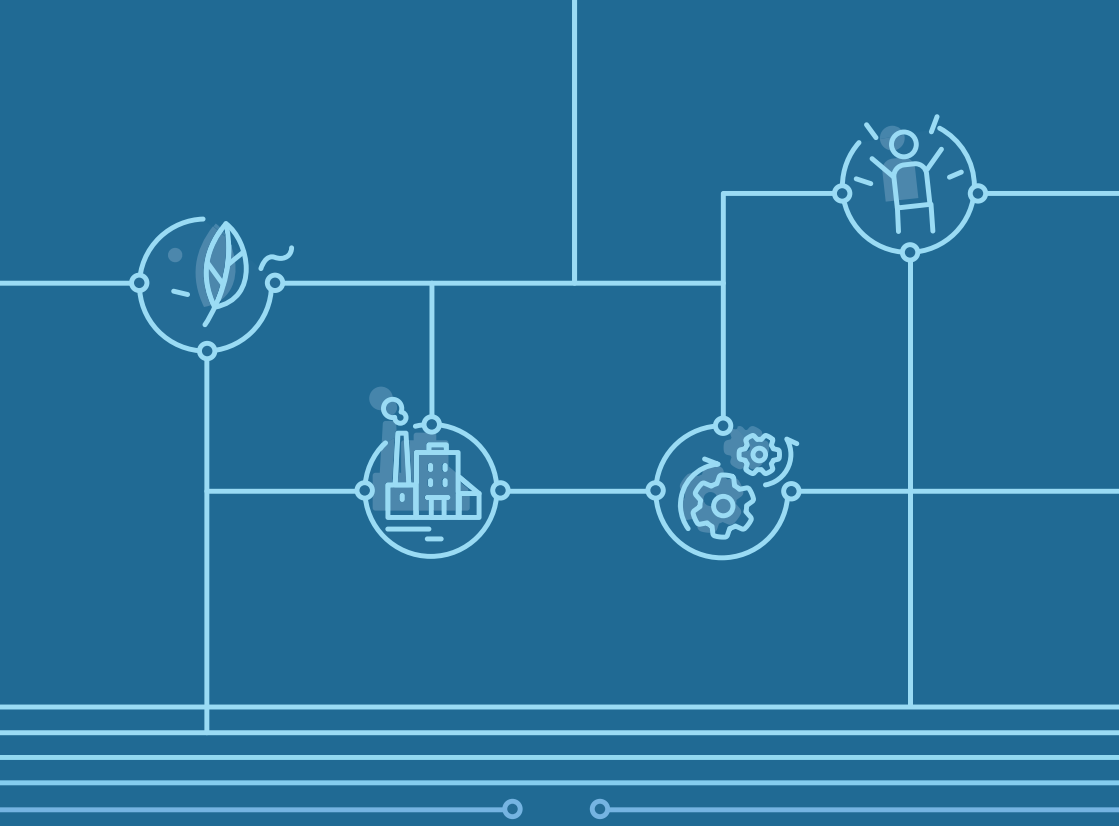


Nachhaltiges Geschäftsprozess-management mit JSON-Netzen

Andreas Fritsch

Nachhaltiges Geschäftsprozessmanagement mit JSON-Netzen

von Andreas Fritsch

Nachhaltiges Geschäftsprozess-management mit JSON-Netzen

Andreas Fritsch

Dissertation, genehmigt von der KIT-Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT), 2025

Tag der mündlichen Prüfung: 7. Oktober 2024
Referent: Prof. Dr. Andreas Oberweis
Korreferentin: Prof. Dr. Birgit Penzenstadler

Impressum

Autor: Andreas Fritsch
Covergrafik: Samuel Lutzweiler – https://animation.duwee.de/
Stand: 29. Oktober 2024

DOI: 10.5445/IR/1000178542

Kurzfassung

Angesichts sich verschärfender globaler Herausforderungen wie der Klimakrise wird die Forderung nach mehr Nachhaltigkeit von verschiedenen Seiten an Unternehmen herangetragen. Aus Unternehmenssicht geht es bei dieser Forderung darum, (negative) Auswirkungen der eigenen Geschäftsprozesse auf Mensch und Umwelt zu verringern. Damit Unternehmen die Nachhaltigkeitsauswirkungen ihrer Geschäftsprozesse effektiv managen und verbessern können, müssen in den Informationssystemen der Unternehmen Nachhaltigkeitsdaten in geeigneter Form für Entscheiderinnen und Entscheider zur Verfügung gestellt werden.

