prüfung zu verstehen, ob das Produkt für seinen Einsatzzweck geeignet ist bzw. den gewünschten Wert erzielt. Hier geht die Erwartungshaltung des Fachexperten und des Anwenders ein. Die Validierung beinhaltet z.B. die Prüfung, ob die Beschreibung eines Algorithmus mit dem zu lösenden Problem übereinstimmt. Sie ist im Allgemeinen nicht formal durchzuführen. Umgangssprachlich ist die Validierung die Beantwortung der Frage: Wird das richtige Produkt entwickelt?*

Definition **Verifikation** nach VDI 2206:

> *Verifikation meint allgemein den Nachweis der Wahrheit von Aussagen. Übertragen auf technische Systeme ist hierunter die Überprüfung zu verstehen, ob eine Realisierung (z.B. ein Software-Programm) mit der Spezifikation (in diesem Fall mit der Algorithmenbeschreibung) übereinstimmt. Bei der Überprüfung der Gültigkeit eines Programms wird auch von der Programmverifikation gesprochen. Die Verifikation wird im Allgemeinen formal realisiert. Umgangssprachlich ist die Verifikation die Beantwortung der Frage: Wird ein korrektes Produkt entwickelt?*

Das heißt in Summe, dass die Verifikation im Laufe des PEP überprüft, ob die Spezifikation die gestellten Anforderungen – hier sind hauptsächlich interne oder abgeleitete Anforderungen gemeint – erfüllt. Mittels der Validierung wird geprüft, ob die Kundenanforderungen mit dem entwickelten Produkt erfüllt sind. Je nachdem, wo die Grenzen des eigenen Systems gezogen werden, kann eine Verifikation auch zur Validierung werden, wenn zum Beispiel die Entwicklungsabteilung Anforderungen aus dem Vertrieb als externe Anforderungen definiert.

Grundsätzlich gibt es zwei Möglichkeiten Anforderungen zu überprüfen – mittels einer Simulation oder eines Versuchs, um gewünschte Eigenschaften abzusichern (Paetzold, 2017).

Der Zweck eines Versuchs zielt darauf ab, ein gefordertes Verhalten oder auch Struktur zu beobachten. Dafür muss ein entsprechender Versuchsaufbau das gewünschte Verhalten erzeugen bzw. geeignet sein, die Struktur zu überprüfen. Im Falle einer Funktion eignet sich beispielsweise eine zeitliche Beobachtung und entsprechende Aufzeichnung. Für eine Strukturanalyse - zum Beispiel eine Festigkeitsanalyse – reicht es, nach Ablauf des Versuchs das Endergebnis zu messen. In jedem Fall müssen die Messdaten interpretiert werden und liefern Rückschlüsse auf die Erfüllung der Produktanforderungen. In Summe bleibt somit der Versuch ein einmaliges Ereignis, dessen Aufbau kosten- und zeitintensiv und nur bedingt wiederholbar ist, weil die Randbedingungen sich leicht variieren können. Die Ergebnisgüte ist zudem direkt vom Prototypen und der eingesetzten Messtechnik abhängig (Schönwald *u. a.*, 2019).

Die Simulation unterscheidet sich grundlegend vom Versuch, weil hier ein Modell eingesetzt wird. Diese ist als ein vereinfachtes Abbild der Wirklichkeit (Stachowiak, 1973) und gibt exklusiv zuvor definierte Eigenschaften wider. Das heißt es bezieht sich auf ein bestimmtes Verhalten oder Eigenschaften des Produktes und schließt Erkenntnisse darüber hinaus aus. Der Ablauf einer Simulation ist somit vorher definiert und beschränkt sich durch definierte Testszenarien und Algorithmen auf den bekannten Lösungsraum (Schönwald *u. a.*, 2019). Dieses erlaubt eine beliebige Wiederholung, weil das Modell und die Randbedingungen identisch sein können. Das heißt im Umkehrschluss, dass eine

Simulation kostengünstiger bleibt, die Testfrequenz unbeschränkt ist und Ergebnisse eine starke Abhängigkeit von der Modellgüte und den Testszenarien und Algorithmen aufweisen (Reitmeier, Chahin und Paetzold, 2015).

In Bezug auf den PEP muss der Entwickler auf Grundlage der vorhanden D&I zum Produkt, aber auch zu Abhängigkeiten des Produktes zur Organisation, zu vor- und nachgelagerten Prozessen und zu den Anforderungen entscheiden können, wann ein Versuch oder eine Simulation einen höheren Mehrwert ergibt. Diese Frage bleibt somit eine Entscheidung, welche nur auf dem Verständnis des Entwicklers über die DIF des PEP beruht.

Für eine Eigenschaftsabsicherung mittels Simulation – sei es eine Verifikation oder Validierung – bieten sich verschiedene Möglichkeiten an (Schick *u. a.*, 2008; Nibert, Herniter und Chambers, 2012):

- Model in the Loop (MiL) entspricht hauptsächlich der bereits erklärten Simulation, wie beispielsweise dem Einsatz eines CAD-Modells in einer FE-Berechnung
- Software in the Loop (SiL) beschreibt eine Software, das anstelle des Modells zum Einsatz kommt, wie zum Beispiel die Software eines Notbremsassistenten zur Reaktion auf ein Hindernis
- Hardware in the Loop (HiL) ist die Erweiterung von SiL, indem Bestandteile des Testings durch physische Hardware ersetzt werden. Dies kann zum Beispiel das Steuergerät des Notbremsassistenten sein, das nun in einer Regelstrecke mittels simulierter Einflussgrößen auf sein Verhalten überprüft wird.

Exemplarisch wird hier der Übergang von MiL zum SiL erläutert (rein virtuell). Abbildung 11 (Sandmann, 2020) zeigt im MiL ein Funktionsmodell und eine Signalvernetzung der einzelnen System-Elemente.

Im SiL werden diese mit einem Quellcode unterlegt, womit es nun möglich ist zu überprüfen, wie die einzelnen Programm-Elemente (Funktionen) über die vordefinierte Schaltung miteinander agieren. Damit lässt sich das Verhalten testen. SiL liefert noch vor dem Bau von Prototypen Erkenntnisse über evtl. Fehler.

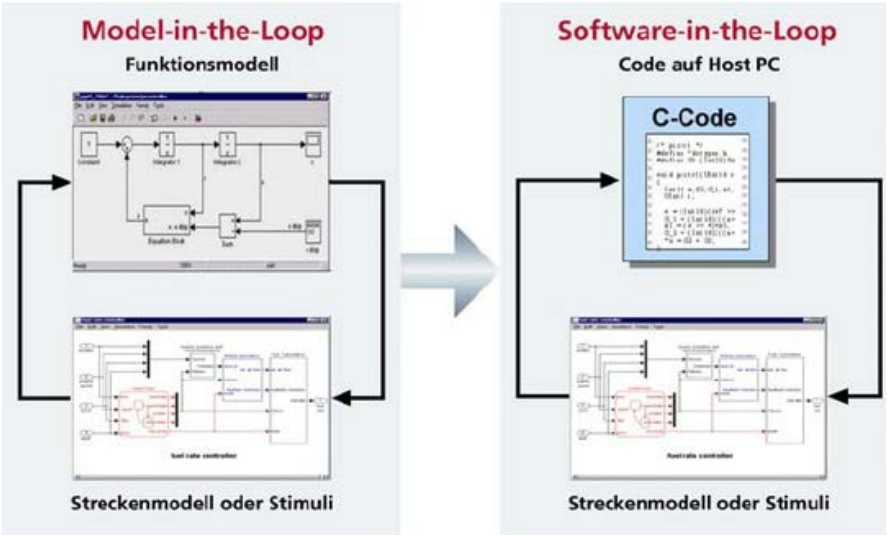


Abbildung 11: Beispiel für MiL und SiL nach Sandmann (Sandmann, 2020)

Allgemein lässt sich auch ein Übergang von MiL zu SiL beobachten. Wendet man beispielsweise am PEP die genannten Prüf-Methoden an, kann man aus der Praxis folgern, dass in der frühen Phase des PEP hauptsächlich MiL betrieben wird. Mit immer wachsender Produktreife verlagert sich die Eigenschaftsabsicherung von MiL zu SiL und schließlich zu HiL - vergleiche Abbildung 12.

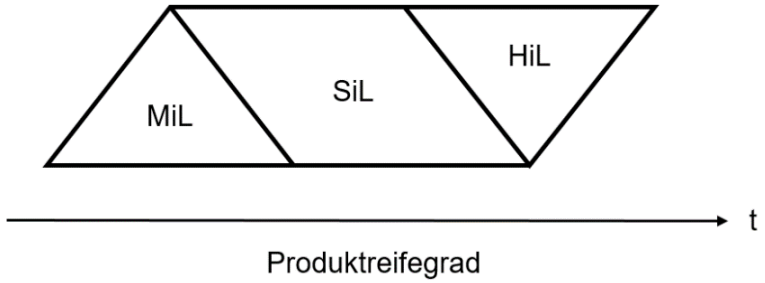


Abbildung 12: Übergang von MiL über SiL zu HiL mit voranschreitender Produktreife

Folgt man demzufolge der Eigenschaftsabsicherung entlang des PEP, kann man die DIF aus Sicht der Absicherung betrachten. Dies hat zum Vorteil, dass ständig ein Abgleich mit den zu erfüllenden Anforderungen erfolgt, womit die Analyse des technisch-physikalischen Systems abgedeckt ist. Gleichzeitig können daraus abgeleitete Prozesse analysiert und verfolgt werden. Dies erfüllt den prozessualen Aspekt des SE.

Im weiteren Verlauf dieser Arbeit wird exemplarisch ein SiL betrachtet und modelliert, um den Rahmen nicht unnötig zu erweitern.

Zwischenfazit: Systemischer Ansatz

In Summe liefert der systemische Ansatz – sei es die technisch-physikalische Betrachtung oder die Beleuchtung der Management-Prozesse – eine gute Basis, die DIF entlang des

Lebenszyklus während der Entwicklung zu identifizieren und nachzuverfolgen. Die bisher gezeigten Ansätze beleuchten den PE-Prozess im Allgemeinen, wobei der SE-Ansatz den System-Gedanken fokussiert und durch den top-down-Ansatz erlaubt die DIF systematisch zu verfolgen. Zudem integriert der SE-Ansatz nicht nur das zu entwickelnde System, sondern auch relevante Prozesse und Randbedingungen. Allerdings bleiben diese Ansätze zu generisch, um den DIF in seiner vollständigen Komplexität wiederzugeben. Vielmehr geben sie einen groben Rahmen, welcher für eine detaillierte Modellierung verwendet werden kann. Die Schwierigkeit liegt in der Definition des Systems und dessen Systemgrenzen. Wird diese Definition zu eng gehalten, besteht die Gefahr, relevante Aspekte des Systems zu vernachlässigen, was die Interpretation der Zusammenhänge verfälscht. Diese Gefahr bezieht sich sowohl auf das System selbst, als auch auf die Umgebung. Das heißt die Aussagekraft bzgl. der Zusammenhänge der DIF ist direkt von der Annahme hinsichtlich des Systems, bzw. der Umgebung, abhängig. Diese Einschränkung lässt sich auf die prozessuale Betrachtung analog übertragen und reicht somit nicht aus, um die die DIF in ihrer Gesamtheit zu identifizieren und differenzieren.

2.1.2 Prozessmodelle für die Darstellung von Produktentwicklungsprozesse

Der systematische Ansatz reduziert die Gefahr die DIF nicht zu erfassen, weil nicht nur das Produkt, sondern durch die Systemgrenzen der In- und Output aufgenommen wird. Um die DIF ganzheitlich betrachten zu können, muss der PEP unter verschiedenen Blickwinkeln analysiert werden. Dabei spielt der Auflösungsgrad eine maßgebende Rolle. Die PE kann auf strategischer Ebene betrachtet werden – Makrologik- oder auf elementarer Ebene – Mikrologik (Lindemann, 2009)

Die Makrologik reduziert den PEP auf einen relativ niedrigen Auflösungsgrad und gibt einen guten Überblick über den Ablauf, womit die Anzahl der Elemente vermindert und somit die Komplexität auf einem Minimum gehalten wird (Lindemann, 2009)

In Abhängigkeit von den Anwendungen existieren eine Vielzahl von graphischen Repräsentationen, die eine Problemstellung induzieren. Das **V-Modell** stellt in diesem Sinne den zeitlichen Bezug zur **Makrosicht** her und teilt den PEP entsprechend in drei Haupt-Phasen (4Soft GmbH, Bund und Innern, 2009):

- System-Entwurf
- Domänenspezifischen Entwurf bzw. Spezifikation
- System-Integration

In der VDI 2206 sind diese Phasen weiter spezifiziert und geben einen guten zeitlichen Bezug der Phasen wieder, anhand dessen die DIF ebenfalls gut nachzuvollziehen sind. So

werden die Anforderungen als Start und die Eigenschaftsabsicherung als Folge der Systemintegration nach dem Systementwurf aufgeführt – vergleiche vorheriges Kapitel. Für den Einstieg in den Entwicklungsprozess steht im ersten Zyklus ein domänenübergreifendes Lösungskonzept zur Verfügung, das physikalische und logische Schnittstellen definiert – vergleiche Abbildung 13:

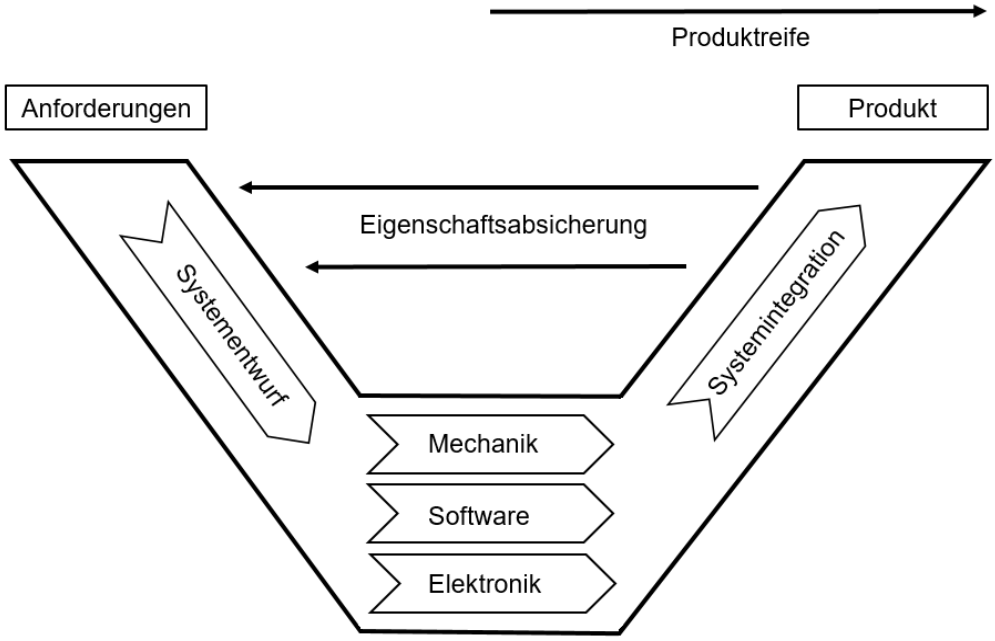


Abbildung 13: V-Modell nach VDI 2206 (VDI 2206, 2004)

Anschließend wird die Modellbildung und -analyse domänenspezifisch gestaltet. Mit dem Übergang von der waagrechten- zum rechten Ast, nimmt die Produktreife zu. Die Eigenschaftsabsicherung erfolgt durch einen Abgleich nach jeder Phase mittels Verifikation und Validierung. Die Verifikation beantwortet somit die Frage nach der richtigen Entwicklung und die Validierung die Frage nach dem richtigen Produkt *(genaue Definition später im Kapitel Requirements-Engineering).*

Die Verfolgung des Produktreifegrades anhand der Anforderungen und deren Absicherung helfen zu verstehen, wie die DIF wachsen und welche Entscheidungen auf deren Grundlage getroffen werden. Dabei ist es wesentlich zu verfolgen, wie sich Änderungen der DIF auf die Entscheidungen (Design, Management, etc.) auswirken.

Der **Mikrozyklus** beschreibt den Problemlösungszyklus, bestehend aus Analyse, Synthese und Bewertung (Lindemann, 2009), welcher sich im PEP wiederholt und keiner bestimmten Phase zugeordnet wird – vergleiche Abbildung 14:

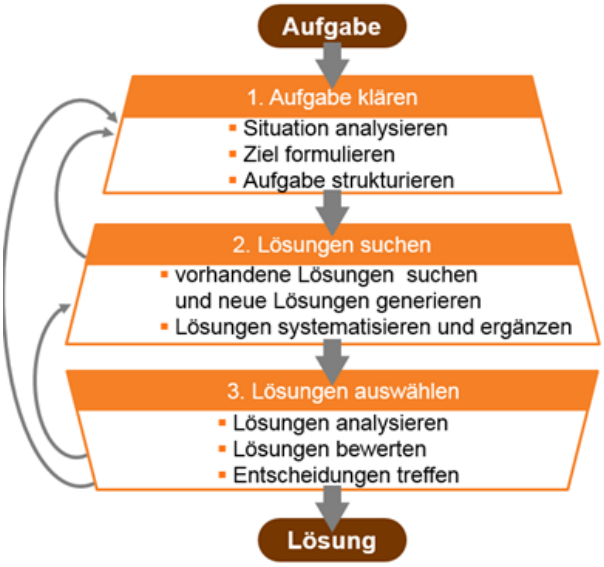


Abbildung 14: Problemlösungszyklus nach Ehrlenspiel (Ehrlenspiel *u. a.*, 2013)

Greift man den System-Gedanken aus dem SE wieder auf, bzw. überträgt dessen Denkweise, handelt es sich bei dem Wechsel zwischen Analyse und Synthese um die Übertragung von Anforderung von einer funktionalen auf eine physische Lösung- vergleiche Mikroprozess wie folgt - Abbildung 15:

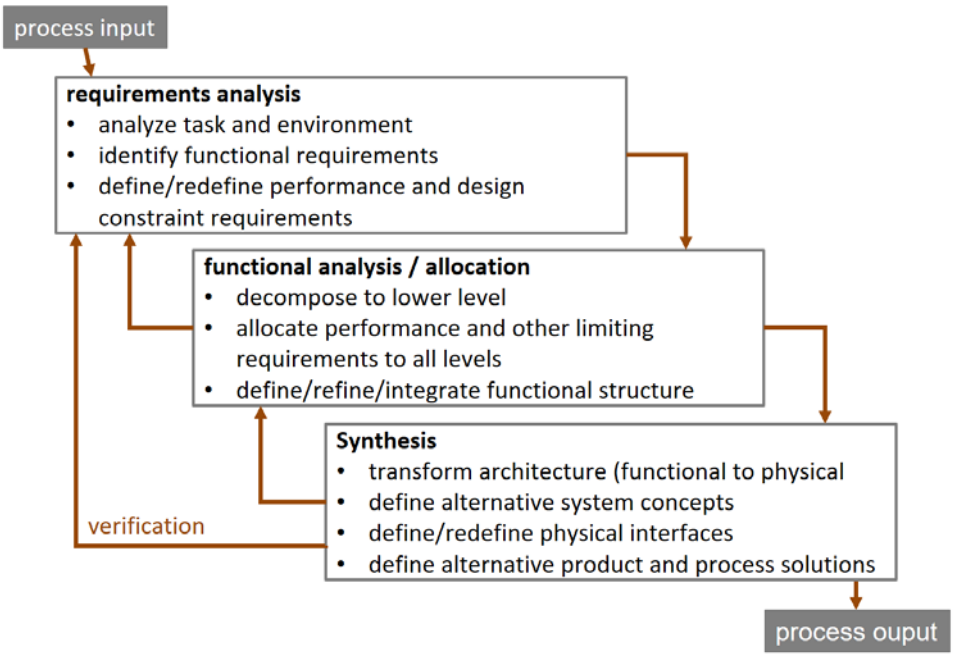


Abbildung 15: Mikroprozess nach Paetzold in Anlehnung an Kossiakoff (Kossiakoff *u. a.*, 2011)

Paetzold greift den Problemlösezyklus nach Kossiakoff auf und differenziert diesen, indem sie die Daten- und Informationsflüsse hervorhebt – vergleiche Abbildung 16:

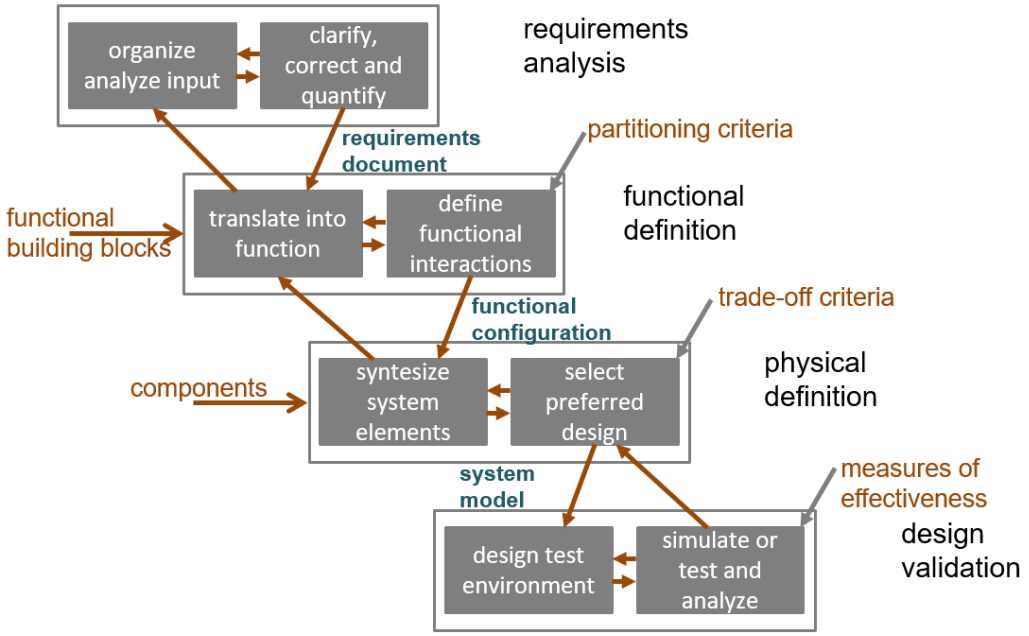


Abbildung 16: Problemlösungs-Zyklus nach Paetzold in Anlehnung an Kossiakoff (Kossiakoff *u. a.*, 2011)

In Summe beinhaltet der Mikroprozess hauptsächlich einen Wechsel von Synthese und Analyse –vergleiche Tabelle 1. Einfach ausgedrückt, wird im „ersten“ Schritt geschaut, was benötigt wird und dafür eine Lösung vorgeschlagen und im Umkehrschluss überprüft, ob die Lösung den Anforderungen entspricht. Diese Erkenntnis liefert wieder Input für eine weitere Synthese, usw.

Tabelle 1: Unterscheidung von Synthese und Analyse

Synthese:	**Analyse:**
- Alle die Tätigkeiten, die den Entwickler zu alternativen Lösungen führen - Kreativer Prozess - Ziel ist es, einen breiten Lösungsraum zu erarbeiten	- Prozess der systematischen Untersuchung von Lösungen hinsichtlich ihrer Eigenschaften - Ziel der Analyse ist die Absicherung der Produktfunktionalität entsprechend der Anforderungen

Somit wird deutlich, dass jede Handlung den DIF formt und entsprechend festlegt. Im Umkehrschluss ist das Zusammenspiel von Synthese und Analyse ein ausschlaggebender Faktor für die Änderung der DIF.

Um aus der operativen Sicht wieder in die strategische zu wechseln, findet unter Anwendung des Systemgedankens ein iterativer Wechsel der Betrachtungsweise statt– vergleiche Abbildung 17:

Abbildung 17: Mikro- und Makrozyklus nach Lindemann (Lindemann, 2009)

Die Verarbeitung von DIF wird durch den Wechsel der Sichtweisen nicht nachvollziehbar. Vielmehr ist es notwendig die beiden Perspektiven zu kombinieren. Genau diese Kombination wird im **integrierten Produktentstehungsmodell** (**iPeM**) nach Albers deutlich.

Im Detail ermöglicht es Akteuren verschiedener Disziplinen zusammenzuarbeiten und gliedert den PEP in drei Hauptbereiche: Dem Zielsystem, welches durch die Anforderungen formuliert wird, dem Handlungssystem, das die Rahmenbedingungen beschreibt und dem Objektsystem, welches durch das zu entwickelnde Produkt beschrieben wird. Dabei beschreibt das Vorgehen „SPALTEN“ den Mikroprozess zu Lösungsfindung mit den Schritten: Situationsanalyse, Problemeingrenzung, alternative Lösungen generieren, Lösung auswählen, Tragweite analysieren, entscheiden und ersetzen, nacharbeiten und lernen. Die Validierung der Anforderungen fällt hier im übergeordneten Sinn unter die Aktivitäten der Produktentstehung, wird jedoch nicht weiter konkretisiert - Abbildung 18.

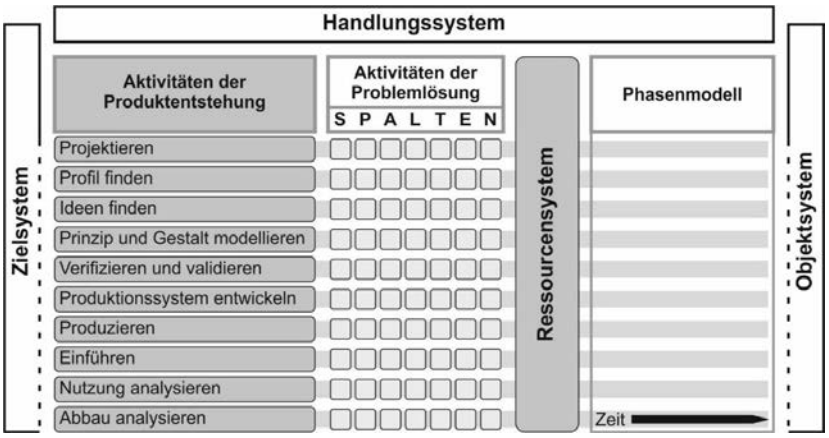


Abbildung 18: Integrierte Produktentstehungsmodell nach Albers (Meyer-schwickerath, 2015)

Damit stellt das iPeM eine Kombination der Makro- und Mikro-Logik dar und konzentriert sich auf die Aktivitäten entlang des PEP und damit den Verlauf der DIF. Albers betrachtete im Spaltenmodell die Aktivitäten als Oberbegriff für alle Aggregationsebenen der Arbeitseinheiten (Albert und Ebel, 2015). Die Aktivitäten orientieren sich dabei am Produktlebenszyklus. Durch die orthogonale Anordnung der Aktivtäten gegenüber dem Ziel- und Objektsystem, entsteht eine „Aktivitätenmatrix“ – vergleiche Abbildung 18, wodurch sich parallele bzw. iterative Aktivitäten erkennen und in Phasen einteilen lassen.

Diese Einteilung der Aktivitäten unterstützt die Möglichkeiten zu analysieren, an welcher Stelle, welche Daten verwendet werden und was daraus folgt. Die Kombination aus Mikro- und Makrosicht in Abhängigkeit von der Produktreife ist im Pyramidenmodell nach Ehrlenspiel gegeben – vergleiche Abbildung 19.

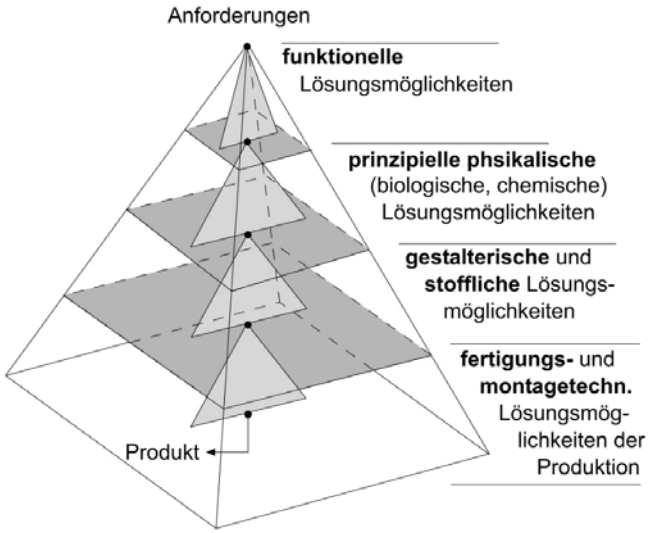


Abbildung 19: Pyramidenmodell der Produktkonkretisierung nach Ehrlenspiel (Ehrlenspiel *u. a.*, 2013)

Zur Beschreibung des zu entwickelnden Systems oder Produkts erlaubt dieses Modell dem Entwickler, das Produkt in verschiedenen Konkretisierungsebenen zu unterteilen. Das Zusammenspiel der Synthese und Analyse kann hier dargestellt werden.

An oberster Stelle stehen die Anforderungen, gefolgt von einer Betrachtung der funktionalen, physikalischen, stofflichen und sogar fertigungs- und montagetechnischen Lösungsmöglichkeiten. Das Model hebt den Einfluss von Anforderungen auf den Fortschritt des PEP und der DIF hervor.

Eine weitere Sicht auf den PEP, die die DIF unter dem Blickwinkel der Aufgaben, Rollen und Entscheidungen fokussiert, ist das **Swimlane-Modell** (Binner, 2017).

Den Ansatz zur Überführung des PEP in Form eines Swimlane-Modells liefert Binner in den 80er Jahren. Er ermöglicht somit die graphische Darstellung der Geschäftsprozesse und setzt die Organisation des Prozesses in den Mittelpunkt. Die Swimlane-Prozessdarstellung beinhaltet Aufgabenstellungen und Arbeitsabläufe in Abhängigkeit von ihrer Rolle im Unternehmen und definiert die Rechte und Pflichten von Mitarbeitern bzw. Teams usw.

Leistung-, Führungs- und Unterstützungsprozesse sind horizontal zeitlich aufgeführt und beschreiben vertikal die Organisation untereinander, wodurch betriebswirtschaftliche Details, wie Funktionen, Ressourcen, Schnittstellen, usw. definiert werden –vergleiche Abbildung 20:

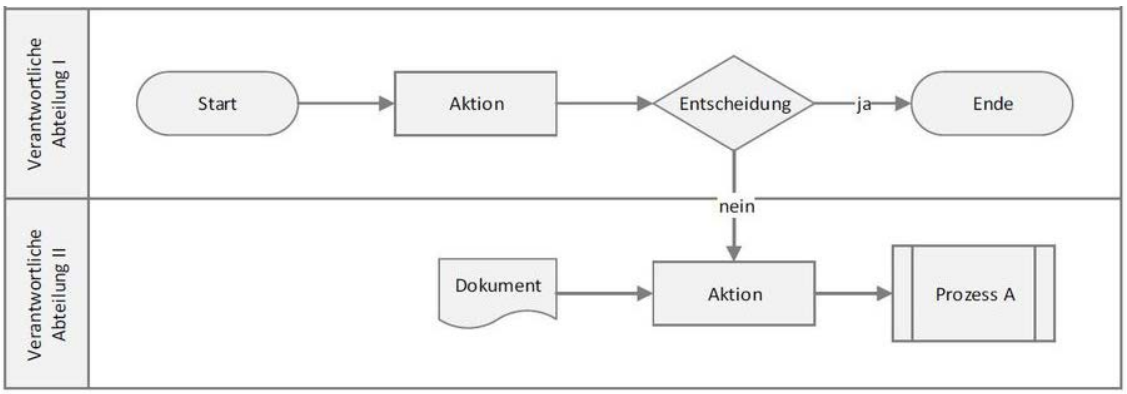


Abbildung 20: Beispiel einer Swimlane-Darstellung

Wie aus der Beschreibung und Anwendung ersichtlich ist, hat das Swimlane-Modell nicht den Anspruch beispielsweise die Planbarkeit des PEP zu konkretisieren, sondern erleichtert die Definition von Schnittstellen und gibt eine bessere Übersicht über die Organisation eines Entwicklungsprozesses auf übergeordneter Ebene. Diese Erkenntnis ist hilfreich, um zu erkennen, welche Rolle, Aufgabe oder Entscheidung die DIF beeinflusst bzw. triggert.

In Summe ist über die Makro-Sicht aus der Mirko-Logik zu folgern, dass in den Iterationen zwischen Analyse und Synthese die DIF –abhängig von Ebene und Kategorie (funktional, physikalisch, etc.) – als Anforderungen eingehen und mit voranschreitendem Produktreifegrad in das gewünschte Produkt überführt werden. Im Umkehrschluss wird das Produkt fortlaufend in Bezug auf die gestellten Anforderungen überprüft. Das heißt, der Absicherungsprozess bietet sich als gute Möglichkeit an, um die DIF zu verfolgen. Er ermöglicht es nachzuvollziehen, wie die D&I verarbeitet werden und in welchen Schritten sie den PEP beeinflussen.

Dieser Abgleich der Anforderungen gegenüber dem Produkt wird im CP-Modell (Characteristics-Properties Modelling) nach Weber (Weber, 2012) in dem **Property-Driven Development/Design (PDD)** ebenfalls angesprochen. Weber liefert ein Systemmodell, das die Anforderungen und die resultierende Merkmalsausprägung gegeneinander aufstellt und zudem die Synthese und Analyse wiedergibt. Diese Erweiterung der genannten Ansätze beschreibt das entwickelte System detaillierter. Das Produkt wird dabei in Merkmale und Eigenschaften unterschieden.

Diese Unterscheidung in Merkmale (Characterstics: C) und Eigenschaften (Properties: P) erlaubt eine Ursache-Wirkungsanalyse des Designs, da C vom Entwickler festgelegt wird und P daraus resultiert. C entspricht zum Beispiel Maße Zylinder und P beschreibt das resultierende Volumen.

Die Relation, welche zwischen C (bzw. C_i für i aus $\mathbb{N}$) und P (bzw. P_i für i aus $\mathbb{N}$) herrscht wird durch die Relation R (bzw. R_i für i aus $\mathbb{N}$) konkretisiert. Sind noch weitere äußere Einflüsse bekannt, können diese mit EC (bzw. EC_i für i aus $\mathbb{N}$) berücksichtig werden – vergleiche Abbildung 21:

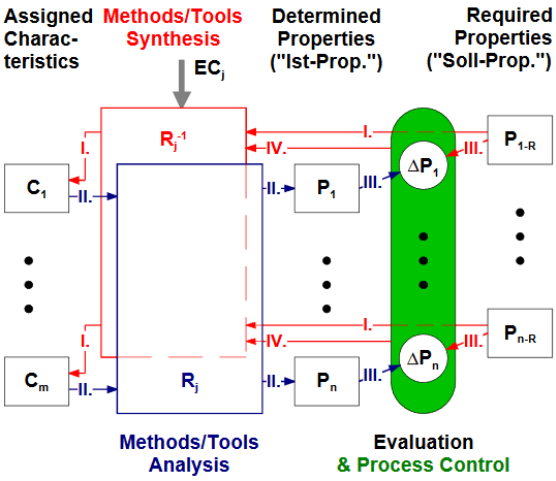


Abbildung 21: Auszug aus dem Weber-Modell - hier die Evaluation (Weber und Deubel, 2003)

Wie aus Abbildung 21 ersichtlich ist, können die tatsächlich erreichten Eigenschaften den gewollten Eigenschaften („Required Properties“) gegenübergestellt werden. Somit lässt sich ein Delta (Δ P) zwischen den Soll- und Ist-Eigenschaften messen, das den Entwicklern hilft zu erkennen, ob die gewünschten Eigenschaften erreicht wurden oder nicht und wie hoch der Unterschied ist.

Ein weiterer Vorteil an diesem Modell ist die Integration des Mikroprozesses, der Analyse (Festlegen von Merkmalen zum Beispiel Durchmesser einer Welle) und Synthese (Überprüfung der Anforderungen anhand der resultierenden Eigenschaften, zum Beispiel Festigkeit der Welle). Das ermittelte Delta legt den Handlungsbedarf fest. (Wie hoch ist der errechnete Widerstandsmoment und ist es höher als das geforderte Torsionsmoment?) –vergleiche Abbildung 22:

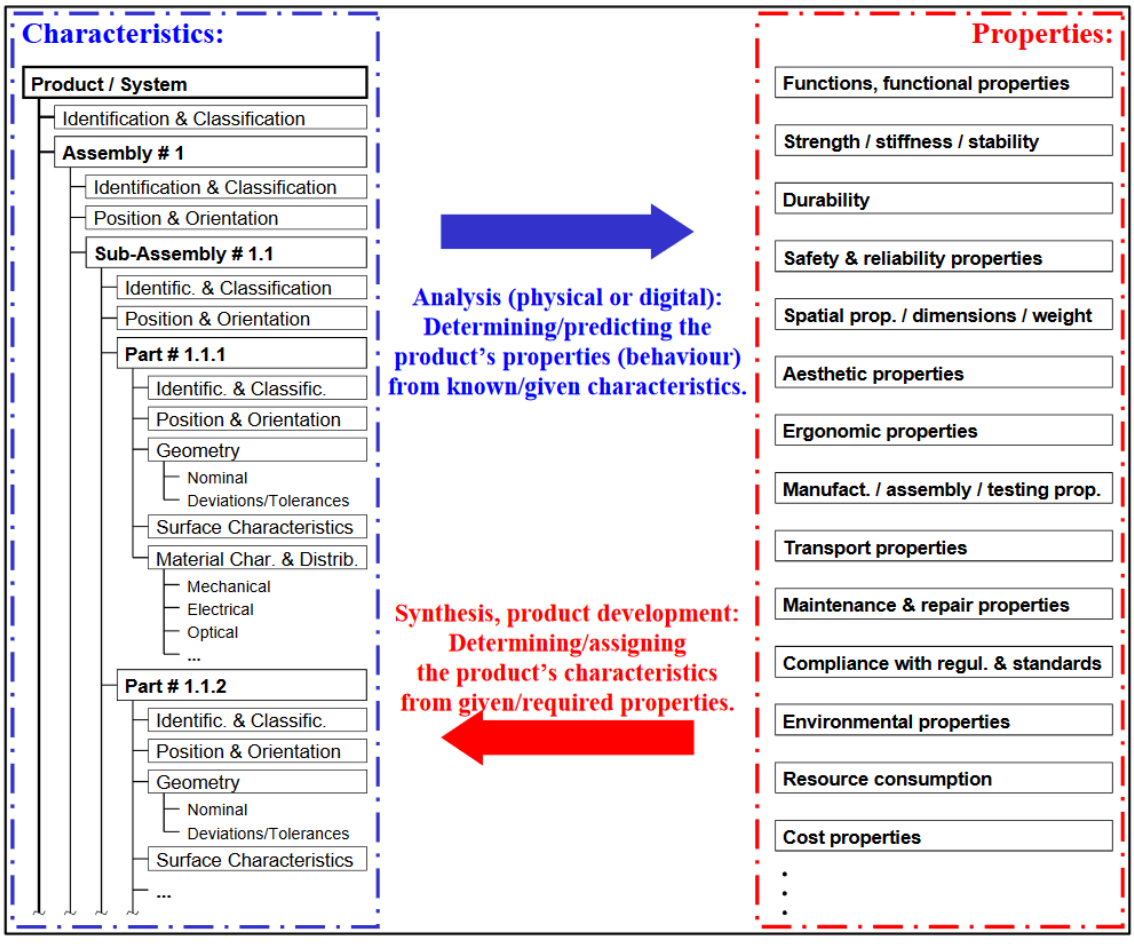


Abbildung 22: Unterscheidung von Analyse und Synthese im Weber-Modell (Weber, 2012)

Zusammengefasst liefert das Weber-Modell die Möglichkeit, das entstandene Delta zwischen der aktuellen Entwicklungssituation und den Systemanforderungen zu identifizieren und zu messen. Diese Differenzierung der Daten- und Informationsflüsse hilft die

Abhängigkeiten besser zu identifizieren und hier speziell den Trigger für weitere Entwicklungsschritte und bereits Erreichtes zu erkennen.

