Wissengraphbasierte Bewertung von Auswirkungsmustern zwischen Testergebnissen und Anforderungsänderungen im MBSE

Knowlegde graph supported assessment of impact patterns between test results and requirement changes in MBSE

Iris Gräßler[1], Marcel Ebel[1,*], Thomas Hesse[1]

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

** Korrespondierender Autor:*
Marcel Ebel
Heinz Nixdorf Institut, Universität Paderborn
Fürstenallee 11
33102 Paderborn
☎ *+49 5251 60 6263*
✉ *marcel.ebel@hni.upb.de*

Abstract

Existing approaches do not adequately address the estimation of engineering costs resulting from failed tests. An analysis of related work reveals three key deficits: insufficient investigation of test-induced changes, the lack of a systematic categorization of the scope of changes, and a failure to derive concrete action steps. This paper presents a knowledge graph-based approach that captures and evaluates impact patterns between test results and requirement changes. In a seven-step prescriptive research approach, impact patterns are derived from the literature, a criticality analysis is conducted, and, based on this, action measures are proposed. The approach supports engineers in estimating the potential for changes and the resulting effort at an early stage, thereby enabling them to make informed decisions about necessary engineering iterations resulting from test-induced changes.

Keywords

Verification & Validation, Impact Patterns, Test Results Analysis, Model-Based Systems Engineering, Knowledge Graph

 | DOI: 10.35199/dfx2026.9

1. Einleitung

Die Entwicklung komplexer technischer Systeme wird zunehmend durch Digital Engineering geprägt, wodurch eine durchgängige, modellbasierte Repräsentation von Entwicklungsartefakten ermöglicht wird [1]. Mithilfe von Model-Based Systems Engineering (MBSE) werden Abhängigkeiten zwischen diesen Artefakten strukturiert abgebildet [2], wodurch algorithmisch analysierbare Wirkungsbeziehungen identifiziert werden können. Beispiele für relevante Modellelemente sind Anforderungen, Anwendungsfälle, Testfälle und Testszenarien, deren Wechselwirkungen entscheidend zur Eigenschaftsabsicherung beitragen [3]. Tests dienen primär der Absicherung, können jedoch auch neue Anforderungen induzieren, Änderungen auslösen und sogar weitere Entwicklungsiterationen begründen. Eine frühzeitige Eigenschaftsabsicherung hilft dabei, Störungen und Fehlentwicklungen im Produktentwicklungsprozess zu identifizieren und trägt dadurch zur Reduktion von Entwicklungsaufwänden bei. Aktuell fehlen jedoch methodische Ansätze zur formalen Repräsentation und maschinellen Erkennung von Auswirkungsmustern zwischen Testfällen, Testszenarien und Anforderungsänderungen.

Obwohl etablierte Methoden zur modellbasierten Entwicklung (z.B. [2], [4], [5], [6]) existieren, ist Testen zum Wissensaufbau [7] und zur frühzeitigen Eigenschaftsabsicherung [8] nur unzureichend abgebildet. Durch frühzeitige Eigenschaftsabsicherung können Fehler in frühen Phasen der Entwicklung entdeckt und Entwicklungsiterationen begründet werden. Um Auswirkungen durchgeführter Tests zu bewerten, ist ein systematisches Vorgehen erforderlich. Voraussetzung dafür ist eine relationale Abbildung zwischen Entwicklungsartefakten. Ziel dieser Arbeit ist die Entwicklung eines Ansatzes zur maschinellen Identifikation, Klassifikation und Bewertung von Auswirkungsmustern als Grundlage menschlicher Entscheidungen in der Produktentwicklung. Der Fokus liegt dabei auf der Analyse von Testergebnissen und deren Auswirkungen auf den weiteren Verlauf der Produktentwicklung. Der entwickelte Ansatz zielt auf die modellbasierte Identifikation, Analyse und Bewertung von Auswirkungsmustern zwischen Testergebnissen und Anforderungsänderungen ab. Aus der Zielsetzung ergeben sich folgende Forschungsfragen (FF):

1. Wodurch lassen sich Abhängigkeiten zwischen Testfällen, Testszenarien, Testergebnissen und Anforderungsänderungen abbilden, um Wirkungspfade transparent darzustellen? **(FF1)**
2. Wodurch kann eine Bewertung der Kritikalität von Änderungswirkungen ausgehend von durchgeführten Tests dargestellt werden? **(FF2)**
3. Welche Handlungsmaßnahmen ergeben sich aus identifizierten Änderungen? **(FF3)**

2. Forschungsvorgehen

Zur Beantwortung der Forschungsfragen wird ein siebenschrittiges Forschungsvorgehen zugrunde gelegt, welches sich an der etablierten Design Research Methodology nach BLESSING und CHAKRARBARTI orientiert [9]. Zunächst werden verwandte Arbeiten in einer systematischen Literaturrecherche analysiert (1). Auf Grundlage der Literaturergebnisse werden Abhängigkeitsbeziehungen zwischen Anforderungen, Testfällen, Testszenarien sowie Testergebnissen formal definiert und Auswirkungsmuster identifiziert (2). Für die identifizierten Auswirkungsmuster wird eine Kritikalitätsanalyse vorgenommen (3). Um eine Entscheidungsunterstützung herbeizuführen, werden Handlungsmaßnahmen für Entwicklungsingenieure abgeleitet (4). Der Ansatz wird anschließend mithilfe eines Wissensgraphen prototypisch implementiert (5) und Ergebnisse in übersichtlichen Kennzahlenübersichten visualisert. Die Anwendbarkeit und Relevanz der implementierten Methode wird im Anschluss durch die Anwendung in einem Fallbeispiel untersucht (6) und kritisch diskutiert (7).