Methoden und Werkzeuge des Nachhaltigen Geschäftsprozessmanagements können Unternehmen bei dieser Aufgabe unterstützen. Bestehende Ansätze des Nachhaltigen Geschäftsprozessmanagements erlauben jedoch nur eine eingeschränkte Betrachtung möglicher Nachhaltigkeitsauswirkungen. In der vorliegenden interdisziplinären Arbeit werden daher systematisch Konzepte der Nachhaltigkeitsanalyse, des Geschäftsprozessmanagements und der Informatik verknüpft. Das Ergebnis des verfolgten gestaltungsorientierten Forschungsansatzes ist eine Modellierungsmethode, die aus drei Komponenten besteht: (1) einem Vorgehensmodell, (2) einer Modellierungssprache und (3) einem Modellierungswerkzeug.

Die erste Komponente, das Vorgehensmodell, beschreibt, wie Unternehmen die Nachhaltigkeitsauswirkungen ihrer Geschäftsprozesse analysieren und verbessern können. Mit JSON-Netzen wird als zweite Komponente eine neue, formale und datenorientierte Prozessmodellierungssprache vorgestellt. Sie unterstützt sowohl die Simulation und Ausführung von Geschäftsprozessen als auch den Austausch von (Nachhaltigkeits-)Daten mit (externen) Informationssystemen. Das Modellierungswerkzeug stellt als dritte Komponente Mechanismen zur Anwendung der Prozessmodellierungssprache und des Vorgehensmodells bereit. Die Modellierungsmethode wurde im praktischen Einsatz in Nachhaltigkeitsschulungen für Unternehmen evaluiert. Sie ermöglicht eine umfangreiche Betrachtung von Nachhaltigkeitsauswirkungen und bietet darüber hinaus IT-Unterstützung für das Management und die Verbesserung der Nachhaltigkeitsauswirkungen von Geschäftsprozessen.

Inhaltsverzeichnis

Diskussion der Ergebnisse

Notation

Netzstrukturen

S	Menge von Stellen einer Netzstruktur
T	Menge von Transitionen einer Netzstruktur
F	Flussrelation einer Netzstruktur (Menge von Kanten)
s	Stelle (Element der Menge von Stellen einer Netzstruktur)
t	Transition (Element der Menge von Transitionen einer Netzstruktur)
(x,y)	Kante (Element der Flussrelation einer Netzstruktur)
${}^\bullet x$	Vorbereich einer Stelle oder Transition
$x^\bullet$	Nachbereich einer Stelle oder Transition

JSON-Datenmodell

d	JSON-Dokument
j	JSON-Baum
w	Pfadausdruck
$\mathit{depth}(w)$	Anzahl der Teil-Zeichenketten eines Pfadausdrucks
$\mathit{key}(w)$	Letzte Teil-Zeichenkette eines Pfadausdrucks
$\mathit{parent}(w)$	Pfadausdruck ohne die letzte Teil-Zeichenkette
$\mathit{env}(w)$	Erste Teil-Zeichenkette eines Pfadausdrucks
p	Filterausdruck oder Schema
$p(d)$	Ergebnis eines Filterausdrucks (Menge von Pfadausdrücken)

JSON-Netze

$SB(s)$	Stellenbeschriftung der Stelle s
$KB(x,y)$	Kantenbeschriftung der Kante (x,y)
$TB(t)$	Transitionsbeschriftung der Transition t
$M(s)$	Markierung der Stelle s
p_s	Schema der Stelle s
$p_{x,y}$	Filterausdruck der Kante (x,y)
$v_{x,y}$	Wertvariable der Kante (x,y)
$k_{x,y}$	Schlüsselvariable der Kante (x,y)
$d_{x,y}$	Dokumentvariable der Kante (x,y)
$\beta(p)$	Belegung eines Filterausdrucks (Pfadausdruck)
$\beta(k)$	Belegung einer Schlüsselvariablen (Wert vom Typ String oder Number)
$\beta(v)$	Belegung einer Wertvariablen
$\beta(d)$	Belegung einer Dokumentvariablen

Einführung und Grundlagen

1 Einleitung

Zur Einführung in die vorliegende Arbeit wird zunächst das Thema Nachhaltiges Geschäftsprozessmanagement motiviert[1] und die Zielsetzung der Arbeit vorgestellt. Es folgen eine Beschreibung der Forschungsmethode, mit der die Zielsetzung bearbeitet wurde, und eine Übersicht der Struktur und Zusammenhänge der einzelnen Kapitel.