Weber stellt hier fest, dass das entstandene Delta den tatsächlichen Impuls für den PEP liefert (Weber und Deubel, 2003), was im weiteren Verlauf dieser Arbeit eine bedeutende Rolle spielen wird. Allerdings reicht auch diese Betrachtung allein nicht aus, um die Abhängigkeiten von DIF sehr detailliert in weitere Faktoren zu zerlegen. Es fehlt darüber hinaus das Verständnis von der Überführung der initialen Anforderungen (Kundenanforderungen) in die spezifischen Produktanforderungen (Durchmesser einer Welle), den daraus resultierenden Abhängigkeiten der Anforderungen untereinander, der Abhängigkeiten von Produkteigenschaften untereinander – wie zum Beispiel der Relation von Gewicht und Durchmesser der Welle, der Abhängigkeit zwischen Mikroprozess und dem ausführenden Entwickler bzw. der Rolle, usw.

Zwischenfazit:

Mit dem Input, dass die Komplexität in der heutigen PE weiter steigt und die Handhabbarkeit nicht mehr mit den aktuellen Mitteln zu bewältigen ist, werden die DIF in einem PEP nicht mehr sichtbar. Die betrachteten Methoden der Literatur geben darüber Aufschluss, dass der systemische Ansatz versucht die Komplexität auf ein bewältigbares Niveau zu reduzieren. Dies wird durch die Differenzierung in ein technisch physikalisches System und die übergreifenden Managementprozesse des PEP des Systems erreicht. Dabei haben die Betrachtungen von Risiko- und Qualitätsmanagement bzw. der Eigenschaftsabsicherung jeweils eine spezielle Blickrichtung auf die DIF, die hilft, die Komplexität ebenfalls zu reduzieren. Die Erkenntnisse aus der Mako- bzw. Miko-Analyse führen zum Weber-Ansatz, welcher diese kombiniert und – ähnlich wie aus der Eigenschaftsabsicherung – die Anforderungen bzw. das Delta zwischen Anforderungen und erreichter Produktreife als Treiber für den PEP identifiziert, an dem die Verarbeitung der D&I nachverfolgt werden kann. Da jedoch die gezeigten Methoden zu generisch sind, um auf die eigene Situation angewendet zu werden und nicht miteinander verknüpft sind, müssen weitere Betrachtungen herangezogen werden.

Deshalb wird im Weiteren analysiert, welche Rolle Anforderungen im PEP spielen, wie diese verarbeitet und nachverfolgt werden.

2.2 Anforderungsmanagement zur Komplexitätsbewältigung

In diesem Abschnitt wird dargelegt, wie genau das Anforderungsmanagement dabei helfen kann, die DIF aufzudecken. Die Relevanz des Anforderungsmanagements bei der

Komplexitätsbewältigung wird in der Arbeit von Brauns (Christoph Brauns, 2016) verdeutlicht. Demnach ist die Hauptaufgabe des Anforderungsmanagements, die PE während des gesamten Produktlebenszyklus zu begleiten – angefangen bei der Erfassung der Anforderung, über die stetige Absicherung der geforderten Eigenschaften, bis hin zur Abnahme des Produktes. Dabei ist das Anforderungsmanagement kein alleinstehender Managementprozess. Vielmehr können aus diesem Blickwinkel wichtige Schnittstellen zu den bereits erläuterten Management-Prozessen der PEP ganzheitlich erfasst werden (Boehm, 1984). Aus diesem Grund eignet es sich sehr gut, um die DIF entlang des PEP zu verfolgen und aufzudecken. Dies kann zur Beherrschung der Komplexität beitragen.

2.2.1 Definition des Requirements-Engineerings und -Managements

In Folgendem wird das Anforderungsmanagement analysiert und differenziert, um zu erkennen, welche Aspekte bei der Komplexitätsbewältigung eine Rolle spielen.

Im Deutschen spricht man übergreifend von Anforderungsmanagement (Christoph Brauns, 2016), allerdings findet man im Englischen eine Differenzierung dieses Begriffes in Anforderungsmanagement und Anforderungs-„Engineering“. Letzteres lässt sich nicht eindeutig ins Deutsche übersetzen, weshalb im weiteren Verlauf von Requirements Engineering (RE) und Requirements Management (RM) gesprochen wird.

Zusammenfassend lässt sich das Anforderungsmanagement unter Nutzung der englischen Bezeichnungen RE/ RM wie folgt definieren:

> *„Requirements Engineering umfasst sämtliche Tätigkeiten, die erforderlich sind, um (Produkt- und Projekt-) Anforderungen zu erheben, zu analysieren, zu verstehen und zu dokumentieren. Schließlich sind auch Tätigkeiten zur Auflösung von Unstimmigkeiten, zur Verifikation und zur Validierung von Anforderungen (z. B. Anforderungsreviews) Teil des Requirements Engineerings“* (S.9 (Valentini *u. a.*, 2013))

Requirements Management umfasst somit unter anderem alle Tätigkeiten, um

> *„die verwalteten Anforderungen allen anderen Disziplinen der Projektdurchführung und allen Stakeholdern zur Verfügung zu stellen und an diese zu kommunizieren, gegebenenfalls auch zielgruppenspezifisch aufbereitet,*
>
> *Änderungs- und Konfigurationsverwaltung für Anforderungen durchzuführen, z. B. durch Versionsverwaltung und Vorabschätzung der Einflüsse von Anforderungsänderungen,*

> *die Anforderungsentwicklung anhand von Anforderungsattributen […] oder Listen offener Punkte zu verfolgen und zu steuern,*
>
> *die Beziehungen zwischen den Anforderungen sowie zwischen Anforderungen und anderen Projektergebnissen u. a. zum Zwecke der Rückverfolgbarkeit zu pflegen"* (S.9f (Valentini u. a., 2013))

Nach Grande sorgen Anforderungen dafür, dass der Kunde das Produkt bekommt, das er sich wirklich wünscht und benötigt. Anforderungen bilden ebenfalls die Basis für Tests und sind die Basis für alle weiteren Aktivitäten und Entwicklungsschritte (Grande und Grande, 2011).

Nach Grande und Hood umfassen die Tätigkeiten des AM unter anderem folgende Aspekte (Hood *u. a.*, 2008):

- Prüfen und Abstimmen von Anforderungen
- Validieren von Anforderungen
- Eigenschaftsabsicherung

Das Handbuch IT-Projektmanagement definiert RE als (Tiemeyer, 2013) einen kooperativen, iterativen inkrementellen Prozess, dessen Ziel es ist zu gewährleisten, dass:

1. Alle relevanten Anforderungen bekannt und in dem erforderlichen Detaillierungsgrad verstanden sind
2. Die involvierenden Stakeholder eine ausreichende Übereinstimmung über die bekannten Anforderungen erzielen
3. Alle Anforderungen konform zu den Dokumentationsvorschriften dokumentiert bzw. konform zu den Spezifikationsvorschriften spezifiziert sind.

Anforderungen werden weiter in „Funktionale-" und „Nicht-Funktionale Anforderungen" unterschieden, wobei Qualitätsanforderungen und Randbedingungen unter die Nicht-funktionalen fallen. Für alle gilt, dass es eine „zu erfüllenden Eigenschaft oder zu erbringende Leistung eines Produktes" ist (Rupp *u. a.*, 2014). Das heißt das gewünschte Produkt muss „ein fachliches oder technisches Leistungsmerkmal" aufweisen (Glinz, 2007).

Diese Einteilung ist für die eingangs geschilderten und vom Kunden geforderten Funktionen hilfreich, weil sie einen direkten Bezug herstellt.

Im Detail definiert Rupp Funktionale Anforderung als:

> *„eine Anforderung bezüglich des Ergebnisses eines Verhaltens, das von einer Funktion des Systems (oder einer Komponente einem Service) bereitgestellt werden soll."* (S.17 (Rupp u. a., 2014))

Ergänzt werden Funktionale Anforderungen durch Nicht-Funktionale Anforderungen (NFA), welche unterschiedlich definiert und interpretiert werden. In Summe herrscht ein Konsens darüber, dass alle Anforderungen, welche nicht die oben genannten Eigenschaften haben, als NFA bzw. als Qualitätskriterien verstanden werden (Pohl und Rupp, 2015). Allerding werden auch Randbedingungen (Constraints) als NFA gesehen (Pohl, 2008). Diese können jedoch funktional oder nicht-funktional ausfallen. Daher kann eine scharfe Abgrenzung nicht eindeutig getroffen werden. Es lässt sich aber festhalten, dass NFA all jene Anforderungen sind, die einen Bezug zu funktionalen Anforderungen haben können, selbst aber nicht als funktional gelten.

Prozessual unterlaufen die Anforderungen verschiedene Phasen im PEP und tracken dabei die PE – genauer in diesen Stufen:

1. Anforderungsermittlung: Aufnahme von Anforderungen
2. Anforderungsdokumentation: Spezifizieren von Anforderungen
3. Anforderungsvalidierung: V&V, Testing, Korrigieren von Anforderungen
4. Anforderungsverwaltung: Planung, Steuerung und Kontrolle von Anforderungen (Management der Anforderungen)

Dabei sind Stufe 1. Bis 3. als eine Schleife zu sehen. Wobei 4. einen ständigen Prozess darstellt (Pohl, 2008).

Das heißt im Kontext dieser Arbeit sind Anforderungen ein guter Indikator, um den DIF zu verfolgen. Auch hilft die Unterscheidung der Arten von Anforderungen, diese danach zu differenzieren. Um hier die D&I noch weiter auszudetaillieren, müssen die Phasen des Requirements-Engineerings weiter vertieft werden.

2.2.2 Erfassungsmethoden für Anforderungen

2.2.2.1 Anforderungen erfassen

Eine gute Übersicht über die Quelle der Anforderungen ist entsprechend (Klaus Pohl, 2011) wie folgt dargestellt – vergleiche Tabelle 2:

Tabelle 2: Definition und mögliche Stakeholder

Definition Stakeholder nach Pohl in Anlehnung an (Hartley und Robertson, 2006; Suzanne Robertson, 2006)
„Ein Stakeholder ist eine Person oder eine Organisation, die ein potenzielles Interesse an dem zukünftigen System hat und somit in der Regel auch Anforderungen an das System stellt.“ (S.56 (Pohl, 2008))

Typische Stakeholder:	Indirekte Stakeholder:
- Kunde/ Nutzer - Wartung - Interne Abteilungen - Gesetzgeber	- Marketing - Unternehmens-Strategie - Qualitäts-Abteilung - Patentamt

Demzufolge gibt es offensichtliche Stakeholder und zudem solche, die nicht auf den ersten Blick zu erfassen sind. In Summe ist alles und jeder, welches bzw. welcher Anforderungen an das Produkt stellt, ein Stakeholder, dessen Input berücksichtigt und entsprechend der Wichtig- und Dringlichkeit zu verarbeiten ist (Glinz und Wieringa, 2007).

Rupp stellt eine gute Übersicht in Form einer Stakeholder-Liste mit Beispielen zusammen, welche dabei helfen kann, die Anforderungen in Bezug zu ihrer Quelle zu stellen und in der Anforderungserfassung vollständig zu sein. Die Vervollständigung der D&I für den PEP erlaubt es, den Verlauf der jeweiligen Anforderung und deren Lösung von der Quelle aus nachzuverfolgen. Auszugsweise einige Stakeholder, die auf den ersten Blick nicht immer im Fokus stehen, jedoch für die PE entscheidend sind und damit einen Einfluss auf die DIF haben (Rupp *u. a.*, 2014):

- Marketing- und Vertriebsabteilung
    - Marketing und Vertrieb repräsentieren den Kunden, durch die Ziele und gesetzten Ziele des Unternehmens zum Beispiel gegenüber von Konkurrenten.
- Safety und Security
    - Safety legt fest, welche Sicherheitsanforderungen an das Produkt für die Nutzung gestellt werden, um zum Beispiel den Anwender physisch nicht zu schaden
    - Security stellt Anforderungen an das Produkt, um es von ungewollter Fremdeinwirkung zu schützen zum Beispiel vor ein mechatronisches System vor Hackern
- Standards
    - Externe oder firmeninterne Standards wie das Projektmanagementhandbuch, die gewählte Arbeitsmethode, die Corporate Identity oder Richtlinien, die einen Einfluss auf das Produkt haben
- Controlling
    - Interne Finanzabteilungen, die finanzielle Rahmenbedingungen des Projekts oder des Produkts bestimmen

Allgemein gilt, dass erst im Zuge der Entwicklung eines Produktes Anforderungen durch die Festlegung von Lösungen bzw. Teillösungen voneinander abhängig werden. Der Fokus geht fließend von den Anforderungen in die festgelegte Produktarchitektur, bis hin zum fertigen Produkt über. Dabei muss der Entwickler in Abhängigkeit von der Lösung abwägen, welche der Anforderung zu Lasten weiterer Anforderungen erfüllt wird. Das heißt im Umkehrschluss, dass gewisse Anforderungen besser als andere erfüllt werden. Dieses Abwägen wird als Trade-Off bezeichnet.

Ein bekanntes Ergebnis eines solchen Trade-Offs sind die Handtrockner, welche oft in öffentlichen Einrichtungen für das Händetrocknen in Waschräumen zu finden sind. Hier hat der Entwickler nur eine beschränkte elektrische Leistung zur Verfügung. Es gibt nun die Möglichkeit die Hände des Kunden durch heiße Luft oder Gebläse zu trocknen. Geräte der Vergangenheit teilten die Energie auf beide Aspekte auf. Doch neue Handtrockner setzen lediglich auf Gebläse und nutzen keine Heizung mehr. Dieses „Abwägen“ wird mit voranschreitender Entwicklung immer präsenter, schränkt den Handlungsraum ein und stellt eine Abhängigkeit zwischen den Anforderungen aufgrund der gewählten Lösung her (Friedman, 1968).

In jeder Entwicklung existieren zu Beginn unabhängige Anforderungen, welche im Laufe des voranschreitenden PEP in Lösungen übergehen - hier als funktionale oder technische Spezifikationen definiert. Abbildung 23 verdeutlicht diesen Vorgang und zeigt, dass die Anzahl der Anforderungen reduziert wird, die Menge an Spezifikation entsprechend größer wird und die Anforderungen durch Trade-Offs zueinander in Abhängigkeit gestellt werden.

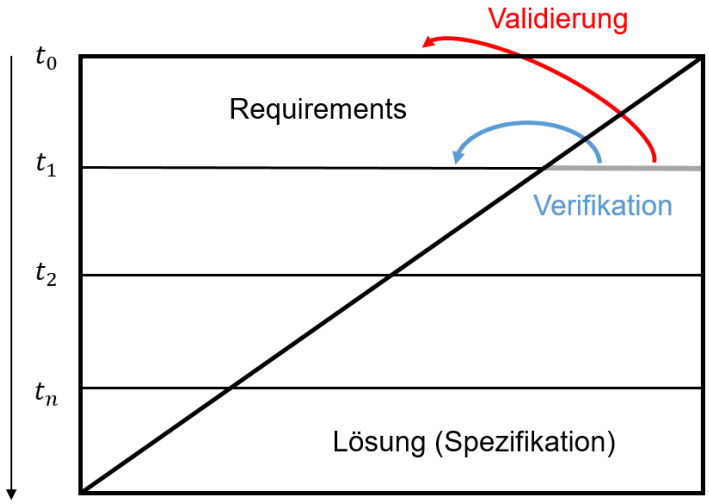


Abbildung 23: Übergang von Anforderungen zur Lösung (Spezifikation)

Das heißt während bei t_0 die Anforderungen noch unabhängig voneinander erfasst sind, ist bei t_1 ein Teil der Lösung spezifiziert. Damit wird der Lösungsraum eingeschränkt

und der Entwickler muss abwägen, welche Anforderungen nun zu Lasten von anderen Anforderungen bevorzugt werden – vergleiche Trade-Off des Handtrockner-Beispiels. Umgekehrt kann nun abgeglichen werden, ob die aktuelle Lösung zum Zeitpunkt t_1 den gestellten Anforderungen entspricht – vergleiche Verifikation. Stellt man die endgültige Lösung den Anforderungen bei t_0 gegenüber, kann eine Validierung stattfinden.

Da wir hier von einer Anpassungs- bzw. Variationsentwicklung ausgehen, sind die Anforderung auch in der grobsten Form voneinander abhängig. Nur in einer kompletten Neuentwicklung können Anforderungen ohne das Vorhandensein einer Lösung – auch wenn dieses nicht dokumentiert ist – unabhängig sein.

Eine weitere und sehr wichtige Maßnahme ist die Prüfung, ob sich Anforderungen grundsätzlich widersprechen. Dabei geht es nicht um Anforderungen, welche sich aufgrund der Lösung ausschließen. Aufgrund von Physik, Logik, etc. kollidiert beispielsweise in der heutigen Fahrzeugentwicklung die Anforderung, dass das Auto schneller sein muss, mit der Anforderung des Leichtbaus, da ein stärkeres Auto – aufgrund der heutigen Möglichkeiten – immer eine Zunahme des Gewichtes (größerer Motor usw.) nach sich zieht.

Diese Anforderungen gilt es zu identifizieren, da sie auf den PEP und folglich auf die DIF bereits in frühen Phasen der Entwicklung Einfluss haben. Werden diese Gegensätze zu spät entdeckt, hat dies folgenschwere und damit kostspielige Folgen –vergleiche Abbildung 24:

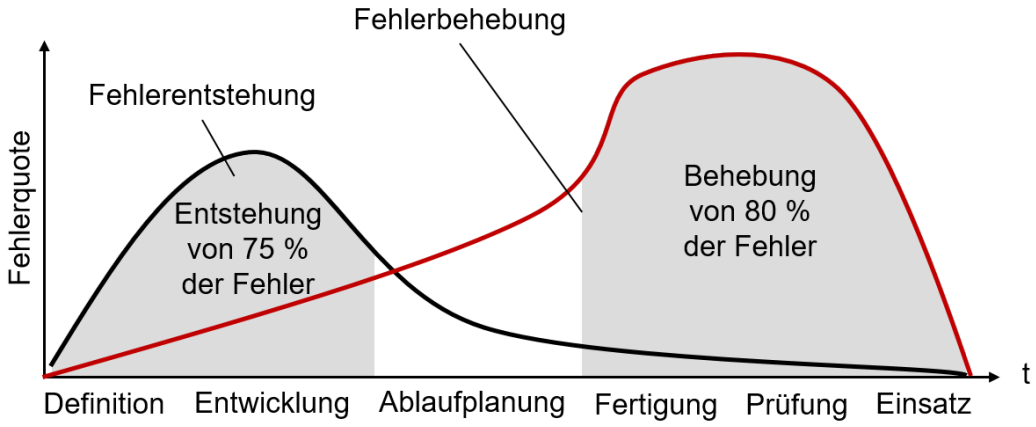


Abbildung 24: Fehlerentstehung und –Behebung nach Schwankl (Schwankl, 2002)

Um diese Gegensätze zu erkennen, eigenen sich die Methoden der Anforderungserfassung, welche sich, mit Fokus auf die Identifikation von Widersprüchen, genauso gut durchführen lassen.

Aus den gängigen Methoden der Anforderungserfassung, eignen sich folgende, um die Widersprüche in den Anforderungen und damit folglich in den DIF zu erkennen (Klaus Pohl, 2011):

- Perspektivenbasiertes Lesen, da der Entwickler jeweils die Anforderungen aus einer anderen „Brille“ liest und Widersprüche dabei erkennt
- Die Sichtung von existierenden Dokumenten, weil diese aus der Historie Hinweise geben, welche Widersprüche bereits bekannt sind
- Fragetechniken, um mit den Stakeholdern eigene und fremde Anforderungen zu diskutieren und so ebenfalls Gegensätze aufzudecken.

2.2.2.2 Anforderungen dokumentieren

Ähnlich wie bei der Anforderungserhebung hat die Dokumentation der Anforderungen großen Einfluss auf die D&I. Denn je nachdem, wie „gut“ die Information der gewünschten Anforderung, deren Quelle, dessen Dringlichkeit, etc. dokumentiert wurde, sind die D&I direkt davon abhängig. Die Güte der Information kann im Nachhinein dabei nicht höher werden als eingangs dokumentiert. Deshalb ist es bedeutsam, die „richtige“ Information in der entsprechenden Form zu dokumentieren (IEEE, 1990). Es wird aber hier vorausgesetzt, dass die Dokumentation in digitaler und nicht in physischer Form vorliegt, da dies heutzutage zeitgemäß ist.

Grundsätzlich gibt es dazu verschiedene Möglichkeiten (Pohl und Rupp, 2015):

- Natürlich-sprachlich
    - Mit Anforderungsschablone
    - Ohne Schablone und frei formuliert
- Modellbasiert
    - Strukturmodelle
    - Verhaltens- und Funktionsmodelle
- Kombination aus natürlich-sprachlich und modellbasiert
    - Use-Case-Diagramme
    - Zustands-Diagramme

Allgemein ist die Erfassung von Anforderungen in einer freien Sprache die einfachste Form. Das heißt die Erfassung und Dokumentation findet in frei formulierten Sätzen ohne eine inhaltliche und strukturelle Vorgabe statt. Textuelle Dokumente können auf bereits vorhandenen (alten) Dokumenten basieren und mit Texten oder Bildern erweitert werden. Die Alternative liegt in der Nutzung von Modellen. Dabei versteht sich ein Modell als ein vereinfachtes Abbild der Wirklichkeit. Nach Stachowiak besitzt ein Modell folgende Merkmale (Stachowiak, 1973):

- Abbildung: Repräsentation des Originals
- Verkürzung: Reduzierung der Attribute des Originals auf die relevante Menge
- Pragmatismus: Das Modell erfüllt eine Ersetzungsfunktion für ein bestimmtes Subjekt, Zeitintervall oder eine Operation

Vor allem erfüllt ein Modell immer einen bestimmten Zweck. Entsprechend haben bestimmte Modelle einen Fokus und eignen sich entweder zur Darstellung von Strukturen oder von Verhalten. Die Modellelemente können entsprechend wiederverwendet werden, weil sie eine eindeutige ID, Attribute, etc. haben und erlauben dem Anwender Modelle oder -Elemente untereinander eindeutig zu verlinken.

Im Sinne des Systems Engineerings werden Modelle ebenfalls eingesetzt, um bestimmte Aspekte besser darzustellen. Dies führt zu Model-based Systems Engineering (MBSE) (INCOSE, 2007). Der folgende Vergleich aus dem MBSE verdeutlicht den Unterscheid zwischen der natürlich-sprachlichen (dokumentenbasierten) und modellbasierten Dokumentation bzw. Entwicklung –vergleiche Tabelle 3:

Tabelle 3: Vergleich zwischen Dokumenten- und Modellbasierter -Entwicklung

Bezug	**Dokumentenbasiert**	**Modellbasiert**
Darstellung	Text und Bilder	Verlinkte Modellelemente (eindeutig durch ID, Attribute, etc.)
Basiert auf	Vorangegangene Dokumentation	Abgeleitete Modelle höherer Ebenen, Hierarchie
Architektur	Unabhängige Bilder und Dokumente	Konsistente, erweiterbare Modelle
Änderungen	Schwer abschätzbar	Durch Verknüpfung nachvollziehbar
Abbildung	Statisch	Dynamisch

Zusammenfassend lässt sich schlussfolgern, dass die sprachliche bzw. textuelle Dokumentation im direkten Vergleich zur Verwendung von Modellen leichter in der Anwendung ist, weil sie nicht explizit erlernt werden muss, jedoch wesentlich mehr Nachteile zur modellbasierten PE aufweist. Allerdings darf nicht vernachlässigt werden, dass bei der Nutzung von Modellen ein initialer Mehraufwand entsteht, da die Modellsprache zu erlernen ist. Dies lohnt sich jedoch auf lange Sicht.

Eine dritte Alternative empfiehlt die Kombination von Text und Modell, um die Vorteile der jeweiligen Dokumentationsform hervorzuheben und gleichzeitig die Nachteile zu minimieren – vergleich SysMl-Sprache (Weilkiens, 2007).

Eine solche Kombination stellt die Strukturierung der natürlichen Sprache durch Satzbauschablonen dar, weil die Sätze einer bestimmten Logik – ähnlich wie ein Modell –

folgen. Demnach können bestimmte Bausteine und eine Satzbauschablone dem Entwickler helfen, in erster Linie Anforderungen textuell zu dokumentieren und dabei die Variation zu reduzieren. Zeitleich sind Anforderungen nach der Satzbauschablone besser miteinander zu vergleichen, da diese eine weitaus reduziertere Form der natürlich-sprachlichen Dokumentation darstellen (Pohl, 2008).

Rupp & die Sophisten geben für unterschiedliche Anforderungen Schablonen vor, die für die Formulierung von Anforderungen als Grundlage nützlich sein können. Sie sind zwar hauptsächlich für die Softwareentwicklung ausgelegt, können jedoch auch als Ausgangspunkt hergenommen und entsprechend der eigenen Situation angepasst werden.

Im Folgenden ist ein Auszug zu finden, nach dem Rupp abhängig von der Detaillierung und Art der Anforderung eine gesonderte – aber in der Logik ähnliche – Formulierung vorschlägt. Wie bereits angeführt, wird nach Funktionalen – und Nicht-Funktionalen Anforderungen unterschieden.

Exemplarisch wird hier eine Funktionelle Anforderung ohne Bedingung herausgegriffen – vergleiche Abbildung 25:

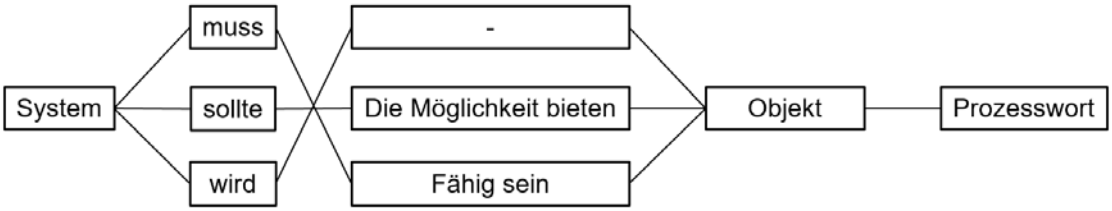


Abbildung 25: Schatzbauschablone für funktionale Anforderungen nach Rupp (Rupp *u. a.*, 2014)

Folgt man dieser Schablone kann man Anforderungen sehr leicht nach deren Verbindlichkeit unterscheiden. Hier unterscheidet man nach der Pflicht, dem Wunsch oder nach der Absicht einer Anforderung. Die deutschsprachigen Adjektive (englisch in Klammern) sind (Team, 2006):

- Muss (Shall)
- Sollte (Should)
- Wird (Will)

An diese Unterscheidung sind **rechtliche Verbindlichkeiten** gebunden. Zum Beispiel sind gesetzliche Anforderungen immer als Muss-Anforderungen zu formulieren, da das Produkt für die Zulassung diese ausnahmslos erfüllen muss. Andererseits führt die Nichterfüllung von Sollte-Anforderungen – zum Beispiel Useability-Anforderungen - in der Regel nicht zu Vertragsstrafen, da diese gut wären, aber nicht zwingend notwendig sind. Für die Validierung ist dementsprechend die Absicherung der jeweiligen Anforderung

entsprechend zu behandeln. Rupp schlägt folgende Hinweise für die drei unterschiedlichen Verbindlichkeiten vor – vergleiche Tabelle 4:

Tabelle 4: Rechtliche Verbindlichkeit von Schlüsselwörtern in Anforderungen (Rupp *u. a.*, 2014)

Rechtliche Verbindlichkeit	**Schlüsselwort**
- Anforderung ist verpflichtend - Erfüllung im Produkt ist verpflichtend - Bei Nichterfüllung droht zum Beispiel die Verweigerung der Zulassung	Muss
- Anforderungen sind nicht verpflichtend - Müssen nicht erfüllt werden - Erhöhen die Akzeptanz des Produktes	Sollte
- Zukünftige Anforderungen sind verpflichtend - Könnten in Zukunft die Integration erleichtern	Wird

Diese Unterscheidung wird für die Auswahl der jeweiligen Validierung im späteren Verlauf eine wichtige Rolle spielen.

Zu den genannten Vorteilen der Satzbauschablone gibt es demgegenüber auch Qualitätsanforderungen an die Anforderungen selbst. Auch hier gibt Rupp einige mit auf den Weg.

Demnach müssen alle Anforderungen gewisse **Qualitätsanforderungen** erfüllen, um Doppeldeutungen, konträre, fehlerhafte oder unvollständige Anforderungen zu vermeiden. Die Qualitätsanforderungen nach Rupp und IEEE sind (IEEE-Computer-Society, 1998; Rupp *u. a.*, 2014):

- Vollständig
- Atomar
- Technisch lösungsneutral
- Konsistent
- Prüfbar
- Notwendig
- Verfolgbar
- Realisierbar
- Eindeutig

Führt man weiter auf, welche Informationen Anforderungen zwecks der Dokumentation liefern müssen, kommen gewisse Attribute hinzu, die notwendig sind, um Anforderungen

zu identifizieren, nachzuverfolgen und zu bearbeiten. Auch hier lassen sich einige Beispiele aufzählen – vergleiche Tabelle 5:

Tabelle 5: Beispiel-Attribute von Anforderungen und Bedeutung (Klaus Pohl, 2011)

Attribut	**Bedeutung**
Autor	Name und evtl. Kontakt des Erstellers (der Ersteller)
Quelle	Quelle der Anforderung nach Stakeholder-Liste
Priorität	Definiert die Reihenfolge der Umsetzung dieser Anforderungen im Bezug zu weiteren Anforderungen

Da Anforderungen im Laufe des PEP verändert und erweitert werden, ist eine Versionierung wichtig und für die Entwicklung von DIF essenziell. Die Relation zwischen Anforderungs- und **Konfigurationsmanagement** (KM) wurde bereits in Abbildung 8 angesprochen und zeigt, dass das KM eine eigene Disziplin darstellt, jedoch einen Einfluss auf Anforderungen hat. Generell umfasst das KM vier Hauptaktivitäten (ISO/BS, 2003):

- Konfigurationsidentifikation: Eindeutigen Identifizierung von Konfigurationseinheiten
- Konfigurationsüberwachung: Kontrolle und Steuerung von Änderungen (Änderungsmanagement)
- Konfigurationsbuchführung: Dokumentation von Änderungen
- Konfigurationsaudit: Überprüfung von Konfigurationen hinsichtlich, ob eine Bezugskonfiguration mit der jeweiligen Produktreife übereinstimmt

All diese Aktivitäten haben einen Einfluss auf die Dokumentation von Anforderungen und folglich auf die DIF. Zuerst ist festzuhalten, dass sowohl Anforderungen als auch die Produktbeschreibung bzw. Spezifikation als Konfigurationseinheiten identifiziert sind, da diese sich ändern können bzw. sich im Verlauf der fortschreitenden Produktreife weiterentwickeln. Das heißt diese Veränderung muss überwacht und kontrolliert werden, da sie einen direkten Bezug zur Änderung der DIF hat. Am Beispiel der Änderungen der Anforderungen kann diese gut beobachtet werden.

Hier kann man zwischen kleinen Änderungen – hier als Inkremente bezeichnet- und Versionen – bei größeren Änderungen – unterscheiden. Diese praxisnahe Unterscheidung ist hilfreich, da es sehr oft vorkommt, dass Anforderungen leicht ergänzt werden, um kleine Missverständnisse zu klären und es nicht sinnvoll ist, eine neue Version der besagten Anforderung zu formulieren. Allerdings sind Änderungen, die weitere Anforderungen

betreffen können, stärker hervorzuheben und entsprechend zu behandeln. In solchen Situationen sollte man zu einer neuen Version greifen – vergleiche Abbildung 26:

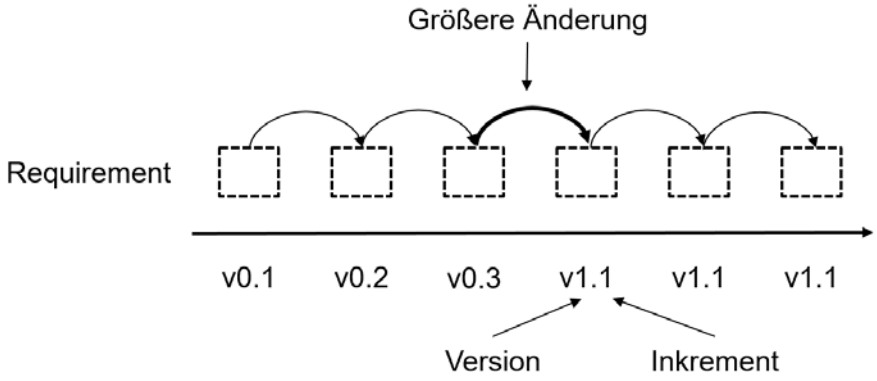


Abbildung 26: Zeitlicher Fortschritt von Anforderungen über Inkremente zu Versionen (Höft und Schaller, 1985; Pohl und Rupp, 2015)

Über die Nummerierung kann eine eindeutige Dokumentation erfolgen. Natürlich kann die Dokumentation auch Namenskonventionen und Ähnliches vorgeben. Für die Dokumentation und Überwachung von Anforderungen und den abhängigen DIF ist die Nummerierung vorerst ausreichend.

Die letzte und ausschlaggebendste Aktivität aus dem KM, nämlich das Konfigurationsaudit, hat einen großen Einfluss auf die Relation zwischen Anforderungen und der Spezifikation und somit den DIF, da nur die „richtige“ Kombination im Verlauf des PEP - zum Beispiel die Grundlage für eine Verifikation bzw. Validierung - herangezogen werden kann (Conradi und Westfechtel, 1998). Das Handling der jeweiligen Relationen erfordert eine Strategie und erhöht unnötig die Komplexität, sollte dies unkoordiniert angegangen werden, da die Anzahl und möglichen Kombinationen von Konfigurationseinheiten zusätzlich untereinander erhöht wird.

Zusammenfassend wurde hier herausgestellt, dass das Anforderungsmanagement einen großen Einfluss auf den Erfolg eines PEP hat und dass dieses bereits für frühe Phasen der Entwicklung entscheidend ist. Es wurde aufgedeckt, wie anhand der Anforderungen nachvollzogen werden kann, welche Entscheidungen, Handlungen und schlussendlich D&I daraus resultieren. Die Sichtung des State of the Art in Bezug auf Requirements-Engineering verdeutlichte, welche Möglichkeiten zur Dokumentation von Anforderungen zur Verfügung stehen und folglich, wie die Information in Bezug auf die DIF festgehalten werden sollten. Dabei wurde der Vergleich von natürlich sprachlicher und modellunterstützter Dokumentation und die Kombination aus beidem analysiert und herausgestellt, welche Dokumentationsart sich am besten für die Analyse von DIF eignet. Die

Relation zum Konfigurationsmanagement hat aufgezeigt, dass die Verwaltung und Dokumentation von Änderungen von Anforderungen und Spezifikationen eine große Rolle im PEP spielen und nur durch eine entsprechende Strategie das Verständnis über die resultierende DIF entstehen kann.

2.2.2.3 Anforderungen nachweisen

Nachdem nun erläutert wurde, welche Rolle Anforderung und deren Dokumentation spielen, ist es notwendig zu verfolgen, wie diese Anforderungen nachgewiesen werden und welche Maßnahmen im PLZ aufgrund des Grades der erreichten Anforderungen getroffen werden. Auch diese Analyse hilft zu erkennen, welche D&I zu welchen Aktionen und Entscheidungen führen und im Umkehrschluss, welche neuen DIF daraus entstehen. Nach Forsteneichner ist die Bereitstellung von Daten und Informationen in Entscheidungssituationen der Schlüssel für effiziente und effektive PEP (Forsteneichner, Paetzold und Metschkoll, 2018). Diese D&I sind folglich bei den Entscheidungen für bzw. gegen Absicherungsmaßnahmen von substanzieller Bedeutung. Denn nur durch die Verifikation bzw. auf Produktebene der Validierung – vergleiche vorheriges Kapitel – können Funktion und Qualität gewährleistet werden. Es gilt demnach die im Laufe des PEP entstehenden Daten und Informationen gezielt zu nutzen, um diese Entscheidung zu unterstützen. Da jedoch die D&I einer ständigen Umwandlung und Transformation unterliegen, ist es zuallererst wichtig, diese zu identifizieren und entsprechende Rückkopplungen zu ermöglichen. Um dies zu realisieren, muss analysiert werden, welche Daten im Rahmen der Absicherung genutzt und übersetzt werden.

Die Verifikations-Matrix koppelt Anforderungen und Absicherungsmaßnahme und hilft somit, die Absicherung zu planen und nächste Schritte vorzubereiten. Diese führt die jeweiligen Anforderungen in Bezug auf Absicherungskriterium, Stadium, Dringlichkeit, Testergebnisse, etc. auf und bietet dem Entwickler so relevante Informationen auf einen Blick – vergleiche Abbildung 27:

Requirement No.[a]	Document[b]	Paragraph[c]	Shall Statement[d]	Verification Success Criteria[e]	Verification Method[f]	Facility or Lab[g]	Phase[h]	Acceptance Requirement?[i]	Preflight Acceptance?[j]	Performing Organization[k]	Results[l]
P-1	xxx	3.2.1.1 Capability: Support Uplinked Data (LDR)	System X shall provide a max. ground-to-station uplink of…	1. System X locks to forward link at the min and max data rate tolerances 2. System X locks to the forward link at the min and max operating frequency tolerances	Test	xxx	5			xxx	TPS xxxx
P-i	xxx	Other paragraphs	Other "shalls" in PTRS	Other criteria	xxx	xxx	xxx			xxx	Memo xxx
S-i or other unique designator	xxxxx (other specs, ICDs, etc.)	Other paragraphs	Other "shalls" in specs, ICDs, etc.	Other criteria	xxx	xxx	xxx			xxx	Report xxx

Abbildung 27: Beispiel einer Verifikations-Matrix (INCOSE, 2007)

Ein weiterer Ansatz die Absicherungsmaßnahmen zu planen und bewerten ist es, mittels der Coordination Theory aufzuführen, welche Aktivitäten (Absicherungsmaßnahmen) in Abhängigkeit von notwendigen „Inputs“ bzw. der Produkt- und Datenreife geplant werden und zu welchem Ergebnis diese führen (Schönwald *u. a.*, 2018).