3. Grundlagen und Stand der Technik

Der nachfolgende Abschnitt befasst sich mit den Grundlagen, welche für die Ableitung und Bewertung von Auswirkungsmustern im Produktentwicklungsprozesses erforderlich sind. Zudem werden verwandte Arbeiten aus diesem Bereich durch eine systematische Literaturrecherche identifziert und im Anschluss analysiert.

3.1. Grundlagen

Die VDI/VDE 2206:2021 beschreibt die Verifikation als Prozess, bei dem der aktuelle Entwicklungsstand in Bezug auf die Systemanforderungen überprüft wird [10]. Verifikation ist entscheidend für die Freigabe zur Integration von Teilsystemen in höhere Systemebenen [10]. MBSE formalisiert den Entwicklungsprozess durch die Erstellung eines formalen Systemmodells unter Verwendung einer Modellierungssprache [11]. Für die Definition eines Systemmodells sind eine Modellierungsmethode, eine Modellierungssprache und ein Modellierungswerkzeug erforderlich [12]. Beispiele für Modellierungsmethoden sind die Object-Oriented Systems Engineering Method (OOSEM) [6], SysMOD [5] oder der Rational Unified Process for Systems Engineering (RUP-SE) [13]. Beispiele für Modellierungswerkzeuge sind Cameo Systèmes Modeler von D'assault Systèmes, Rhapsody von IBM und Enterprise Architect von Sparx Systems. Eine beispielhafte Modellierungssprache ist SysML, ein Profil der Unified Modeling Language (UML) 2.0 [14]. Die Erstellung eines Systemmodells mit SysML ermöglicht es dem Ingenieur, die Systemkomplexität des zu entwerfenden komplexen technischen Systems zu bewältigen [15], [16]. Im Rahmen der Produktplanung sowie in einer frühen Phase des Produktentwicklungs-prozesses erfolgt die Ableitung von Anforderungen aus den identifizierten Kundenbedürfnissen [17]. Diese Ableitung stellt den Ausgangspunkt für ein fundiertes Requirements Engineering dar. Innerhalb der modellbasierten Entwicklung finden Ansätze wie RFLPV^2 nach Gräßler et al. Anwendung [18]. Dieser Ansatz integriert die wesentlichen Entwicklungsartefakte Anforderungen (R), Funktionen (F), logische Elemente (L), physische Elemente (P) sowie Verifikation und Validierung (V&V) [18]. Der RFLPV^2-Ansatz bietet somit eine geeignete Grundlage zur Verknüpfung von Anforderungen mit den entwickelten Systemelementen sowie mit V&V-Artefakten wie Testfällen und Testszenarien. Ein Testfall ist dabei definiert als ein Verfahren, welches dazu dient, eine Eigenschaft eines Testobjekts zu überprüfen, die als Anforderung festgelegt wurde [3]. Die Beschreibung eines Testfalls muss eine Reihe von Vorbedingungen, Eingaben und erwarteten Ergebnissen enthalten. Sie kann erforderliche Testmethoden, Nachbedingungen, Testhilfsmittel sowie bestimmte Ziele umfassen [3]. Ein Testszenario hingegen ist eine Kombination aus zwei oder mehr Testfällen, die eine Ausführung unter denselben Umgebungsbedingungen gewährleisten [3]. Das Entwicklungsänderungsmanagement (engl.: Engineering Change Management) beschreibt den Prozess der Änderung von Entwicklungsartefakten. Nach JARRET [19] gliedert sich der generische Engineering-Change-Prozess in sechs Schritte mit vier Haltepunkten für Abstimmungen bzw. Freigaben sowie zwei Iterationsschleifen für die wiederholbare Lösungsfindung und Risikoanalyse; die Schritte lassen sich zudem danach unterscheiden, ob der Engineering Change vor der Beantragung, in der Abwicklung oder bereits im beantragten Stadium liegt.

3.2. Stand der Technik

Zur Identifikation von verwandten Ansätzen, welche die Identifikation und Bewertung von Auswirkungsmustern im Engineering Change Management behandeln, wird zur Dokumentation des Literatureanalyse-Prozesses das PRISMA-Vorgehen [20] verwendet. Mit der zu Beginn festgelegten Zielsetzung wird folgender Suchstring hergeleitet und in den Datenbanken Scopus, Web of Science, IEEE Xplore sowie Google Scholar verwendet:

("test" OR "Use Case" OR "requirement*" OR "Verification" OR "Validation") AND ("model*" OR "systems engineering" OR "Product development") AND ("impact analysis")*

Nach der Bereinigung um Dubletten reduziert sich der Datensatz von 2.065 auf 883 Publikationen. Diese werden in einem ersten Selektionsschritt einem Titel-Screening unterzogen, wodurch 55 potenziell relevante Beiträge identifiziert werden. Im darauffolgenden Abstract-Screening erfolgt eine inhaltliche Vorprüfung, sodass 22 Publikationen für die weitere Analyse verbleiben. Diese werden anschließend einer Volltext-Analyse unterzogen, in deren Rahmen 13 Arbeiten hinsichtlich ihrer methodischen und inhaltlichen Eignung eingehend geprüft werden. Sieben Publikationen erweisen sich dabei als relevant.