1.1 Motivation und Zielsetzung

Die Forderung nach *mehr Nachhaltigkeit* wird aktuell von verschiedenen Seiten an Unternehmen herangetragen, sei es von der Politik, Nichtregierungsorganisationen, Kundinnen und Kunden oder der eigenen Belegschaft [Hal19, Amr14, Ern22]. Der Begriff Nachhaltigkeit oder Nachhaltige Entwicklung steht dabei für eine gesamtgesellschaftliche und langfristige Perspektive mit dem Ziel, angesichts sich verschärfender globaler Herausforderungen wie der Klimakrise, menschliche Bedürfnisse heute und in Zukunft zu befriedigen [Uni87]. Bei der Erreichung dieses Ziels werden Fragen des Umweltschutzes und sozialer Gerechtigkeit als zentral erachtet [Uni87]. Aus Unternehmenssicht geht es bei der Forderung nach mehr Nachhaltigkeit also um (negative) Auswirkungen des unternehmerischen Handelns auf Mensch (soziale Gerechtigkeit) und Umwelt (Umweltschutz). In der Literatur finden sich Beispiele für die Verwendung des Begriffs Nachhaltigkeit, die nicht dem hier vertretenen Begriffsverständnis entsprechen: beispielsweise, wenn von „nachhaltigen Wettbewerbsvorteilen“, oder „nachhaltigem Unternehmenswachstum“ gesprochen wird und dabei lediglich ein langfristiger finanzieller Erfolg des Unternehmens gemeint ist (zum Beispiel [Hua15, Rah17]). Es kann durchaus Zusammenhänge und Synergien zwischen beispielsweise Umweltschutz und dem finanziellen Erfolg eines Unternehmens geben [Mir18, Duq21, Hol01]. Doch aus Sicht einer globalen Nachhaltigen Entwicklung, wie sie dem Nachhaltigkeitsverständnis im weiteren Verlauf dieser Arbeit zugrunde gelegt wird, geht es nicht um den langfristigen Erfolg eines Unternehmens, sondern um den langfristigen Erhalt

[1] Teile der Einleitung wurden aus den Veröffentlichungen [Fri23a, Fri22] übernommen.

des Planeten und seiner Fähigkeit, menschliche Bedürfnisse zu befriedigen [Hil15]. Der Fokus liegt also im Folgenden darauf, wie ein Unternehmen seinen Beitrag zu (globaler) sozialer Gerechtigkeit und (globalem) Umweltschutz verstehen und verbessern kann. Eine Konsequenz aus der hier vertretenen Auffassung des Nachhaltigkeitsbegriffs ist, dass sich der Verantwortungsbereich für unternehmerisches Handeln erweitert [Hei02b]. Es reicht nicht aus, nur die unmittelbaren Auswirkungen innerhalb der Unternehmensgrenzen, wie beispielsweise Löhne an die eigene Belegschaft oder Schadstoffemissionen auf dem Betriebsgelände, zu betrachten. Vielmehr müssen auch die Auswirkungen der durch die unternehmerischen Handlungen ausgelösten Stoffströme berücksichtigt werden [Möl95].[1] Zum Beispiel trägt ein Unternehmen, das einen Server betreibt, auch Verantwortung für die Treibhausgasemissionen, die durch die Erzeugung der für den Betrieb des Servers benötigten Energie in einem Kohlekraftwerk entstehen. Ebenso trägt ein Unternehmen, das Batterien herstellt, Verantwortung für den Lithium-Abbau und die damit einhergehende Kontamination der Wasservorkommen lokaler Gemeinschaften [Deu19]. Dieser Gedanke ist ein zentraler Bestandteil international anerkannter und standardisierter Methoden der Nachhaltigkeitsanalyse, wie dem Treibhausgasprotokoll und der Lebenszyklusanalyse [ISO06a, UNE15, WBC04]. Er findet sich auch in aktueller Nachhaltigkeitsgesetzgebung wie dem Lieferkettengesetz[2], das von Unternehmen verlangt, Rechenschaft über soziale und ökologische Auswirkungen entlang ihrer Lieferkette abzulegen.