Diese Theorie verwendet eine formalisierte Sprache bzw. ein Modell aus vier Elementen:

- Aktivitäten
- Ziele
- Actor bzw. Trigger
- Abhängigkeiten

Folglich hat jede Aktivität einen Input, Output und ein definiertes Ziel, wobei die Ziele untereinander hierarchisiert werden können. Mehrere Ziele können auch ein gemeinsames Ziel verfolgen, zum Beispiel ein Verifikationskriterium – vergleiche Abbildung 28:

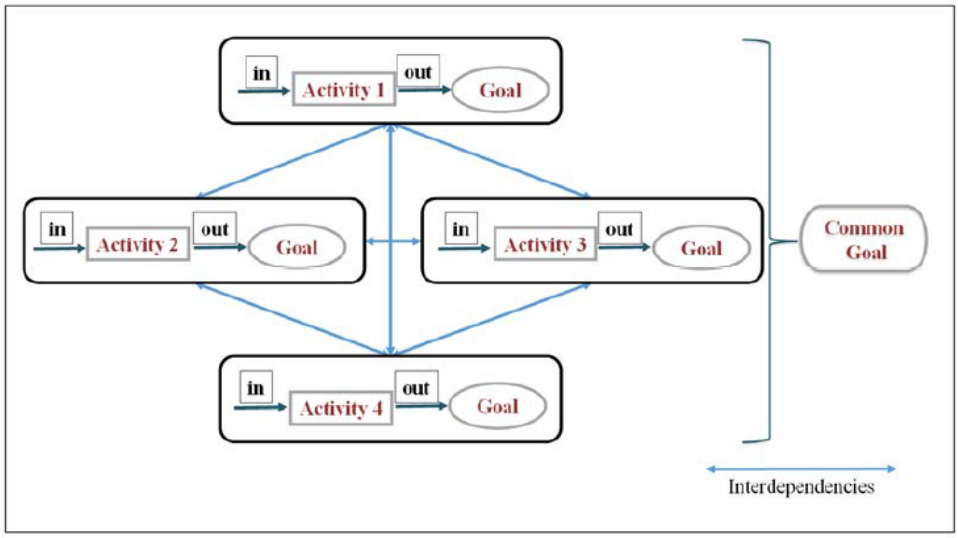


Abbildung 28: Prozess-Modell in Aktivität, In- und Output nach Malone (Malone und Crowston, 1990)

Mit dieser Aufstellung (Modellierung) der Absicherungsmaßnahmen verspricht man sich die Kommunikation, Kooperation und Kollaboration zu verbessern und somit die Verifikation und Validierung effizienter zu planen (Schönwald *u. a.*, 2018). Umgekehrt hilft sie dem Entwickler/ Management die D&I besser zu bewerten und die Entscheidung für oder gegen eine Absicherungsmaßnahme zu treffen. Das heißt die Abhängigkeit der D&I im Hinblick auf die Anforderungen und Absicherungsmaßnahmen werden deutlicher und interpretierbarer.

Parraguez sieht einen ähnlichen Ansatz in der Netzwerk-Theorie. Hier wird die Abhängigkeit von D&I in mathematische Netzwerke übersetzt und die Abhängigkeit von DIF visualisiert (Parraguez und Maier, 2015; Parraguez, Eppinger und Maier, 2015).

Zwischenfazit:

Aus dem Anforderungsmanagement heraus wurde erkannt, dass Absicherungsmaßnahmen der Schlüssel für den Erfolg der PE sind. Zusätzlich kann durch die Beobachtung von Absicherungsmaßnahmen, die in vorangegangen Kapitel zu Verifikation- und Validierungsmaßnahmen erörtert wurden, die Verarbeitung und der Einfluss von DIF fortlaufend beobachtet werden. Um schließlich die DIF zu identifizieren und zu verfolgen, können exemplarisch gezeigte Methoden verwendet werden, welche die D&I formalisiert erfassen und analysieren. Das heißt die Nachverfolgung von Absicherungsmaßnahmen während des PEP hilft die DIF zu erkennen, diese zu bewerten und somit den PEP zu optimieren – hinsichtlich Ressourcen, Zeit, usw.

2.2.3 Traceability im Anforderungsmanagement und in der Produktentwicklung

Das vorige Kapitel hat gezeigt, was mit den Daten und Informationen abhängig vom Fortschritt des PEPs passiert und welche Abhängigkeiten entstehen. Die sogenannte Nachverfolgung bzw. die Traceability kann dabei helfen, die DIF zu verfolgen und bietet die Möglichkeit, Entscheidungen während des PLZ nachzuvollziehen.

Somit ist die Nachverfolgung der Anforderungen erforderlich, um die DIF in der Komplexität der heutigen PEPs zu erfassen und um diese zu analysieren und mit dem Ziel der Optimierung verwenden zu können.

Lindemann verknüpft den Begriff „Traceability“ mit der Rückverfolgbarkeit im RE und definiert sie als die Aufgabe sowohl die Abhängigkeiten zwischen den Anforderungen, als auch die Relation zwischen den Anforderungen und dem Produkt abzubilden. Beide Betrachtungen sollen über den gesamten Produktlebenszyklus hinweg betrieben werden, um eine ganzheitliche Rückverfolgbarkeit zu ermöglichen und um die Abhängigkeiten nachvollziehen zu können (Lindemann und Lindemann, 2016).

Analog zu Lindemann betrachtet auch Rupp die Nachverfolgbarkeit von Anforderungen, geht jedoch in Bezug auf den Lebenszyklus weiter ins Detail und definiert Traceability mit den Worten: „Traceability ist die Fähigkeit, Verbindungen und Abhängigkeiten zwischen Informationen, welche während der Analyse, Entwicklung, Wartung und Weiterentwicklung eines Systems anfallen, jederzeit nachvollziehen zu können.“ (S. 421 (Rupp *u. a.*, 2014))

Einer der ältesten und bekanntesten Definitionen der Traceability im Zusammenhang mit dem AM ist nach Gotel und Finkelstein (Gotel und Finkelstein, 1994) in der Originalsprache: „Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction“

Pinheiro definiert ähnlich zu Gotel und Finkelstein Traceability als: “the ability to define capture, and follow the traces left by requirements“ (S.93 (Pinheiro, 2004)).

Grande bestätigt ebenfalls, dass unter anderem die „Anforderungs-Nachverfolgbarkeits-Matrix“ für die Nachverfolgbarkeit als Mittel genutzt werden kann, um Ziele wie das des gemeinsamen Verständnisses und des Wissenstransfers zu erreichen oder um geplante Zeitrahmen einhalten zu können (Grande und Grande, 2011).

Eine der aktuellsten Definitionen, welche laut Pohl dennoch an Gotel und Finkelstein angelehnt ist, beschreibt Traceability unter dem Begriff der „Anforderungsnachvollziehbarkeit“ wie folgt: „Die Nachvollziehbarkeit einer Anforderung ist die Fähigkeit, den Lebenszyklus der Anforderung vom Ursprung der Anforderung über die verschiedenen Verfeinerungs- und Spezifikationsschritte bis hin zur Berücksichtigung der Anforderung in nachgelagerten Entwicklungsartefakten verfolgen zu können“ (S.503 (Klaus Pohl, 2011)).

Pohl erwähnt verschiedene Disziplinen in der PE, welche durch die Nachverfolgbarkeit unterstützt werden. Dazu gehören unter anderem die Nachweisbarkeit und Akzeptanz, das Änderungsmanagement, das Reengineering, usw. Weitere Autoren, wie Ramesh (Ramesh *u. a.*, 1995) oder Jarke (Ramesh und Jarke, 2001) erwähnen zusätzliche Ausprägungen der Traceability, wie zum Beispiel die Nachvollziehbarkeit zwischen Anforderungsartefakten und nachgelagerten Artefakten im PEP. Fischer, Conklin und Bergemann (Fischer *u. a.*, 1998), (Conklin und Begeman, 1988) setzten die Nachverfolgbarkeit ein, um Entscheidungen, Alternativen und Begründungen für eine bestimmte Wahl zu untermauern.

Partsch (Partsch, 1998) und Pohl (Pohl, 2008) bzw. Gotel und Finkelstein (Gotel und Finkelstein, 1994) differenzieren Traceability noch weiter, indem sie zeitlich, zwischen eine Pre- von einer Post-Traceability unterscheiden. Demnach beinhaltet die Pre-Traceability Artefakte, die im Laufe eines PEP Quelle oder Ursprung einer Anforderung sind. Dagegen beschreiben Artefakte wie zum Beispiel Komponenten, Implementierungen oder Testfälle, die im Projektverlauf zeitlich nachgelagert sind, die Post-Traceability – vergleiche Abbildung 29:

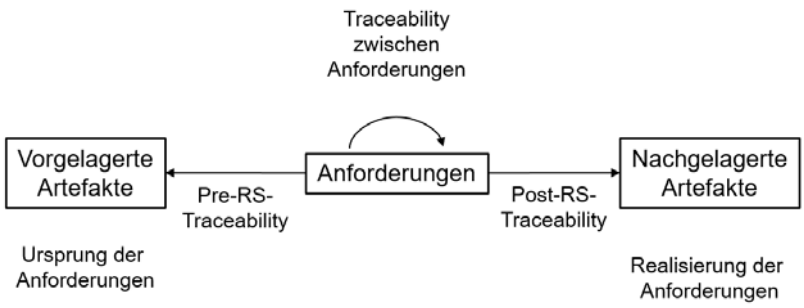


Abbildung 29: Traceability zwischen Anforderungen nach Rupp (Rupp *u. a.*, 2014)

Eine weitere Differenzierung der Traceability ist die Einteilung nach Eben (Eben, Daniilidis und Lindemann, 2010) in eine horizontale- und vertikale Nachverfolgbarkeit. In dieser Einteilung steht nicht der zeitliche Aspekt im Vordergrund, sondern der Detailierungsgrad bzw. die hierarchische Abhängigkeit. Konkret werden Zusammenhänge -sei es zwischen Artefakten und Anforderungen oder nur zwischen Anforderungen - einer Entwicklungsstufe als horizontale Traceability bezeichnet. Eine Abhängigkeit über verschiedene Entwicklungsstufen hinweg wird als vertikale Traceability definiert und hier in Weiterem in diesem Sinne verwendet. Ähnlich verhält es sich in Bezug auf den Vertiefungsgrad. Der Literatur ist zu entnehmen, dass dies grundsätzlich möglich ist. Die gängigsten Quellen verweisen jedoch darauf, dass der Aufwand ein Traceability-Modell zu erstellen dem Nutzen nicht gleichkommt (Pohl, 2008; Rupp *u. a.*, 2014).

Als Vorteile der Traceability werden die Potentiale gesehen, welche durch das bessere Verständnis über die Abhängigkeit der Informationen im AM erst möglich werden. Darunter werden folgende Vorteile erwähnt:

- Nachweisbarkeit von besonders wichtigen Anforderungen – vergleiche sicherheitsrelevante - funktionale Anforderungen nach ISO 26262 (Standard, 2011)
- Auswirkungsanalyse bei Ausfällen von bestimmten Bausteinen und Folgen für die Gesamtfunktion des Produktes
- Wartung und Pflege des Systems bei Anforderungsänderungen

Um jedoch eine detaillierte und nützliche Aussage zu erhalten, empfiehlt Rupp sechs Schritte, welche dem Anwender helfen, ein Verfolgbarkeitsmodell zu erstellen. Diese sind (Rupp *u. a.*, 2014):

Schritt 1: Das Ziel definieren, das das Traceability-Modell erfüllen soll.

Schritt 2: Welche Informationen müssen deshalb miteinander verknüpft werden?

Schritt 3: Welche Bedeutung haben die Traces?

Schritt 4: Definieren der max. und min. möglichen Anzahl an Traces.

Schritt 5: Wie wird die Information der Traces dokumentiert?

Schritt 6: Definieren des Verfolgbarkeitsmodelles im Informationsmodell.

Damit lassen sich die genannten Vorteile der Traceability nutzen, um die DIF sichtbar werden zu lassen.

Zwischenfazit:

Die Betrachtung der Möglichkeiten der Nachverfolgung im Anwendungsfall der Anforderungsabsicherung erweist sich als eine vielversprechende Möglichkeit die DIF zu iden-

tifizieren. Dabei wurde die Relation zwischen Traceability und Anforderungsmanagement hervorgehoben und herausgestellt, welche Ausprägung die Nachverfolgung haben kann und welche für die Identifikation von DIF relevant sind.

2.3 Möglichkeiten für die Abbildungen von Zusammenhängen

Da die Abhängigkeiten zwischen Anforderungen untereinander und zwischen Anforderungen und Spezifikation sehr ausgeprägt und unterschiedlich sein werden, ist es erforderlich die Zusammenhänge abzubilden, um die Traceability nachvollziehen zu können. Ein Modell ist die relativ „banale“ Methode den Fluss und die Verarbeitung von D&I aufzeigen und kann dem Entwickler helfen die Zusammenhänge nachzuvollziehen, wodurch die Komplexität auf ein handhabbares Niveau reduziert werden kann.

Aus dem Requirements-Engineering ist die Verwendung von Modellen ebenfalls bekannt. Es ist nachgewiesen, dass bildhafte Darstellungen schneller erfasst werden und besser in Erinnerung bleiben als Informationen in natürlicher Sprache (Cheng *u. a.*, 1986), (Kosslyn, 1988), (Mietzel, 2017).

Ein weiterer Vorteil von Modellen ist die Reduzierung der komplexen Realität auf ein Minimum. Somit wird nur ein bestimmter Sachverhalt wiedergegeben, welcher analysiert werden soll. Exemplarisch wird hier das bereits erläuterte Systems Engineering bzw. hier unter der Verwendung von Modellen das MBSE erläutert – vergleiche Vor- und Nachteile von Modellen in Kapitel 2.2.2.2.

Das MBSE besteht dabei aus drei Elementen, die es erlauben, Zusammenhänge in Modellen auszudrücken– vergleich Abbildung 30.

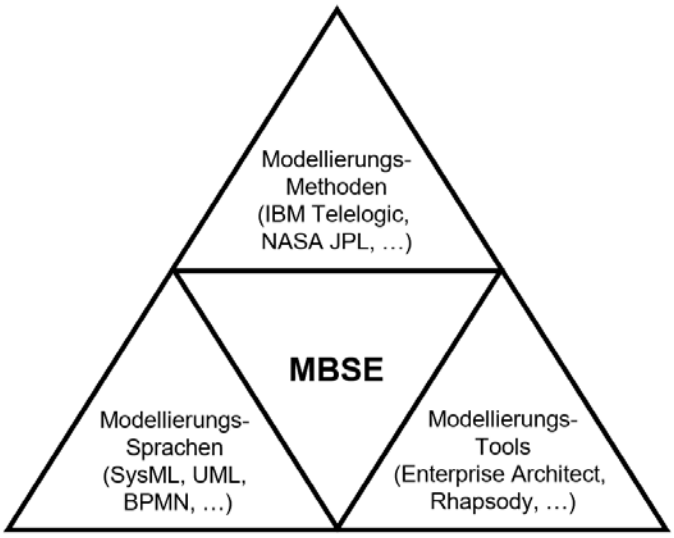


Abbildung 30: MBSE-Dreieck nach Rudolph (Rudolph *u. a.*, 1999)

Dazu bedarf es in erster Linie einer Methode, die in Bezug auf die Problemstellung dieser Arbeit zum Beispiel aus dem Anforderungsmanagement – vergleiche vorherige Kapitel – entnommen werden kann. Aufbauend auf einer Methode bedarf es einer Modellierungssprache. Diese besteht aus einem vordefinierten Regelwerk und dient einem bestimmten Zweck. Übliche Modellierungssprachen aus dem MBSE sind UML, SysMl, BPMN, etc. Da die Modellierung komplexer Zusammenhänge einheitlich und nicht mehr „händisch" in einem vertretbaren Umfang und innerhalb eines akzeptablen Zeitrahmens bewerkstelligt werden kann, geschieht die Modellierung heutzutage mittels eines Tools, das auf einer Methode basiert und dessen Notation einer Modellierungssprache folgt. Beispiele sind die Sprache SysMl MagicDraw, Enterprise Architect, Rhapsody, etc.

Am Beispiel der SysMl-Sprache sieht man, wie Diagramme verwendet werden können, um bestimmte Sachverhalte hinsichtlich der Anforderungen an das Produkt zu verdeutlichen und bestimmte Aspekte der D&I zu repräsentieren. – vergleiche Abbildung 31.

Dazu kann man die möglichen Diagramme in drei Hauptkategorien unterteilen, die jeweils einen Fokus haben – das Verhalten, die Struktur des Produktes und die Anforderungen an das Produkt.

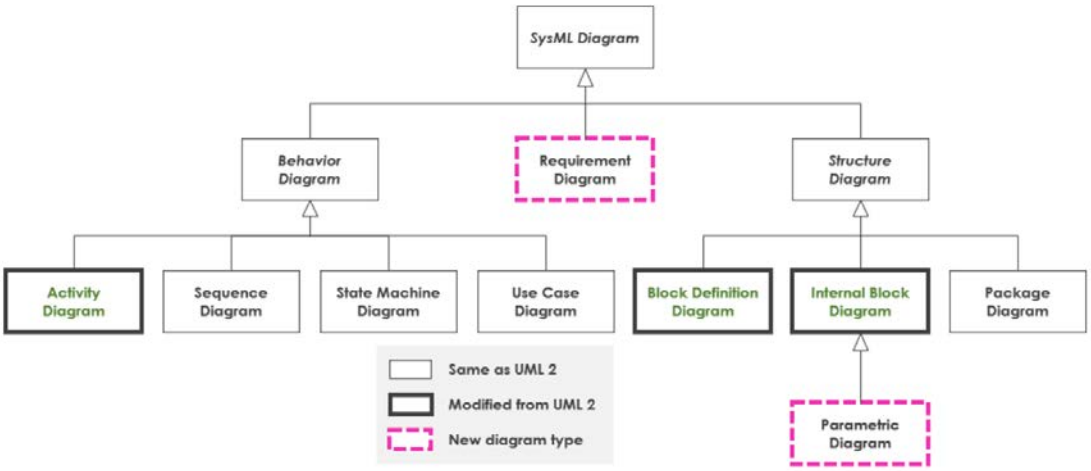


Abbildung 31: SysMl-Diagramme - mit Unterscheidung des Informationsgehaltes nach Friedenthal (Friedenthal, 2009)

Natürlich können die Diagramme, wie nach Paetzold aufgezeigt, für die Darstellung der Produkthierarchie verwendet werden –vergleiche Abbildung 32:

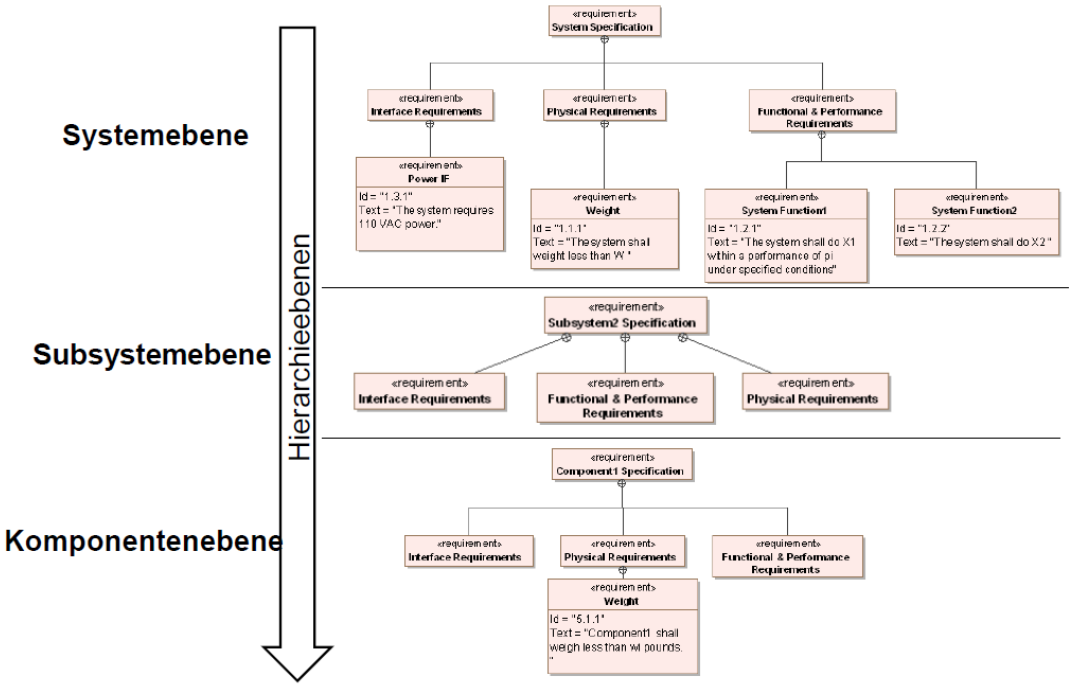


Abbildung 32: SysMl-Daigramm angeordnet nach der Produkt-Hierarchie

Ungeachtet der Tatsache, dass jedes der aufgeführten Modelle eine bestimmte Sichtweise behandelt, ist die Aussagekraft solch einer Darstellung in der Aufnahme von Information begrenzt, wenn sie für den Entwickler noch verständlich bleiben soll. Angelehnt an die Millersche Zahl aus den Geisteswissenschaften, kann der Mensch 7±2 „Aspekte“ gleichzeitig überblicken (Miller, 1956). Diese Zahl findet sich ebenfalls zum Beispiel in der Modellierung von Use Cases und besagt, dass ein Modell max. 7±2 Elemente aufweisen soll, damit es noch verständlich bzw. handhabbar bleibt (Gomaa und Gomaa, 2012). Das heißt, die gezeigten Modelle müssen hierarchisiert werden, um eine größere Informationsfülle aufnehmen zu können. Da es hier jedoch um die ganzheitliche Abbildung von komplexen DIF geht, sind diese Darstellungsformen nur bedingt nützlich. Sie eignen sich für die Erfassung von maximal komplizierten Sachverhalten, aber nicht für die Erfassung und verständliche Darstellung von komplexe DIF.

Zwischenfazit:

Dieses Kapitel konnte zeigen, wie am Beispiel des MBSE D&I dargestellt werden können und abhängig von der gewählten Methode und Sprache ein bestimmter Zweck betrachtet werden kann. Die DIF im PEP können somit über die Prozesse und die Dokumentation relevanter Anforderungen nachverfolgt werden. Allerdings sind die vereinzelten Methoden aus dem SE bzw. dem MBSE, den Sprachen und sogar den Tools nur „Insellösungen“, die ein bestimmten Fokus haben und nicht den DIF ganzheitlich beschreiben. Die Abhängigkeiten bleiben unsichtbar bzw. sind nur vereinzelt erkennbar.

2.4 Möglichkeiten der Analyse von Abhängigkeiten

Die bisher beschriebenen Methoden der Darstellung des PEP bzw. der Relation von Anforderungen und Spezifikation/ Testmethode erlauben die DIF zu verstehen. Verglichen mit der ASPICE-Bewertung stellt das Level 1 dar – vergleiche Abbildung 3. Da jedoch angestrebt wird, die DIF nicht nur darzustellen, sondern zu verstehen, um den PEP zu steuern (ASPICE-Level 2), ist es essenziell die DIF analysieren zu können.

Geht man jedoch noch weiter und möchte die Abhängigkeiten nicht nur identifizieren, sondern auswerten und optimieren (ASPICE-Level 5), muss die Traceability – ähnlich wie die Erkenntnisse aus der der Anwendung der Coordination Theory – auswertbar werden. Das heißt die DIF müssen formalisiert werden, um deren Abhängigkeit entweder optisch oder in Kennzahlen auswertbar zu machen.

Diese Erweiterung der Traceability – sei es ein graphisches Modell oder eine Matrix – ist die Analyse der abgebildeten oder entstandenen Abhängigkeiten. Übersetzt man die Traceability in eine mathematische Umgebung, lässt sich eine Fülle an mathematischen Methoden anwenden, die aus der Graphen- und Netzwerktheorie stammen (Song *u. a.*, 2017). Dadurch wird es nicht nur möglich, die Traceability abzubilden, sondern darüber hinaus komplexe Zusammenhänge mittels Suchalgorithmen zu detektieren und zu verstehen, da die Information in einer neutralen Sprache analysiert werden (Sebastian *u. a.*, 2017; Chahin und Paetzold, 2018).

Zwar wurden bisher keine umfassenden Versuche unternommen, die Möglichkeiten der Graphen- und Netzwerktheorie auf dem Gebiet der Komplexitätsbewältigung auszuschöpfen, aber in anderen Bereichen der PE geschieht dies bereits.

Allgemein bestehen Graphen aus einer Menge von Knoten und Kanten zur Verbindung zweier Knoten, die sowohl als Graph als auch als Matrix darstellbar sind. Die Bedeutungen der Knoten und auch der Kanten ergeben sich aus dem Anwendungskontext. Knoten zum Beispiel können Mitarbeiter symbolisieren, die Aufgaben (als Kanten) im Sinne eines vordefinierten Workflows zu bearbeiten haben. Als Ergebnis können Abhängigkeiten in der Aufgabenbearbeitung dargestellt werden. Über Hierarchiebildung ist nicht nur die Abbildung von Mikro- und Makroebenen möglich, sondern es können auch unterschiedliche Bedeutungen von Knoten und Kanten berücksichtigt werden (Browning, 2001). Die Graphentheorie hinterlegt eine fundierte mathematische Beschreibung sowohl zur Modellierung als auch zur Auswertung dieser, sodass auf Basis der Netzwerke Analysen vorgenommen werden können. Die Herausforderung liegt in der Interpretation, da die Analyseergebnisse im Kontext der Modellbildung zu bewerten sind (Parraguez, Eppinger und Maier, 2015). Abhängig von der Anwendung können Knoten und Kanten eines Teil- oder Gesamtgraphen mit beliebig vielen Informationen belegt werden. Dies erlaubt dem Nutzer, sich nicht auf lediglich eine Informationsart zu beschränken und dennoch bekannte

mathematische Analysen aus der Netzwerktheorie anzuwenden, da die Informationen auf verschiedenen Ebenen geclustert werden und sich somit vergleichen lassen.

Die Graphentheorie findet in der Produktentwicklung seit geraumer Zeit in Form von Design Structure Matrizen (DSM) Anwendung (Eppinger und Browning, 2012). Für die genannte Hierarchisierung werden diese in Multiple Domain Matrizen (MDM) zusammengefasst (Lindemann und Lindemann, 2016), hier helfen die mathematischen Grundlagen, Abhängigkeiten zwischen Teilmatrizen aus den Modellen abzuleiten. Humpert nutzt die Modellierung der Anforderungen als Basis für ein Requirements Engineering, um Rückschlüsse auf die Kritikalität sowie mögliche Zielkonflikte aus Anforderungsabhängigkeiten zu ziehen. Es werden jedoch keine Vorgehensempfehlungen im Sinne von Reaktionsstrategien für betroffene Entwickler entlang der Produktentwicklungsprozesses abgeleitet (Gausemeier *u. a.*, 1995).

Clarkson verwenden Design Structure Matrizen (DSMs), um die Fortplanzungen von Änderungen in komplexen Systemen zu untersuchen (Wynn und Clarkson, 2005). Mittels der Change Prediction Method (CPM) werden Risiken für die Produktentwicklung antizipiert. Hierbei wird zwischen initialen und resultierenden prädiktierten Veränderungen unterschieden (Clarkson, Simons und Eckert, 2004). Die Änderungsfortpflanzung kann neben DSMs auch mittels qualitativer Beziehungen ausgedrückt werden (Purchase, James und Cohen, 1997).

Sinha ergänzt die Netzwerkanalyse indem er die strukturelle Beschaffenheit des Netzwerks um weitere Eigenschaften, wie die Anzahl oder die Schnittstellen der Komponenten, erweitert, um Kennzahlen über das Produkt zu generieren (Sinha *u. a.*, 2015). Die Analyse dieser Zahlen und deren Entwicklung lassen Rückschlüsse auf das Produkt und demzufolge die Absicherungsmaßnahmen zu, welche trotz der vielen Einzelteile und der hohen Anzahl an integrierten Funktionen übersichtlich erscheinen.

In weiteren Anwendungen werden Netzwerke mit unterschiedlichen Bedeutungen erstellt und im Hinblick auf die zu untersuchende Parameter optimiert (Halu *u. a.*, 2016). Zudem gibt es sowohl in der Industrie als auch in wissenschaftlichen Anwendungen ein breites Spektrum an Tools, die es entweder erlauben, eindeutige Netzwerke zu entwickeln, um diese auf ihre Struktur zu analysieren, oder Netzwerke im Sinne der Traceability zu erstellen, um Wirkketten zu erschließen und ihre Abhängigkeiten zu verdeutlichen (Danilovic und Browning, 2007).

Zwischenfazit:

Die Möglichkeiten aus der Netzwerktheorie versprechen die DIF nicht nur zu identifizieren, sondern auch zu analysieren und für Prozessoptimierung nutzen zu können. Erste Anwendungen bei ähnlichen Problemstellungen stellen sich als sehr vorteilhaft heraus

und lassen eine analoge Übertragung auf die Problemstellung zu. Die Möglichkeiten der mathematischen Analysen untermauern zudem die Chancen, die Abhängigkeiten messen und auswerten zu können. Verschiedene Anwendungen der Graphen- und Netzwerktheorie zeigen, dass es möglich ist, komplexe Abhängigkeiten zu formalisieren und im Hinblick auf definierte Faktoren zu optimieren. Demnach ist es theoretisch möglich auch diese Methode für die Problemstellung dieser Arbeit anzuwenden.

3 Herleitung der Forschungsfrage

3.1 Zusammenfassung der Erkenntnisse

Hinsichtlich der anfänglichen Frage, welche Methoden dabei helfen die Komplexität zu reduzieren bzw. diese beherrschbar zu machen, um die DIF ganzheitlich zu beschreiben, zu analysieren und den PEP zu optimieren, wurden folgende Erkenntnisse gewonnen:

Systemischer Ansatz:

- Hilft das Produkt bzw. System durch die Definition des Systems, der Elemente und der Umwelt zu erkennen und verbindet die Prozess- und Produktsicht
- Offen: Wie wird das System definiert und unter welchen Aspekten?

ZOPH-Ansatz:

- Teilt das System in vier Sichtweisen auf, die helfen die Elemente des Systems einzuordnen und ordnet den PEP Entwicklungsrandbedingungen zu, die bei Entscheidungen nützlich sind
- Offen: Wie ist mit Veränderungen der Teilsysteme umzugehen (Dynamik des Systems). Die Abgrenzung der ist nicht exakt möglich

Systems Engineering:

- Die technisch physikalische Betrachtung des Systems hilft das Produkt zu hierarchisieren und zu beschreiben
- Die Betrachtung des Systems in Bezug auf die verschiedenen Management-Prozesse, hilft zu erkennen, was relevant ist, um gewünschte DIF aufzudecken und herauszustellen, wie die D&I beeinflusst werden
- Jede Betrachtung liefert Erkenntnisse über die Verarbeitung von D&I aus einer Sicht:
    - des Risikomanagements
    - der Produkt- und Qualitätssicherung
    - des Absicherungsmanagements
    - des Anforderungsmanagements & -Engineering
    - des Konfigurationsmanagements/ Änderungswesen
- Offen: Eine ganzheitliche und vollständige Betrachtung der DIF ist nicht gegeben, da die Methoden nicht detailliert genug sind, D&I vollständig zu analysieren

Prozessmodelle der methodischen Produktentwicklung

- Makrologik zeigt einige Modelle, die helfen, die D&I entlang des PLZ zu verfolgen
- Mikrologik deckt den Problem-Lösungs-Zyklus auf

- Synthese und Analyse - im Weber-Modell kombiniert - beschreiben aus der Mikrologik das Entstehen des Produktes und dessen Hierarchie und identifizieren die Charakteristik der Daten
- Aus einem Delta zwischen Anforderungen und Lösung kann gefolgert werden, welche Prozesse getriggert werden und somit D&I verfolgt werden
- Offen: Adaption und Kombination der Modelle auf eine sehr detaillierte Ebene, um die DIF ganzheitlich zu erfassen

Anforderungsmanagement

- Anforderungsmanagement stellt sich aus dem Systems Engineering als Schlüsseldisziplin zur Erkennung und Nachverfolgung von DIF heraus, da es den PEP bei der Entstehung (Definition der Anforderungen) über den gesamten PLZ, bis hin zur Validierung des Produktes begleitet
- Die Erfassung und Dokumentation von Anforderungen hat großen Einfluss auf den Informationsgehalt
- Die Anforderungsverifikation, erlaubt es zu erkennen, wie DIF entlang des PLZ mittels Beobachtung der Anforderungen entstehen
- Um DIF anhand der Anforderungen zu erkennen, wird mittels Traceability detailliert
- Traceability deckt auf, welche Methoden dabei helfen die DIF nachzuverfolgen und welche Vorteile entstehen
- Darstellungsarten zeigen verschieden Möglichkeiten DIF zu visualisieren und damit zu erkennen
- Die Graphen- und Netzwerktheorie zeigt, wie die DIF formalisiert und analysiert werden können
- Offen: Wie müssen diese Ansätze verknüpft werden, um DIF zu erkennen und analysieren

In Summe konnte herausgearbeitet werden, welche Aspekte beachtet werden müssen, um die DIF zu erfassen bzw. zu formulieren, welche Sichtweisen theoretisch helfen können das Produkt und den Prozess in der vollständigen Komplexität zu erfassen und welche Möglichkeiten es gibt, die D&I nachzuverfolgen und zu analysieren. Dabei erwies sich aus dem Systems Engineering das Anforderungsmanagement als zielführend, da es die DIF während des gesamten Produktlebenszyklus begleitet. Die Betrachtung des Verlaufes von Anforderung und deren Verifikation im Absicherungsprozess hat Potential die DIF während des gesamten PLZ aufzudecken und zu verfolgen und könnte somit die Verarbeitung und Entstehung von D&I erläutern. Es wurde jedoch kein Ansatz gefunden, welcher alle notwendigen Gesichtspunkte verknüpft, um die DIF zu erfassen, zu analysieren und folglich den PEP zu optimieren.

3.2 Definition der Forschungsfragen

Um diese Situation zu ändern, müssen die vorhanden Methoden abgestimmt, kombiniert und erweitert werden. Ziel ist es die DIF für den einzelnen Entwickler und das Unternehmen sichtbar und verständlich zu machen. Nur dadurch kann die Komplexität auf ein bewältigbares Niveau reduziert werden.

Aus dem Ziel die DIF aufzudecken und diese zu analysieren, um den PEP optimieren zu können und um Fehlentwicklungen zu verhindern, resultiert die Nachfrage nach einem Framework, das es erlaubt, vorhandene Ansätze und Methoden zu verwenden, zu kombinieren und zu erweitern. Ziel ist es, DIF zu verstehen, die Komplexität auf das nötigste zu reduzieren und den PEP gezielt zu steuern. Daraus ergibt sich folgende Forschungslücke bzw. folgende -Fragen:

1. Wie können die Abhängigkeiten zwischen Produkt und Prozess am Beispiel des Absicherungsmanagements sichtbar werden?

2. Wie müssen die Daten und Informationen aufbereitet werden, damit eine Analyse des PEP möglich wird, um den Entwickler in seinen Handlungen und Entscheidungen unterstützen?

3. Wie ist ein solches Framework umzusetzen?
    a. Welche Daten sind dafür notwendig?
    b. Wo sind die Daten üblicherweise in einem Unternehmen zu finden?
    c. Wie könnte eine Implementierung des Frameworks aussehen?
    d. Welche Erkenntnisse sind beispielsweise mit Hilfe des Frameworks möglich?
    e. Wie kann der PEP mittels des Frameworks gesteuert und optimiert werden?

4 Herleitung und Grundlagen des Frameworks

Um die Forschungsfrage zu beantworten, werden die relevanten Ansätze aus der bisherigen Analyse aufgegriffen, verarbeitet und verknüpft. Ziel ist es, ein Framework aufzustellen, das es erlaubt, die DIF aufzunehmen, zu strukturieren und durch geeignet Analysen aufzudecken. Das Framework soll dazu dienen, die DIF zu verstehen, um trotz der komplexen Zusammenhänge Prozessoptimierungen zu erlauben. Dazu werden die Methoden der Graphen- und Netzwerktheorie verwendet, um die Abhängigkeiten in den DIF zu erkennen, auszuwerten und zu deuten.

4.1 Allgemeine Erläuterung zum Aufbau des Frameworks

Das Framework wird vergleichbar zum Dreieck des MBSE entwickelt, welches eine Methode, eine entsprechende Sprache und die Modellierung in einem Tool fordert, – vergleiche Abbildung 33:

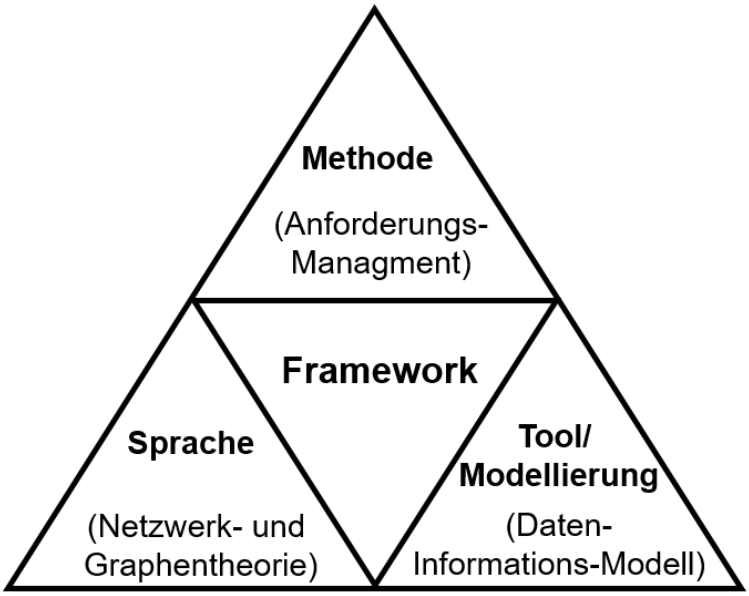


Abbildung 33: Framework in Anlehnung an das MBSE-Dreieck - aufgeteilt in drei Einzelteile

Das Pendant zur Methode stellen die Erkenntnisse aus dem Stand der Technik dar –vergleiche Zusammenfassung in Kapitel 3.1. Daraus lässt sich erkennen, dass das Anforderungsmanagement die Möglichkeit bietet, die DIF entlang des PEP bereits von der Definition der ersten Anforderungen über den gesamten PLZ bis hin zur Produktabnahme zu verfolgen. Da die DIF untereinander in Abhängigkeit stehen und sich nicht linear, sondern netzwerkartig verbinden, kommt die Graphen- und Netzwerktheorie (NWT) als formali-

sierte Sprache zum Einsatz (Chahin und Paetzold, 2015). Erste verwandte Fragestellungen haben ergeben, dass sich die NWT gut eignet, die Abhängigkeiten zu erkennen und zu analysieren. Die Mathematik dient hier als Sprache und ist im MBSE-Dreieck zum Beispiel SysMl. Die Möglichkeiten der mathematischen Netzwerktheorie ermöglichen es, die komplexen Abhängigkeiten der DIF mit Hilfe einer neutralen Sprache zu erkennen und mit Kennzahlen zu untermauern (Chahin, Hoffmeister, *u. a.*, 2016; Chahin *u. a.*, 2017; Salehi *u. a.*, 2017; Schmidt *u. a.*, 2017). Das dritte und letzte Element ist die Durchführung, welches hier das Daten-Informations-Modell darstellt. Dieses erlaubt es, die notwendigen Daten strukturiert aufzunehmen und ermöglicht gleichzeitig mathematische Analysen. Es wird in dieser Arbeit kein bestimmtes Tool vorgestellt oder entwickelt. Es wird jedoch aufgezeigt, welche Möglichkeiten es gibt, das Framework in bestehende Tools zu integrieren.