Im Folgenden werden die identifizierten Ansätze eingeordnet. Der Ansatz nach ELLSEL et al. [21] führt die Auswirkungsanalyse mittels eines SysML-Modells durch, indem Abhängigkeiten zwischen Entwicklungsartefakten sowie deren Änderungsaufwand über vier Änderungstypen kategorisiert und mit dem PERT-Ansatz berechnet werden. LI und CHEN [22] verfolgen hingegen einen musterbasierten Ansatz auf Basis der Designabhängigkeits-Matrix, bei dem entstehende Strukturmuster mit einer Bibliothek abgeglichen werden, um darauf aufbauend eine Neugestaltungsstrategie abzuleiten. REDDI und MOON [23] entwickeln ein Tool zum Management von Propagationseffekten, das auf einer während der Entwurfsphase aufgebauten Datenbank aus Änderungsklassen, Auslösern und Wahrscheinlichkeiten basiert und nach Eintritt einer Änderung automatisiert eine Liste betroffener Elemente ausgibt. INTANA et al. [24] analysieren die Auswirkung von Anforderungsänderungen auf Testfälle, indem Anwendungsfälle vor und nach der Änderung verglichen und darauf basierend betroffene Testfälle identifiziert oder neu abgeleitet werden. LIN et al. [25] kombinieren Modellgetriebenes Engineering mit dem Management von Änderungsanträgen in einem siebenstufigen, SysML-gestützten Vorgehen, das betroffene Anforderungen und Systemelemente über trace-, satisfy- und allocate-Relationen identifiziert und in einem Lösungsmodell zusammenführt. YANG und DUAN [26] stellen eine algorithmische Methode zur Identifikation von Propagationspfaden anhand von Parameter-Verbindungen in einem Netzwerkmodell vor, die iterativ Ziel- und direkte Parameter sowie deren Machbarkeit prüft. DEUBEL et al. [27] präsentieren mit dem CIRA-Verfahren einen sechsstufigen Ansatz zur Auswirkungsanalyse, der kritische Eigenschaften über Relevanzklassifikation, Lösungssynthese, Einflussanalyse und Risikobewertung untersucht.

In der Analyse der verwandten Arbeiten sind Analyse testinduzierter Änderungen, eine Kategorisierung des Änderungsumfangs sowie die Ableitung von Maßnahmen als Defizite identifiziert worden, welche in der Entwicklung des eigenen Ansatzes daher besonders berücksichtigt werden.

4. Ableitung und Bewertung von Auswirkungsmustern zwischen Testergebnissen und Anforderungsänderungen

Um aufbauend auf den identifizierten verwandten Arbeiten Auswirkungsmuster, deren Kritikalität sowie Handlungsmaßnahmen abzuleiten, wird der Ansatz zunächst die in [8] vorgestellte Methode zum test-orientierten resilienten Requirements Engineering (ToRRE) eingeordnet (siehe Abbildung 1). Die Methode besteht aus vier Schritten: der Planung von V&V-Aktivitäten (1), der Durchführung und Dokumentation von V&V-Aktivitäten (2), der Generierung von Thesen aus Testergebnissen (3) sowie der Analyse von Ergebnissen aus V&V-Aktivitäten. Der in dieser Arbeit beschriebene Ansatz setzt im vierten Schritt der ToRRE-Methode an. Geplante V&V-Aktivitäten sind bereits durchgeführt. Das Testergebnis ist dokumentiert und tatsächliche Ergebnisse des Testfalls im SysML-Modell abgebildet. Entspricht ein tatsächliches Ergebnis eines Tests dem erwarteten Ergebnis des Testfalls, so ist die zugrundeliegende Anforderung verifiziert [28]. Weicht das tatsächliche Ergebnis vom erwarteten Ergebnis jedoch ab, kann das einerseits trotzdem zur Validierung führen [28] andererseits auch Entwicklungsiterationen auslösen. Testinduzierte Änderungen prägen sich

durch die Änderung der Entwicklungsartefakte (Anforderungen, Anwendungsfälle, Funktionen, physische Elemente) aus. Durch die Verknüpfung der Entwicklungsartefakte nach dem RFLPV^2-Ansatz [18] werden die Wirkketten ausgehend von dem Testfall zurückverfolgt.

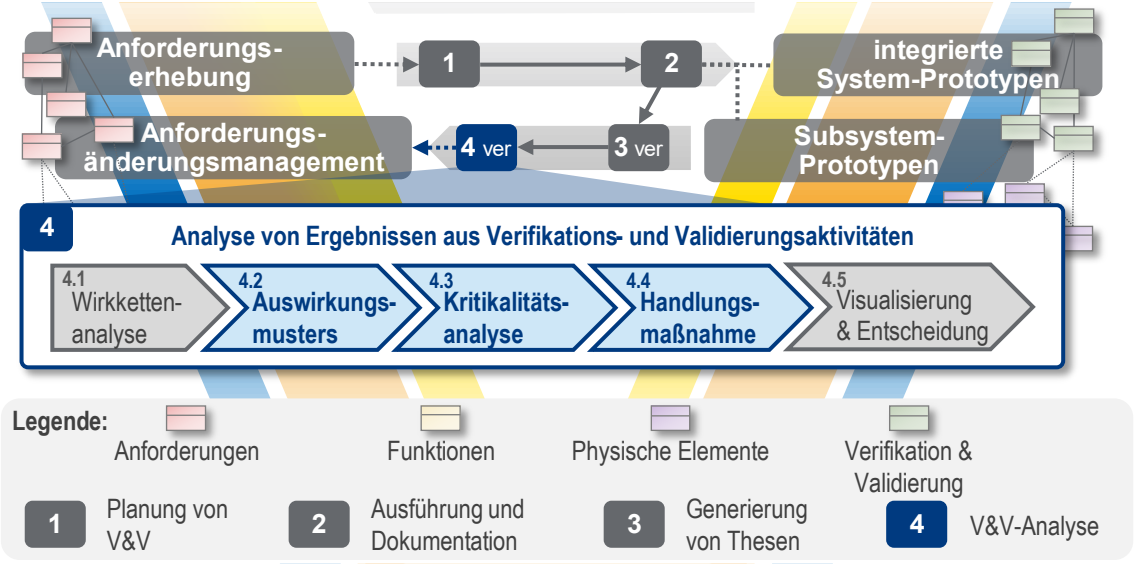

Abbildung 1: Einordnung des Ansatzes in die ToRRE-Methode nach GRÄßLER und EBEL [8]