Eine weitverbreitete Anwendung des Konzepts der Nachhaltigen Entwicklung auf den Unternehmenskontext ist das der „Triple Bottom Line“ [Elk97]. Demnach sollen Unternehmen nicht nur über ihre finanziellen Kennzahlen Rechenschaft ablegen, sondern auch über die sozialen und ökologischen Auswirkungen ihrer Tätigkeiten berichten. Während das Konzept breite Anwendung in der Nachhaltigkeitsdiskussion gefunden hat, wurde es zuletzt vom ursprünglichen Autor selbst kritisiert. So merkt [Elk18] an, dass das Konzept hauptsächlich dazu verwendet wird, Berichte für externe Interessengruppen zu schreiben, anstatt notwendige Daten zu Entscheiderinnen und Entscheidern im Unternehmen zu bringen und die Nachhaltigkeitsauswirkungen der Unternehmensaktivitäten nachzuverfolgen und zu verstehen. Es kann hier festgehalten werden, dass weniger das Konzept selbst kritisiert wird, als vielmehr die Tatsache, dass sich das Verhalten von Unternehmen nicht (in der notwendigen Art und Weise) verändert hat. In anderen Worten: die weite Verbreitung des *Triple Bottom Line*-Konzepts

[1] Der Begriff Stoffstrom wird hier in einem umfassenden Sinn verwendet und meint den Austausch von Materialien, Energie und Produkten von Unternehmen entlang von Wertschöpfungsketten und mit der natürlichen Umwelt. Eine umfangreichere Begriffsklärung findet in Unterabschnitt 3.3.2 statt.

[2] Lieferkettensorgfaltspflichtengesetz vom 16. Juli 2021 (BGBl. I S. 2959).

zeigt, dass bei vielen Unternehmen ein Bewusstsein für Nachhaltigkeitsthemen entstanden ist. Die aktuelle Herausforderung besteht nun darin, nicht nur gesetzliche Berichtspflichten zu erfüllen, sondern die Nachhaltigkeitsauswirkungen des unternehmerischen Handelns effektiv zu managen.

Der Leitgedanke dieser Arbeit ist, dass das Geschäftsprozessmanagement als Forschungsdisziplin und Anwendungsgebiet hier einen wichtigen Beitrag leisten kann. Ziel des Geschäftsprozessmanagements ist es, die Aktivitäten von Unternehmen zu verstehen und zu verbessern [Wes19]. Während hier traditionell Verbesserungen in Bezug auf Prozesskosten, Durchlaufzeiten und Fehlerraten angestrebt werden [Dum18], haben in den letzten Jahren Forschende unter dem Schlagwort Grünes oder Nachhaltiges Geschäftsprozessmanagement damit begonnen, auch Nachhaltigkeitsaspekte in ihre Ansätze miteinzubeziehen [Cou19a, Bet14]. Der Beitrag der vorliegenden Arbeit liegt darin, dass sie systematisch Konzepte der Nachhaltigkeitsanalyse mit Konzepten des Geschäftsprozessmanagements und der Informatik verknüpft. Die Konzepte der Nachhaltigkeitsanalyse helfen dabei, belastbare Nachhaltigkeitsdaten zu erheben. Diese Nachhaltigkeitsdaten müssen schließlich aggregiert, analysiert und den Entscheiderinnen und Entscheidern auf verschiedenen Ebenen des Unternehmens zur Verfügung gestellt werden, um eine effektive Steuerung der Unternehmensaktivitäten zu ermöglichen. Konzepte des Geschäftsprozessmanagements und der Informatik unterstützen dabei, Informationssysteme so zu gestalten, dass diese Daten zur Steuerung und Verbesserung der Unternehmensaktivitäten genutzt werden können.

Zur Umsetzung dieses Leitgedankens wird eine Modellierungsmethode [Kar02] für Nachhaltiges Geschäftsprozessmanagement entworfen. Diese Modellierungsmethode soll Unternehmen bei der Verbesserung der sozialen und ökologischen Auswirkungen ihrer Geschäftsprozesse unterstützen. Sie besteht aus drei Komponenten, (1) einem Vorgehensmodell, (2) einer Modellierungssprache und (3) einem unterstützenden Modellierungswerkzeug:

1 Das *Vorgehensmodell* beschreibt allgemein und unabhängig von konkreten Modellierungssprachen oder -werkzeugen, wie Unternehmen ihre Geschäftsprozesse und deren soziale und ökologische Auswirkungen modellieren, analysieren und verbessern können.