4.2 Datenstruktur für Erklärung des Frameworks

Um die Verständlichkeit zu erleichtern, wird das zu Beginn eingeleitete Beispiel der Komplexitätssteigerung durch autonome Systeme aufgegriffen und verwendet. Dabei sind die Beispiele generisch gehalten, um lösungsneutral zu bleiben. Gemäß den bisherigen Erkenntnissen werden die DIF von funktionalen Anforderungen abgeleitet – vergleiche Abbildung 34:

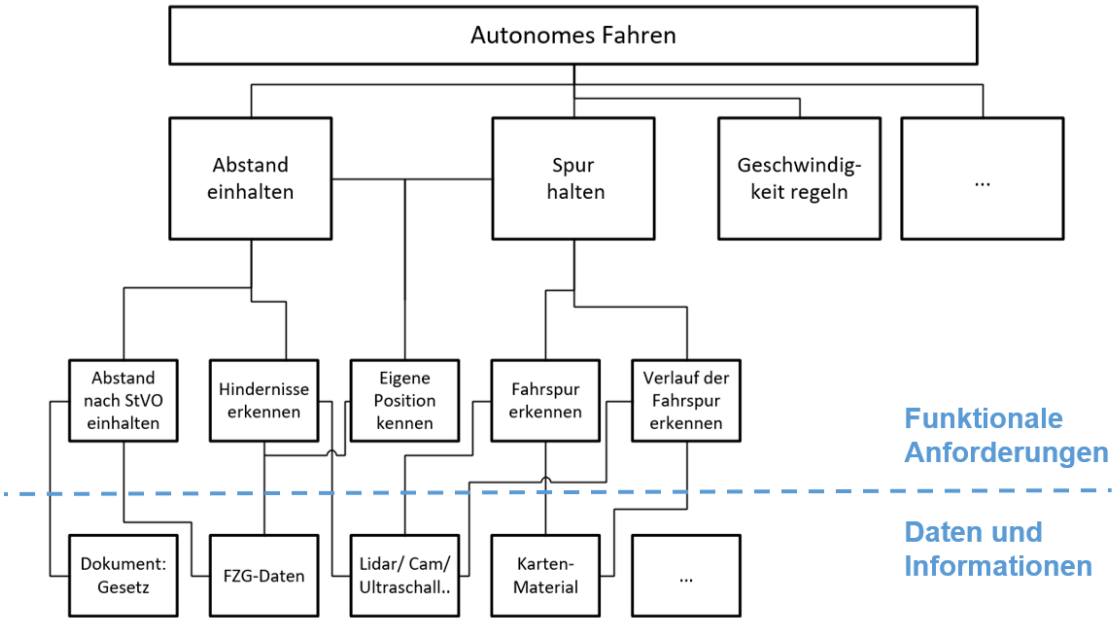


Abbildung 34: Beispiel des autonomen Fahrens

Wie im Datenmodell beispielhaft veranschaulicht, besteht das autonome Fahren aus einer Vielzahl von funktionalen Anforderungen, die zum DIF führen und deshalb in Summe das Daten und Informations-Modell darstellen. Zur Veranschaulichung sind hier drei Anforderungen ausdetailliert und zwei dieser Anforderungen bis auf eine mögliche technische Realisierung ausgeführt. Wie in diesem Teilausschnitt bereits ersichtlich wurde, verwendet das Beispiel den Top-down-Ansatz (vergleiche Systems Engineering) und beginnt mit funktionalen Anforderungen, die auf der hier dargestellten letzten Ebene schließlich auf technische bzw. juristische Daten und Informationen verweisen. Auch als D&I sind beispielsweise die Folgenden zu verstehen:

- Technische-physikalische Eigenschaften des Systems/ Teilsystem
- Randbedingungen aus der Systemgrenzen
- Normen und gesetzliche Vorgaben
- Abgeleitete, interne Anforderungen – vergleiche Abbildung 23

Da diese D&I verschiedene Attribute aufweisen, die einer Versionierung unterliegen – vergleiche Abbildung 26 und zueinander in Relation stehen, entstehen im Laufe des PEP die DIF, die es hier zu modellieren gilt.

Bereits bei der kleinen Anzahl an Funktionen ist ersichtlich, dass eine Komponente (zum Beispiel laser detection and ranging bzw. LiDAR) Daten für mehrere Anforderungen liefern kann. Zudem ist es durch dieses Vorgehen möglich, eine technische Realisierung auszutauschen – beispielsweise durch eine Innovation – ohne das gesamte Produkt neu aufsetzen zu müssen. Vielmehr kann man erkennen, welchen Funktionen durch diese Komponente betroffen sind – vergleiche (Chahin, Paetzold, *u. a.*, 2016).

4.3 Netzwerk-Theorie im Kontext des Frameworks

Dieses Kapitel greift nun das Beispiel des autonomen Fahrens auf und zeigt, wie die DIF mittels der Graphen- und Netzwerktheorie abgebildet werden können, um die D&I nachvollziehen und deren Abhängigkeit interpretieren zu können. Um dieses Potenzial nutzen zu können, müssen die DIF neutral formuliert werden. Dadurch werden die Analysen der mathemaischen Netzwerktheorie anwendbar. Dazu wird jedes mathematische Element einzeln eingeführt und in den Kontext gesetzt. Dabei fungiert die folgende Beschreibung als Guide-Line, um nachzuvollziehen, wofür das jeweilige Element verwendet werden kann, um es auf die eigene Situation anwenden zu können. Aus diesem Grund wird jedes mathematische Element

- zuerst neutral **definiert**
- in einem **neutralen Beispiel** erläutert
- und im Anschluss in den **Kontext des Beispiels des autonomen Fahrens** gesetzt

Um diese Definitionen treffen zu können, sind grundsätzlich Quellen, wie (Walther, 1979; Nitzsche, 2009; Turau, 2009; Büsing, 2010) notwendig. Aus den genannten Quellen sticht das Buch nach (Krischke, André, 2014) heraus, das sehr aktuell ist und als Grundlage fungiert.

In einem Vergleich der Notationen sind allerdings sowohl im englischen als auch im deutschen Sprachgebrauch die verwendeten Definitionen identisch.

Die grundlegendste Definition ist die des **Graphen G**, welcher ein Paar von Mengen beschreibt:

$$G = (V, E) \qquad (1)$$

Wobei V die Menge der Knoten darstellt, die durch die Menge E, den Kanten, verbunden werden. Die Buchstaben beschreiben Abkürzungen aus dem Englischen. Graph entspricht G, Vertex entspricht V und Edge entspricht E. Für die Anwendung in dieser Arbeit sind alle drei genannten Mengen als ein endlicher Raum zu verstehen.

Abbildung 35 zeigt ein Beispiel eines Graphen mit sechs Knoten - $v1$ bis $v6$, welche durch die Kanten $e1$ bis $e6$ unterschiedlich verbunden sind.

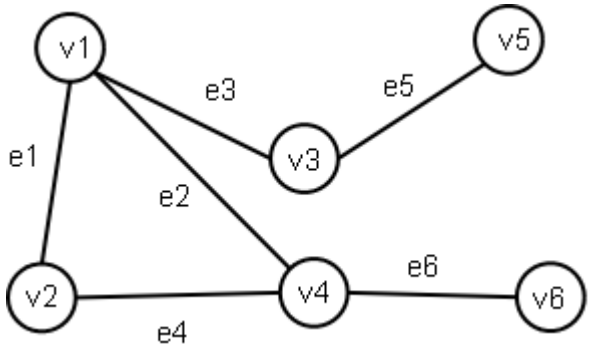


Abbildung 35: Ein Beispiel-Graph mit sechs Knoten

Analog zum mathematischen Graphen wird ein Ausschnitt des Datenmodells als Graph dargestellt. Hier bilden die drei angeleiteten Funktionen und die technische Realisierung bzw. Datenquelle (LiDAR) die Knoten, die entsprechend durch Kanten verknüpft sind – vergleiche Abbildung 36. Damit werden die ersten Bestandteile der D&I in ein mathematisches Netzwerk überführt.

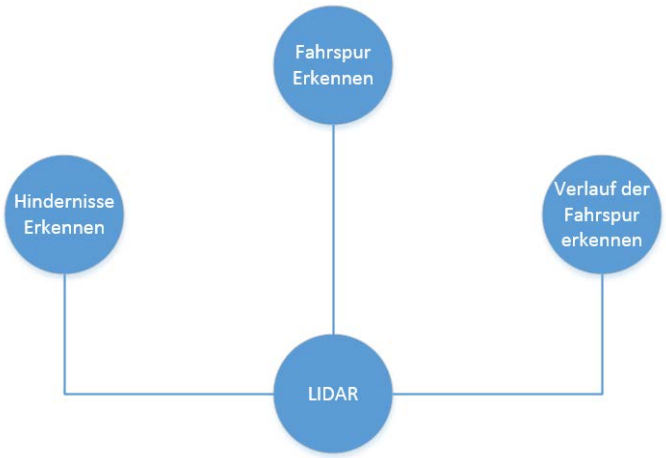


Abbildung 36: Graph mit vier Knoten aus dem Anwendungsbeispiel des autonomen Fahrens

Zu beachten ist hierbei, dass die hier verwendeten Knoten nicht mit sich selbst verbunden werden dürfen. Eine Kante muss immer zu einem Knoten führen und muss mind. immer einen Start- und mind. einen Endknoten besitzen. Somit gilt $G > V > E$ bei $N \in \mathbb{N}$.

Das heißt ein **vollständiger Graph** bei dem alle Knoten miteinander durch Kanten verbunden sind, kann bis zu $\binom{n}{2}$ Kanten aufweisen:

$$\binom{n}{2} = \frac{n(n-1)}{2} \qquad (2)$$

Betrachtet man zwei Knoten, die über eine Kante verbunden sind, nennt man diese ein **Tupel**:

$$V(P_{n=1}) = \{v_0, v_1\}\,; wobei\ P_n\ die\ Menge\ der\ Pfade\ ist \qquad (3)$$

Für Gewöhnlich können **Kanten bewertet** werden, um die Relation zwischen Knoten zu gewichten. Für das vorgestellte Modell müssen die Knoten ebenfalls eine Bewertung erhalten. Somit können sowohl Kanten als auch Knoten mit einer Zahl aus dem Raum $\mathbb{R}$ belegt werden. Das heißt für eine bewertete Kante gilt:

$$b(u, v)\ wobei\ b \in \mathbb{R}\ ist \qquad (4)$$

Ein **Weg** oder **Pfad** ist folglich die endliche Anreihung von Knoten und Kanten, die in Verbindung stehen. Diese Knotenmenge nennt man:

$$E(P_n) = \{\{v_0, v_1\}, \{v_1, v_2\}, \dots, \{v_{n-1}, v_n\}\} \quad (5)$$

Dabei ist es jedoch nicht vorgesehen, dass Knoten mehrfach durchlaufen werden. Ein Beispiel für verschiedene Pfade ist in Abbildung 37 gezeigt:

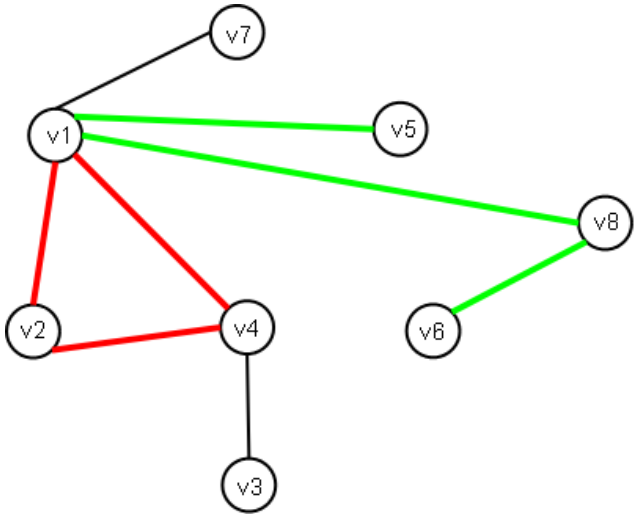


Abbildung 37: Beispiel-Graph mit verschiedenen Pfaden

Einzelne Pfade sind zum Beispiel $\{v_6, v_8\}$, $\{v_8, v_1\}$ oder $\{v_1, v_5\}$ – hier grün dargestellt. Es sind nicht nur einzelne Pfade erkennbar, sondern Zyklen zum Beispiel die Pfade $\{v_1, v_2\}$, $\{v_2, v_4\}$ und $\{v_4, v_1\}$ – hier rot dargestellt- welche zusammen einen Zyklus bilden. Zu beachten ist, dass ein Zyklus oder ein Kreis in einem ungerichteten Graphen durch einen Pfad, der wieder zum Startknoten führt, definiert ist.

Entsprechend kann ebenfalls die **Gewichtung der Knoten** vorgenommen werden, um auszudrücken, wie die Knoten zueinander in Relation stehen:

$$c(V)\ wobei\ c\ \in\ \mathbb{R}\ ist \quad (6)$$

Für einen Graphen mit sechs Knoten V_1 bis V_6 können somit die Kanten bewertet und somit zueinander gewichtet werden- vergleiche Abbildung 38.

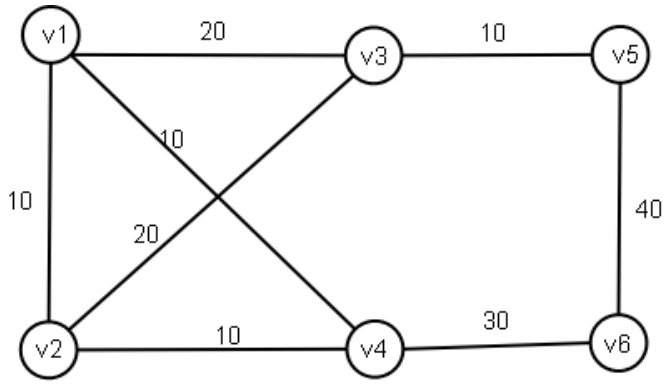


Abbildung 38: Graph mit sechs Knoten und gewichteten Kanten

Folglich ist die **Länge $l(w)$ eines Wegs** nur in einem bewerteten Graphen als die Summe der Kantenbewertungen eines Pfades $w = (v_0, v_1, \dots, v_n)$ dargestellt:

$$l(w) = \sum_{k=0}^{n-1} b(v_k, v_{k+1}) \; ; k \in \mathbb{N} \tag{7}$$

Greift man den Graphen aus dem Anwendungsbeispiel wieder auf, kann man die Kanten entsprechend ihrer Wichtigkeit bewerten. Hier sieht man beispielsweise, dass das LiDAR für die Hinderniserkennung auf einer Skala von 0 (unwichtig) bis 10 (sehr wichtig) mit der Bewertung 10 sehr bedeutsam ist – vergleiche Abbildung 39. Entsprechend ist die Abstufung der weiteren Funktionen zu verstehen.

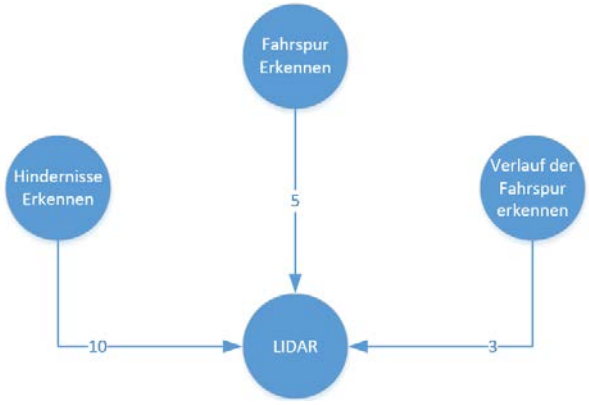


Abbildung 39: Bewerteter Teil-Graph für LiDAR -Funktion

Die bisherigen Definitionen erlauben die Definition von Knoten, Kanten und die Gewichtung der Relation zueinander, womit D&I aufgenommen und in gewichtete Abhängigkeiten gestellt werden können. Für einen Fluss von D&I ist die Definition des gerichteten Graphen notwendig, welche durch die Vorgabe einer Richtung der Kanten entsteht. Dem-

nach kann eine Kante nur von einem Start v_i- auf einen Endkonten v_j gerichtet sein. Graphisch wird dies durch einen Pfeil (v_i, v_j) dargestellt. Geht vom Endkonten wieder ein Pfeil zu dem Startknoten, spricht man von einer bidirektionalen Verbindung – vergleiche Abbildung 40:

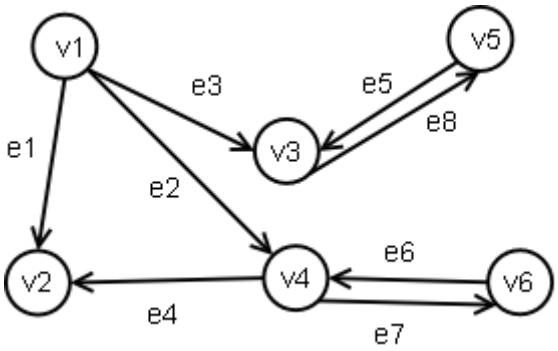


Abbildung 40: Beispiel eines gerichteten Graphen

Für das eingeführte Beispiel kann man die Kanten entsprechend richten, womit ein gerichteter Graph entsteht, welcher erlaubt DIF aufzunehmen. Hier bedeutet die Pfeilrichtung: „wird erfüllt durch“ – vergleiche Abbildung 41:

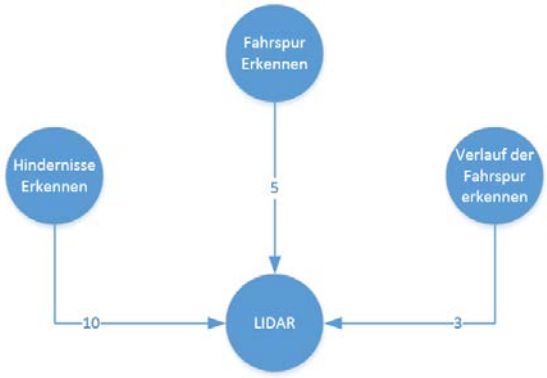


Abbildung 41: Gerichteter und gewichteter Teil-Graph für LiDAR-Funktion

Um die Daten- und Informationsflüsse zu differenzieren, ist die Unterscheidung der Verknüpfung der Knoten notwendig. Dabei müssen die bekannten Detaillierungen und Abhängigkeiten zwischen den Anforderungen und dem Produkt bzw. Systemen, durchgehend nachgebildet werden.

Zusammengefasst liefert die reine Mathematik für die Verknüpfung von Knoten durch Kanten im Allgemeinen diese Möglichkeiten:

- ungerichtet,

- gerichtet,
- gewichtet,
- und die jeweilige Kombination

Für die Gewichtung sind verschiedene Skalen denkbar. Die Vorarbeit „An approach for using the FMEA for the development of a new and revolutionary rotation piston engine“ zeigt eine Möglichkeit der Gewichtung auf, ermöglicht aber auch -lokal für einen Teilgraphen oder global für den Gesamt-Graphen - diese zu kombinieren und zum Beispiel durch Operationen, wie der Summe, Produkt, Differenz, usw. zu verknüpfen (Chahin, Paetzold, *u. a.*, 2016). Dadurch lassen sich Aussagen zum Beispiel über die relative Gewichtung treffen und entsprechend interpretieren.

Für die Aufnahme der Abhängigkeiten der DIF sind die getroffenen Kantendefinitionen unzureichend. Deshalb werden hier noch weitere Verbindungsarten verwendet, die teilweise aus SysMl oder aus Requirements-Management-Tools, wie DOORS oder DOORS-NEXT-Generation, entlehnt werden. Natürlich ist es auch möglich persönliche Definitionen für die Kanten zu formulieren.

Welchen Text man wählt, hängt von der Unternehmenssituation ab. Allerdings sollte man hier eine einheitliche und verständliche Definition wählen. Diese Entscheidung obliegt im Normalfall dem Anwender. Hier werden jedoch einige Beispiele gezeigt, wie die Daten- und Informationsflüsse bzw. die Abhängigkeiten weiter detailliert werden können.

Aus SysMl lassen sich im Kontext dieser Arbeit die Verbindungen aus dem Anforderungsdiagramm gut nutzen. Diese sind allgemein – vergleiche Tabelle 6.

Tabelle 6: Link-Typen aus der SysMl-Sprache nach Weilkiens (Weilkiens, 2007; Alt, 2012; Kapos *u. a.*, 2014)

Link-Typ	**Erklärung**
Copy	Ist eine Anforderung eine Kopie einer anderen Anforderung, wird die Beziehung zwischen diesen als "Copy Relationship" bezeichnet. Sie ist von der Kopie zum Original gerichtet und wird mit «copy» gekennzeichnet.
Trace	Besteht eine Beziehung zwischen einer Anforderung und einem beliebigen Modellelement, handelt es sich um eine "Trace Relationship". Sie wird mit «trace» gekennzeichnet. Sie ist beispielsweise von einer funktionalen Anforderung auf eine nicht-funktionale Anforderung gerichtet.

Refine	Wird eine Anforderung durch weitere Anforderungen/Modellelemente detaillierter beschrieben, so spricht man von "Refine Relationship". Sie ist von der verfeinerten/detaillierteren Anforderung/Modellelement auf die zu verfeinernde/allgemeinere Anforderung gerichtet und wird mit «refine» gekennzeichnet.
Satisfiy	Wird eine Anforderung von einem Designelement erfüllt, handelt es sich um eine "Satisfy Relationship". Sie ist vom Designelement zur Anforderung gerichtet und wird mit «satisfy» gekennzeichnet.
Verfiy	Kann eine Anforderung durch einen Test verifiziert werden, spricht man von "Verify Relationship". Sie ist vom Test auf die zu verifizierende Anforderung gerichtet und wird mit «verify» gekennzeichnet.

Aus Doors-Next-Generation (DNG) werden hier nur einige Linktypen erwähnt, um zu zeigen, wie detailliert Kanten definiert werden können. Diese sind – vergleiche Tabelle 7.

Tabelle 7: Link-Typen aus dem Tool Doors-Next-Generation (Engineering, 2020)

Link-Typ	**Dt. Bez.**	**Bez.-Beziehungen**	**Beschreibung**
Affected By	Beeinflusst durch/ von	Affects/ Affected By	Captures the relationship between a Requirements Management artifact and a Change Management item that has an effect on the implementation of the requirement artifact (e.g., a Defect in Rational Team Concert). Links of this type appear as 'Affects' in Change Management.
Decomposition	Erbt/ vererbt	Parent Of/ Child Of	Captures part-whole relationships between Requirements Management artifacts. These types of links are typically used to represent artifact hierarchies.

Derives Architecture Element	Leitet sich von ... ab/ leitet ... an	Derives From Architecture Element/ Derives Architecture Element	Captures the relationship between a Requirements Management artifact and an Architecture Management item that represents a model of the requirement artifact (e.g., a UML Use Case in Rational Design Manager). Links of this type appear as 'Derives From' in Architecture Management.
Embeds	Eingeschlossen in/ durch	Embedded In / Embeds	Tracks a containment relationship between Requirements Management artifacts. These types of relationships occur when performing operations such as 'Insert Artifact' and 'Insert Image' for a Text artifact.
Extraction	Ausgelagert durch/ von	Extracted From/ Extracted	Captures when the content of a Requirements Management artifact has been created from content of another Requirements Management artifact. This type of link is created when performing extraction-based operations, for example, the 'Save As New' operation for a Text artifact.
Implemented By	Implementiert/ impl. durch	Implements/ Implemented By	Captures the relationship between a Requirements Management artifact and a Change Management item that describes the implementation of the requirement artifact (e.g., a Story in Rational Team Concert). Links of this type appear as 'Implements' in Change Management.
References	Referenziert auf/ druch	Referenced By / References	Captures the relationship between a Requirements Management artifact and another Requirements Management artifact in a different RM instance or external project area.

Die getroffenen Definitionen und Beispiele erlauben es nun, die DIF zu beschreiben und in Relation zu setzen. Allerdings ist es auch notwendig, gewisse Aspekte des Graphen zu betrachten und den nicht relevanten „Rest“ auszublenden. Dafür ist die Definition des Teilgraphen $\boldsymbol{G_n}$; $\boldsymbol{n} \in \mathbb{N}$ notwendig, welche eine Untermenge aus einem Graph G be-

schreibt. Folglich sind die Knoten und Kanten von $\boldsymbol{G_n}$ eine Teilmenge von G. In der Literatur spricht man hier oft vom Teilgraph mit der Abkürzung H. Hier wird jedoch der Teilgraph als $\boldsymbol{G_n}$ bezeichnet – vergleiche Abbildung 42.

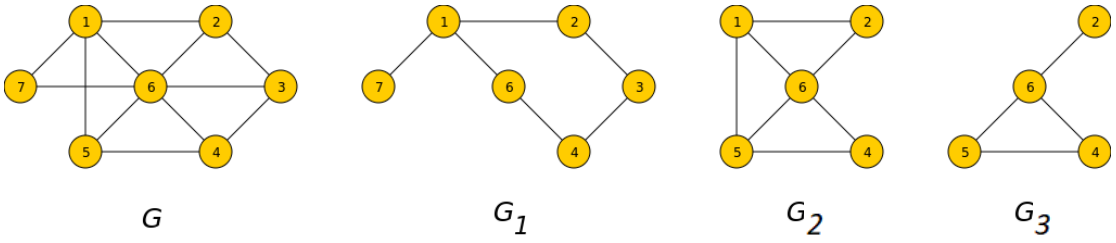


Abbildung 42: Drei Teilgraphen aus Gesamt-Graph G

Die Aufteilung des Graphen in Teilgraphen findet Anwendung, wenn aus dem Graph eine Untermenge selektiert wird. Die „überflüssigen“ Knoten des Graphen werden somit ausgeblendet. Eine bewusste Selektion ist darüber hinaus maßgebend, wenn nur die Knoten mit bestimmten Eigenschaften betrachtet werden sollen. Zieht man hier das eingeführte Beispiel heran, dann wäre die Funktionen des „Abstandhaltens“ eine mögliche vertikale Selektion – vergleiche Abbildung 43:

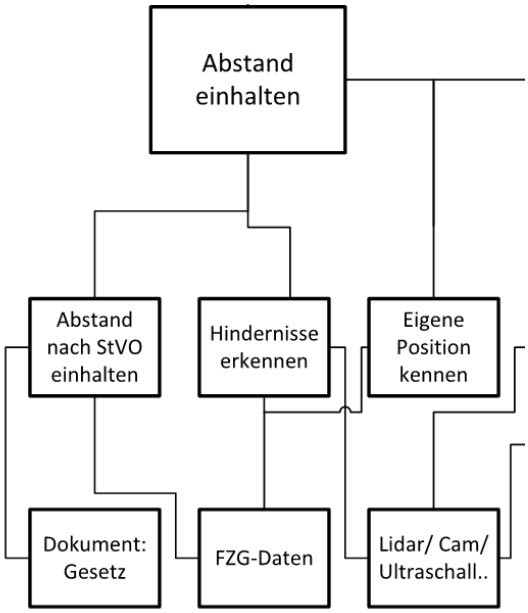


Abbildung 43: Teil-Graph der Funktion „Abstand einhalten“

Da die graphische Darstellung von Abhängigkeiten „nur“ eine visuelle Darstellung ist, kann man die Graphen-Information $G = (V, E)$ in andere Formen überführen, die mathematisch auswertbar bzw. maschinenlesbar sind. Die **Adjazenz-Matrix** ist eine Darstellung des Graphen in Matrixform, wobei $A(G)$ als eine $(n\ x\ n)$ Matrix definiert ist:

$$A(G) = \left(d_{ij}\right); wobei\ d_{ij} = d(v_i, v_j) \qquad (8)$$

Die Matrix wird üblicherweise von links nach rechts gelesen und beschreibt die Verbindung zweier Knoten. Die Eintrag-Zeile ist mit der Eintrag-Spalte verbunden. Handelt es sich um einen ungewichteten Graphen, werden die Einträge meistens mit einem „x“ oder einer „1“ belegt. In einem gewichteten Graphen trägt der Eintrag die Gewichtungszahl.

Ist es ein ungerichteter Graph, werden die Einträge an der Diagonalen gespiegelt. Die Diagonale selbst bleibt für gewöhnlich unbelegt, da dies andernfalls eine Verbindung mit demselben Knoten bedeuten würde. Allerdings kann die Diagonale dafür verwendet werden, die Knoten mit einer eindeutigen Zahl zu definieren. Dieser Ansatz wird im Weiteren verfolgt und ausgebaut.

Die Adjazenz-Matrix des ungerichteten Graphen aus Abbildung 35 ist somit eine an der Diagonalen (graue Felder) gespiegelte Matrix - vergeliche Abbildung 44:

	v1	v2	v3	v4	v5	v6
v1		1	1	1		
v2	1			1		
v3	1				1	
v4	1	1				1
v5			1			
v6				1		

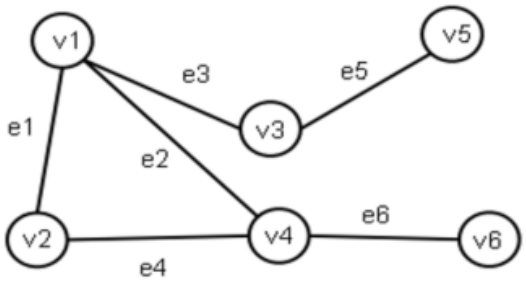


Abbildung 44: Adjazenz-Matrix und zugehöriger Graph

Die Adjazenz-Matrix ist lediglich eine „andere“ Darstellung des Graphen, welche die mathematischen Berechnungen erleichtert und für die Datenverarbeitung relevant ist.

Im Falle eines gerichteten Graphen, ist die Matrix nicht mehr gespiegelt und die Einträge beschreiben entsprechend die Richtung des Pfeiles – vergleiche Abbildung 45.

	v1	v2	v3	v4	v5	v6
v1		1	1	1		
v2						
v3					1	
v4		1				1
v5			1			
v6				1		

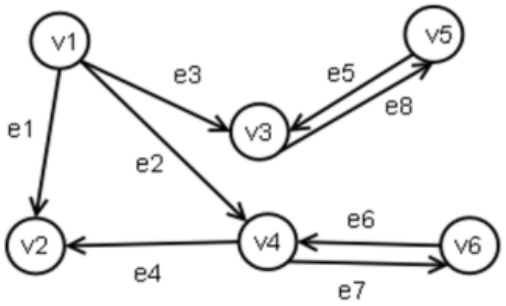


Abbildung 45: Adjazenz-Matrix eines gerichteten Graphen

Multidimensionale Erweiterung des Graphen:

Für die vorgeführte Anwendung reicht die einfache bzw. eindimensionale Definition eines Graphen nicht aus, um die gewünschte Detailierung der DIF zu modellieren (Chahin *u. a.*, 2017). Die zu betrachtenden D&I sind unterschiedlich in ihrer:

- Art/ Darstellung
- Detailtiefe
- Quelle
- Attribuierung
- Relation zueinander (Abhängigkeiten die zu komplexen DIF führt)

Folglich handelt es sich bei dem aufzubauenden Framework um eine heterogene Menge. All diese Information in einen Graphen zu integrieren, würde eine mathematische Analyse unmöglich machen. Deshalb muss die bisher geschaffene Grundlage erweitert werden.

Um das Netzwerk-Modell nach seinen Eigenschaften unterscheiden zu können, wird die mathematische Definition des Teilgraphen herangezogen. Somit entsteht aus dem Gesamt-Graphen ein **Multi-Netzwerk**, das „künstlich“ aufgrund der unterschiedlichen Eigenschaften seiner Knoten getrennt bzw. zusammengefasst wird.

Abbildung 46 zeigt ein Beispiel, in dem die Kanten innerhalb eines Teilgraphen eine Verbindungsart beschreiben. Zusätzlich sind innerhalb des Gesamtgraphen Verbindungen möglich, die zum Beispiel von einem Layer zum anderen führen können.

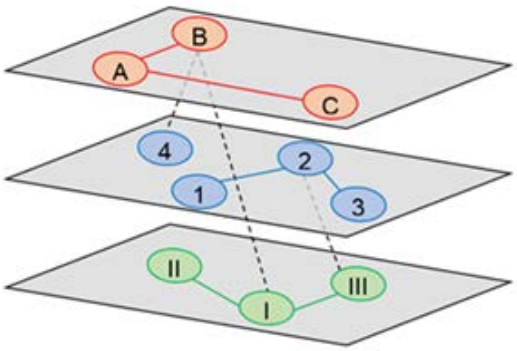


Abbildung 46: Multi-Layer eines Graphen

Bereits in eingeführten Bespiel ist eine Trennung nach Layern möglich. So stellt das Datenmodell allein ein Multi-Layer dar, wenn man die vertikale Detaillierung als Layer interpretiert. Deutliche Layer sind die Ebenen: „Funktionale Anforderung“ und „Technische und Juristische Daten“, die aufgrund des definierten Inhalts der Knoten als zwei unterschiedliche Layer zu deuten sind – vergleiche Abbildung 47:

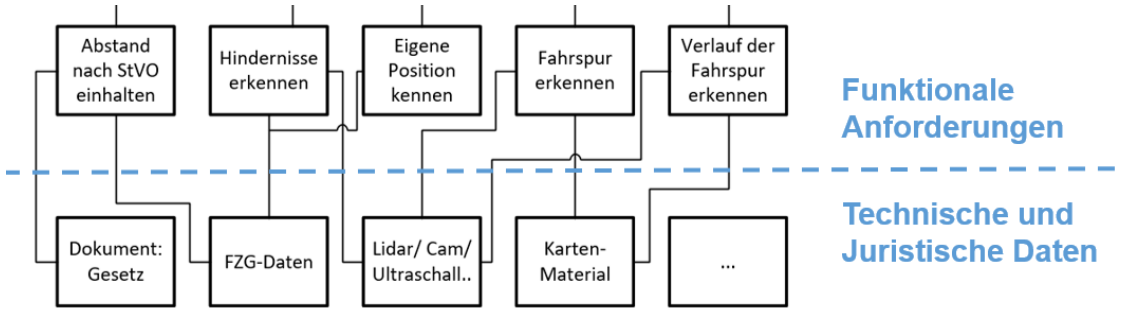


Abbildung 47: Multi-Layer der funktionalen Anforderungen und technischen und juristischen Daten

In der Vorarbeit (Chahin, Hoffmeister, *u. a.*, 2016) ist bereits herausgearbeitet worden, dass es für die Aufnahme der Daten- und Informationen, welche zum Beispiel für die Planbarkeit von Absicherungsmaßnahmen notwendig sind, nicht ausreicht die Knoten und Kanten mit nur einer einzigen Information oder Identifikation zu belegen. Folglich müssen alle Knoten und Kanten mit beliebig vielen Informationen belegbar sein. Diese Erweiterung des Graphen nennt man **Multiplex** - vergleiche Abbildung 48.

Dabei ist es nicht zwingend notwendig, dass alle Knoten oder Kanten mit allen Informationen belegt sind. Es ist auch nicht vorgeschrieben, dass alle möglichen Verbindungen vorhanden sein müssen. Lediglich die Reihenfolge der möglichen Kategorien der Knoten und Kanten ist im vornherein bekannt, definiert und sortiert. Im Grunde handelt es sich um n-Netzwerke des gleichen Netzwerks, die überlagert werden.

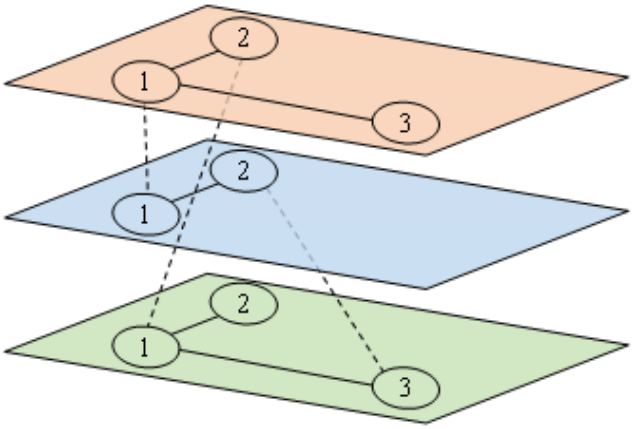


Abbildung 48: Multiplexer Graph

Zusätzlich muss es möglich sein, jeden beliebigen Knoten oder jede Kante nicht nur mit einer endlichen Zahl aus $\mathbb{R}$ zu belegen, sondern beispielsweise mit einem Text oder String.

Am Beispiel der funktionalen Anforderungen zeigt sich, dass es nicht ausreicht, die Knoten mit einer Zusammenfassung einer Anforderung (vergleiche „Hindernisse erkennen“) zu belegen. Vielmehr muss hier der Wortlaut der Anforderung nach der Satzschablone formuliert werden und der Bearbeiter, der Status, die Dringlichkeit, etc. als multiplexe Knoten definiert sein.

Greift man nun hier nur die beiden folgende „Knoten“ heraus – vergleiche Abbildung 49 - können die Knoten um weitere notwendige Informationen erweitert werden – vergleiche Abbildung 50.

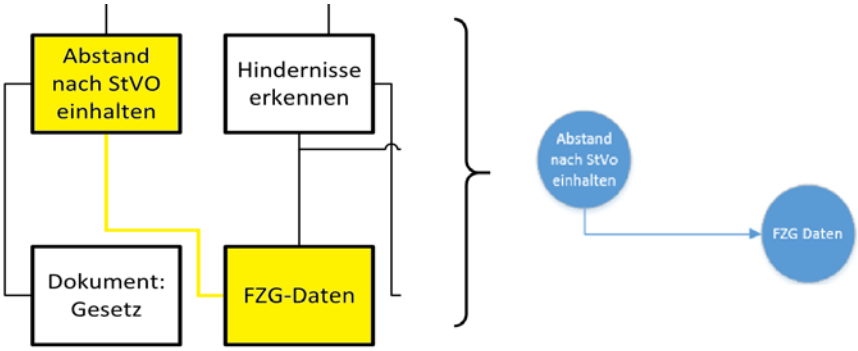


Abbildung 49: Teil-Graph "Hindernisse erkennen"

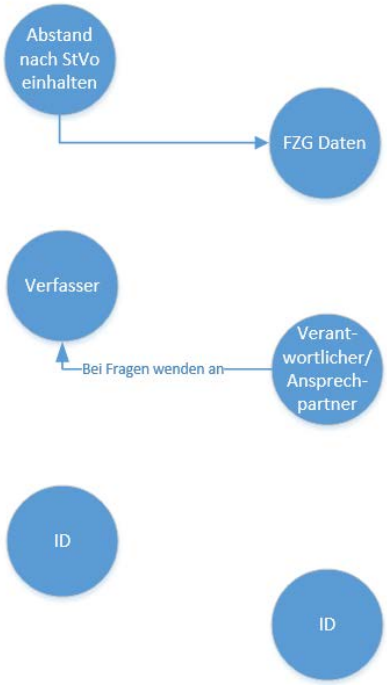


Abbildung 50: Multiplexe Darstellung der Funktion "Hindernisse erkennen"

Demnach beschreibt das zweite Level die ausführlichen Anforderungen nach der Satzbauschablone. Das dritte Level bezieht sich auf die verantwortliche Person und das vierte auf die eindeutige ID, usw.

Aufgrund der zu beherrschenden Komplexität und Größe des Netz-Modelles ist es einerseits notwendig Informationen zu bündeln oder zusammenzufassen und andererseits Knoten zu entfalten – **Multi-Level**. Dabei werden Teilnetzwerke durch einzelne Knoten repräsentiert und damit das Netzwerk übersichtlicher modelliert. Umgekehrt können Knoten detailliert werden, indem diese in einem Level höherer Granularität ein eigenes Netzwerk darstellen:

$$I_n(G) = (V_I\,, E_I)\,;\;\; n \epsilon \mathbb{N} \qquad (9)$$

Die Untermenge $\boldsymbol{I_n}$ erweitert G um weitere Knoten V_I und Kanten E_I und bilden somit neue Graphen $I_n = I_0, I_1, \dots, I_n\,, n \epsilon \mathbb{N}$, welche alle bisherigen Definitionen enthalten können – vergleiche Abbildung 51.