Im vorgestellten Ansatz werden zunächst die Wirkketten auf Grundlage der MECA-Methode [11] analysiert. Betroffene Systemelemente werden durch Propagationspfade identifiziert. Die Auswirkungsmuster adressieren die möglichen Ausprägungen eines Propagationspfades ausgehend von einem durchgeführten Test. Daraus entstehen 20 verschiedene mögliche **Auswirkungsmuster**. Da keine bestehenden Ansätze vorliegen, werden diese Muster aus einer Kombination theoretischer und praktischer Modellierungsansätze entwickelt. Die Auswirkungsmuster basieren auf Entwicklungsartefakten und deren Relationen. Änderungen werden mithilfe von Testfällen analysiert, indem betroffene Artefakte – direkt oder indirekt – markiert, anschließend geclustert und zusammengeführt werden. Zur besseren Systematisierung erfolgt eine Klassifikation in direkte und indirekte Auswirkungsmuster. Direkte Auswirkungsmuster beziehen sich auf Testergebnisse, die sich über Anforderungen unmittelbar auf Bezugs-Testobjekte bzw. -Systemelemente auswirken. Im Gegensatz dazu beschreiben indirekte Auswirkungsmuster Effekte, die sich durch Propagationseinflüsse mittelbar auf Systemelemente übertragen. Die resultierenden Muster zeigen, wie sich Änderungen entlang von Abhängigkeiten propagieren. Zur strukturierten Darstellung werden Steckbriefe verwendet, die sich am Aufbau etablierter Systemelementbeschreibungen orientieren. Jeder Steckbrief enthält eine eindeutige Bezeichnung, die sich aus drei Komponenten zusammensetzt: der Kategorie (systemebenen-spezifisch oder systemebenen-übergreifend), der Art der Eigenschaftsabsicherung (Validierung oder Verifikation) sowie der Art der vorgesehenen Änderung. Im Zentrum steht die grafische Darstellung des Auswirkungsmusters entlang des V-Modells, die sowohl Auswirkungen innerhalb einer Ebene als auch über mehrere Systemebenen hinweg zeigt. Farbmarkierungen verdeutlichen das initiale Artefakt, die geplante Änderung sowie direkt und indirekt betroffene Elemente und definieren damit den Änderungspfad. Die Steckbriefe enthalten zudem Listen direkt betroffener Artefakte, aus denen sich weitere potenzielle Auswirkungsmuster ableiten lassen. Durch die Anzahl und Art der betroffenen Elemente kann das Änderungsverhalten eingeschätzt werden, insbesondere ob sich das Änderungsvolumen im Prozessverlauf erhöht. In Bezug auf das potenzielle Änderungsverhalten wird der Unterteilung von Eckert et al. in Konstanten, Absorber, Träger und Vervielfältiger gefolgt [29]. Ergänzend beinhalten die Steckbriefe konkrete Handlungsmaßnahmen, relevante Randbedingungen sowie eine Bewertung des jeweiligen Musters.

Die Bewertung der **Kritikalität** von Auswirkungen erfolgt nach dem Ansatz von DEUBEL et al. [27], bei dem ein Kennwert durch die Multiplikation mehrerer Faktoren bestimmt wird. Zur Ermittlung der Kritikalität werden drei Faktoren herangezogen (siehe : ein Auswirkungsmuster (Faktor A), eine Bewertung des Verhaltens des Systemelements nach [29] (Faktor B) sowie der Entwicklungsstand des Systems (Faktor C). Die Bewertung des Auswirkungsmusters erfolgt auf einer vierstufigen Skala von 1 bis 4. Dabei steht 1 für eine unkritische Änderung mit vernachlässigbaren Auswirkungen auf das System. Die Stufe 2 beschreibt gering kritische Änderungen mit begrenzten und gut beherrschbaren Auswirkungen. Eine Bewertung von 3 kennzeichnet kritische Änderungen, die signifikante Auswirkungen auf mehrere Systembereiche haben können und eine erhöhte Aufmerksamkeit erfordern. Die höchste Stufe 4 steht für sehr kritische Änderungen, die weitreichende, systemübergreifende Auswirkungen besitzen und potenziell die Funktionalität oder Sicherheit des Systems erheblich beeinträchtigen.

Tabelle 1: Faktoren und ihre Abstufungen zur Kritikalitätsbewertung

Faktor	1	2	3	4
A Auswirkungsmuster	nicht kritisch	geringfügig kritisch	kritisch	sehr kritisch
B Verhalten	Absorber	Konstante	Träger	Vervielfältiger
C Entwicklungsreife	A-Muster	B-Muster	C-Muster	D-Muster

Aus der Kombination dieser Faktoren wird eine Kritikalitätskennzahl berechnet, die zur Einordnung der Änderungsart dient. Durch Multiplikation kann ein Maximalwert von 64 erreicht werden. Eine Kritikalitätskennzahl von 1-4 entspricht dabei einer *einfachen Änderung*, 5-16 entspricht einer *umfassenden Änderung* und 17-64 resultiert in einer *komplexen Änderung*. Ziel der Bewertung ist dabei nicht die exakte Quantifizierung, sondern die qualitative Einschätzung der potenziellen Auswirkungen einer Änderung.