2 Mit JSON-Netzen wird eine *Modellierungssprache* bereitgestellt, welche sowohl die Darstellung der betrachteten Phänomene (Geschäftsprozesse) als auch die Integration und den Austausch von (Nachhaltigkeits-)Daten unterstützt.

3 Das *Modellierungswerkzeug* stellt unterstützende Mechanismen zur Anwendung der Modellierungssprache bereit, um anhand des Vorgehensmodells Geschäftsprozesse unter Einbeziehung ihrer ökologischen und sozialen Auswirkungen zu modellieren und zu analysieren.

Im folgenden Abschnitt werden in der Literatur empfohlene Vorgehensweisen zur Entwicklung von Modellierungsmethoden vorgestellt und ihre Anwendung in der vorliegenden Arbeit dargelegt.

1.2 Forschungsmethode und Kapitelstruktur

Die Arbeiten [Kar15, Kar16, Fra13] beschreiben Vorgehensweisen und Richtlinien für die Erstellung von Modellierungsmethoden beziehungsweise Modellierungssprachen. Das Vorgehen bei der Entwicklung einer Modellierungsmethode kann nach [Kar15, Kar16, Öst11] in die Phasen (1) Analyse, (2) Entwurf, (3) Implementierung und (4) Evaluation unterteilt werden. Wie in Tabelle 1.1 dargestellt, lassen sich die Ergebnisse und Kapitel der vorliegenden Arbeit den verschiedenen Phasen zuordnen. Wie [Fra13] festhält, sollten die Empfehlungen der Quellen nicht als „Kochbücher“ angesehen werden, denen strikt zu folgen ist. Vielmehr sind sie als Orientierungshilfe zu sehen, müssen reflektiert und an die jeweilige Situation angepasst werden. Im Folgenden werden die Phasen und Ergebnisse vorgestellt und es wird dargelegt, welche Empfehlungen aus den genannten Quellen bei den Arbeiten herangezogen wurden.

Tabelle 1.1: Übersicht der Teilergebnisse und Kapitelstruktur.

Phase	Ergebnisse	Kap.
Analyse	Grundlagen des Geschäftsprozessmanagements	2
	Grundlagen der Nachhaltigkeitsanalyse	3
	Definition für Nachhaltiges Geschäftsprozessmanagement	4
	Verwandte Arbeiten	5
	Anforderungsspezifikation	6
Entwurf	Vorgehensmodell	7
	Modellierungssprache	8
Implementierung	Modellierungswerkzeug	9
Evaluation	Beispielszenario	10
	Technische Evaluation und Fallstudie	11
	Beitrag und Ausblick	12

In der *Analyse*-Phase geht es nach [Kar15, Kar16] um das Sammeln von Wissen und die Erhebung von Anforderungen. Das gesammelte Wissen, das die Grundlage der vorliegenden Arbeit darstellt, ist in den Kapiteln zu Geschäftsprozessmanagement und Nachhaltigkeitsanalysen (Kapitel 2, 3 und 4) beschrieben. Weiterhin wurden bestehende verwandte Arbeiten untersucht (Kapitel 5). Zur Ermittlung von verwandten Arbeiten wurde eine systematische Literaturrecherche nach [Kit07] durchgeführt. Auf methodische Details der Literaturrecherche wird im entsprechenden Kapitel eingegangen. Basierend auf den Grundlagen und verwandten Arbeiten werden in Kapitel 6 Anforderungen an die Modellierungsmethode spezifiziert. Entsprechend der Empfehlung in [Fra13] findet dabei eine Klärung des Umfangs und Zwecks der Modellierungsmethode statt.[1] Die Anforderungsspezifikation stellt das zentrale Ergebnis der Analyse-Phase dar und bildet eine Brücke zwischen den Kapiteln 2 bis 5 und den folgenden Arbeitsergebnissen.

Nach [Kar15, Kar16] wird in der *Entwurf*-Phase die Modellierungssprache und das Vorgehensmodell einer Modellierungsmethode spezifiziert und präzise beschrieben. Gestaltungsentscheidungen sollten dabei in einer strukturierten Form festgehalten und begründet werden [Fra13]. In den Kapiteln 7 und 8 werden das entworfene Vorgehensmodell und die Modellierungssprache sowie die getroffenen Gestaltungsentscheidungen vorgestellt. In der Phase *Implementierung* erfolgt schließlich die Implementierung eines Modellierungswerkzeugs für die Modellierungsmethode [Kar15, Kar16, Fra13], das in Kapitel 9 vorgestellt wird. Weiterhin wird empfohlen, die Entwicklung einer Modellierungsmethode mit Beispielszenarios zu unterstützen [Fra13]. Entsprechend wird in Kapitel 10 ein Beispielszenario beschrieben, das die Anwendung der bereitgestellten Modellierungsmethode veranschaulicht.