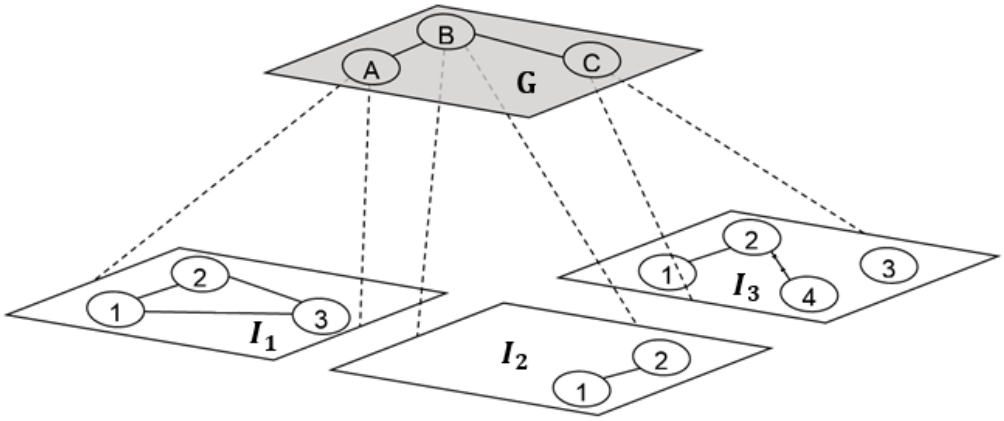


Abbildung 51: Multi-Level eines Graphen

Auch diese Einteilung ist bereits im gezeigten Beispiel zu finden. So lassen sich die Detaillierungen der funktionalen Anforderungen, wie im Beispiel des „Abstandhaltens“ in drei weitere Anforderungen aufspalten, denen auch bestimmte Parameter (zum Beispiel in $x\ [m] \pm y[m]$) zugeordnet werden können. Dieser Teilgraph der drei Anforderungen stellt für sich ein Subnetzwerk an Knoten dar, womit der Graph Multilevel besitzt – vergleiche Abbildung 52:

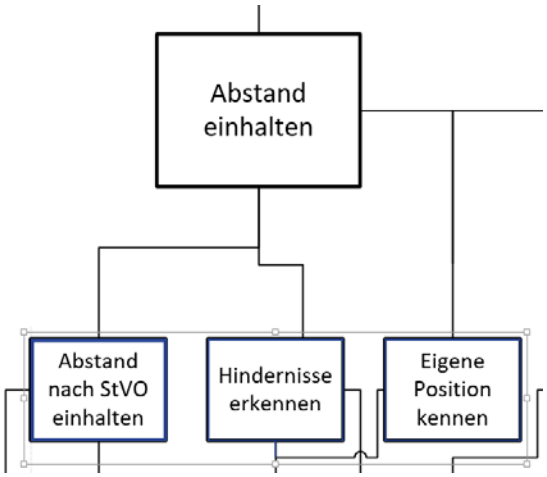


Abbildung 52: Multi-Level der Funktion "Abstand einhalten"

Eine Verknüpfung ist zugleich auf allen anderen - bereits definierten - Ebenen möglich. So kann beispielsweise auf einer tieferen Ebene desselben Knotens der zulässige Parameterbereich einer Anforderung mit einer anderen verknüpft werden. Eine solche Verknüpfung kann sich aufgrund der physikalischen Abhängigkeit ergeben und durch eine Formel beschrieben sein. Ob es eine gerichtete oder ungerichtete Kante ist, hängt ebenfalls vom Anwendungsfall ab. Der Entwickler hat dadurch die Möglichkeit bekannte Abhängigkeiten zu ergänzen, um den Informationsfluss zu vervollständigen.

Somit sind Kanten auch auf der multiplexen Ebene möglich – jedoch nicht zwingend erforderlich. Je genauer eine Verknüpfung ist, umso aussagekräftiger ist die Beschreibung der Abhängigkeit.

Gesamtfazit:

Am Beispiel des autonomen Fahrens wurde in diesem Kapitel als Pendant zur Sprache des MBSE-Dreiecks die Graphen- und Netzwerktheorie eingeführt, die notwendig ist, um die DIF eines komplexen Systems zu modellieren. Hierfür wurden allgemeine Notationen aus der Graphen-Theorie verwendet, gleichzeitig erweitert und jeweils am Beispiel erläutert. Zudem wurde der Fokus auf Multi-Netzwerke gelegt, um die notwendige Detailtiefe der DIF aufnehmen zu können. Diese Erkenntnisse gilt es nun zu verwenden, um einen ganzheitlichen Ansatz aufzustellen, welcher die notwendigen D&I strukturiert und Optimierungen erlaubt.

5 Detaillierter Aufbau des Drei-Layer-Daten-Informations-Frameworks

5.1 Allgemeine Erläuterung zum Daten-Informations-Framework

Dieses Kapitel greift die Erkenntnisse der bisherigen Ausführungen auf und fasst diese zu einem Daten-Informations-Framework zusammen, welches die eingangs gestellte Forschungsfrage beantworten kann. Dabei werden die gesetzten Grundlagen für die Erstellung eines Graphen aufgegriffen und zu einem Framework verarbeitet. Da nicht alle Anwendungen explizit besprochen werden können, wird an dieser Stelle die zu Beginn erwähnte Anwendung – die Komplexitätsbewältigung bei der Planung von Eigenschaftsabsicherung – exemplarisch herausgegriffen.

Im Detail werden folgende Ansätze verwendet und verknüpft:

- Das Modell basiert auf der mathematischen Lehre der Netzwerk- und Graphentheorie und ermöglicht die Übertragung von Netzwerkanalysen auf die Problemstellung – vergleiche vorangegangenes Kapitel.
- Die Prozesslogik und der Top-down Ansatz wird aus dem Systems Engineering entnommen
- Für die Definition, Struktur und Handhabung von Anforderungen werden die Regeln nach Rupp verwendet. Diese Ansätze stammen zwar aus der Softwareentwicklung, sind aber auch für die physische oder mechatronische PE verwendbar. Zudem ist der Ansatz sehr aktuell, vereint einen Großteil des theoretischen bzw. wissenschaftlichen Anforderungsmanagements und beinhaltet zudem Aspekte, die aus der Zusammenarbeit mit der Industrie stammten.
- Ebenfalls werden die Anforderungen nach der Logik des Systems-Engineering von den Gesamtanforderungen bis hin zu detaillierten Einzelanforderungen hierarchisch gegliedert.
- Die Beschreibung des Produktes geschieht nach Weber. Dort wird das Produkt in Merkmale und Eigenschaften unterteilt und über den CPM/ PDD-Ansatz spezifiziert.
- Die Detailierung des Produktes wird mit Hilfe des Webermodelles - ausgehen vom Gesamtsystem bis hin zu den Komponenten – definiert und spezifiziert.
- Um die Workflows abzubilden und später auch zu analysieren werden Beispielprozesse aus der V&V herangezogen.

Das heißt, das im vorherigen Kapitel eingeführte Beispiel des autonomen Fahrens, welches bisher „nur“ die Relation zwischen Anforderungen und technischer Überführung

aufgezeigt hat, wird hier um weitere Aspekte wie zum Beispiel die operationale oder prozessuale Betrachtung erweitert. Diese erlauben es zu validieren, ob die DIF ganzheitlich aufgefasst werden können und wie weit die Analysemöglichkeiten der NWT für Prozessoptimierungen Anwendung finden.

Die Summe ist ein ganzheitliches Netzwerk-Modell, das in der Lage ist, die notwendigen D&I aufzunehmen und durch entsprechende Verknüpfung eine Nachverfolgbarkeit erlaubt. Da die D&I in drei Hauptkategorien – Entwicklungsartefakte, Organisation und Prozess - gegliedert werden können, stellt das entwickelte Framework ein **Drei-Layer-Daten-Informations-Framework** dar – vergleiche Abbildung 53 bzw. Anhang 1:

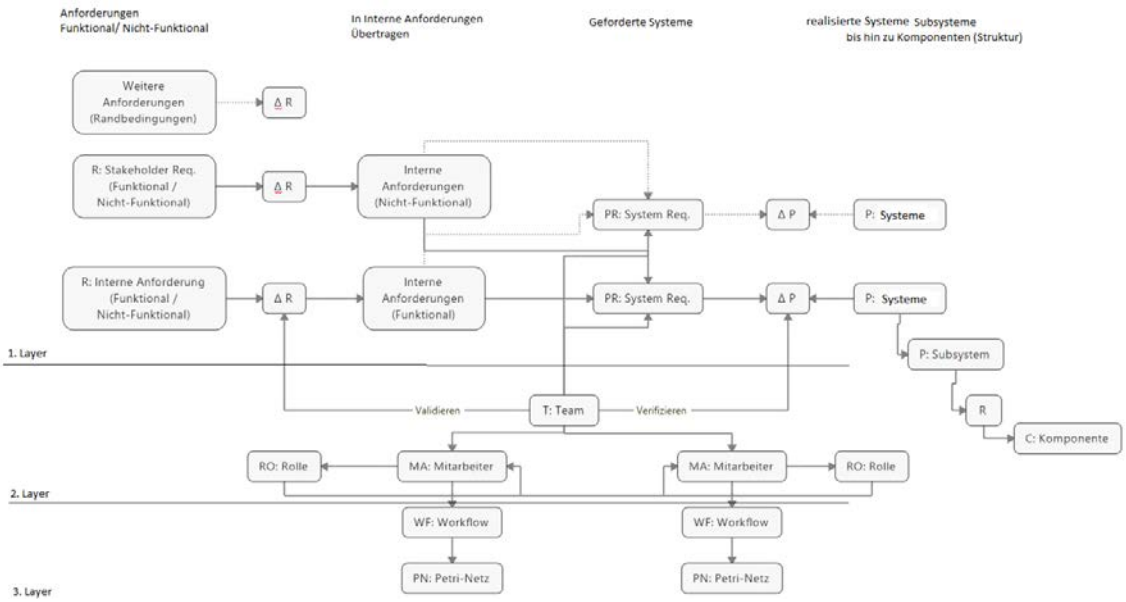


Abbildung 53: Das Drei-Layer-Daten-Informations-Framework

Beim ersten Layer handelt es sich um alle im Entwicklungsprozess verwendeten Artefakte – unabhängig von ihrer Detailierung, Aufteilung und deren Status. Dazu gehören nicht nur Dokumente, sondern alle Anforderungen, Daten, wie CAD-Daten, Prototypen, usw.

Das zweite Layer beschreibt die Unternehmensstruktur. Dies beinhaltet die Organisationsstruktur des Unternehmens, seine Aufteilung vom Team bis hin zu den einzelnen Mitarbeitern und -falls für die Analyse notwendig- etwaige Zulieferer. Zusätzlich können die Rollen von Mitarbeitern oder Teams dargestellt werden.

Das dritte Layer beinhaltet die Aufgaben bzw. die Aktionen, die im Laufe des PEP getätigt werden. Um die statische Abbildung des Netzwerks zu umgehen, können Workflows in Petri-Netzen überführt werden.

Die drei dargelegten Layer (Artefakte, Organisation und Prozess) beschreiben ein Multi-Layer-Netzwerk, da die Knoten und Kanten aus einem Netzwerk „künstlich“ kategorisiert

werden. Hier findet die Trennung aufgrund der vorgenommen inhaltlichen Unterscheidung statt. Es wäre grundsätzlich möglich, alle Layer in einem Netzwerk darzustellen. Für eine bessere Übersichtlichkeit und eine separat anzuwendende Analyse wird jedoch davon abgesehen.

Analog zu den mathematischen Definitionen aus dem vorangegangenen Kapitel werden die Knoten und Kanten der jeweiligen Elemente definiert. Folgende Übersicht zeigt, welche Notation für die Teilgraphen festgelegt wurde und wie entsprechend rekursiv die Untermenge dieser definiert ist – vergleiche Abbildung 54 bzw. Anhang 2:

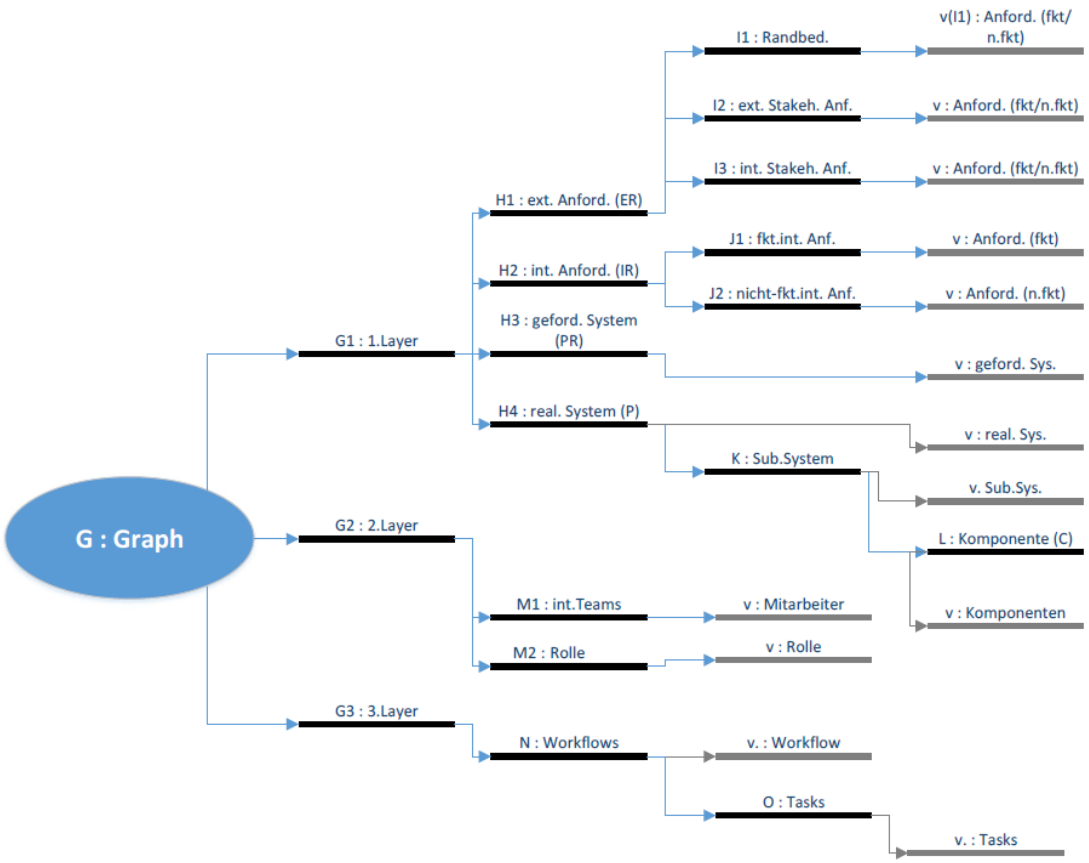


Abbildung 54: Notation der Teilgraphen des Drei-Layer-Daten-Informations-Frameworks

In Folgendem werden die einzelnen Layer (Teilgraphen) erläutert. Dabei wird der erste Layer am detailliertesten ausgeführt. Die beiden weiteren Layer bleiben absichtlich allgemeiner gefasst, da diese auf die spezifische Unternehmenssituation bzw. den Prozess bezogen werden müssen. Es werden auch nur Layer-interne Verknüpfungen eingeführt. Dies schließt jedoch nicht aus, dass auch Layer-übergreifende Verbindungen möglich

sind. Im Prinzip werden die mathematischen Definitionen verwendet, gegebenenfalls erweitert und am Beispiel erklärt. Ziel ist es, die Guideline zu erweitern. Dies beinhaltet das Erstellen des Frameworks und macht eine Anpassung der Anwendung in Bezug auf die eigene Situation notwendig.

5.2 Erstes Layer des Daten-Informations-Frameworks: Artefakte und Aufbau

Das erste Layer stellt dabei einen Teilgraphen G_1 aus G dar, wobei G den Gesamtgraphen abbildet und das gesamte Framework beinhaltet.

Da die Anforderungen den PEP triggern, ist es folgerichtig damit zu beginnen. Die erste Erweiterung zu den bekannten Modellen (aus dem Stand der Technik) ist die zusätzliche Unterscheidung hinsichtlich der Quelle der Anforderungen. Deshalb werden zu Beginn alle Anforderungen als „externe“ Anforderungen gesehen. Hier erfolgt eine Anwendung der Definition nach Rupp, da diese danach unterscheidet, welche Elemente außerhalb des eigenen und damit beeinflussbaren Kontextes liegen. Trotzdem sind die Stakeholder-Anforderungen nach unternehmensinternen bzw. -Externen – wie Kundenanforderungen, Marktanforderungen, etc. – zu unterscheiden. Weitere Randbedingungen – nach der Definition nach Rupp (Rupp *u. a.*, 2014) – sind gesondert zu führen. Somit sind die drei Hauptquellen für Anforderungen referenziert - jedoch noch nicht speziell in funktional bzw. nicht-funktional gesplittet – vergleiche Abbildung 55:

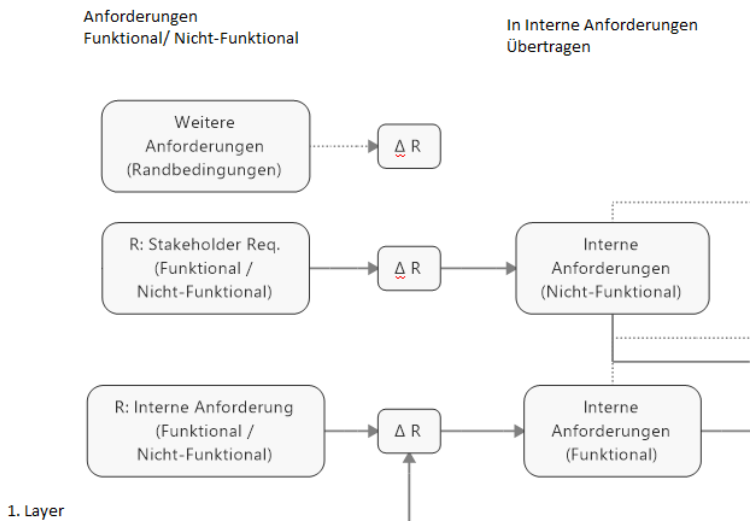


Abbildung 55: Externe und interne Anforderungen modelliert im ersten Layer

Sollte bekannt sein, um welche Art von Anforderungen es sich handelt, kann eine Mehrfachbelegung der Knoteninformation vorgenommen werden – vergleiche Mulitiplex

(vorheriges Kapitel). Diese Separierung wird aber bewusst auf die nächste Stufe verlegt, weil dort die Anforderungen in „interne“ Anforderungen uminterpretiert und möglichweise ergänzt und umsortiert werden. Hier wird unterschieden, welche Anforderungen explizit funktional und welche nicht-funktional sind.

Da die Anforderungen als extern betrachtet werden, können diese nach dem jeweiligen Stakeholder unterschieden werden. Die Unterscheidung nach externen und internen Anforderungen, erlaubt es zu prüfen, ob und in welcher Form ein Delta (Δ) zwischen den jeweiligen Anforderungen existiert und wie sich das Delta zeitlich verändert. Es ist zu erwarten, dass hier eine $n:m$ –Beziehung zwischen externen und internen Anforderungen entsteht. Die Beziehung hilft nachzuverfolgen, wie die externen Anforderungen in interne überführt worden sind und welches Delta über die Entwicklungszeit gemessen werden kann. Dieses Delta ist äquivalent zum Delta nach Weber - vergleiche Abbildung 21- definiert, zielt jedoch auf die Validierung ab – vergleiche Kapitel 2.2.2.3.

An dieser Stelle ist zu erinnern, dass die repräsentativen Knoten, wie zum Beispiel „Stakeholder Req.“, um mehrere Level erweitert werden können, um die Informationen auf der richtigen Ebene zu verankern. Rupp stellt hier eine mögliche Hierarchie für die Spezifikation von Anforderungen auf, welche praxisnah ist und in der Industrie Anwendung findet – vergleiche Abbildung 56:

Ebenen	Zugeordnete Begrifflichkeiten				
	aus herkömmlichen Vorgehen, wie z. B. RUP oder V-Modell	aus agilen Vorgehen			
		Scrum	Adaptive Software Development	Lean Software Development	Crystal
Spezifikationslevel 0	Grobe Systembeschreibung, Systemziele, Systemüberblick, Vision, Introduction, Mission Statement	Sprint Goal Product Vision	Project Vision, Project Data Sheet	Release Plan	Mission Statement
Spezifikationslevel 1	Anwendungsfall (Use-Case), User Story, (Anwendungs-) Szenario, Fachkonzept, Funktionsbeschreibung, Funktionsgliederung, fachliche Anforderung, Organisational Requirement, Featureliste, Kontextabgrenzung	Epic Theme	Product Specification Outline	User Story, Backlog Item, Use-Case	Actor Goal List, User Role Model, Use-Case
Spezifikationslevel 2	Anwenderforderung, Nutzeranforderung, Operational Concept Description, Operational Requirements Description, Interface Requirements Specification, Lastenheft, Sollkonzept, Grobspezifikation, Operational Requirement, betriebliche Anforderung, Testfälle, Featureliste	User Story, Technical User Story, Backlog Item, Acceptance Criteria, Definition of Done	Product Specification Outline	User Story, Backlog Item, Use-Case, Iteration Plan, Qualifier, Test Case	Common Domain Model, Test Case, Iteration Plan
Spezifikationslevel 3	Detaillierte Anwenderforderungen, Technische Anforderung, Schnittstellenübersicht, Schnittstellenbeschreibung, System Segment Specification, Interface Requirements Specification, Systemanforderung, Feinspezifikation, Testfälle, Featureliste	Technical User Story, Backlog Item, Acceptance Criteria, Definition of Done	Technical User Story, Backlog Item, Acceptance Criteria, Definition of Done	Qualifier, Test Case	Architecture Description, Test Case
Spezifikationslevel 4	Komponentenanforderungen, Technische Anforderung, Schnittstellenübersicht, Schnittstellenbeschreibung, Software Requirement Specification, Interface Design Description, Pflichtenheft, Feinspezifikation, Modulanforderung, Testfälle	Technical User Story, Backlog Item, Acceptance Criteria	Technical User Story, Backlog Item, Acceptance Criteria	Test Case	Architecture Description, Test Case

Abbildung 56: Ebenen der Anforderungsspezifikation nach Rupp (Rupp *u. a.*, 2014)

Allein die Knotenbezeichnung wird natürlich nicht ausreichen, um alle notwendigen Daten aufzunehmen und D&I zu repräsentieren. Deshalb muss man hier auf die mathematische Erweiterung des „**Multiplexen**“- Netzwerks (vergleiche vorheriges Kapitel) zurückgreifen. Diese Erweiterung erlaubt nun die Aufnahme weiterer Information zu jeder Anforderung. Mögliche Erweiterungen sind:

- ID der Anforderung
- Text der Anforderung nach der Satzbauschablone
- Datum der Erfassung und Gültigkeit
- Dringlichkeit

Mit welchen Informationen nun die Knoten der externen Anforderungen zu belegen sind, ist vom Modellzweck abhängig. Um die DIF für das Absicherungsmanagement zu modellieren, werden wie eingangs beschrieben Schablonen nach Rupp verwendet.

Für die weitere Beschreibung des Produktes in seiner Hierarchie und Detaillierung wird das Weber-Modell herangezogen, weil es die Verifikation darstellen kann und mit dem Top-down-Ansatz des SE zu vereinbaren ist.

Folglich ist den in Interne übersetzten Anforderungen die Spezifikation auf systemebene gegenübergestellt. Analog zu der Nomenklatur nach Weber sind diese vergleichbar mit den „geforderten Systemen“ (PR – required Property) – vergleiche Abbildung 57:

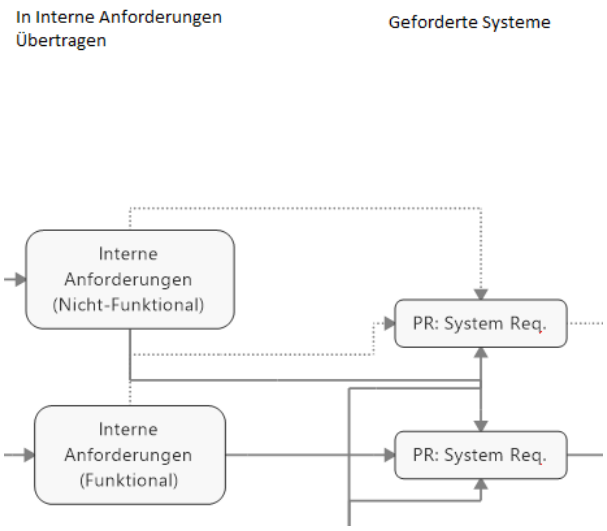


Abbildung 57: Interne Anforderungen überführt in Geforderte Systeme

Die geforderten Systeme, die notwendig sind, um die gestellten Anforderungen zu erfüllen, werden hier in den Knoten auf Ebene H3 definiert:

$$H_3(G) = \left(V_{H_3}, E_{H_3}\right) \quad (10)$$

Mögliche Kanten zu H_3 entstehen damit zwischen:

$$E(G_1) = \{\{v_{J_1}, v_{H_3}\}, \{v_{J_2}, v_{H_3}\}\} \quad (11)$$

Bezogen auf das aufgeführte Beispiel sind das:

$$V_1(H_3) \triangleq \text{Abstandmessendes-System}$$
$$V_2(H_3) \triangleq \text{Abstandregelndes-System}$$
$$V_3(H_3) \triangleq \text{Abstands-einstellendes-System}$$

Der Einfachheit halber verfolgt dieser Bericht nur einen Strang weiter - nämlich die Abstandsmessung.

An dieser Stelle findet der Übergang der funktionalen Anforderung in eine mögliche technische Lösung statt. Dies hat den Vorteil, dass das „reale“ und damit technische System austauschbar ist. Sollte es in der Zukunft eine Innovation geben, kann die technische Lösung ausgetauscht werden. Gleichzeitig ist genau zu verfolgen, welche Systeme einen Beitrag zu welchen geforderten Funktionen liefern.

Als abstandmessendes System wird zum Beispiel ein LiDAR-System ausgewählt. Vergleichbar zum Weber-Modell handelt sich hierbei um eine Eigenschaft (englisch P für Property), die das Produkt besitzen muss.

Durch die Verwendung des Weberansatzes zur Erfassung von $\Delta P = P - PR$ ist es möglich zu messen, wie weit die selbstgestellten Anforderungen an die Teilsysteme erfüllt sind. Dabei handelt es sich um die in Kapitel 2.2.2.3 definierte Verifizierung. Im Gegensatz dazu, ist die Abfrage nach ΔR die Messung, ob die intern angeforderten Systeme die Anforderungen an das Produkt erfüllen. Dieser Vorgang beschreibt die Validierung des Produktes. Beide Werte und das resultierende Delta (ΔP) müssen vom Entwickler selbst erfasst werden. Handelt es sich um genaue Parameter kann das ΔP gegen 0 streben und optimiert werden.

Aus mathematischer Sicht sind folgende Definitionen notwendig:

$$H_4(G) = \left(V_{H_4}, E_{H_4}\right) \quad (12)$$

$$E(G_1) = \{\{v_{H_3}, v_{H_4}\}\} \quad (13)$$

Bezogen auf das aufgeführte Beispiel ist das:

$$V_1(H_4) \triangleq \text{LiDAR}$$

In Weiterem werden nach der Logik des SE Systeme definiert, die die Anforderungen erfüllen sollen. Diese werden in Teil- bzw. Sub-Systemen bis hin zu den einzelnen Komponenten modelliert.

Die Unterscheidung nach Weber in Produkteigenschaften und Merkmale wird hier übernommen, da es die Abfrage nach der Verifikation - hauptsächlich an die Merkmale gerichtet – und nach der Validierung deutlich erleichtert. Mit Hilfe der eigenen Erweiterung, welche durch die Modellierung der externen und internen Anforderungen erfolgt, kann die Frage der nach der Validierung von der Frage nach der Verifikation getrennt werden. Dafür wird der Soll-Ist-Abgleich ΔP_{jB} der geforderten Eigenschaften verwendet und adaptiert – vergleiche Abbildung 58:

B3 Wie im Zyklus A folgt eine Gegenüberstellung der Ist-Eigenschaften ($\mathbf{P_{jB}}$) mit den Anforderungen bzw. Soll-Eigenschaften ($\mathbf{PR_j}$) zur Bestimmung der nunmehr vorliegenden Einzelabweichungen ($\mathbf{\Delta P_{jB}}$). Gegenüber dem vorangegangenen Zyklus haben sich die Soll-Eigenschaften/Anforderungen in der Regel nicht verändert, es gibt aber auf Seiten der Ist-Eigenschaften eine veränderte (hoffentlich verbesserte) Situation, woraus neue (hoffentlich verkleinerte) Werte für die Abweichungen zwischen Soll und Ist resultieren.

B4 Auf Basis der neu bestimmten Einzelabweichungen ($\mathbf{\Delta P_{jB}}$) muss im letzten Schritt des Zyklus B eine neue Gesamtevaluation vorgenommen werden, aus der wiederum die Schlussfolgerungen für den nächsten Zyklus abzuleiten sind (Gesamtevaluation als Prozess-„Treiber").

Abbildung 58: Gegenüberstellung von Ist- und Soll-Eigenschaften des geforderten Systems nach Weber (Weber, 2012)

Entsprechend bezeichnet Weber das ΔP als den Treiber für den PEP, welcher in der Analyse entsprechend gemessen wird – vergleiche Abbildung 59.

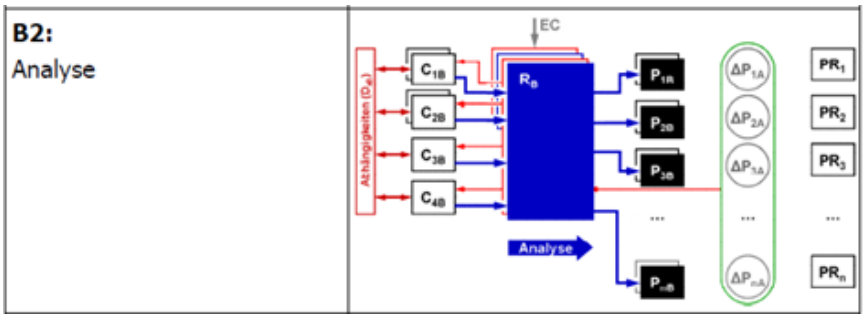


Abbildung 59: Analyse nach Weber mit Delta P als Messgröße (Weber, 2012)

Die Definition nach Weber ist auf der Ebene der Teilgraphen H_3 und H_4 identisch – vergleiche Abbildung 60:

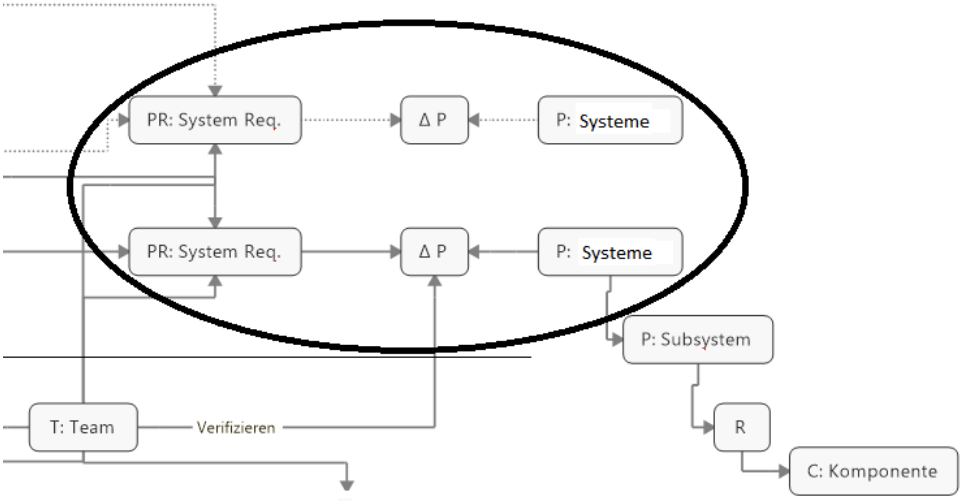


Abbildung 60: Identische Notation zum Weber-Modell im ersten Layer

Ebenfalls an Weber angelehnt, ist die Notation der Teilgraphen L - der Komponenten. Dabei ist nur zu beachten, dass bewusst eine Zwischenebene „Sub-System“ eingebaut wird. Diese Detaillierung kann beliebig weitergeführt werden und hängt von der Produktstruktur, -Hierarchie und -Komplexität ab. Hier wird eine Ebene exemplarisch eingeführt, um zu zeigen, welche Schritte notwendig sind. Gleichzeitig bietet diese n: m – Beziehung die Möglichkeit die Produktstruktur „umzusortieren“. Der Systemarchitekt hat somit die Möglichkeit, die Produktstruktur zu optimieren und anders aufzubauen. Die Knoten der jeweiligen Teilgraphen sind von der Notation nach Weber losgelöst.

Diese Detaillierung stellt eine multi-level-Erweiterung dar, da mathematisch die Menge der Sub-Systeme und später der Komponenten Untermengen darstellen.

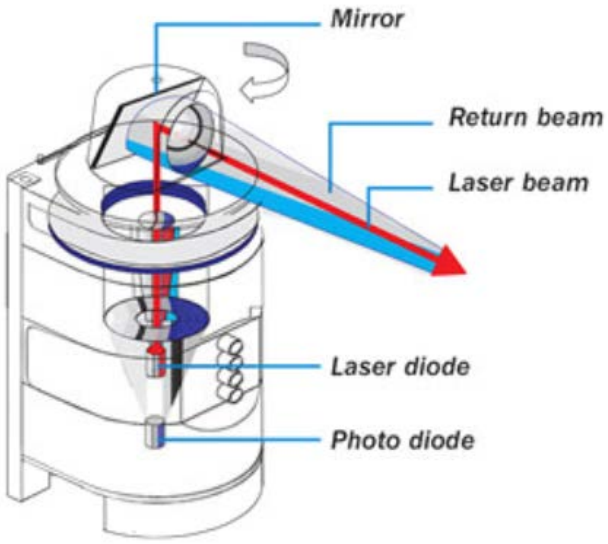
Aus mathematischer Sicht sind folgende Definitionen notwendig:

$$K(H_4) = (V_K\,, E_K) \quad (14)$$

$$L(K) = (V_L\,, E_L) \quad (15)$$

$$E(G_1) = \{\{v_{H_4}, v_K\}, \{v_{H_4}, v_L\}, \{v_K, v_L\}\} \quad (16)$$

Bezogen auf das aufgeführte Beispiel sind $V_1(K)$ bis $V_3(K)$ die entsprechenden Knoten des Sub-Systems K – vergleiche Abbildung 61.



$V_1(H_4) \triangleq$ LiDAR

$V_1(K) \triangleq$ Photo diode

$V_2(K) \triangleq$ Laser diode

$V_3(K) \triangleq$ Mirror

Abbildung 61: Beispielaufbau eines LiDAR nach Richter (Patrick Richter, 2016)

Unterschiede zum Weber-Modell existieren in der Betrachtung von externen Einflüssen. Diese werden im hier gezeigten Modell als externe Anforderungen modelliert, da diese von Anfang an mit den Kundenanforderungen gleichzusetzen sind. Andernfalls werden die äußeren Einflüsse nicht berücksichtig oder zu spät wahrgenommen, was einen steigenden Aufwand zur Folge hätte.

Ein weiterer markanter Unterschied ist die Behandlung des Übergangs der Eigenschaften in die Merkmale. Dieser wird im Weber-Modell in der Analyse als „R“ bzw. in der für diesen Bericht relevanten Situation in der Synthese „R^{-1}“ behandelt – vergleiche Abbildung 62.

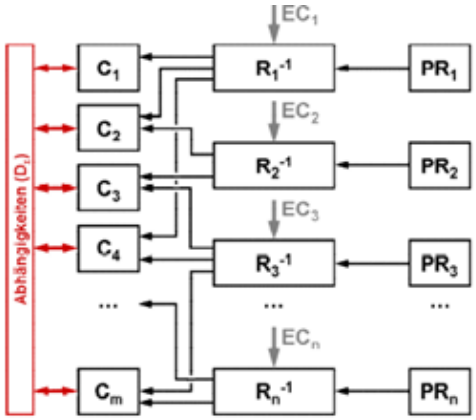


Abbildung 62: Synthese nach Weber (Weber, 2012)

Um die Abhängigkeiten zwischen den Anforderungen, Eigenschaften und Merkmalen nachvollziehen zu können, werden für die Modellierung des Produktes die Ansätze aus dem Systems-Engineering entlehnt und erst ein Übergang von den Subsystemen zu den Komponenten geschaffen. Das heißt statt von PR der Systeme direkt auf die Merkmale C der Komponenten zu überführen, wird vom P der Subsysteme über R auf die Merkmale C der Komponenten übergeleitet.

Dieser Übergang der Eigenschaften der Subsysteme in Merkmale der Komponenten, wird nach Weber -und ebenfalls hier- durch „R“ definiert – vergleiche (16).

Bezogen auf das aufgeführte Beispiel – hier der $V_1(K) \triangleq$ Photo diode - ist das:

$V_1(L) \triangleq$ Merkmale der Photodiode, wie zum Beispiel Bauraum, Material, etc.

$$E(H_4) = \left\{\left\{v_{1_K}, v_{1_L}\right\}\right\} \triangleq R$$

Natürlich handelt es sich bei der Detailierung des Gesamtsystems bis hin zu den Merkmalen der Komponenten nicht um eine starre Modellierung. Es wird hier lediglich ein Vorschlag gemacht, wie der Anwender unter Berücksichtigung seiner Situation die für seine Zwecke richtige Detaillierung vornehmen kann. Die Modellierung der spezifischen Situation soll entsprechend dem gezeigten Vorgehen vorgenommen werden. Dabei ist die Wahl der Anzahl der Stufen von der Komplexität des Produktes abhängig. Allgemein gilt: je höher die Komplexität, desto mehr Stufen sollten modelliert werden. Fest ist lediglich die Unterteilung in Systeme und die Modellierung auf letzter Ebene der Merkmale.

Jeder der hier aufgeführten und definierten Knoten ist natürlich - analog zur Erweiterung der Anforderungen - multiplex-erweiterbar, um die notwendigen Daten und Informationen aufzunehmen.

Zwischenfazit:

Die bisherigen Definitionen ermöglichen die Erstellung eines Netzwerkmodells für das erste Layer, das dem Anwender erlaubt die notwendigen Daten in Knoten und Kanten zu modellieren, welche sich auf die Anforderungen und Struktur des Produktes beziehen. Kongruent können weitere relevante Artefakte wie CAD, Dokumente, etc. modelliert werden.