Um auf potenzielle Änderungen von Entwicklungsartefakten systematisch reagieren zu können, werden **Handlungsmaßnahmen** benötigt, die dem Entwickler ein standardisiertes Vorgehen für die jeweiligen Auswirkungsmuster bieten. Die Handlungsmaßnahmen werden in Form von Algorithmen umgesetzt. Diese Algorithmen sind durch Randbedingungen ergänzt, welche zusätzliche Anforderungen und Abhängigkeiten während der Durchführung spezifizieren. Insgesamt werden sieben Algorithmen verwendet, die in SysML-Aktivitätsdiagrammen modelliert sind. Die Algorithmen folgen dabei den Entwicklungsartefakten, die in den Auswirkungsmustern enthalten sind und führen zu einer Prüfung, ob das jeweilige Artefakt – bedingt durch die Änderung des auslösenden Artefakts infolge des Tests – angepasst werden muss. Der Ablauf folgt einem einheitlichen Schema: Zunächst wird geprüft, ob ein Entwicklungsartefakt bereits durch dieselbe Änderung bearbeitet wurde, um redundante Bearbeitung und Endlosschleifen zu vermeiden. Andernfalls wird die Änderung implementiert und hinsichtlich der weiterhin gewährleisteten Abbildung von Stakeholderanforderungen überprüft. Anschließend erfolgt eine rekursive Analyse verknüpfter Systemanforderungen, wobei entsprechende Algorithmen erneut ausgeführt werden. Nachgelagert werden verknüpfte Anwendungsfälle identifiziert und bei Bedarf durch weitere Algorithmen berücksichtigt. Der Prozess endet, sobald keine weiteren abhängigen Artefakte vorhanden sind. Zentral ist zudem die konsistente Anpassung der Eigenschaftsabsicherung sowie die Sicherstellung der Prüfbarkeit mit den vorgesehenen Methoden und Prüfständen. Die Randbedingungen definieren darüber hinaus relevante Abhängigkeiten zwischen Auswirkungsmustern und heben besonders kritische Artefakte hervor, insbesondere solche mit hohem Vervielfältigungspotenzial sowie abgeleitete und hierarchisch verknüpfte Artefakte.

5. Protoypische Implementierung

Die prototypische Implementierung der Wirkkettenanalyse erfolgt aufbauend auf der Wirkkettenanalyse nach WIECHEL [30]. Dem entwickelten SysML-Metadatenmodell muss zuvor der Testfall sowie das tatsächliche Testergebnis hinzugefügt werden, um ausgehend von einem Testergebnis Wirkketten zu analysieren. Die testorientierte Wirkkettenanalyse stellt dabei den vorbereitenden Ausgangspunkt für die Wahl des Auswirkungsmusters und somit der Kritikalitätsanalyse sowie dem Vorschlag der Handlungsmaßnahmen dar.

Das Metamodell folgt der formalen Notation eines attributierten, gerichteten Graphen: V bezeichnet die Menge der Knotentypen (Artefaktklassen), E die Menge der Kantentypen (Relationsklassen), ρ die konkrete Ausprägung der Relationen zwischen den Knotentypen, und σ die Attributzuordnung je Knotentyp. Zur Integration von Testfällen als eigenständige Klasse mit Testergebnis als Attribut sind Erweiterungen in V, σ sowie E und ρ erforderlich. Die Knotenmenge wird um die Klasse TestCase ergänzt: V = {Requirement, Function, LogicalElement, PhysicalElement, Verification, Validation, TestCase}. In σ wird dieser Klasse folgender Attributsatz zugeordnet: TestCase {testCaseID, testCaseName, testCaseDescription, testItem, testProcedure, testExecutionDate, testExecutedBy, testResult, testResultDetails}. Das Attribut testResult repräsentiert das Testergebnis (z. B. als Enum: passed, failed, blocked, not executed). Zur Einbettung in den bestehenden Graphen von WIECHEL [30] wird E um den Relationstyp Tests erweitert. Da Verification und Validation bereits die methodischen Prüfaktivitäten gegenüber Requirement repräsentieren, wird TestCase als deren konkrete Durchführungsebene modelliert: (rel15:Tests) = (TestCase, Verification), (rel16:Tests) = (TestCase, Validation).

Zur Identifikation aller Requirement-, LogicalElement- und PhysicalElement-Knoten, die mit Testfällen eines bestimmten testResult-Werts direkt oder indirekt verknüpft sind, wird folgende Cypher-Abfrage verwendet und an die Kennzahlenübersicht in Neo4j weitergegeben:

```
MATCH (tc:TestCase {testResult: "failed"})
MATCH path = (tc)-
[:Tests|Verifies|Validates|Satisfies|Specifies|Realizes|Requires|Refines|Derives|Decompose*1..10]-(element)
WHERE element:Requirement OR element:LogicalElement OR element:PhysicalElement
RETURN DISTINCT tc.testCaseID, element
```

Die Kennzahlenübersicht umfasst folgende Informationskategorien: Projektmetadaten, die Einordnung in die ToRRE-Methode, nutzerspezifische Daten, die Visualisierung der Wirkkettenanalyse in Form eines gerichteten Graphen, eine Auflistung der betroffenen Systemelemente, den Steckbrief des ausgewählten Auswirkungsmusters, eine Visualisierung der Kritikalität sowie eine Auflistung der vorgeschlagenen Handlungsmaßnahmen.