Mit dem Modellierungswerkzeug kann die Modellierungsmethode in der Phase *Evaluation* potentiellen Anwendenden vorgestellt werden und die Erfüllung der aufgestellten Anforderungen überprüft werden [Kar15, Kar16, Fra13]. Im Rahmen der Evaluation sollen darüber hinaus Verbesserungspotentiale für zukünftige Arbeiten identifiziert werden [Fra13]. Das Beispielszenario in Kapitel 10 dient zum einen der Veranschaulichung der Modellierungsmethode und zum anderen als erste Evaluation, indem die Anwendbarkeit der Modellierungsmethode gezeigt wird. Daneben wurden weitere Evaluationsmethoden zur technischen Evaluation eingesetzt sowie eine Fallstudie durchgeführt, deren Ergebnisse in Kapitel 11 beschrieben werden. Das verfolgte

[1] [Fra13] unterscheidet nicht wie [Kar02] zwischen Modellierungs*sprache* und Modellierungs*methode*. Im Wortlaut adressiert [Fra13] nur die Gestaltung von Modellierungssprachen, die Empfehlungen lassen sich aber auf die Gestaltung von Modellierungsmethoden im Sinn von [Kar02] übertragen.

Vorgehen bei der Evaluation sowie die eingesetzten Evaluationsmethoden werden im entsprechenden Kapitel näher beschrieben. Beiträge und ein Ausblick zu Weiterentwicklungen und anknüpfenden Arbeiten werden schließlich in Kapitel 12 zusammengefasst und vorgestellt.

2 Grundlagen des Geschäftsprozessmanagements

In Abschnitt 2.1 wird zur Darlegung der Grundlagen der vorliegenden Arbeit das Forschungs- und Anwendungsgebiet Geschäftsprozessmanagement vorgestellt. Es folgt in Abschnitt 2.2 eine tiefergehende Beschreibung der Begriffe und Konzepte des Geschäftsprozessmanagements zur Vorbereitung der Gegenüberstellung von Geschäftsprozessmanagement und Nachhaltigkeitsanalyse in Kapitel 4. Als Grundlage für die Entwicklung einer Prozessmodellierungssprache (siehe Kapitel 8) wird in Abschnitt 2.3 ein Überblick über bestehende Prozessmodellierungssprachen gegeben.

2.1 Betrachtungsgegenstand

Im Geschäftsprozessmanagement geht es in der Forschung und Anwendung darum, die Geschäftsprozesse, das heißt die in einem Unternehmen ausgeführten Aktivitäten [Obe96a],[1] und damit die Leistung eines Unternehmens zu verbessern [Wes19, Dum18]. [Har15] zeichnet die Geschichte des Geschäftsprozessmanagements über historische Figuren wie Henry Ford, und einflussreiche Veröffentlichungen wie Taylors *Principles of Scientific Management* von 1911 [Tay11] und Porters *Competitive Advantage* von 1985 [Por85] nach. Dabei ist festzuhalten, dass, lange bevor der Begriff „Geschäftsprozessmanagement“ geprägt wurde, viele Personen in Leitungsfunktionen darum bemüht waren, die Leistung ihrer Mitarbeiterinnen und Mitarbeiter zu verbessern. So gesehen ließe sich die Geschichte des Geschäftsprozessmanagements bis in die Antike zurückverfolgen. In der jüngeren Geschichte jedoch identifiziert [Har15] drei Traditionen des Geschäftsprozessmanagements, jede mit ihren eigenen Zielgruppen und Methoden. Zur Einordnung dieser Arbeit in den wissenschaftlichen Kontext wird die Unterscheidung zwischen der Qualitätssicherungs-Tradition, der Management-Tradition und der IT-Tradition des Geschäftsprozessmanagements nach [Har15] im Folgenden knapp wiedergegeben. Abbildung 2.1 dient dabei der Orientierung und stellt den zeitlichen Ablauf der Entwicklung der Traditionen dar.

[1] Die Definition des Begriffs Geschäftsprozess wird in Unterabschnitt 2.2.1 vertieft betrachtet.