5.3 Zweites Layer des Daten-Informations-Frameworks: Unternehmensorganisation

Wie bereits eingangs erklärt, beschreibt das zweite Layer die Unternehmensorganisation. Hier ist es essenziell festzuhalten, welcher Mitarbeiter für welches System, Teilsystem bzw. welche Komponente verantwortlich ist. Demgegenüber kann die Verantwortung auch über eine bestimmte Rolle definiert werden. Da es in der Regel keine 1:1- Beziehung zwischen Rolle und Mitarbeiter gibt, ist es ratsam beides zu führen. Die genaue Definition und Ausgestaltung des zweiten Layers bleibt bewusst offen, damit dieses Framework auf die eigene Unternehmenssituation angepasst werden kann. Natürlich muss nicht das gesamte Unternehmen mit all seinen Mitarbeitern modelliert werden. Es muss lediglich mit den notwendigen Informationen befüllt werden, die für den Modellzweck unabdinglich sind. Bezogen auf die Eigenschaftsabsicherung sind das zum Beispiel folgende Rollen:

- System-Verantwortlicher: trägt die technische Verantwortung des Systems
- Tester: Führt die Eigenschaftsabsicherung durch (SIL, HIL)
- Test-Manager: Koordiniert die verschiedenen Tests und überwacht die Testergebnisse
- Konfigurations-Manger: Stellt sicher, dass die richtige Version des Systems getestet wird und verwaltet die verschiedenen Versionen

Auf Mitarbeiter-Ebene sind es beispielsweise folgende Personen:

$$V_1(M_1) \triangleq \text{Mitarbeiter „Max Mustermann“}$$

$$V_2(M_1) \triangleq \text{Mitarbeiter „Hans Sonnenschein“}$$

Die Rollen sind, wie folgt zu beschreiben

$$V_1(M_2) \triangleq \text{Rolle des System-Verantwortlichen}$$

$$V_2(M_2) \triangleq \text{Rolle des Testers}$$

$$V_3(M_2) \triangleq \text{Rolle des Konfigurations-Verantwortlichen}$$

Dabei kann beispielsweise Hans Günther sowohl die Rolle $V_2(M_2)$ als auch $V_3(M_2)$ innehaben.

Mathematisch ist zusätzlich folgende Definition bzgl. der Kanten notwendig:

$$E(G_2) = \left\{\left\{v_{M_1}, v_{M_2}\right\}\right\} \triangleq der\ Rollenzuordnung$$

Zwischenfazit:

Die hier getroffenen Definitionen bleiben bewusst offen, erlauben jedoch die Aufnahme der Unternehmenssituation in Form von Mitarbeitern und Rollen. Auch hier lassen sich die Definitionen der Multi-level-Erweiterung - äquivalent zur Detaillierung des Systems – anwenden, um die Unternehmenshierarchie abzubilden.

5.4 Drittes Layer des Daten-Informations-Frameworks: Workflows

Auf dem dritten Layer sind die Informationen über die Workflows verankert. Dabei werden mehrere Aufgaben zu einem Workflow zusammengefasst. Auch hier ist die Definition bewusst schlank gehalten, damit der Anwender das Framework mit Bezug auf den Zweck befüllen und ausgestalten kann. Es wird lediglich anhand des eingeführten Beispiels gezeigt, wie ein solcher Workflow erstellt wird und welche mathematischen Definitionen dafür notwendig sind.

Die Eigenschaftsabsicherung könnte folgende Aufgaben enthalten. Dabei ist der Prozess als SIL oder HIL frei zu gestalten. Abbildung 63 zeigt einen Test-Prozess, welcher in der Swimlane-Darstellung die Aktivitäten aufzeigt und entsprechend der Rollen gesplittet ist.

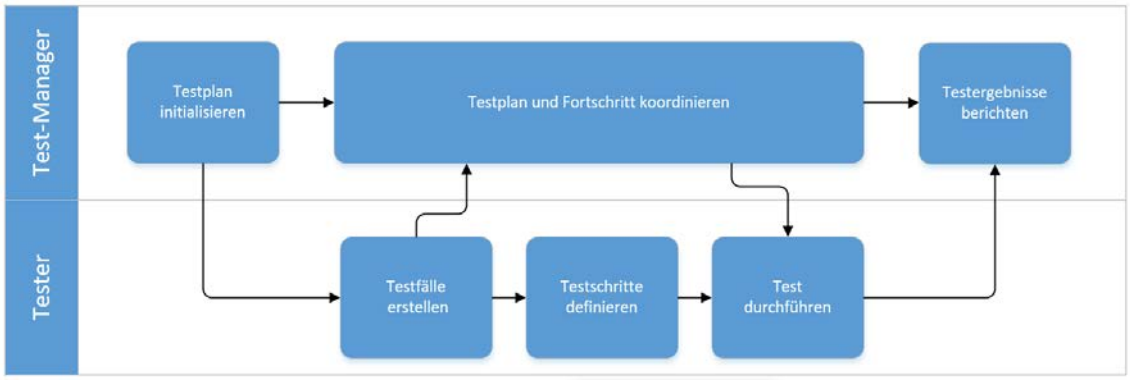


Abbildung 63: Beispiel eines Test-Prozesses mit entsprechenden Rollen

Der Workflow des Testers zeigt zum Beispiel auf, dass verschiede Aktivitäten zusammengefasst werden können und aus der Sicht der Rolle wieder beliebig detaillierbar sind.

Mathematisch sind entsprechend folgende Definitionen notwendig:

$$N(G_3) = (V_N\ , E_N) \quad (17)$$

$$O(N) = (V_O\ , E_O) \quad (18)$$

$$E(G_3) = \{\{v_{G_3}, v_N\}, \{v_N, v_O\}\} \quad (19)$$

Zwischenfazit:

Analog zu den Definitionen des zweiten Layers, erlauben die hier gezeigten Beispiele die Aufnahme von situationsabhängigen Prozessen und eine beliebige Detaillierung. Ebenso lässt sich hier die Definitionen der Multi-level-Erweiterung anwenden.

5.5 Layer-übergreifende Verknüpfungen

Um nun die Artefakte des ersten Layers unter die Verantwortung einer Rolle bzw. eines Mitarbeiters zu setzen oder einer Rolle einen Workflow zuzuordnen, sind noch layerübergreifende Verknüpfungen notwendig.

Sämtliche Aspekte, angefangen bei den Anforderungen, die unternehmensintern zu verantworten sind, bis hin zu allen Artefakten jeglicher Ebenen (System-level, Sub-System, Komponenten, etc.), müssen durch eine Rolle – besser noch durch eine Person – abgedeckt sein. Deshalb muss es möglich sein, jeden Knoten aus dem ersten Layer mit einer oder mehreren Rollen/ Personen und damit entsprechenden Knoten zu verknüpfen:

$$E(G) = \{\{v_{G_1}, v_{G_2}\}\} \quad (20)$$

Die Definition bezieht sich dabei auf jeden Knoten der Untermenge von G_1 und G_2.

Bei der Zuordnung einer Rolle zu einem Workflow, ist die Verknüpfung und damit auch die Anzahl der möglichen Kanten zwischen G_2 zu G_3 eingeschränkter. Hier sollte eine Person bzw. ein Mitarbeiter nicht auf einen Workflow verweisen, da dieser durch die Verknüpfung (Einteilung) zwischen Rolle und Mitarbeiter bereits eindeutig getroffen ist. Somit ist eine Person über die Rolle für einen Workflow verantwortlich – jedoch nicht über eine Kante direkt damit verknüpft, um die Eindeutigkeit zu bewahren. Somit ergibt sich lediglich die Verknüpfung von:

$$E(G) = \{\{v_{M_2}, v_N\}\} \quad (21)$$

Dem Anwender des Frameworks ist an dieser Stelle die Freiheit gegeben, selbst zu entscheiden, ob die getroffene Einschränkung genügt. Man kann sich auch für das Gegenteil entscheiden und entsprechend den Workflow mit dem Mitarbeiter direkt verknüpfen. Auf jeden Fall sollte man sich aus Gründen der Eindeutigkeit für eine der beiden Möglichkeiten entscheiden. Die Praxis zeigt jedoch, dass Mitarbeiter häufiger wechseln und die Rolle eher langsamer verändert wird. Deshalb sollte die Rate der Änderung in Betracht gezogen werden und entsprechend die Option gewählt werden, welche stabiler ist.

Die dritte und letzte layerübergreifend mögliche Verbindung ist die Verknüpfung von Knoten des ersten Layers mit Knoten des dritten Layers. Auch hier würde es sich um eine indirekte Verknüpfung handeln, wenn der Workflow über eine Rolle mit einem Artefakt verbunden ist. Folglich ist es nicht notwendig ein Artefakt direkt mit einem Workflow oder einer Task zu verbinden. Sollte es dennoch notwendig sein, muss sichergestellt sein, dass die Information dieser Kante nicht bereits abgedeckt ist. Andernfalls modellieren zwei verschiedene Wege (Kanten) den gleichen DIF, was es zu verhindern gilt.

5.6 Datenbeschaffung für die Anwendung Drei-Layer-Daten-Informations-Frameworks

In den vorangegangenen Kapiteln wurde das Framework generisch erläutert. Allerdings ist es für die Anwendung des Ansatzes notwendig zu wissen, wo die Daten- und Informationsgrundlagen zu finden sind und wie diese sichtbar werden. Zu beachten ist, dass die DIF unternehmensspezifisch sind und diese unter anderem von

- der Unternehmensstrategie allgemeinen
- der Entwicklungsaufgabe
- der Organisation der Entwicklung
- dem spezifischen Variantenmanagement
- der IT-Landschaft und Tools

abhängig sind.

Grundsätzlich ergeben sind vier Quellen, die sich gut als Datengrundlage für das vorgestellte Framework eignen:

- Textuelle Anforderungen
- Modelle zur Struktur, Funktion und Verhalten
- Tool-Landschaft und PDM-Systeme
- Unternehmensdaten zur operationellen und prozessualen Aufstellung

Textuelle Anforderungen:

Wie im Grundlagenkapitel bereits ausführlich besprochen, sind Anforderungen meistens in textueller Form festgehalten, da sie keiner speziellen Modellierung-Syntax folgen müssen. Sind die Anforderungen frei formuliert, müssen die relevanten Informationen, die für das Framework notwendig sind, explizit extrahiert werden. Das könnte ein Auszug aus einem Tool sein – hier IBM Rational Doors Abbildung 64:

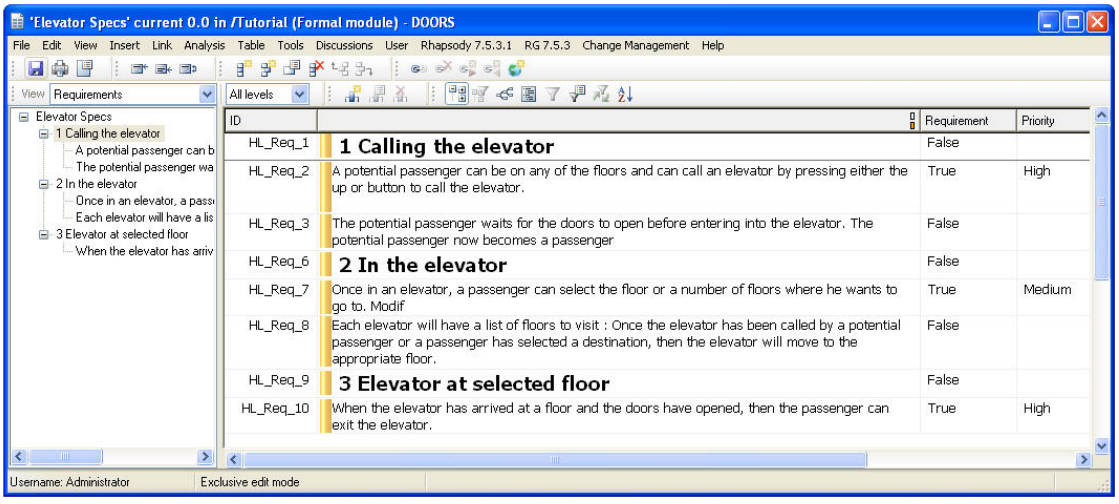


Abbildung 64: Auszug von textuellen Anforderungen aus DOORS (IBM Knwledge Center, 2020)

Mit solch einer Datengrundlage können wichtige Daten für das erste Layer des Frameworks gewonnen werden. So lassen sich aus dem gezeigten Beispiel nicht nur der textuelle Anforderungstext entnehmen, sondern darüber hinaus weitere Informationen aus den Attributen. Hier sieht man die ID, Dringlichkeit, den Status etc. der Anforderung, welche

in den Knotendaten verankert werden können. Die Verlinkung der jeweiligen Anforderungen lassen sich meistens aus der Ordner- oder Kapitelstruktur herleiten. Es steht jedoch dem Entwickler frei, diese in die vorgeschlagene Framework-struktur zu übertragen.

Es fällt leichter relevante Informationen zu extrahieren, wenn die Anforderungen nach den Satzbauschablonen nach Rupp aufgestellt sind. Damit lassen sich Informationen für Level 1 und evtl. sogar für Level 2 ableiten. Hilfreich ist, dass die Satzbauschablone bereits immer häufiger in Unternehmen der Automobilbranche Verwendung findet.

Da in den meisten Fällen in den Anforderungsdokumenten aufgelistet ist, wer für die jeweilige Anforderung verantwortlich ist, kann auch diese Information verwendet werden, um die Verknüpfung von Layer 1 und Layer 2 zu erstellen.

Vielfältiger ist es, Informationen aus Modellen zu extrahieren. Exemplarisch werden hier einige gängige Modelle aufgezeigt, welche Informationen liefern –vergleiche Abbildung 65:

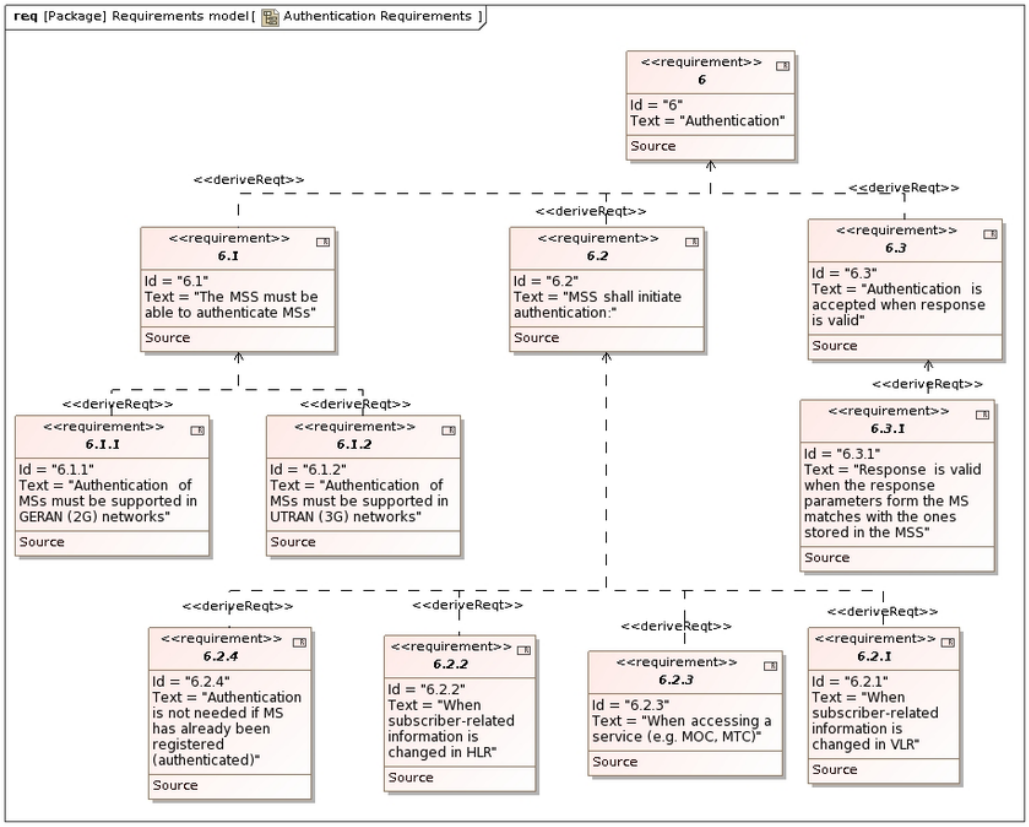


Abbildung 65: Beispiel eines Requirements-Diagramms aus SysMl (Dragos Truscan, 2019)

Die notwendigen Daten aus beispielsweise dem hier gezeigten Requirements-Diagramm (SysMl) zu entnehmen gestaltet sich noch einfacher, da die Relationen der Anforderungen zueinander 1:1 übernommen werden können. Ist die Struktur ähnlich zu dem Framework,

kann das Diagramm automatisch überführt werden. Denkbar sind auch automatisierte Übersetzungen, die einer Regel folgen und die Daten auf das Framework verteilen.

Systemstruktur:

Tools werden von den Entwicklern für jede erdenkliche Aufgabe benutzt. Es beginnt mit den einfachsten Tools wie zum Beispiel MS Office-Tools und endet mit sehr spezifischen Einzellösungen wie zum Beispiel Altair Inspire für Topologieoptimierung. Deshalb werden hier zwei gängige und zeitgleich sehr anpassbare Tools herausgegriffen, an denen erläutert wird, wie die Produktinformation zu extrahieren ist, um das erste Layer mit den notwendigen Daten zu „realisiertes System“ bzw. Sub-Systemen oder mit Komponenten zu befüllen. Da es nicht mehr zeitgemäß ist, dass Produktstrukturen in Papierform gehandhabt werden, muss der Entwickler hier nur das jeweilige Tool identifizieren, in dem die notwendige Produktarchitektur aufgeführt ist. Auch hier bietet SysMl einige Diagramme, die sich dafür eignen, die Struktur eines Systems bzw. Teilsystems zu modellieren. In Folgendem ist ein solches Beispiel aufgeführt, das erkennen lässt, wie das System – hier Vehicle – in Subsysteme zerlegt ist und welche Relation zwischen den Subsystemen herrscht. Vergleichbar zum Requriements-Diagram können die Subsysteme, ihre Hierarchie und Abhängigkeit untereinander, in das Framework übertragen werden –vergleiche Abbildung 66:

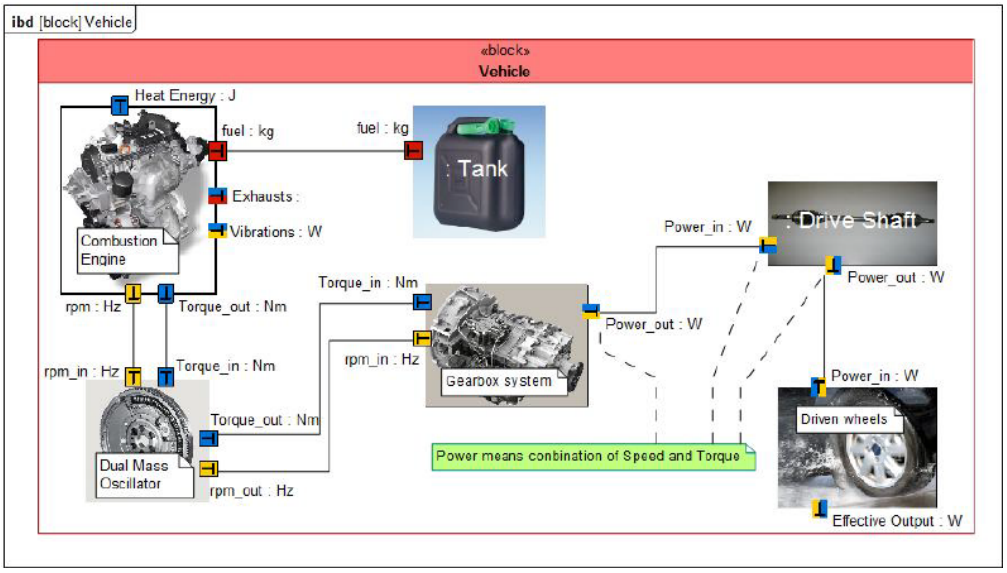


Abbildung 66: Beispiel eines Internal-Block-Definition Diagramms aus SysMl (Christian Zingel, 2020)

Die Anwendung des Frameworks ist nicht für eine spezielle Produktdetaillierung definiert. Vielmehr lässt sich das Framework auf die gewünschte Detailebene anpassen und mit entsprechenden Daten befüllen. So kann man die Produktstruktur auch aus einen

CAD oder PDM-System entnehmen und damit das Framework befüllen. Wie im oberen Beispiel kann hier die Stückliste als Grundlage für das System, Teilsystem oder für die Komponenten dienen, welche durch die Produkthierarchie zueinander in Relation gesetzt werden können – vergleiche Abbildung 67:

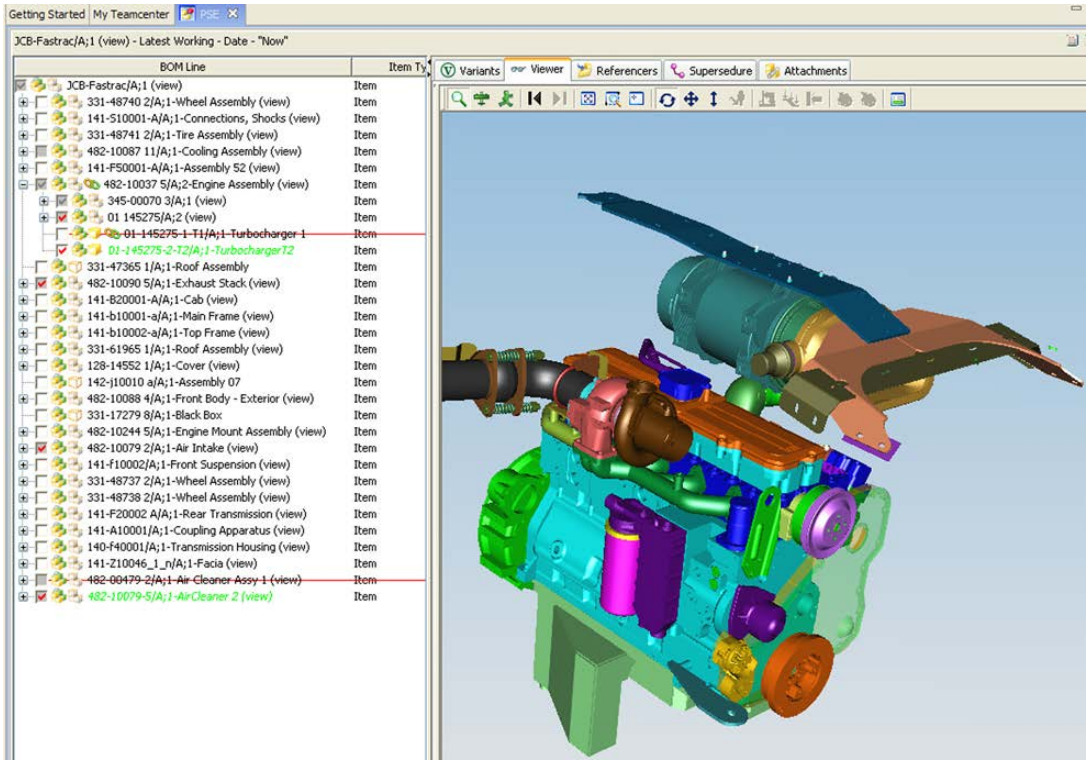


Abbildung 67: Auszug aus einem PDM-System - hier Stückliste eines Motors (Siemens, 2019)

Unternehmensdaten:

Daten für das zweite Layer sind am einfachsten aus den Daten des Human Regulation zu entnehmen. Erdenklich ist auch die Analyse des Email-Verkehrs, usw. Allerdings sind hier die Möglichkeiten der Datenerhebung durch die nationalen und internationalen Datenschutzverordnungen geregelt. In diesem Fall liegt die Schwierigkeit darin, nicht die relevanten Daten zu „finden“, sondern in den begrenzten Möglichkeiten Daten zu erheben. Einfachste Informationen zum Sachbearbeiter oder Verantwortlichen sind jedoch auch in den Anforderungstools zu finden. Alles was darüber hinaus notwendig ist, muss entweder manuell erhoben werden oder entsprechend freigegeben werden – vergleiche Abbildung 68.

Personen (Datenschutzmodus aktiviert)

Interne Personennummer : *

Suchen | Zurücksetzen | Neu | Bearbeiten | Gutscheine kaufen | Import

Bild	Nachname	Vorname	Personalnummer	Benutzerkennung	Organisationseinheit	Land für gesetzliche Regelungen	Währung
	K*****	K*****	0*****	k*****	Marketing	DEU	EUR
	K*****	N*****	0*****	k*****	Bestandskunden	DEU	EUR
	L*****	M*****	0*****	m*****	Neukunden	DEU	EUR
	L*****	J*****	1*****	j*****	Marketing	DEU	EUR
	M*****	N*****	0*****	n*****	Entwicklung	DEU	EUR

Abbildung 68: Auszug von Personaldaten (HRworks, 2018)

Workflows/ Tasks:

Betroffene Prozesse können entweder festgelegt oder entwickelt werden, um diese in Layer 3 zu übertragen. Da meistens jedoch in einem größeren Unternehmen bestimmte Prozesse in einer Prozesslandkarte existieren, muss nur der zu betrachtende Prozess identifiziert und in das Netzwerk übertragen werden. Die relevante Detailtiefe wird auch hier über den Zweck der Erhebung definiert. Gängige Quellen sind unter anderem:

- Verfahrensanweisungen
- Meilenstein-Planungen
- Absicherungsprozesse

Fazit:

Die Daten und Informationen, welche notwendig sind, um das Framework zu befüllen und um Analysen durchzuführen, existieren zwar in den unterschiedlichsten Formen, sind jedoch meistens vorhanden. In der heutigen Zeit stellt sich aufgrund der aktiven Datensammlung nicht die Frage, ob die Daten vorhanden sind, sondern, wo und in welcher Form die Daten und Informationen zu entnehmen sind. Hat man die relevanten Daten extrahiert, kann ein entsprechender Mehrwert daraus gewonnen werden, diese auszuwerten.

5.7 Möglichkeiten der Umsetzung des Drei-Layer-Daten-Informations-Frameworks

Nachdem nun das Framework, die Möglichkeiten und die Datenbeschaffung gezeigt wurden, kann in diesem Kapitel analysiert werden, wie die technische Umsetzung aussehen kann. Grundsätzlich ergeben sich drei Möglichkeiten, das Framework in einem Unternehmen anzusiedeln. Natürlich hängt dies von der Unternehmenssituation und dem

Zweck ab. Deshalb wird hier kein Beispiel ausgearbeitet, sondern allgemeiner plausibilisiert, welche Möglichkeiten es gibt und die jeweiligen Vor- und Nachteile besprochen. Somit kann ein Anwender diese Analyse auf die eigene Situation adaptieren und entsprechend wählen.

Darüber hinaus ist es ratsam die Tool-Landschaft in einer Plattform zu vereinen. Denn Insellösungen beziehen sich meistens auf eine Anwendung und erfüllen einen spezifischen Zweck. Um jedoch ganzheitlich die DIF zu erfassen und zu analysieren, bedarf es der Sammlung und Zugänglichkeit der DIF. Eine Plattform impliziert, dass Tools miteinander kommunizieren können und über entsprechende Schnittstellen zueinander in Relation stehen. In diesem Sinne ergeben sich zwei sehr wichtige Aspekte – die Versionierung von Elementen und die Variantenbildung von Komponenten bzw. Systemen.

Dabei ist die Versionierung von der zeitlichen Weiterentwicklung des Produktes/ Elementes abhängig und beschreibt den Stand zu einem bestimmten Zeitpunkt – vergleiche Kapitel 2.2.2.2. Das heißt die Daten (Anforderungen, Produktspezifikation, Unternehmenssituation, usw.), die in den Tools fortlaufend bearbeitet werden, sind als „Stände“ bzw. „Versionen“ zu bestimmten Intervallen „gefreezt“, müssen übergreifend zusammengeführt werden und bilden somit eine Konfiguration.

Die Variante eines Produktes oder Systems beschreibt ein „ähnliches“ Produkt, das zeitgleich parallel existiert und eine große Schnittmenge an Gemeinsamkeiten aufweist. Das heißt zwei Produkte oder Systeme können eine bestimmte Menge an Komponenten aufweisen, die identisch sind, wobei sich jedoch die jeweilige Funktion unterscheiden darf. Ein solches Beispiel ist die ca. 80%- Überdeckung von Mercedes-Benz C- und E-Klasse.

Stehen nun mehrere Tools in Abhängigkeit zueinander, muss sichergestellt werden, dass die Versionierung und das Varianten-Management toolübergreifend gewährleistet ist. Dies wird durch eine Plattform realisiert, die die jeweiligen Tools beinhaltet – vergleiche IMB Jaz-Platform, Siemens Teamcenter, etc. Für das Framework ist es essenziell, dass die Tools über die jeweiligen Schnittstellen verfügen und somit die Abhängigkeiten vorhanden sind. Deshalb wird davon ausgegangen, dass die Tools in einer Plattform verknüpft sind. Ist das situationsbedingt nicht der Fall, müssen entsprechende Workarounds gegangen oder auf Teile der Frameworks verzichtet werden, womit Teile der DIF nicht sichtbar sind.

Die **erste Möglichkeit** ist, das Framework extern zu einer bestehenden Tool-Landschaft zu betreiben. Das heißt das Framework kann zum Beispiel in einem rein mathematischen Tool aufgestellt sein, muss jedoch mit den Daten aus der Tool-Landschaft bzw. Plattform befüllt werden. Natürlich ist eine manuelle Befüllung möglich, jedoch nicht mehr zeitgemäß. Die Schnittstelle zu den Tools muss automatisiert und entsprechend implementiert werden. Nur dann hat das Framework die notwendige Fülle an DIF darstellen, um eine

aussagekräftige Analyse zu liefern. Diese muss interpretiert und in den PEP zurückgespielt werden.

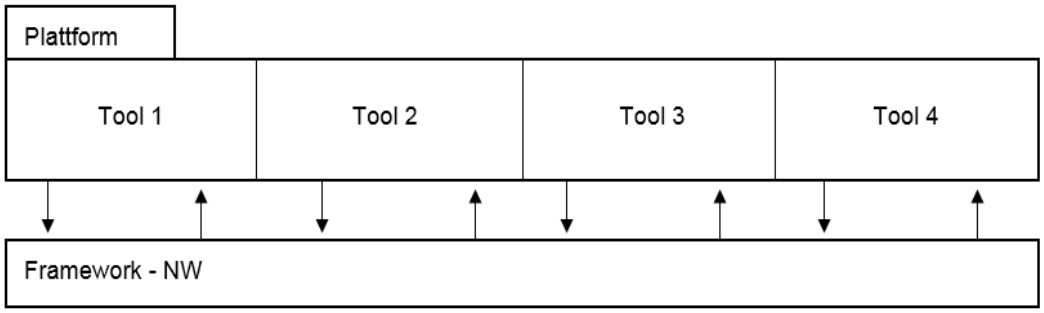


Abbildung 69: Erste Möglichkeit der Umsetzung des Frameworks - hier externe Durchführung

Eine solche Umsetzung gestaltet sich als sehr einfach, da die Tool-Landschaft unberührt bleibt. Das Framework wird extern implementiert und über die Schnittstellen mit relevanten Daten gespeist. Allerdings muss berücksichtig werden, dass die Daten zu einem bestimmten Zeitpunkt überspielt werden und sich die Analyse bzw. Aussage des Frameworks auf diesen Zeitpunkt bezieht. Die fortlaufende Analyse gestaltet sich schwierig, weil die Konfiguration der Plattform nicht mit der des Frameworks übereinstimmt.

Zwischenfazit:

Das Framework aus der Plattform auszulagern, hat folgende Vor- und Nachteile:

+ die Umsetzung ist im Verhältnis zum Aufwand einfach
+ nur relevante Daten müssen extrahieren werden
- keine Konfiguration der Plattform: nur statische Analyse

Die **zweite Möglichkeit** das Framework umzusetzen ist, es in einem Tool zu integrieren. So lässt sich zum Beispiel das Framework im Hintergrund zu Doors o.Ä. aufsetzen, das mit den Daten des jeweiligen Tools gespeist wird. Somit kann gleichzeitig die mathematische Analyse im Hintergrund laufen und dem Entwickler die Interpretation in Form von „Hinweisen“ ermöglichen.

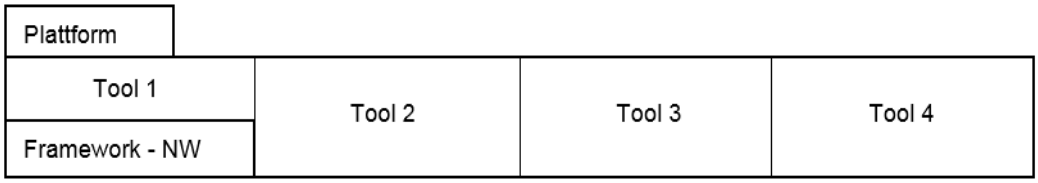


Abbildung 70: Zweite Möglichkeit der Umsetzung des Frameworks - hier Tool-interne Durchführung

Diese Lösung ist in der Einführungsphase aufwändiger, kann jedoch fortlaufend mit Daten befüllt werden und hat eine Echtzeit-Interpretation zur Folge. Nachteilig ist, dass das Framework auf die Daten und Informationen des Tools begrenzt ist, womit nicht das gesamte Framework befüllt werden kann.

Zwischenfazit:

Das Framework in einem Tool anzusetzen, hat folgende Vor- und Nachteile:

+ speziell angepasst für ein Tool und Anwendung

- nur innerhalb des Tool-Kontextes anwendbar

- nicht alle Daten für das Framework vorhanden

Die **dritte Möglichkeit** das Framework umzusetzen ist, an der Plattform jedem Tool den jeweiligen Anteil des Frameworks aufzusetzen. Damit wird das gesamte Framework abgedeckt und auf die entsprechenden Tools der Plattform verteilt. Abgesehen von der Verteilung ist diese Variante analog zur zweiten Möglichkeit. Die Umsetzung entspricht der vorangegangenen Darstellung.

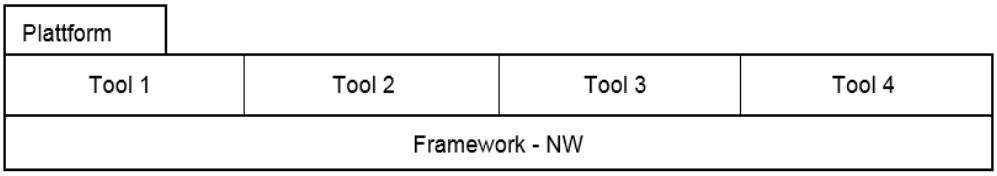


Abbildung 71: Dritte Möglichkeit der Umsetzung des Frameworks - hier Plattform-integrierte Durchführung

Diese Lösung erlaubt es, das Framework mit allen notwendigen Analysen zu betreiben, die DIF parallel und aktuell aufzuzeigen und zu analysieren. Dementsprechend ist jedoch der Aufwand diese Variante umzusetzen höher.

Zwischenfazit:

Das Framework übergreifend auf der Plattform aufzusetzen, hat folgende Vor- und Nachteile:

+ Toolübergreifend

+ alle Daten für Framework auffindbar

+ Alle Daten konfiguriert

- sehr aufwendig

Gesamtfazit:

In diesem Kapitel wurden drei verschiedene Umsetzungsstrategien aufgelistet und bewertet. Je nach Zweck und der subjektiven Situation muss entschieden werden, welche der Varianten am vielversprechendsten ist. In Summe lässt sich folgende Aussage treffen:

- Erste Möglichkeit anzuwenden bei schnellen Analysen
- Zweite Möglichkeit anzuwenden bei einem bestimmten Zweck und für ein Bereich des Frameworks
- Ditte Möglichkeit anzuwenden zur Analyse eines ganzheitlichen DIF

6 Anwendung des Frameworks am Beispiel des Absicherungs-Managements

Für die Beantwortung der Forschungsfrage wurde erläutert, welches Framework die Möglichkeit bietet die substanziellen D&I aufzunehmen. Die vorangegangenen Kapitel haben ausgeführt, welche Mittel dafür notwendig sind, welche Ansätze dabei verwendet werden sollten, dass an einigen Stellen eine Erweiterung vorgenommen werden muss und wo die D&I bzw. Artefakte üblicherweise zu finden sind.

In diesem abschließenden Kapitel wird plausibilisiert, wie das Framework verwendet werden kann, um die DIF ganzheitlich aufzudecken, womit die Komplexität auf ein handhabbares Niveau reduziert wird und welche Optimierungen möglich werden.

Am Beispiel der Eigenschaftsabsicherung wird deshalb erläutert, welche Netzwerkanalysen möglich sind, wie diese zur Komplexitätsbewältigung beitragen können und welche Prozessoptimierungen sie erlauben. Es wird vorausgesetzt, dass die Anwendung der Graphentheorie für ähnliche Gebiete, wie das des Qualitäts- oder Risikomanagements, äquivalent auszuführen ist. Da es jedoch den Rahmen dieser Arbeit sprengen würde, für alle Anwendungen genau zu erläutern, wie das Framework abzuwandeln ist, werden für die Eigenschaftsabsicherung Kennzahlen erst neutral und im Anschluss erläutert.

6.1 Identifikation von Daten- und Informationsflüssen

Um die Komplexität zu reduzieren ist es eine Möglichkeit, die Reduzierung von Elementen – hier Informationen – auf ein notwendiges Minimum vorzunehmen. Dies geschieht in Folgendem durch die Darstellung bzw. der Suche nach bestimmten D&I. Im Beispiel der Eigenschaftsabsicherung liegt der Fokus auf den Anforderungen, der Spezifikation, den bearbeitenden Mitarbeitern, den Rollen und der entsprechenden Verifikation bzw. Validierung.

6.1.1 Darstellung von Daten- und Informationsflüssen innerhalb eines Layers

Das Framework ist in der Lage die D&I ganzheitlich aufzunehmen und ist beliebig erweiterbar. Allerdings muss es möglich sein, „nur“ relevante DIF zu betrachten. Dies ist die einfachste und nun fast triviale Analyse -nämlich die Darstellung eines Traces bzw. Weges. Allerdings ist das nur möglich, weil die jeweiligen Elemente des PEPs miteinan-

der verknüpft und um die jeweilige Information angereichert sind. So lässt sich zum Beispiel im Anwendungsfall der Eigenschaftsabsicherung darstellen, welche Anforderungen in welcher Komponente erfüllt werden – vergleiche Abbildung 72:

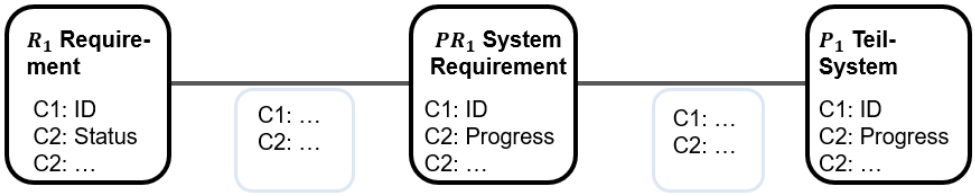


Abbildung 72: Trace einer Anforderung zum erfüllenden Teilsystem

Durch die mulitiplexe Erweiterung der Knoteninformation ist das Framework in der Lage, die notwendigen Daten und Informationen aufzunehmen. So sind nicht nur die Knoten miteinander verknüpft, sondern um elementrare Informationen erweitert –vergleiche. Trace in Abbildung 73:

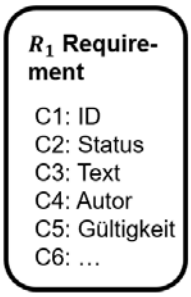


Abbildung 73: Multiplexe Erweiterung eines Anforderungs-Knotens

Somit ist es dem Anwender möglich nach essenziellen Folgen zu suchen und diese entsprechend zu identifizieren und zu bewerten.