6. Anwendung im Fallbeispiel

Als Fallbeispiel dient dasFunktionsmodell einer Windkraftanlage. Eine exemplarisch betrachtete Anforderung ist die *Leistung der Windkraftanlage*. Aus der Gesamtleistung resultieren weitere Anforderungen, wie die *erforderliche Generatorleistung*, die *Flügellänge* sowie die *Größe des Turms* der Windkraftanlage. Das Systemelement *Generator* stellt das Testobjekt dar. Um die Erfüllung der Anforderung zu überprüfen, wird als Testfall eine *Leistungsmessung* im Testszenario *Belastung durch Wind* an einem physischen Funktionsprototypen durchgeführt. Das erwartete Ergebnis wird durch den Funktionstest nicht erreicht. Dies löst eine Anpassung der zugrundeliegenden Anforderung aus. Die *erforderliche Generatorleistung* wird erhöht. Dadurch wird eine Neudimensionierung des physischen Elements *Generator* begründet. Das physische Element *Generator* wird auf Nema 17 Schrittmotor geändert. Durch die Änderung ergibt sich der Propagationspfad entlang der Schnittstellenbauteile *Flügel*, *Rotornabe*, *Gondel* und *Turm*, welche ebenfalls angepasst werden müssen. Durch den Propagationspfad kann das Auswirkungsmuster *CL-SU-CP*

identifiziert werden. Das initial betroffene Systemelement ist das physische Element *Generator*. Durch Änderung des Attributs in der Anforderung auf Systemebene sind weitere Subsysteme Turm, Gondel und Flügel betroffen. Dadurch wird das potenzielle Änderungsverhalten nach ECKERT et al. [29] als „Vervielfältiger“ eingestuft. Die Kritikalitätsbewertung setzt sich wie folgt zusammen: Es handelt sich um ein kritisches Auswirkungsmuster, da mehrere Subsysteme von der Änderung betroffen sind. Daher wird Faktor A mit 3 bewertet. Faktor B wird als Vervielfältiger mit 4 bewertet. Da der Test mit einem Funktionsprototyp durchgeführt wurde, wird der Faktor C als B-Muster bzw. mit 2 bewertet. Aus dem Produkt der drei Faktoren ergibt sich dann eine Kritikalitätskennzahl von 24, was einer komplexen Änderung entspricht. Mit dem ausgewählten Auswirkungsmuster wird die entsprechende Handlungsmaßnahme in Form eines Algorithmus vorgeschlagen. Der Algorithmus startet mit der Prüfung, ob die Erhöhung der Nennleistung des Generators bereits im Rahmen des Änderungsprozesses berücksichtigt wurde, um Mehrfachbearbeitungen zu vermeiden. Nach Umsetzung der Leistungsanpassung werden die zugrunde liegenden Berechnungszusammenhänge analysiert, wobei sich zeigt, dass die erhöhte Nennleistung außerhalb bestehender Toleranzen liegt und somit eine Anpassung abhängiger Variablen erforderlich ist, was konkret die Neudimensionierung der Rotorblätter (Flügel), der Gondelstruktur sowie des Turms nach sich zieht. Im weiteren Verlauf werden diese abhängigen Attribute iterativ untersucht und optimiert, bis alle betroffenen Subsysteme konsistent angepasst sind und die Gesamtauslegung der Windkraftanlage wieder innerhalb zulässiger Betriebs- und Belastungsgrenzen liegt. Die nachfolgende Abbildung 2 zeigt schematisch die Auswirkungsanalyse nach dem durchgeführten Test.

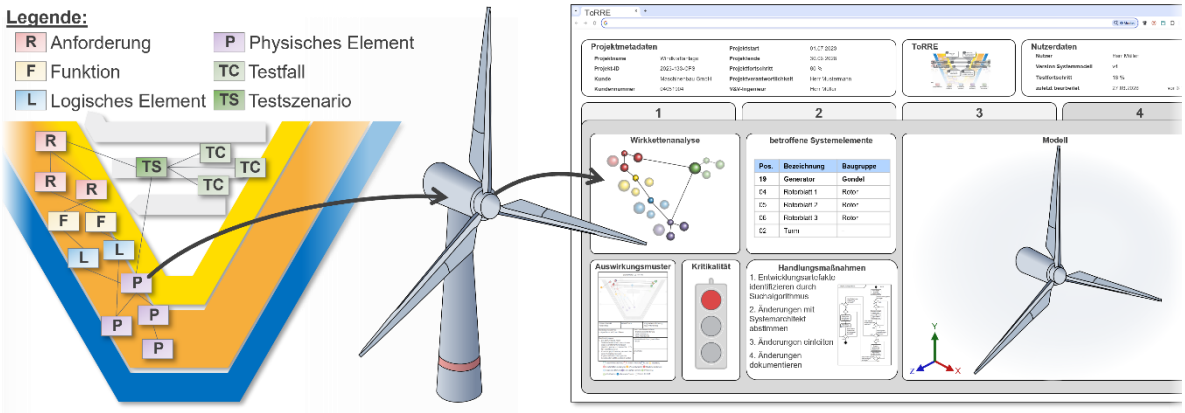

Abbildung 2: Analyse von Auswirkungsmustern auf Grundlage durchgeführter Tests

Ausgehend vom physischen Element *Generator* wird die wissengraphbasierte Auswirkungsanalyse genutzt, um die betroffenen Systemelemente *Flügel*, *Rotornabe*, *Gondel* und *Turm* zu identifizieren und das passende Auswirkungsmuster auszuwählen. Mit dem Auswirkungsmuster *CL-SU-CP* wird die Handlungsmaßnahme *Parameteränderung* vorgeschlagen und die Entscheidung des Ingenieurs dahingehend unterstützt, dass die Kritikalität und das Volumen der Änderung der Anforderung abschätzbar sind. Die Entscheidung, eine Entwicklungsiteration anzustoßen, liegt nach wie vor beim Ingenieur.