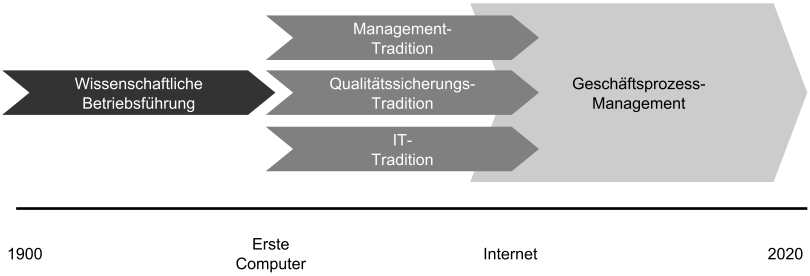


Abbildung 2.1: Zeitliche Entwicklung des Geschäftsprozessmanagements, in Anlehnung an [Har15, S. 39].

Die *Qualitätssicherungs-Tradition* hat sich aus der Wissenschaftlichen Betriebsführung [Tay11] entwickelt und umfasst Methoden wie Total Quality Management und Six Sigma [Ten01]. Wie der Name schon sagt, sind Beschäftigte in der Qualitätssicherung sowie Ingenieurinnen und Ingenieure die Zielgruppe dieser Tradition, wohingegen auf Management-Ebene in Unternehmen Methoden aus der *Management-Tradition* bevorzugt werden. Im Gegensatz zur Qualitätssicherungs-Tradition wird hier weniger die Leistung einzelner Herstellungsprozesse, als vielmehr das gesamte Unternehmen in den Blick genommen. Typische Methoden und Konzepte dieser Tradition sind Porters Value Chain [Por85], die Balanced Scorecard [Kap96] sowie Methoden und Konzepte zur Prozessverbesserung nach [Rum90]. Die *IT-Tradition* des Geschäftsprozessmanagements hat sich mit dem Aufkommen der Informationstechnologie und insbesondere der Verbreitung des Internets entwickelt. Der Fokus liegt hier auf der Automatisierung und Unterstützung von Geschäftsprozessen mit Hilfe von Informationssystemen.

Schließlich betont [Har15], dass Unternehmen in unserer modernen, immer komplexer werdenden Welt nicht erfolgreich sein können, wenn sie nur in Methoden einer dieser drei Traditionen stark sind: Prozessüberwachung allein, ohne Blick auf die beteiligten Beschäftigten, ist nicht ausreichend. Ein Managementansatz kann schnell von modernen Technologien überholt werden. Und auch die Verbesserung eines Prozesses erfordert eine klare Vorstellung davon, wie der Prozess zum Erfolg des gesamten Unternehmens beiträgt. Aus diesem Grund gibt es nach [Har15] vermehrt Bestrebungen, die verschiedenen Traditionen zusammenzuführen. In Abbildung 2.1 ist die Zusammenführung dadurch dargestellt, dass das Feld Geschäftsprozessmanagement die Pfeile der drei vorangegangenen Traditionen umfasst. Diese Gesamtsicht auf Unternehmen ist es, was [Har15] unter „modernem“ Geschäftsprozessmanagement versteht.

Konzepte, Werkzeuge und Methoden des Geschäftsprozessmanagements können in Management-orientierte Ansätze und technischen Ansätze eingeteilt werden [Cou19b]. Management-orientierte Ansätze adressieren Themen wie die Organisationsstruktur, Werte und Verantwortlichkeiten im Unternehmen [Van14]. Technische Ansätze lassen sich anhand des Geschäftsprozesslebenszyklus-Konzepts kategorisieren [Cou19b, Van14]. Definitionen des Geschäftsprozesslebenszyklus finden sich bei [Mue05, Net06, Kan08, Dum18, Wes19, Van14]. Diese Definitionen variieren in Details, aber ihnen gemein ist die Vorstellung, dass Geschäftsprozesse Zyklen durchlaufen, in denen sie kontinuierlich verbessert werden.