6.1.2 Layer-übergreifende Darstellung von Daten- und Informationsflüsse

Natürlich beschränkt sich die Identifikation und Darstellung eines Traces nicht nur auf ein Layer. Durch die getroffenen Definitionen sind layerübergreifende Traces möglich und folglich darstellbar. Dies könnte beispielsweise die Information sein, welche Anforderung, von welchem Mitarbeiter (MA) in einem bestimmten Workflow nachgewiesen wird. Ein solcher DIF zieht sich über die drei Layer hinweg und könnte folgende Information enthalten – vergleiche Abbildung 74.

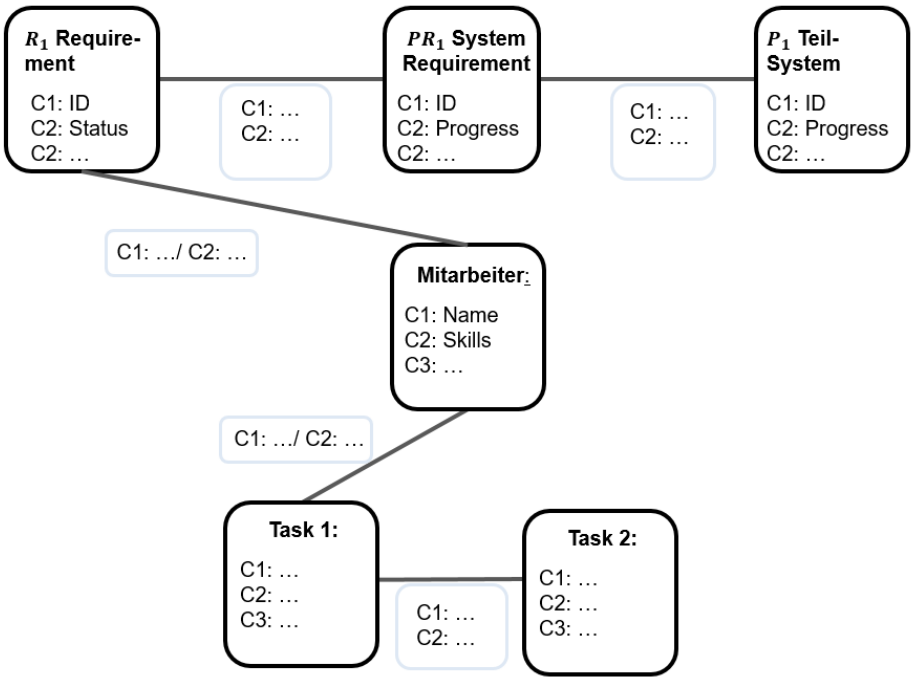


Abbildung 74: Ebenen-übergreifender Trace im Drei-Layer-Daten-Informations-Frameworks

Natürlich wird es nicht nur lineare Traces geben. Die D&I werden unterschiedlich verzweigt sein. Allerdings ist auch hier möglich bestimmte Inhalte auszublenden und nur DIF in Abhängigkeit von dem intendierten Zweck zu betrachten.

6.1.3 Filterung der Daten- und Informationsflüsse nach bestimmten Inhalten

Da das Framework hierarchisch aufgestellt ist und die Knoten- bzw. Kanten-Informationen multiplex erweitert sind, kann der Anwender nicht nur bestimmte Traces begutachten, sondern einerseits die jeweilige Untermenge eines Graphen eine Stufe tiefer betrachten (beispielsweise vom System in den jeweiligen Subs-Systemen eintauchen) und andererseits die multiplexen Informationen „ausklappen“ bzw. verbergen.

Das heißt die Hierarchie hilft dem Anwender auf eine höhere Ebene zu abstrahieren und gleichzeitig in die jeweiligen notwendigen Teilgraphen zu „zoomen“. Diese Funktion ist für die Reduzierung der Komplexität wesentlich – vergleich Abbildung 75.

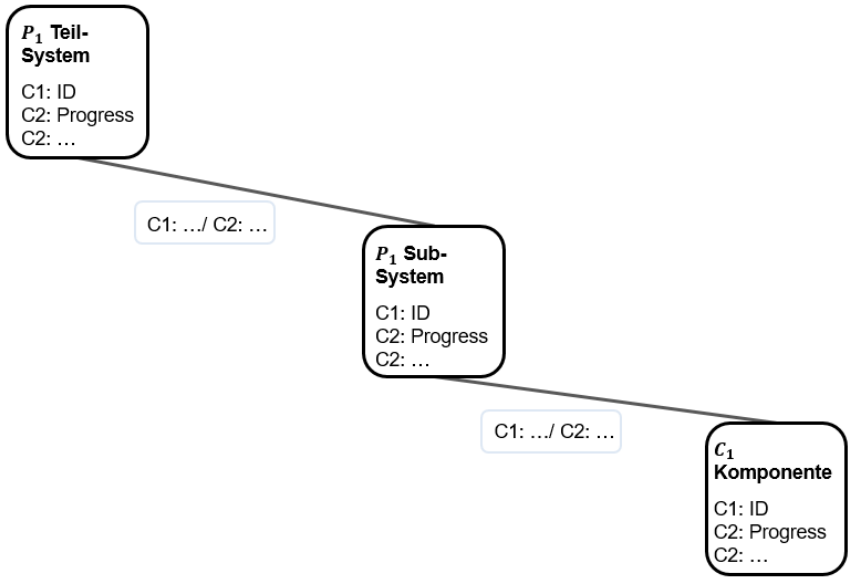


Abbildung 75: System-Hierarchie eines Teilsystems

Analog kann nach bestimmten Attributen von Knoten oder nach einem Strang gesucht werden, welcher von einem bestimmten Knoten mit entsprechenden Attributen ausgeht.

Damit können zum Beispiel Anforderungen relevant sein, die bestimmte Informationen aufweisen – zum Beispiel Anforderungen, die mit bestimmten Sicherheitsaspekten belegt sind – vergleiche Abbildung 76:

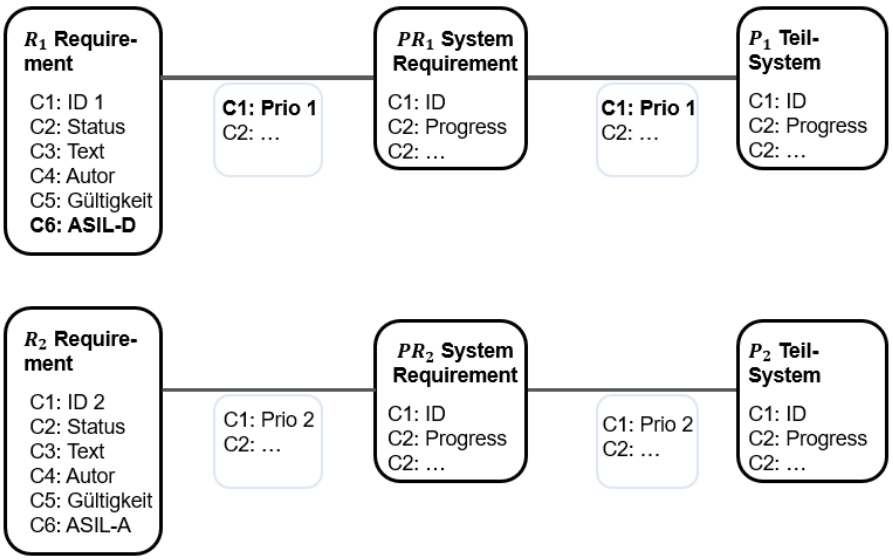


Abbildung 76: Zwei Anforderungs-Traces mit verschiedenen Sicherheits-Leveln (ASIL-Level)

Zudem erlaubt diese Form der Selektion auch die Kombination von Selektionen bzw. das Ausschließen von Knoten/ Kanten mit bestimmten Attributen. Konkret kann man damit

zum Beispiel nicht nur nach Anforderungen mit einem kritischen Sicherheitslevel suchen, sondern in einer Teilmenge (Teilgraph) nach Anforderungen suchen, die beispielsweise folgende Attribute aufweisen:

- ASIL-Level: höher A
- Status: rot
- Dinglichkeit: muss-Anforderungen
- Quelle: Kunde
- etc.

Somit kann die Anzahl der D&I auf genau den notwendigen Inhalt und somit die Komplexität reduziert werden.

6.1.4 Identifikation von indirekten Abhängigkeiten

Da das Framework die möglichen Verknüpfungen der Knoten miteinander vorgibt, sind die möglichen Abhängigkeiten bekannt. Es wurde zudem an manchen Stellen zwar ausgeschlossen, den identischen DIF über zwei Wege zu modellieren, allerdings kann es immer noch zu indirekten Abhängigkeiten kommen. Dies kommt vor, wenn – vergleiche Abbildung 77- eine Kette wie A zu B zu C existiert, welche eine indirekte Abhängigkeit zwischen A und C darstellt.

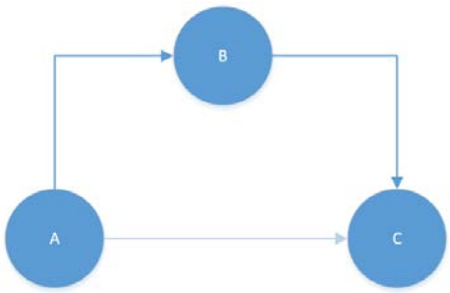


Abbildung 77: Beispiel indirekter Abhängigkeit zwischen Knoten A und C

Durch die Überführung der DIF in neutrale Knoten und Kanten im Framework sind solche Pfade identifizierbar und eindeutig zu erkennen. Für die Suche nach solchen Pfaden gibt es in der Graphen- und Netzwerktheorie verschiede Algorithmen (Turau, 2009). Die Auswahl dieser, wird in Abhängigkeit von dem zu erwartenden Aufwand getroffen. Da jedoch die gängigen Tools im Allgemeinen diese Funktion aufweisen, wird an dieser Stelle auf die Fachliteratur verwiesen, die nur in Frage kommt, wenn die Berechnungszeit eine wichtige Rolle spielt (Turau, 2009).

6.1.5 Suche nach bestimmten Mustern in den Daten- und Informationsflüssen

Die erste Auswertung in Richtung der strukturellen Gegebenheit des mathematischen Netzwerkes wurde anhand der Suche nach indirekten Abhängigkeiten aufgezeigt. Dies ist natürlich nicht die einzige mögliche Abhängigkeit die relevant sein kann. Vielmehr erlaubt das gerichtete Netzwerk nach jeglicher Kombination von Knoten und Kanten zu suchen, die in einer bestimmten Art und Weise und somit in einer relevanten Struktur miteinander verknüpft sind. Zieht man hier das Beispiel von Anforderungen an das System heran, können zum Beispiel folgende Abhängigkeiten gesucht und entsprechend interpretiert werden – vergleiche Abbildung 78:

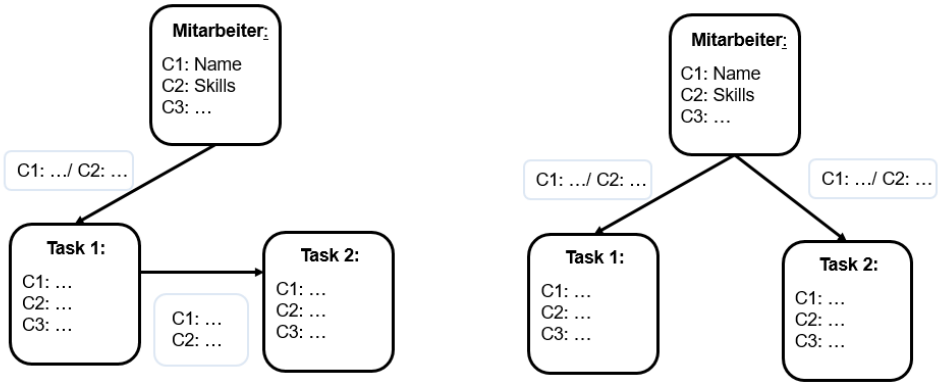


Abbildung 78: Zwei Beispiele von Zusammenhängen von Mitarbeiter und Aufgaben

Im linken Bespiel kann der Mitarbeiter die Aufgaben nur in einer Sequenz abarbeiten. Dagegen zeigt das rechte Beispiel eine mögliche Parallelisierung und somit erste Optimierung zur linken Struktur von Knoten und Kanten.

Dies ist jedoch nur ein Auszug aus den möglichen Abhängigkeiten und soll zeigen, wie nach einer bestimmten Struktur gesucht werden kann und welche Interpretation daraus folgt. Die Identifikation solcher Strukturen kann bei Entscheidungen herangezogen werden und hilft dem Anwender wichtige Abhängigkeiten nicht zu vernachlässigen.

6.2 Optimierungen anhand von Kennzahlen

Abgesehen von der Identifikation und Interpretation der DIF, welche nun ganzheitlich durch das Framework betrachtet werden können, ermöglicht das Framework zudem die Berechnung von Kennzahlen, welche ebenfalls die DIF aufzeigen, aber auch Optimierungen zulassen. In dieser Arbeit werden die Prozessoptimierungen – hier die Eigenschaftsabsicherung - vorrangig behandelt. Es ist jedoch möglich diese Berechnungen auch für Strukturanalysen zu verwenden.

Da das Framework mathematisch definiert und stelleweise erweitert wurde, können Kennzahlen aus der mathematischen Graphen- und Netzwerktheorie berechnet und entsprechend interpretiert werden. Dabei lassen sich Kennzahlen einzelner Knoten, knotenübergreifende Kennzahlen und Kennzahlen zum Netzwerk selbst berechnen.

Diese Erkenntnisse helfen dem Anwender nach bestimmten Knoten und damit Aspekten zu suchen oder gesamte Netzwerke zu interpretieren und reduzieren somit die Komplexität auf neutrale Kennzahlen. Die Kennzahlen beschreiben die Struktur des Netzwerkes oder beziehen sich auch einen einzelnen Knoten im Netzwerk. Die Analyse ist in beiderlei Hinsicht eine strukturelle Analyse.

In diesem Kapitel werden deshalb die wichtigsten und bekanntesten Berechnungen aus der Graphentheorie vorgestellt und angewendet, die im Sinne des Ziels dieses Beitrages stehen – ohne dabei die Vollständigkeit aller Analysen zu behaupten. Dabei werden die bereits getroffenen Definitionen verwendet, womit die folgende Netzwerkanalyse auf die vorherigen Kapitel aufbaut. Es ist jedoch anzumerken, dass die Notationen für die hier verwendeten Analysen im Deutschen, sowie auch im Englischen, weit verbreitet sind und daher als einheitlich gelten. Um nicht die Definitionen für jeden Teilgraphen zu formulieren, werden die folgenden Definitionen auf $G\ (V, E)$ bezogen und sind entsprechend für alle Teilgraphen, Knoten und Kanten gültig.

6.2.1 Identifikation und Interpretation von Knoten mit bestimmten Eigenschaften

In diesem Abschnitt werden Kennzahlen behandelt, die sich direkt auf einzelne Knoten beziehen oder in Relation zum Gesamt- bzw. Teilnetzwerkes stehen.

Die wohl bekannteste Berechnung ist die Frage nach dem **Grad** der jeweiligen **Knoten** in einem Graphen. Für den Graph G wird jedem Knoten $v \in V$ ein Grad $d(v)$ zugeordnet, welcher die Anzahl der Kanten repräsentiert – vergleiche Abbildung 79.

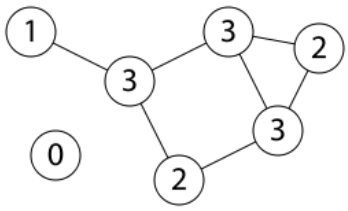


Abbildung 79: Graph mit Knotengrad als Beschriftung der Knoten

Zu beachten ist, dass die Knoten mit einem Knotengrad von 1 in einem ungerichteten Graphen einen Start bzw. Endkonten darstellen. Knoten mit einem Knotengrad von 0

entsprechen einem isolierten Knoten. Da jedoch in den vorangegangenen Kapiteln eine solche Situation ausgeschlossen wurde, kann der Knotengrad nur im Wertebereich $d(v) > \mathbb{N}$ sein.

Da im gezeigten Framework sowohl ungerichtete, als auch gerichtete Graphen bzw. Teilgraphen auftauchen, wird die Definition des Knotengrades erweitert. Demzufolge wird in einem gerichteten Graphen der **Eingangsgrad** als die Anzahl von eingehenden Kanten definiert:

$$d^{-}(V) \triangleq Eingangsgrad$$

Folglich ist der **Ausgangsgrad** als die Anzahl von ausgehenden Kanten eines Knoten definiert:

$$d^{+}(V) \triangleq Ausgangsgrad$$

Das heißt der Knotengrad gibt Aufschluss darüber, wie ein Knoten (Anforderung, System, MA, etc.) in Relation zu den restlichen Knoten des Netzwerks steht. Da es sich jedoch wiederum „nur“ um die Anzahl der Verbindungen handelt, darf diese Aussage nicht überbewertet werden. Es kann zum Beispiel vorkommen, dass ein Bauteil mit allen anderen Bauteilen in Relation steht, weil es beispielsweise die Hülle eines Systems ist und somit den höchsten Knotengrad aufweist, jedoch für die Entwicklung des Systems eine untergeordnete Rolle spielt.

Bei einer ersten Analyse in Bezug auf das Beispiel der Eigenschaftsabsicherung einer Anforderung, eines Systems und ggf. einer Rolle bzw. eines MA ist jedoch ein hoher Knotengrad ein gutes Indiz dafür, dass es sich um einen wichtigeren Knoten handelt. Im Umkehrschluss gilt, dass je kleiner der Knotengrad ist, desto unkritischer ist der Knoten.

Für einen gegebenen Graphen kann man zusätzlich bestimmen, welcher Knoten den **maximalen bzw. welcher den minimalen Knotengrad** besitzt, weshalb die Definition des Knotengrades um drei weitere erweitert wird:

$$\sigma(G) = \min\{\, d(v) \mid v \in V \,\} \triangleq Minimalgrad\ von\ G$$

$$\Delta(G) = \max\{\, d(v) \mid v \in V \,\} \triangleq Maximalgrad\ von\ G$$

$$\bar{d}(G) = \frac{1}{\mid V \mid} \sum_{v \in V} d(v) \triangleq Durchschnittsgrad\ von\ G$$

Die beiden zuerst genannten Knotengrade lassen sich mit dem Index – bzw. + in aus- bzw. eingehende Knotengrade unterscheiden und helfen die restlichen Knotengrade zu

normieren. Somit kann der Anwender einen Wert zwischen 0 und 1 erzeugen, welcher leichter zu interpretieren ist und dem Knoten im Gesamtnetzwerk einen relativen Wert verleiht.

Der Durchschnittsgrad stellt einen Mittelwert dar und hilft dem Anwender zu vergleichen, ob der jeweilige Knotengrad im Netzwerk eher überdurschnittlich verknüpft ist oder nicht.

Hilfreich bei der Interpretation des Knotengrades ist, dass immer folgendes gilt:

$$\sigma(G) \leq d(G) \leq \Delta(G)$$

$$\sum_{v \in V} d(v) = 2 \mid E \mid$$

$$\sigma_d = \sqrt{\frac{1}{n} \sum_{1 \leq i \leq n} (d_i - \bar{d})^2} \triangleq Standartabweichung\ von\ d(G)$$

Beim Beispiel der Eigenschaftsabsicherung hilft es zu erkennen, wo ein Flaschenhalseffekt entsteht, wenn zum Beispiel „MA 1“ im Vergleich zu „MA 2“ mehrere Rollen wahrnehmen muss – vergleiche Abbildung 80:

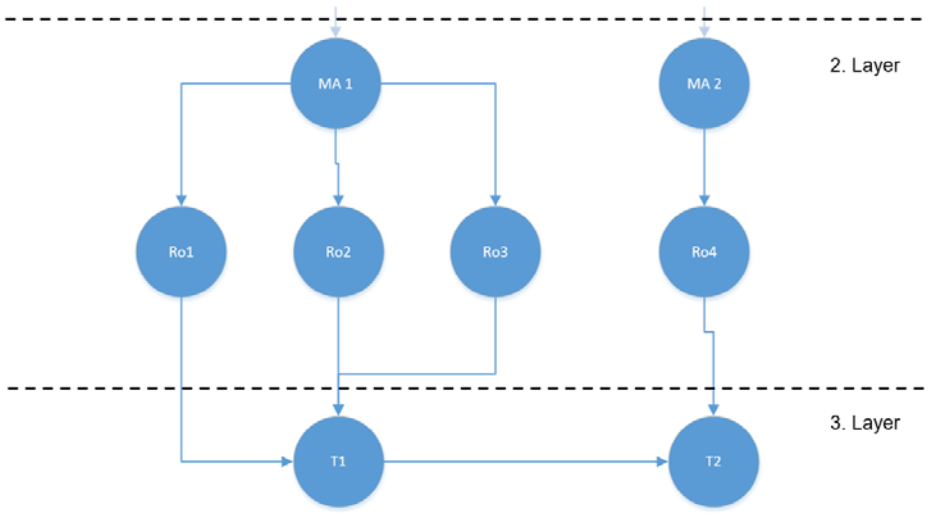


Abbildung 80: Auszug aus einem Graphen mit Beziehung zwischen Mitarbeiter, Rollen und Aufgaben

Hier kann der Knotengrad der jeweiligen Mitarbeiter in Relation zu $\Delta(M_1)$ gesetzt und somit normiert werden, womit die Verantwortung des Mitarbeiters im Vergleich zu den

restlichen betrachtet wird. Am Beispiel ergibt sich ein normierter Ausgangsknotengrad des MA1 und MA2 von 1 für MA1 und $\frac{1}{3}$ für MA2, was die Vermutung der überdurchschnittlichen Belastung von MA1 unterstreicht.

Durch die Verwendung des Ein- bzw. Ausgangsgrades lassen sich Trigger bzw. Endknoten erkennen. Trigger sind Anforderungen, wie das des „Abstand-halten nach StVO“ , welches als Eingangsanforderung – hier gesetzliche Vorgabe – in den Graphen eingeht.

Umgekehrt sind Endknoten mit einem $d^- = 0$, wie das eines Workflows –vergleiche Abbildung 82 –Knoten „T2“, was eine Abschlussaufgabe darstellt.

Um die Auswertung der Zentralität noch weiter zu differenzieren, wird in der Graphen- und Netzwerktheorie zwischen der Zwischenzentralität und Nähezentralität – im Englischen „Closeness Centrality“ und „Betweeness Centrality“ – unterschieden.

Die **Closeness Centrality** beschreibt den gemittelten Kehrwert aller kürzesten Wege eines Knotens $v(i)$ zu allen anderen Knoten $v(j)$ des Graphen bzw. Teilgraphen:

$$C_i^c = \frac{(n-1)}{\sum_{0 \le j \le n} d_{ij}} \; ; j \neq i$$

Dabei muss für jeden Knoten $v(i)$ eine Matrix $D(G) = (d_{ij}) = d(v_i, v_j)$ erstellt werden, welche die Entfernung des Knotens zu allen anderen Knoten beschreibt.

Das heißt mit diesem Analyseverfahren misst man nicht nur die Verbindungen eines Knotens zu unmittelbar naheliegenden Nachbarknoten, sondern zu allen Knoten des Graphen bzw. Teilgraphen. Die Closeness Centrality wird auch als durchschnittliche Pfaddistanz eines Knotens zu den anderen des Netzwerkes bezeichnet.

Die Bedeutung eines Knotens mit Hoher Closeness Centrality wird deutlicher, wenn dieser Knoten aus dem Netzwerk entfernt wird. Denn dann entstehen Teilnetzwerke, die nicht mehr miteinander verknüpft sind. Bezogen auf das Beispiel – vergleiche Abbildung 81 - stellt der Integrationsschritt v_5 (höchste Closeness) im Teilgraph G_3 einen kritischen Workflow dar, da ohne diesen Knoten der Graph in zwei Teilgraphen zerfällt und der Folge-Workflow v_6 nicht weitergeführt werden kann.

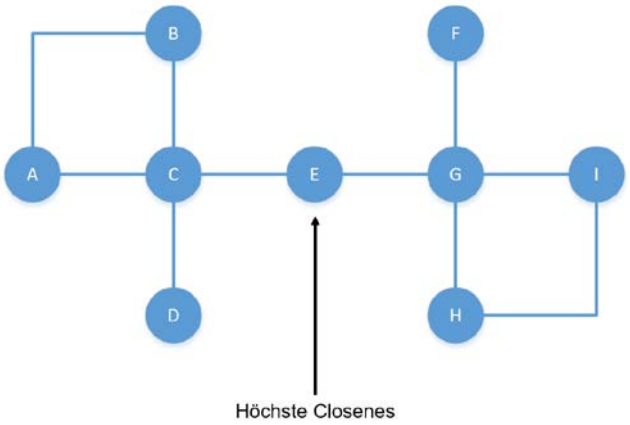


Abbildung 81: Beispiel für höchste Closeness von Knoten "E"

Natürlich ist dieser Faktor auch layerübergreifend interessant. Bezieht man beispielsweise den Closeness-Faktor auf die Verbindung eines Mitarbeiters zu den Systemen, dann sollte dieser eine hohe Closeness im Netzwerk (M_1, H_4 auf System-Ebene) vorweisen, weil dann gesichert ist, dass er eine gute Erreichbarkeit zu den Systemen bzw. Teilsystemen (M_1, K) oder Komponenten (M_1, L) hat.

Dagegen beschreibt die **Betweeness Centrality** das Verhältnis der Anzahl der kürzesten Wege P_{kj}^{i} (P für Pfad) zwischen den Knoten v_k und v_j, auf denen der Knoten v_i liegt, zur Zahl aller kürzesten Wege P_{kj} - wiederum zwischen den Knoten v_k und v_j:

$$C_i^b = \sum \frac{P_{kj}^{i} / P_{kj}}{(n-1)(n-2)/2} \quad wobei\ k \neq j\ und\ \ i \notin \{k, j\}$$

Das heißt die Betweeness Centrality ist in der Analyse der DIF eine sehr wichtige Kennzahl, da sie in erster Linie „wichtige“ Knoten, Mitarbeiter, Systeme, Anforderungen, etc. identifiziert, über die die meisten Daten und Informationen laufen. Im Umkehrschluss kann man hier erkennen, welche Auswirkungen ein solcher Knoten auf das Teil- bzw. Gesamtnetzwerk hat und welche Folgen entstehen. Das kann der Ausfall einer Komponente sein, welche eine Reihe von Funktionen impliziert, ein Mitarbeiter, der für viele Systeme essenziell ist oder einer Task, die viele Folgen nach sich zieht. Diese Knoten sind ein Indikator für Schnittstellen, die im PEP besonders zu beobachten sind –vergleiche Abbildung 82.

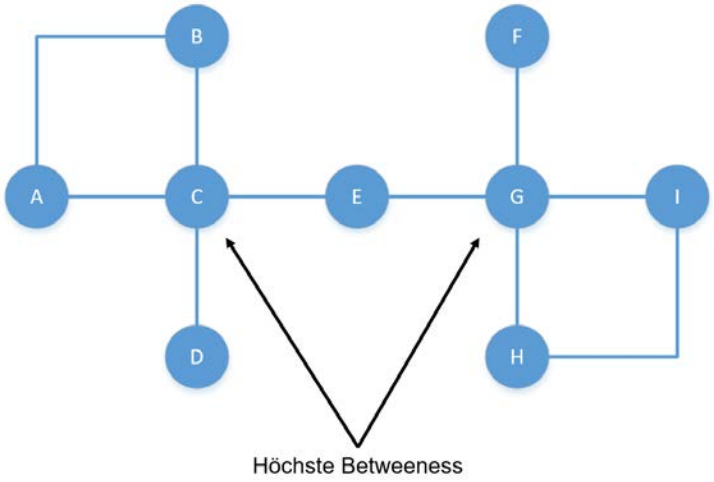


Abbildung 82: Beispiel für höchste Betweeness der Knoten "C" und "G"

Fällt hier der Knoten v_6 aus, können die drei damit verbunden Workflows (v_7, v_8 und v_9) nicht durchgeführt werden.

Auch für die Closeness und Betweeness Centrality gilt es, die Möglichkeit nach eingehender- bzw. ausgehender Centrality zu unterscheiden. Eine solche Unterscheidung erlaubt es nicht nur die gezeigten Analysen zu interpretieren, sondern beantwortet zudem die Frage nach der Relevanz des Knotens **von** (eingehende) bzw. **zu** (ausgehend) den restlichen Knoten des Netzwerks.

Zwischenfazit:

In diesem Abschnitt wurden Kennzahlen der Netzwerktheorie gezeigt, die sich auf einzelne Knoten des Frameworks beziehen. Dabei wurde jeweils erklärt, welche Bedeutung die Kennzahl haben kann. Es handelt sich jedoch lediglich um Beispiele, um die Berechnung auf die eigene Situation abwandeln zu können. Sie zeigt aber, wie die Möglichkeiten der Netzwerk-Theorie verwendet werden können, um Knoten mit hoher Relevanz zu identifizieren. Die Interpretation unterliegt natürlich stets der Daten- und Konsensgrundlage. Denkbar sind zum Beispiel nachfolgende Kennzahlen und Bedeutungen.

6.2.2 Strukturelle Berechnungen und Interpretationen des Netzwerks

Im zweiten Abschnitt werden strukturelle Analysen des Netzwerkes betrachtet. Dabei kann das Gesamtnetzwerk bzw. Teilnetzwerk oder auch eine Zusammenführung von Teilgraphen analysiert werden. Die folgenden Kennzahlen beurteilen somit die Beschaffenheit des Netzes und liefern entsprechend Rückschlüsse.

Die **Gesamtgröße** eines Netzwerkes lässt sich durch die Anzahl der Elemente eines NWs betrachten. Dabei kann die Summe der Knoten, der Kanten, oder beider eine Aussage liefern, mit welcher Komplexität zu handhaben ist. Diese Zahl ist allein genommen noch keine starke Aussage. Allerdings wird die Bedeutung der Größe eines Netzwerks in der zeitlichen Betrachtung wichtiger, da Anhand der Zunahme der Größe gefolgert werden kann, wie hoch die zu erwartende Komplexität zum Zeitpunkt t ist.

Dagegen liefert der **globale Durchmesser** die maximale Entfernung aller Knoten. Das heißt es wird die maximale Anzahl an Kanten auf dem kürzesten Weg zwischen zwei Knoten v_i und v_j gezählt:

$$L(G) = \max(d(i,j))$$

Das bedeutet, je höher der Durchmesser ist, umso weiter sind die Knoten auseinander. Bezieht man das auf die Kommunikation zwischen Mitarbeitern, gilt es, die Zahl zu minimieren, da andernfalls der Kommunikationsweg zu lang ist und die D&I zu lange brauchen, bis sie verteilt sind.

Die **Dichte** eines Netzwerks beschreibt die Anzahl m der Kanten im Verhältnis zu der maximal möglichen Kantenanzahl eines Graphen:

$$\rho(G) = \frac{m}{\max n} = \frac{2m}{n(n-1)}$$

Das heißt die Dichte des Netzwerks bzw. Teil-Graphen ist eine charakteristische Zahl zwischen 0 und 1 und ist ein Indikator für die „Verbundenheit“ des Graphen – wobei 1 für ein vollständiges Netzwerk steht. Allgemein gilt, dass je größer das Netzwerk ist, desto geringer ist die Dichte, weil nicht jeder Knoten mit allen anderen Knoten verknüpft sein muss – vergleiche Abbildung 83:

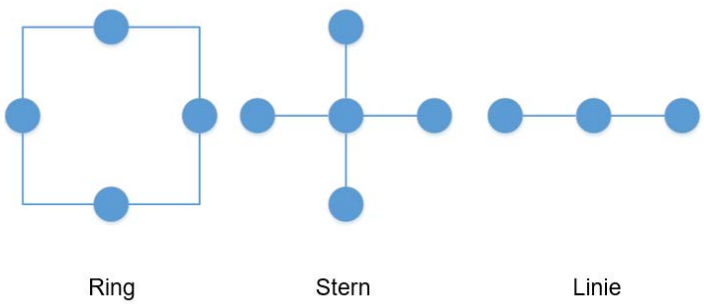


Abbildung 83: Beispiel-Graphen und Bezeichnung

Auch diese Kennzahl liefert eine neutrale Zahl über die Beschaffenheit des Teilnetzwerkes und könnte ein Hinweis für mögliche Optimierungen sein. Wenn man diese zum Beispiel auf die Kommunikation oder auf die Produktstruktur bezieht, ist eine geringe Dichte ein ungünstiger Faktor und sollte erhöht werden.

Ein weiterer globaler Wert ist die **Konnektivität**, welcher das Verhältnis der Kanten zu Knoten beschreibt:

$$\beta(G) = \frac{E}{V}$$

Auch diese Kennzahl liefert eine relative Aussage zur Verbundenheit des Netzwerks und wächst mit „besserer“ (höherer) Verknüpfung. Die Interpretation verhält sich ähnlich zu der Dichte – vergleiche Abbildung 84:

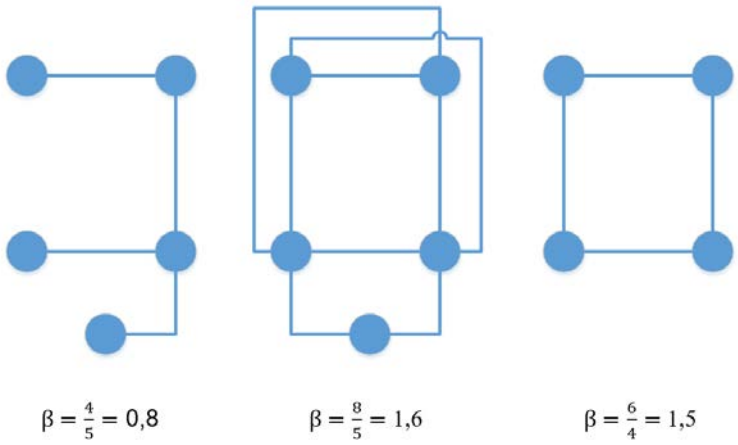


Abbildung 84: Beispiel-Graphen mit unterschiedlicher Konnektivität

Die **Kantenkonnektivität** spielt eine größere Rolle und beschreibt die Anzahl an Kanten, die aus dem Netzwerk entfernt werden müssen, um das Netzwerk zu unterbrechen:

$$k(G) \triangleq Kantenkonnektivität$$

Das heißt je höher die Zahl k ist, desto mehr unabhängige Pfade existieren zwischen den Knoten und damit ist die Wahrscheinlichkeit umso kleiner, kritische Verbindungen zu schaffen – vergleiche Abbildung 85.

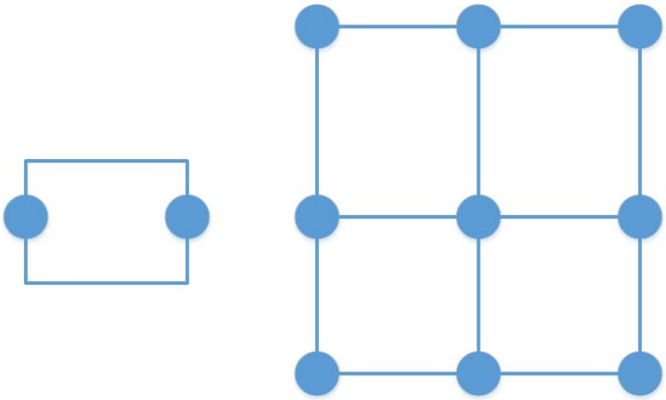


Abbildung 85: Zwei Beispiel-Graphen mit Konnektivität = 2

Liegt die Kantenkonnektivität lediglich bei 1, ist das zum Beispiel ein starkes Indiz für einen kritischen Workflow. Somit gilt es diese Zahl zu maximieren und den Prozess bei Annährung an 1 zu optimieren. Layerübergreifend gilt diese natürlich für den gesamten DIF.

Zwischenfazit:

Die strukturelle Analyse des Netzwerks zeigt sich ebenfalls durch entsprechende globale Kennzahlen. Diese helfen zu identifizieren, ob Handlungsbedarf besteht und der Prozess bzw. der DIF optimiert werden muss. Die gezeigten Berechnungen beziehen sich auch hier auf das gesamte Framework bzw. auf Teil-Graphen. Übereinstimmend sind die Kennzahlen auch auf Multi-Level-Ebene differenziert einsetzbar und liefern Erkenntnisse über die Beschaffenheit eines Sub-Systems, Komponenten, o.Ä.

6.2.3 Kombination von Analysen und Messung von Veränderungen

Die dritte und aussagekräftigste Analyse ist einerseits die Kombination und Interpretation von Kennzahlen und andererseits die Messung von deren Änderungen. Das heißt die Bedeutung von mehreren Kennzahlen kann eine noch tiefere und womöglich neue Interpretation zulassen. Damit werden Gegebenheiten der DIF noch sichtbarer oder tiefer begründet. Bezogen auf das Netzwerk bzw. Teilnetzwerk können bestimmte Kenngrößen zusammen eine stärkere Bedeutung hinsichtlich der Struktur liefern.

In Folgendem werden beispielsweise drei Kennzahlen für drei verschiede Bedeutungen gezeigt, die in der Gesamtheit Auskunft über das Netzwerk geben. Da die Beispiele einfach gehalten sind, kann ihre Verifizierung mit einfachem Hinsehen plausibilisiert werden.

Für die Erklärung werden exemplarisch der Knotengrad, der Durchmesser und die Kantenkonnektivität verwendet, um zu verdeutlichen, wie die gezeigten Kennzahlen kombiniert werden können.

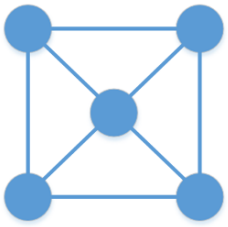
Im ersten Fall haben die drei Faktoren jeweils den Wert 1. Es handelt es sich um eine Kette bzw. einen Bus – vergleiche Abbildung 86:

Knotengrad = 1 Durchmesser = 1 Konnektivität = 1

Abbildung 86: Beispiel einer Kette mit jeweiligen Kennzahlen

Ein solches Netzwerk (auch Teil-Graph möglich) bedeutet zum Beispiel, dass die Tasks einer Folge in einer bestimmten Reihenfolge abgehandelt werden müssen und es keine Alternativen gibt.

Möchte man jedoch die Verknüpfung der Knoten erhöhen, dann sollten sich der Knotengrad und die Kantenkonnektivität entsprechend $n - 1$ annähern, wobei hier der Durchmesser bei 1 bleibt – vergleiche Abbildung 87:



Knotengrad = n-1 Durchmesser = 1 Konnektivität = n-1

Abbildung 87: Beispiel eines vollständigen Graphen mit jeweiligen Kennzahlen

Somit ergibt sich ein vollständiges Netzwerk, in dem die Knoten sehr gut miteinander verknüpft sind. Eine solche Situation sollte beispielsweise unter den Mitarbeitern im zweiten Layer erreicht bzw. angenähert werden.

Auf der anderen Seite handelt es sich beim Netzwerk um einen Ring, wenn der Knotengrad hier bei 2 liegt, die Kantenkonnektivität liegt ebenfalls bei 2 und er Durchmesser bei $n/2$ – vergleiche Abbildung 88.

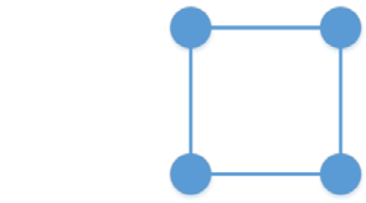


Abbildung 88: Beispiel eines Ringes mit jeweiligen Kennzahlen

Ein Ringschluss kann gewollt sein, wenn zum Beispiel die Informationsweitergabe so gewollt ist. Es kann jedoch sehr kritisch für einen DIF werden, wenn die beteiligten Systeme bzw. Mitarbeiter nicht untereinander kommunizieren.