7. Diskussion

Der entwickelte Ansatz umfasst 20 aus der Literatur abgeleitete Auswirkungsmuster. Diese Auswirkungsmuster sind in standardisierten Steckbriefen dokumentiert, um eine konsistente Beschreibung und Vergleichbarkeit zu gewährleisten **(FF1)**. Eine prototypische Implementierung der vorbereitenden Wirkkettenanalyse ist durch die Graphdatenbank neo4j umgesetzt. Modellbasierte Entwicklungsartefakte werden dabei in Knoten abgebildet. Die Relationen zwischen Artefakten werden in Kanten dargestellt. Dadurch entsteht ein gerichteter

Graph, der die Wirkungspfade abbildet und automatisierte Abfragen durch Cypher ermöglicht. Die identifzierten Propagationspfade werden genutzt, um das passende Auswirkungsmuster auszuwählen. Durch eine qualitätive Bestimmung der Kritikaltität einer potenziellen Änderung, ist das Änderungsvolumen für den Ingenieur abschätzbar **(FF2)**. Generische Handlungsmaßnahmen werden in Form von Algorithmen vorgeschlagen **(FF3)**. Die generierten Informationen werden für den Ingenieur in übersichtlichen Kennzahlenübersichten zusammengeführt und bieten damit eine Entscheidungsunterstützung (siehe Abbildung 3). Am Fallbeispiel des Funktionsmodells wird die erfolgreiche Anwendung der wissensbasierten Bewertung von Auswirkungsmustern gezeigt.

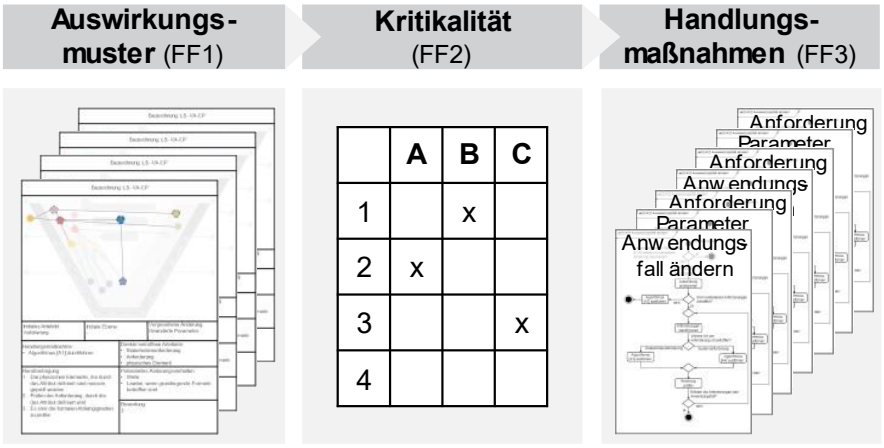

Abbildung 3: Auswirkungsmuster, Kritikalität und Handlungsmaßnahme als Entscheidungsunterstützung

Eine Limitation des Ansatzes liegt darin, dass Propagationspfade und Auswirkungsmuster derzeit manuell zusammengeführt werden müssen, was den Prozess aufwendig und potenziell fehleranfällig macht. Zudem bleiben die abgeleiteten Handlungsempfehlungen generisch und bieten nur begrenzte Unterstützung. Weiterhin berücksichtigt die Kritikalitätsbewertung keine projektbezogenen Entwicklungskosten, wodurch eine wirtschaftliche Einordnung von Änderungen erschwert wird. Nichtsdestotrotz stellt die qualitative Entscheidungsunterstützung eine wertvolle Hilfe für Ingenieure dar, um das Änderungsvolumen nach durchgeführten Tests fundiert abzuschätzen.

8. Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war die Entwicklung eines Ansatzes zur maschinellen Identifikation, Klassifikation und Bewertung von Auswirkungsmustern als Grundlage für menschliche Entscheidungen in der Produktentwicklung. Durch eine systematische Literaturanalyse wurden verwandte Arbeiten identifziert. Auf Grundlage der Literaturergebnisse sind 20 Auswirkungsmuster abgeleitet worden, die zeigen, wie sich testinduzierte Änderungen auf betroffene Entwicklungsartefakte auswirken. Weiter wurde eine qualitative Kritikalitätsanalyse entwickelt, um das Änderungsvolumen einschätzen zu können. Handlungsmaßnahmen wurden entwickelt, um dem Ingenieur Hilfestellungen zu geben Entwicklungsiterationen anzustoßen. Durch Implementierung in einem Wissensgraphen und exemplarische Anwendung auf eine Windkraftanlage wurde die Anwendbarkeit des Ansatzes gezeigt. Zuletzt wurde der Ansatz kritisch diskutiert und Limitationen wie die manuelle Zusammenführung von Propagationspfaden und generische Handlungsempfehlungen identifiziert. Diese Limitationen bieten das Potenzial für zukünftige Arbeiten. Der Ansatz wird in einen automatisierten Workflow – implementiert in Node-RED – überführt, wodurch ein regelbasierter Abgleich zwischen identifziertem Propagationspfad und Auswirkungsmuster hergestellt werden kann. Zudem können spezifischere Handlungsempfehlungen abgeleitet werden, indem eine Vektordatenbank mit Änderungen früherer Entwicklungsprojekte und einem Sprachmodell verknüpft werden.

Literaturverzeichnis

[1] B. Gerschütz *et al.,* “Digital Engineering Methods in Practical Use during Mechatronic Design Processes,” *Designs*, vol. 7, no. 4, p. 93, 2023, doi: 10.3390/designs7040093.

[2] A. M. Madni, N. Augustine, and M. Sievers, Eds. *Handbook of Model-Based Systems Engineering,* 1st ed. Cham: Springer International Publishing; Imprint Springer, 2022.