Eine weitere Form der Kombination von Kennzahlen ist die Erstellung von sogenannten „verdichteten Schichten“ (Klaus Pohl, 2011). Demnach finden diese zwar nur in Bezug auf Anforderungen Anwendung, sind jedoch auf alle möglichen Kennzahlen der Netzwerkanalyse zu übertragen. Sie liefern Auskunft über die DIF im PEP in Kombination und erleichtern somit die Interpretation. Dabei kann die einfache Summe, der relative Anteil an der Gesamtsumme und jede erdenkliche Anwendung der mathematischen Operationen $+, -, *, /$ auf selektierte Daten angewendet werden, um weitere Folgerungen zu ziehen. In der Vorarbeit „An approach for using the FMEA for the development of a new and revolutionary rotation piston engine” wurden ebenfalls Kennzahlen zusammengefügt, die dem Entwickler die Entscheidung erleichtern sollten und somit die Komplexität der Situation wesentlich reduzierten (Chahin, Paetzold, *u. a.*, 2016). Angelehnt an dieses Beispiel kann man die gleichen Berechnungen auf das zweite Layer als Pendant zu den „Funktionalen Anforderungen“ übertragen und zwei Workflows analog für die „Konzepte“ nutzen. Somit würden Functions die Mitarbeiter und Concept A und B verschiede Workflow-Alternativen darstellen – vergleiche Abbildung 89:

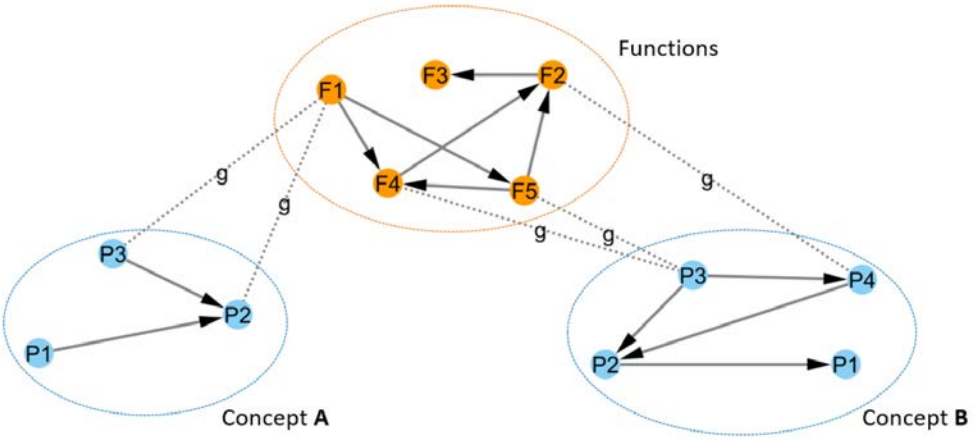


Abbildung 89: Beispiel-Graph als Analogie für zweite und dritte Layer nach Chahin (Chahin, Paetzold, *u. a.*, 2016)

Zieht man dann die gleiche Formel heran, indem die Knotengrade der Teil-Graphen zuerst normiert und anschließend zu einer Zahl zusammengefasst werden, kann vergleichbar dazu der gleiche Schluss gezogen werden. Nämlich, dass je höher K wird, umso kritischer ist der Workflow, weil in solch einem Fall wenige Mitarbeiter für viele stark vernetzte Aufgaben verantwortlich sind – vergleiche Abbildung 90:

$$K(d) = F_j \cdot \sum_i P_i \cdot g_i \ , \ j - node, \ i - connected\ Parts\ to - with\ weight\ g_i$$

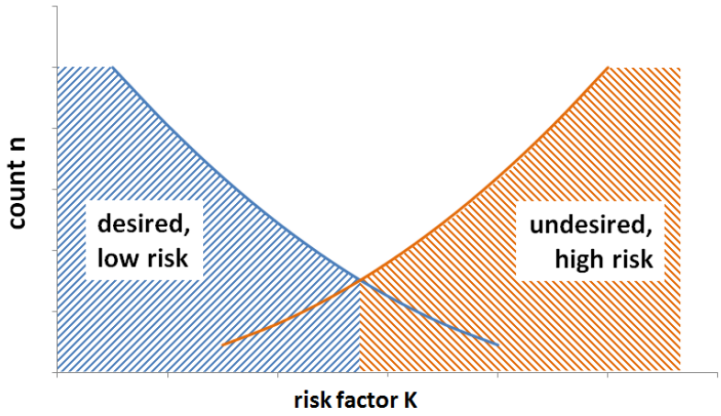


Abbildung 90: Berechnung des Indexes "*K*" und Verteilung nach Chahin (Chahin, Paetzold, *u. a.*, 2016)

Es kann gefolgert werden, welcher – in diesem Fall – Mitarbeiter relativ viele Workflows verantwortet, die wiederum für weitere Workflows entscheidend sind. Das heißt die komponierte Zahl identifiziert kritische Flaschenhalseffekte, ohne dass der Anwender die DIF im Detail kennen muss.

Bisher wurden „nur“ statische Betrachtungen vorgestellt. Eine weitere Möglichkeit ist, zu messen, wie sich Kennzahlen über die Zeit *t* verändern. Dafür können festgelegte Zeitintervalle definiert werden oder ab einer bestimmten prozentualen Änderung regiert werden. So lassen sich Kennzahlen eines Traces zeitlich vergleichen und daraus Handlungen ableiten – etwa eine Eigenschaftsabsicherung ab einem bestimmten Reifegrad eines Systems – vergleiche Abbildung 91:

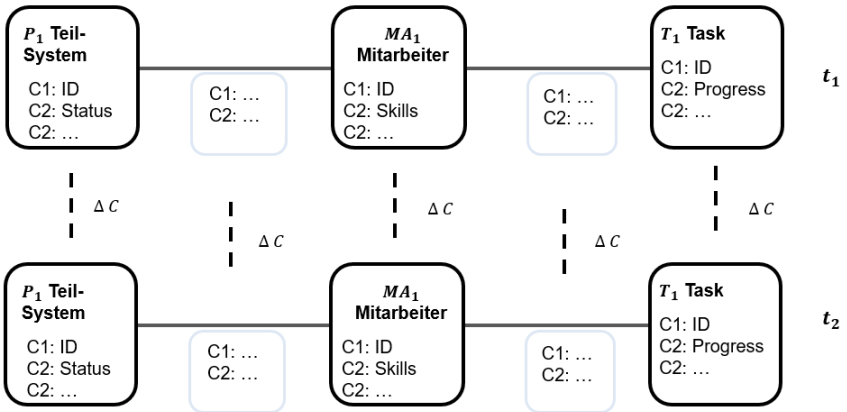


Abbildung 91: Beispiel-Trace zu unterschiedlichen Zeitpunkten t_1 und t_2

Auf der anderen Seite können auch mehrere Kennzahlen zusammen interpretiert werden. Etwa das Heranwachsen eines vollständigen Graphen aus einem Ring – vergleiche Abbildung 92 - wobei „KG“ für Knotengrad, „D“ für Durchmesser, „K“ für Konnektivität und „n“ für die Anzahl der Knoten steht:

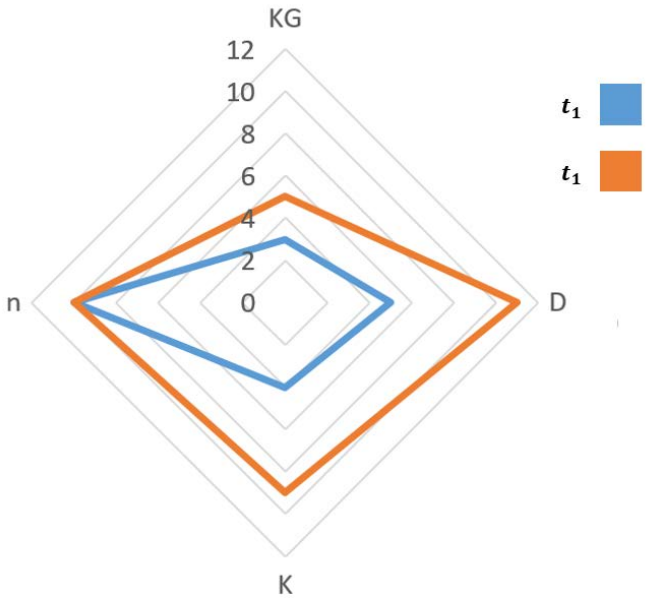


Abbildung 92: Beispiel-Graph eines vollständen Graphen zum Zeitpunkt t_2, das sich aus einem Ring zum Zeitpunkt t_1 entwickelte

Eine solche Entwicklung kann ausschlaggebend für eine Handlung sein, wenn man zum Beispiel wiederum eine Eigenschaftsabsicherung plant und diese in Relation zu der Produktarchitektur stellt, hier zum Beispiel „n“ für die Anzahl der Systeme, die plötzlich ansteigt.

Zwischenfazit:

Aufbauend auf die vorangegangenen zwei Abschnitte (Netzwerke und Knoten) wurden in diesem Abschnitt Möglichkeiten aufgezeigt, die nun mathematisch definiert sind. Somit lassen sich die DIF nicht nur erfassen, sondern im Hinblick der analytischen Möglichkeiten in Kennzahlen fassen. Dabei werden nicht nur einzelne Kennzahlen (KZ) ersichtlich, sondern zudem Kombinationen. Diese Kombinationen ergeben zusammen eine noch genauere Aussage über die Abhängigkeiten im Graphen und beziehen dessen Topologie mit ein.

6.3 Verweis auf Algorithmen für die Berechnungen

Die gezeigten KZ des Graphen bzw. Teilgraphen (einzelner Knoten oder bezogen auf die Struktur) setzen häufig voraus, dass Algorithmen zum Einsatz kommen. Dies zeigt sich beispielsweise in den Summen, die gebildet werden oder weil ein Wert eines Knotens, zum Beispiel der Knotengrad, mit benachbarten oder allen weiteren Knoten verglichen werden muss. Am deutlichsten wird dies bei der Betrachtung der „klassischen“ Weglänge eines gewichteten Netzwerks. Hier sind aus der Graphentheorie die sogenannte Breiten- und Tiefensuche am bekanntesten (Nitzsche, 2009; Krischke, André, 2014). Je nachdem für welchen Algorithmus man sich entscheidet, ist der kürzeste Weg schneller gefunden und damit der Aufwand bzw. die notwendigen Zyklen entsprechend gering. Dieses Phänomen wird in der komplexen Graphen- und Netzwerktheorie und in der Informatik stark untersucht, da der Aufwand entscheidend ist und wissenschaftlich diskutiert wird.

Für die relativ „einfachen“ Berechnungen, die hier gezeigt werden, fällt die Zeit, die notwendig ist, um die diskutierten KZ zu erzeugen, nicht hoch aus. Gängige Tools und die heutigen Rechnerleistungen erlauben den Aufwand in erster Linie zu ignorieren.

Sollte jedoch das Netzwerk über ca. 10^6 Elemente aufweisen und die Abfrage zum Beispiel für zeitliche Delta-Analysen (vergleiche vorangegangenes Kapitel) eine hohe Frequenz aufweisen, kann der genutzte Algorithmus an Bedeutung gewinnen und sollte in diesem Fall näher betrachtet und optimiert werden (Turau, 2009).

7 Zusammenfassung und Ausblick

Diese Arbeit stellt in Summe eine Guideline dar, welche dem Anwender zeigt, wie das Framework aufzusetzen ist, um komplexe Daten- und Informationsflüsse eines Produktentwicklungsprozesses in erster Linie zu erkennen und zu verstehen, aber auch im Sinne einer Prozessoptimierung Kennzahlen abzuleiten und zu interpretieren.

Die eingeführten mathematischen Definitionen können somit verwendet werden, um DIF - analog zu den Methoden des Requirements-Engineering bzw. –Managements - ganzheitlich wiederzugeben und darüber hinaus auszuwerten. Dazu wurden im ersten Abschnitt Graphen eingeführt und ihre Möglichkeiten aufgezeigt, beliebig detaillierte Hierarchien aufnehmen zu können. Es wurde erläutert, welcher Notation zu folgen ist.

Im zweiten Abschnitt wurden neben den bekannten mathematischen Verbindungen (Kanten) der Elemente (Knoten) zueinander weitere Möglichkeiten aus der Praxis verwendet, die es erlauben die Abhängigkeiten der DIF weiter auszudifferenzieren.

Schließlich wurde gezeigt, welche neutralen mathematischen Analysen möglich sind und wie diese interpretiert werden können, um die steigende Komplexität zu reduzieren. Es wurden am Beispiel der Eigenschaftsabsicherung Vorschläge aufgeführt, wie die mathematischen Kennzahlen dem Entwickler bei der Entscheidungsfindung helfen, Handlungen abzuleiten oder zu begründen, ohne die vollständige Komplexität des entstandenen Netzwerks im Detail überblicken zu müssen.

Wie bereits eingangs angeführt, wird in dieser Arbeit anhand von Beispielen plausibilisiert, wie das Framework anzuwenden ist und entsprechend validiert. Weiterführende Arbeiten können diesen Ansatz situations- bzw. unternehmensbezogen anwenden und somit weitere Erkenntnisse gewinnen und Best Practises formulieren. Es ist denkbar, dass sich branchenbezogen bestimmte Analysen etablieren und durch die Häufigkeit der Anwendung und der erzeugten Ergebnisse herauskristallisieren. Ebenfalls lassen sich kombinierte Analysen nur durch die Nutzung und durch Erfahrungswerte etablieren und erweitern. Kennzahlen, wie im letzten Kapitel am Beispiel des Rotationskolbenmotors gezeigt, haben das Potential in bestimmten Situationen als Hilfsmittel zu dienen und können in einer Art Bibliothek bzw. Katalog in einem Unternehmen geführt werden.

Eine ganz andere Art die Bedeutung der gewonnenen Kennzahlen zu steigern, ist die sogenannte „Feinjustierung“ der KZ. Diese geschieht durch die Gewichtung der errechneten Faktoren und steigert bzw. senkt den jeweiligen Wert in der Gleichung. Am Beispiel des Betweeness- bzw. Colseness-Faktors kann es situationsbezogen vorkommen, dass der eine Wert „wichtiger“ als der andere ist und somit in der Gleichung gewichtet werden muss. Dieses Vorgehen ist auf alle gezeigten KZ anzuwenden. Mit diesem Vorgehen steigt die Interpretationsstärke der KZ für die Entscheidungen bzw. die Optimierung. Die

Folge ist die teil- bzw. vollständige Automatisierung der Erzeugung und Interpretation der KZ im PEP.

Quellenverzeichnis

4Soft GmbH, Bund, I. und Innern, B. des B. des (2009) *Einstieg in das V-Modell XT Bund, Das Referenzmodell für Systementwicklungsprojekte in der Bundesverwaltung Version: 2.0*.

Albert, A. und Ebel, B. (2015) „Modellierung von Zielsystemen in der interdisziplinären Produktentstehung Modeling of System of Objectives in Interdisciplinary Product Engineering“, S. 83ff., 89,1ff.

Alt, O. (2012) *Modellbasierte Systementwicklung mit SysML, Modellbasierte Systementwicklung mit SysML*. doi: 10.3139/9783446431270.

Automotive, S. (2010) „Automotive SPICE Process Assessment Model“, *Final Release, v4*.

Binner, H. F. (2017) „Organisation 4.0-Konfigurationskonzept: Neuausrichtung des geschäftsmodells“, *Productivity Management*.

Blessing, L. T. M. und Chakrabarti, A. (2009) *DRM, a design research methodology, DRM, a Design Research Methodology*. doi: 10.1007/978-1-84882-587-1.

Boehm, B. W. (1984) „Verifying and Validating Software Requirements and Design Specifications“, *IEEE Software*. doi: 10.1109/MS.1984.233702.

Booton, R. C. und Ramo, S. (1984) „The Development of Systems Engineering“, *IEEE Transactions on Aerospace and Electronic Systems*. doi: 10.1109/TAES.1984.4502055.

Browning, T. R. (2001) „Applying the design structure matrix to system decomposition and integration problems: A review and new directions“, *IEEE Transactions on Engineering Management*. doi: 10.1109/17.946528.

Büsing, C. (2010) *Graphen- und Netzwerkoptimierung, Graphen- und Netzwerkoptimierung*. doi: 10.1007/978-3-8274-2423-5.

Chahin, A., Hoffmeister, J., *u. a.* (2016) „A Practical Approach to Structure the Product Development Process using Network Theory“, in *DESIGN 2016: 14th International DESIGN Conference*.

Chahin, A., Paetzold, K., *u. a.* (2016) „An approach for using the FMEA and network-theory for the development of a novel rotation piston engine“, in *Proceedings of NordDesign, NordDesign 2016*.

Chahin, A. *u. a.* (2017) „Prerequisites for the Modelling and Analysis of a Product Development Process Using Network Theory“, in *Complex Systems Design & Management*. doi: 10.1007/978-3-319-49103-5_20.

Chahin, A. und Paetzold, K. (2015) „Analyse des Produktentwicklungsprozesses mit Methoden der komplexen Netzwerktheorie“, in Krause, D., Paetzold, K., und Wartzack, S. (Hrsg.) *DFX 2015: 26th Symposium on Design for X*. Herrsching: TuTech Verlag, S. 301–310.

Chahin, A. und Paetzold, K. (2018) „Planning Validation Verification Steps According to the Dependency of Requirements and Product Architecture“, in *2018 IEEE International Conference on Engineering, Technology and Innovation, ICE/ITMC 2018 - Proceedings*. doi: 10.1109/ICE.2018.8436250.

Cheng, P. W. *u. a.* (1986) „Pragmatic versus syntactic approaches to training deductive reasoning“, *Cognitive Psychology*. doi: 10.1016/0010-0285(86)90002-2.

Christian Zingel (2020) *Internal Block Diagramm of the exemplified vehicle assembly*. Verfügbar unter: https://www.researchgate.net/figure/Internal-Block-Diagram-of-the-exemplified-vehicle-assembly_fig7_285582096 (Zugegriffen: 28. Januar 2020).

Christoph Brauns (2016) *REQUIREMENTS ENGINEERING & MANAGEMENT IN DER WEHRTECHNISCHEN BESCHAFFUNG*. Shaker Verlag.

Clarkson, P. J., Simons, C. und Eckert, C. (2004) „Predicting change propagation in complex design“, *Journal of Mechanical Design, Transactions of the ASME*. doi: 10.1115/1.1765117.

Conklin, J. und Begeman, M. L. (1988) „gIBIS: A Hypertext Tool for Exploratory Policy Discussion“, *ACM Transactions on Information Systems (TOIS)*. doi: 10.1145/58566.59297.

Conradi, R. und Westfechtel, B. (1998) „Version Models for Software Configuration Management“, *ACM Computing Surveys*. doi: 10.1145/280277.280280.

Daniel Reh (2009) *Entwicklung einer Methodik zur logistischen Risikoanalyse in Produktions-und Zuliefernetzwerken*. Stuttgart: Fraunhofer-Verlag.

Danilovic, M. und Browning, T. R. (2007) „Managing complex product development projects with design structure matrices and domain mapping matrices“, *International Journal of Project Management*. doi: 10.1016/j.ijproman.2006.11.003.

David McCandless (2020) *Lines Of Code (LOC)*. Verfügbar unter: https://www.techopedia.com/definition/8073/lines-of-code-loc (Zugegriffen: 20. April 2020).

Dragos Truscan (2019) *SysMl requirements diagram*. Verfügbar unter: https://www.researchgate.net/figure/Example-of-a-SysML-requirements-diagram_fig1_31598271 (Zugegriffen: 1. Dezember 2019).

Eben, K. G. M., Daniilidis, C. und Lindemann, U. (2010) „Interrelating and prioritising requirements on multiple hierarchy levels“, *11th International Design Conference, DESIGN 2010*, S. 1055–1064.

Ehrlenspiel, K. *u. a.* (2013) „Methodik der integrierten Produkterstellung IPE in Entwicklung und Konstruktion“, in *Integrierte Produktentwicklung*. doi: 10.3139/9783446436275.006.

Engineering, I. (2020) *Link types in requirements projects*. Verfügbar unter: https://jazz.net/help-dev/clm/index.jsp?re=1&topic=/com.ibm.rational.rrm.help.doc/topics/r_linktypes.html&scope=null (Zugegriffen: 2. Februar 2020).

Eppinger, S. D. und Browning, T. R. (2012) *Design structure matrix methods and applications. MIT press, 2012.*, *MIT Press Books*.

Fischer, G. *u. a.* (1998) „Informing system design through organizational learning“, *Proceedings of ICLS 96. Second Interntional Conference on the Learning Sciences. Evanston, IL, USA. 25-27 July 1996*, S. 52–59.

Forsteneichner, C., Paetzold, K. und Metschkoll, M. (2018) „Continuous integration of model validation into product development“, in *Proceedings of International Design Conference, DESIGN*. doi: 10.21278/idc.2018.0231.

Friedenthal, S. (2009) „SysML: Lessons from Early Applications and Future Directions“, *INSIGHT*. doi: 10.1002/inst.200912410.

Friedman, M. (1968) „The Role of Monetray Policy“, *The American Economic Review*.

Gausemeier, J. *u. a.* (1995) „Integrated Product Development A New Approach for Computer Aided Development in the Early Design Stages“, in *Proceedings of the Third Conference on Mechatronics and Robotics*. doi: 10.1007/978-3-322-91170-4_1.

Gausemeier, J. *u. a.* (2013) „Studie: Systems Engineering in der industriellen Praxis“, in *Tag des Systems Engineering*. doi: 10.3139/9783446439467.012.

Glinz, M. (2007) „On non-functional requirements“, in *Proceedings - 15th IEEE International Requirements Engineering Conference, RE 2007*. doi: 10.1109/RE.2007.42.

Glinz, M. und Wieringa, R. J. (2007) „Stakeholders in requirements engineering“, *IEEE Software*. doi: 10.1109/MS.2007.42.

Gomaa, H. und Gomaa, H. (2012) „Use Case Modeling“, in *Software Modeling and Design*. doi: 10.1017/cbo9780511779183.008.

Gotel, O. C. Z. und Finkelstein, A. C. W. (1994) „Analysis of the requirements traceability problem“, in *Proceedings of the International Conference on Requirements Engineering*. doi: 10.1109/icre.1994.292398.

Grande, M. und Grande, M. (2011) „Reifegradmodelle“, in *100 Minuten für Anforderungs - management*. doi: 10.1007/978-3-8348-8135-9_15.

Haberfellner, R., de Weck, O., Fricke, E., & Vössner, S. (2012) *Systems Engineering: Grundlagen und Anwendung*. Zürich: Orell Füssli.

Halu, A. *u. a.* (2016) „Data-driven modeling of solar-powered urban microgrids“, *Science Advances*. doi: 10.1126/sciadv.1500700.

Hartley, T. W. und Robertson, R. A. (2006) „Stakeholder engagement, cooperative fisheries research and democratic science: The case of the Northeast Consortium“, *Human Ecology Review*.

Höft, D. und Schaller, H. (1985) „Software-Konfigurationsmanagement in großen Softwareprojekten“, *Informatik Spektrum*. doi: 10.1007/BF00425954.

Hood, C. *u. a.* (2008) *Requirements management: The interface between requirements development and all other systems engineering processes, Requirements Management: The Interface Between Requirements Development and All Other Systems Engineering Processes*. doi: 10.1007/978-3-540-68476-3.

HRworks (2018) *DSGVO 2018: HR Software mit starkem Schutz für Ihre personenbezogenen Daten*. Verfügbar unter: https://www.hrworks.de/news/dsgvo-2018-hr-software-mit-starkem-schutz-fuer-ihre-personenbezogenen-daten/ (Zugegriffen: 9. November 2019).

IBM Knwledge Center (2020) *DOORS Totorial*. Verfügbar unter: https://www.ibm.com/support/knowledgecenter/SSB2MU_8.3.0/com.ibm.rhp.doors.tutorial.doc/topics/lesson2_apply.html (Zugegriffen: 3. Mai 2020).

IEEE-Computer-Society (1998) „IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998“, *Electronics*.

IEEE (1990) „IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). Los Alamitos“, *CA: IEEE Computer Society*. doi: 10.1109/IEEESTD.1990.101064.

INCOSE (2007) *Systems engineering handbook: A guide for system life cycle processes*

and activities INCOSE -TP 2003- 002-03.1 August 2007, *Journal of the Franklin Institute*. doi: 10.1016/0016-0032(66)90450-9.

ISO/BS (2003) *Quality management systems: Guidelines for configuration management - BS ISO 10007:2003, BS ISO 10007:2003*.

ISO - The International Organization for Standardization (2009) „ISO 31000:2009 - Risk management - Principles and guidelines“, *ISO 31000:2009*.

Kapos, G. D. *u. a.* (2014) „Model-based system engineering using SysML: Deriving executable simulation models with QVT“, *8th Annual IEEE International Systems Conference, SysCon 2014 - Proceedings*, S. 531–538. doi: 10.1109/SysCon.2014.6819307.

Klaus Pohl, C. R. (2011) *Basiswissen Requirements Engineering, … " Certified Professional for Requirements Engineering*.

Kossiakoff, A. *u. a.* (2011) *Systems Engineering Principles and Practice: Second Edition, Systems Engineering Principles and Practice: Second Edition*. doi: 10.1002/9781118001028.

Kosslyn, S. M. (1988) „Aspects of a cognitive neuroscience of mental imagery“, *Science*. doi: 10.1126/science.3289115.

Krischke, André, and H. R. (2014) *Graphen und Netzwerktheorie: Grundlagen-Methoden-Anwendungen*. München: Hanser Verlag GmbH Co KG.

Lindemann, U. (2009) *Methodische Entwicklung technischer Produkte, Methodische Entwicklung technischer Produkte*. doi: 10.1007/978-3-642-01423-9.

Lindemann, U. und Lindemann, U. (2016) „Handbuch Produktentwicklung“, in *Handbuch Produktentwicklung*. doi: 10.3139/9783446445819.fm.

Malone, T. W. und Crowston, K. (1990) „What is coordination theory and how can it help design cooperative work systems?“, in *Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work, CSCW 1990*. doi: 10.1145/99332.99367.

Marc Neumann (2017) *Ein modellbasierter Ansatz zur risikoorientierten Entwicklung innovativer Produkte*. Shaker Verlag.

Meyer-schwickerath, B. (2015) „Das integrierte Produktentstehungs-Modell (iPeM) als Bezugsrahmen für Vorausschau am Beispiel von Szenariotechnik und strategischer Frühaufklärung“.

Mietzel, G. (2017) *Pädagogische Psychologie des Lernens und Lehrens, Pädagogische Psychologie des Lernens und Lehrens*. doi: 10.1026/02457-000.

Miller, G. A. (1956) „The magical number seven, plus or minus two: some limits on our capacity for processing information“, *Psychological Review*. doi: 10.1037/h0043158.

Negele, H., Fricke, E. und Igenbergs, E. (1997) „ZOPH - A Systemic Approach to the Modeling of Product Development Systems“, *INCOSE International Symposium*. doi: 10.1002/j.2334-5837.1997.tb02181.x.

Nibert, J., Herniter, M. E. und Chambers, Z. (2012) „Model-based system design for MIL, SIL, and HIL“, in *26th Electric Vehicle Symposium 2012, EVS 2012*. doi: 10.3390/wevj5041121.

Nitzsche, M. (2009) *Graphen für Einsteiger, Graphen für Einsteiger*. doi: 10.1007/978-3-8348-9968-2.

Paetzold, K. (2017) „Chapter 2 : Product and Systems Engineering / CA * Tool Chains“.

Parraguez, P., Eppinger, S. D. und Maier, A. M. (2015) „Information Flow Through Stages of Complex Engineering Design Projects: A Dynamic Network Analysis Approach“, *IEEE Transactions on Engineering Management*. doi: 10.1109/TEM.2015.2469680.

Parraguez, P. und Maier, A. M. (2015) „Unfolding the design process architecture: A networked perspective on activities“, in *Proceedings of the International Conference on Engineering Design, ICED*.

Partsch, H. (1998) *Requirements-Engineering systematisch, Requirements-Engineering systematisch*. doi: 10.1007/978-3-662-09758-8.

Patrick Richter (2016) *Illustration of LIDAR sensor demonstraing the time of flight principle*. Verfügbar unter: https://www.htw-mechlab.de/index.php/portfolio/masterarbeit-patrick-richter/ (Zugegriffen: 11. September 2019).

Patzak, G. (1982) *Systemtechnik — Planung komplexer innovativer Systeme, Systemtechnik — Planung komplexer innovativer Systeme*. doi: 10.1007/978-3-642-81893-6.

Pinheiro, F. A. C. (2004) „Requirements Traceability“, in *Perspectives on Software Requirements*. doi: 10.1007/978-1-4615-0465-8_5.

Pohl, K. (2008) „Requirements Engineering: Grundlagen, Prinzipien, Techniken“, *Requirements Engineering: Grundlagen, Prinzipien, Techniken*.

Pohl, K. und Rupp, C. (2015) *Basiswissen Requirements Engineering (4. überarbeitete Auflage), … " Certified Professional for Requirements Engineering*.

Purchase, H. C., James, M. I. und Cohen, R. F. (1997) „An Experimental Study of the Basis for Graph Drawing Algorithms“, *ACM Journal of Experimental Algorithmics*. doi: 10.1145/264216.264222.

Ramesh, B. *u. a.* (1995) „Implementing requirements traceability: a case study“, in *Proceedings of the IEEE International Conference on Requirements Engineering*. doi: 10.1109/isre.1995.512549.

Ramesh, B. und Jarke, M. (2001) „Toward reference models for requirements traceability“, *IEEE Transactions on Software Engineering*. doi: 10.1109/32.895989.

Reitmeier, J., Chahin, A. und Paetzold, K. (2015) „An approach to the property-based planning of simulations“, in *Proceedings of the International Conference on Engineering Design, ICED*.

Rudolph, M. *u. a.* (1999) „4 Design and Implementation of a System Engineering Standard Process for Satellite Development“, *INCOSE International Symposium*. doi: 10.1002/j.2334-5837.1999.tb00288.x.

Rupp, C. *u. a.* (2014) „Requirements-Engineering und -Management“, in *Requirements-Engineering und -Management*. doi: 10.3139/9783446443136.fm.

Salehi, V. *u. a.* (2017) „Integration of Systems Engineering Approach in Product-Lifecycle-Management by Means of a Mechatronic System“, in *Complex Systems Design & Management*. doi: 10.1007/978-3-319-49103-5_18.

Sandmann, G. (2020) „Formal Verification Techniques in a Model-Based Development Process based on TargetLink generated C-Code“. Verfügbar unter:

https://www.researchgate.net/publication/265242221_Formal_Verification_Techniques_in_a_Model-Based_Development_Process_based_on_TargetLink_generated_C-Code.

Schick, B. *u. a.* (2008) „Simulation Methods to Evaluate and Verify Functions, Quality and Safety of Advanced Driver Assistance Systems“, *IPG Technology Conference*. doi: 10.1038/nature13166.

Schmidt, T. S. *u. a.* (2017) „Agile Development and the Constraints of Physicality: A Network Theory-based Cause-and-Effect Analysis“, in *ICED 2017: 21st International Conference on Engineering Design*. Vancouver, Canada, S. 199–208.

Schönwald, J. *u. a.* (2018) „A method for a detailed analysis of verification and validation processes in product development“, *Proceedings of International Design Conference, DESIGN*, 3, S. 1313–1324. doi: 10.21278/idc.2018.0313.

Schönwald, J. R. *u. a.* (2019) „Improvement of collaboration between testing and simulation departments on the example of a motorcycle manufacturer“, *Proceedings of the International Conference on Engineering Design, ICED*, 2019-Augus(AUGUST), S. 149–158. doi: 10.1017/dsi.2019.18.

Schwankl, L. (2002) *Analyse und Dokumentation in den frühen Phasen der Produktentwicklung, Lehrstuhl für Produktentwicklung*.

Sebastian, S. T. *u. a.* (2017) „Agile development and the constraints of physicality: A network theory-based cause-andeffect analysis“, in *Proceedings of the International Conference on Engineering Design, ICED*.

Siemens (2019) *Siemens teamcenter*. Verfügbar unter: http://ukkoladom.ru/siemens-teamcenter/ (Zugegriffen: 3. Oktober 2019).

Sinha, K. *u. a.* (2015) „A simplified mathematical model for two-sided market systems with an intervening engineered platform“, in *Proceedings of the ASME Design Engineering Technical Conference*. doi: 10.1115/DETC201546657.

Song, Y. W. *u. a.* (2017) „Optimierung des Produktentwicklungsprozesses mittels Risikoanalyse vernetzter Anforderungen“, in *DFX 2017: Proceedings of the 28th Symposium Design for X*.

Stachowiak, H. (1973) *Allgemeine Modelltheorie, Allgemeine Modelltheorie*. doi: 10.1007/978-3-7091-8327-4.

Standard, D. I. (2011) „ISO 26262“, *Baseline*.

Suzanne Robertson, J. R. (2006) *Mastering the Requirements Process, The American journal of medicine*.

Taket, A., Checkland, P. und Holwell, S. (1999) „Information, Systems and Information Systems-Making Sense of the Field.“, *The Journal of the Operational Research Society*. doi: 10.2307/3010006.

Team, C. P. (2006) „CMMI for Development, version 1.2“, *Preface, Software Engineering Institute, Carnegie Mellon University, August*.

Tiemeyer, E. (2013) „IT-Projektmanagement“, in *Handbuch IT-Management*. doi: 10.3139/9783446436220.005.

Turau, V. (2009) *Algorithmische Graphentheorie, Algorithmische Graphentheorie*. doi: 10.1524/9783486598520.

Ulrich, H. (1970) „Die Unternehmung als produktives soziales System: Grundlagen der allgeminen Unternehmungslehre“, in *1*.

Valentini, U. *u. a.* (2013) „Grundlagen des professionellen Requirements Engineering & Management“, in. doi: 10.1007/978-3-642-29432-7_3.

VDI (2004) „Entwicklungsmethodik für mechatronische Systeme -- Design methodology for mechatronic systems“, *VDI 2206*. doi: 10.1002/mawe.19740050417.

VDI 2206, V. D. I. (2004) „VDI 2206 - Entwicklungsmethodik für mechatronische Systeme“, *Design*.

Walther, H. (1979) *Anwendungen der Graphentheorie*, *Anwendungen der Graphentheorie*. doi: 10.1007/978-3-322-84013-4.

Weber, C. (2012) „Produkte und Produktentwicklungsprozesse abbilden mit Hilfe von Merkmalen und Eigenschaften - eine kritische Zwischenbilanz“, *DFX 2012: Proceedings of the 23rd Symposium Design for X*, S. 25–62.

Weber, C. und Deubel, T. (2003) „New theory-based concepts for PDM and PLM“, in *Proceedings of the International Conference on Engineering Design, ICED*.

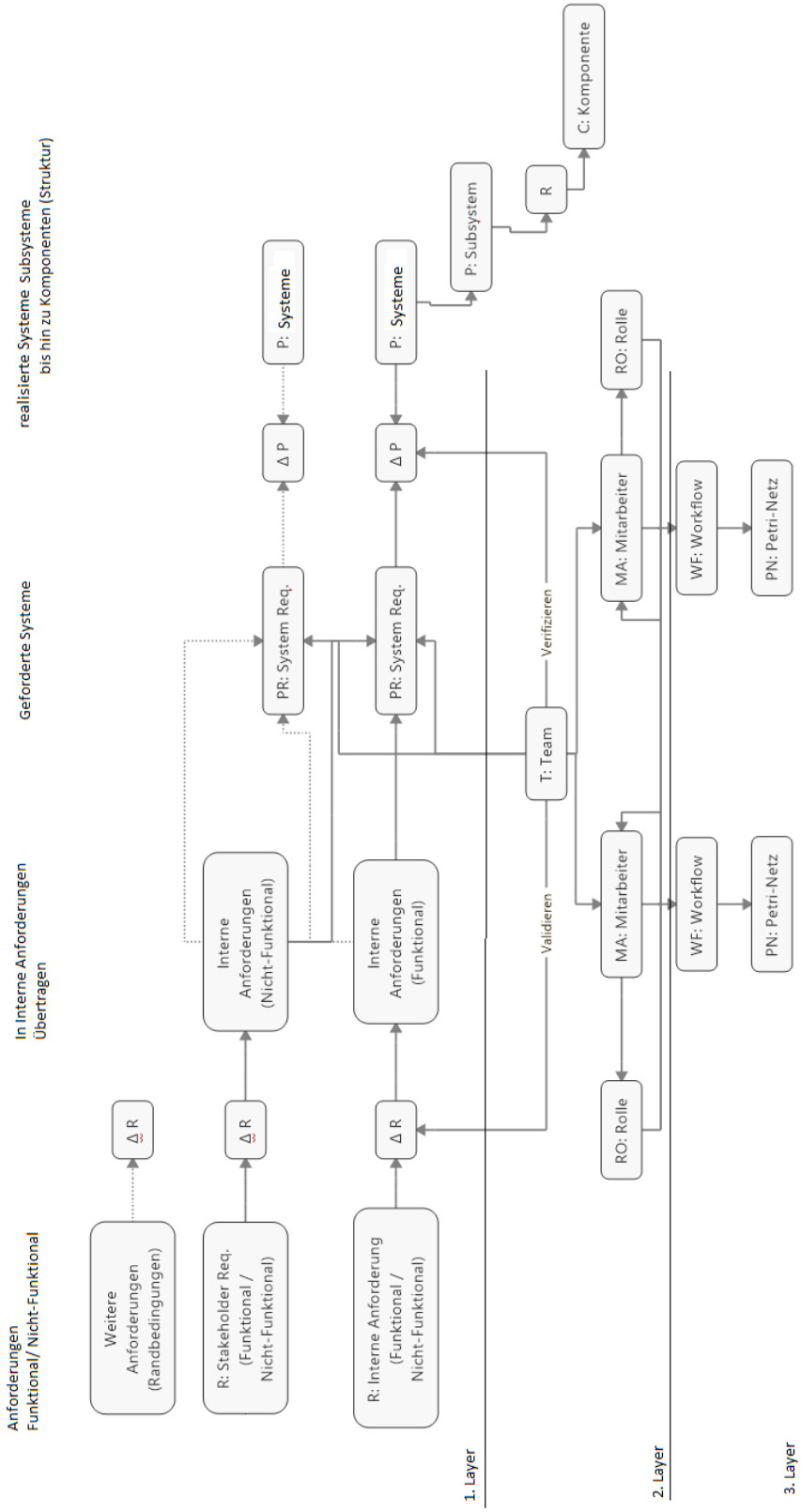
Weilkiens, T. (2007) *Systems Engineering with SysML/UML*, *Systems Engineering with SysML/UML*. doi: 10.1016/B978-0-12-374274-2.X0001-6.

Winzer, P. (2016) *Generic Systems Engineering*, *Generic Systems Engineering*. doi: 10.1007/978-3-662-52893-8.

Wynn, D. und Clarkson, J. (2005) „Models of designing“, in *Design Process Improvement: A Review of Current Practice*. doi: 10.1007/978-1-84628-061-0_2.

Anhang

Anhang 1: Das Drei-Layer-Daten-Informations-Framework



Anhang 2: Notation der Teilgraphen des Drei-Layer-Daten-Informations-Frameworks