[3] I. Graessler, M. Ebel, and J. Pottebaum, “Model-based planning of test cases and test scenarios to support engineering of Cyber-Physical Systems,” in *IEEE ISSE 2024: 10 th IEEE International Symposium on Systems Engineering*, Perugia, Italy, 2024.

[4] W. D. Schindel, “Pattern-Based Methods and MBSE,” in *Handbook of Model-Based Systems Engineering*, A. M. Madni, N. Augustine, and M. Sievers, Eds., 1st ed. Cham: Springer International Publishing; Imprint Springer, 2022, pp. 1–45.

[5] T. Weilkiens, *Systems engineering mit SysML/UML: Modellierung, Analyse, Design,* 1st ed. Heidelberg, 2006

[6] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: The systems modeling language,* 3rd ed. Amsterdam, Waltham, Mass.: Elsevier; Morgan Kaufmann, 2015.

[7] S. Matthiesen and P. Grauberger, *Konstruktionswissen für Ingenieure: Innovative Produkte zielgerichtet entwickeln* (Lehrbuch). Berlin: Springer Vieweg, 2024.

[8] I. Graessler and M. Ebel, “Test-oriented Resilient Requirements Engineering (ToRRE): Extending Model-based Effect Chain Analysis to Verification Objectives,” in *Proceedings of the Design Society Volume 5: ICED25*, 2025, pp. 3031–3040.

[9] L. T. Blessing and A. Chakrabarti, *DRM, a Design Research Methodology*. London: Springer London, 2009.

[10] *VDI/VDE 2206 Entwicklung mechatronischer und cyber-physischer Systeme*, 2206, VDI/VDE, Düsseldorf, 2021.

[11] I. Gräßler, D. Wiechel, A.-S. Koch, D. Preuß, and C. Oleff, “Model-based effect-chain analysis for complex systems,” in *Proceedings of the Design*, Cavtat, Croatia, 2022, pp. 1885–1894, doi: 10.1017/pds.2022.191.

[12] L. Delligatti, *SysML distilled: A brief guide to the systems modeling language*. Upper Saddle River, 2014.

[13] Object Management Group. “IBM Rational Unified Process for Systems Engineering (RUP-SE).”,

[14] Object Management Group. “About the Unified Modeling Language Specification Version 2.5.1.”

[15] I. Gräßler and C. Oleff, *Systems engineering: Verstehen und industriell umsetzen*. Berlin, Heidelberg: Springer Vieweg, 2022.

[16] J. Pottebaum and I. Graessler, “Informationsqualität in der Produktentwicklung: Modellbasiertes Systems Engineering mit expliziter Berücksichtigung von Unsicherheit,” *Konstruktion - Zeitschrift für Produktentwicklung und Ingenieur-Werkstoffe*, 11-12, pp. 76–83, 2020.

[17] K. Pohl, *Requirements engineering: Fundamentals, principles, and techniques*. New York: Springer, 2010.

[18] I. Gräßler, D. Wiechel, and C. Oleff, “Extended RFLP for complex technical systems,” in *Proceedings of 8th IEEE International Symposium on Systems Engineering 2022*, Vienna, Austria, 2022

[19] T. Jarratt, J. Clarkson, and C. Eckert, “Engineering change,” in *Design process improvement: A review of current practice*, P. J. Clarkson and C. Eckert, Eds., London: Springer, 2005, pp. 262–285.

[20] M. J. Page *et al.,* “The PRISMA 2020 statement: an updated guideline for reporting systematic reviews,” *Systematic reviews*, vol. 10, no. 1, p. 89, 2021, doi: 10.1186/s13643-021-01626-4.

[21] C. Ellsel, G. Senkal, and R. Stark, “Using MBSE With SysML to Analyze Change Impacts, Including Efforts, in Technical Systems,” in *2022 IEEE International Symposium on Systems Engineering (ISSE)*, 2022

[22] S. Li and L. Chen, “Pattern-based reasoning for rapid redesign: a proactive approach,” *Res Eng Design*, vol. 21, no. 1, pp. 25–42, 2010. doi: 10.1007/s00163-009-0069-2.

[23] K. R. Reddi and Y. B. Moon, “A framework for managing engineering change propagation,” *IJIL*, 2009.

[24] A. Intana and T. Sriraksa, “Impact Analysis Framework of Test Cases Based on Changes of Use Case Based Requirements,” in *2019 23rd International Computer Science and Engineering Conference (ICSEC)*, 2019

[25] H.-Y. Lin, S. Sierla, N. Papakonstantinou, A. Shalyto, and V. Vyatkin, “Change request management in model-driven engineering of industrial automation software,” in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, 2015, pp. 1186–1191, doi: 10.1109/indin.2015.7281904.

[26] F. Yang and G. Duan, “Developing a parameter linkage-based method for searching change propagation paths,” *Res Eng Design*, vol. 23, no. 4, pp. 353–372, 2012. doi: 10.1007/s00163-011-0124-7.

[27] T. Deubel, J. Conrad, C. Köhler, S. Wanke, and C. Weber, *Change impact and risk analysis (CIRA) : combining the CPM/PDD theory and FMEA-methodology for an improved engineering change management*. Universität des Saarlandes, 2007.

[28] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, and T. M. Shortell, Eds. *Systems engineering handbook: A guide for system life cycle processes and activities*. Hoboken, N.J.: Wiley, 2015.

[29] C. Eckert, P. J. Clarkson, and W. Zanker, “Change and customisation in complex engineering domains,” *Res Eng Design*, vol. 15, no. 1, pp. 1–21, 2004, doi: 10.1007/s00163-003-0031-7.

[30] D. Wiechel, “Modellbasierte Wirkkettenanalyse zur Bewertung technischer Änderungen,” 2025.