Abbildung 2.2 zeigt den Geschäftsprozesslebenszyklus nach [Dum18] in leicht angepasster und vereinfachter Form. In der ersten Phase, *Modellierung*, wird ein Modell eines Geschäftsprozesses erstellt. Anhand des Modells können dann in der *Analyse*-Phase Untersuchungen durchgeführt werden, welche zu Verbesserungen (des Modells) führen. Das verbesserte Prozessmodell kann schließlich zum Beispiel in einem unterstützenden Informationssystem umgesetzt, ausgeführt und dort überwacht werden (Phase *Umsetzung*).[1] Verschiedene (technische) Ansätze des Geschäftsprozessmanagements unterstützen eine oder mehrere Phasen des Geschäftsprozesslebenszyklus. Zum Beispiel unterstützen Modellierungssprachen wie Business Process Model and Notation (BPMN), Ereignisgesteuerte Prozessketten (EPK) oder Petri-Netze die Modellierung eines Geschäftsprozesses [OMG13, Sch13, Kos18]. Modellierungsmethoden wie Horus [Sch12] oder ARIS [Sch13] stellen darüber hinaus Konzepte für die Analyse, Verbesserung und Umsetzung bereit.

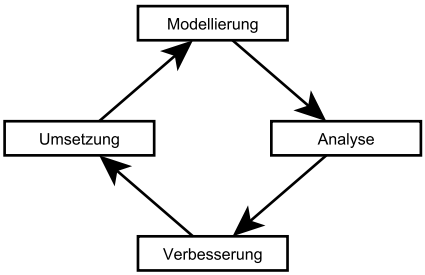


Abbildung 2.2: Phasen des Geschäftsprozesslebenszyklus, in Anlehnung an [Dum18, S. 23].

[1] Manche Quellen [Dum18, Wes19] unterscheiden detaillierter zwischen Phasen wie *Konfiguration*, *Ausführung* und *Überwachung*. Diese Aspekte werden hier vereinfachend unter der Phase *Umsetzung* zusammengefasst.

Die in dieser Arbeit entwickelte Modellierungsmethode ist den technischen Ansätzen des Geschäftsprozessmanagements zuzuordnen. Bei der Untersuchung verwandter Arbeiten in Kapitel 5 sowie bei der Diskussion und im Ausblick (Kapitel 11 und 12) wird die Betrachtung auch auf Management-orientierte Ansätze ausgeweitet.

2.2 Begriffe und Konzepte

In Vorbereitung auf die Gegenüberstellung mit vergleichbaren Begriffen und Konzepten der Nachhaltigkeitsanalyse in Kapitel 4 erfolgt in diesem Abschnitt eine vertiefte Betrachtung der Begriffe und Konzepte des Geschäftsprozessmanagements. Ausgehend von der zu Beginn des Kapitels erwähnten Geschäftsprozessdefinition nach [Obe96a] werden zunächst Grundbegriffe des Geschäftsprozessmanagements vorgestellt (Unterabschnitt 2.2.1). Es folgt eine vertiefte Betrachtung der Themenbereiche Kontroll- und Datenfluss (Unterabschnitt 2.2.2) sowie Leistungsindikatoren (Unterabschnitt 2.2.3), die eine zentrale Rolle bei der Integration von Nachhaltigkeitsaspekten in das Geschäftsprozessmanagement spielen.

2.2.1 Grundbegriffe

Nach [Obe96a, S. 14f] ist ein Geschäftsprozess[1] „eine Menge von [...] Aktivitäten, die in einem [Unternehmen] nach bestimmten Regeln auf ein bestimmtes Ziel hin ausgeführt werden“. Weiterhin hängen die Aktivitäten „über betroffene Personen, Maschinen, Dokumente, Betriebsmittel u.ä. miteinander zusammen“. Beispiele für Geschäftsprozesse nach dieser Definition sind die Bearbeitung einer Bestellung, die Fertigung eines Produkts, oder die Planung und Abwicklung einer Geschäftsreise. Vergleichbare Definitionen für einen Geschäftsprozess finden sich auch bei [Ham94, Dav93, Wes19, Dum18, Fra03]. Für diese Arbeit sind die in den Quellen [Obe96a] und [Sch12] beschriebenen Konzepte maßgeblich. Die anderen genannten Quellen werden ergänzend herangezogen. Zur Übersicht und als Referenz für die folgenden Kapitel sind in Tabelle 2.1 die in diesem Abschnitt beschriebenen Grundbegriffe des Geschäftsprozessmanagements in der Reihenfolge ihres Erscheinens in diesem Unterabschnitt aufgeführt.

[1] [Obe96a, S. 14f] verwendet die Begriffe „betrieblicher Ablauf“ und „Betrieb“ anstelle von „Geschäftsprozess“ und „Unternehmen“, wie sie in dieser Arbeit verwendet werden. Mit dem Begriff „Betrieb“ beziehungsweise „Unternehmen“ sind dabei beispielsweise auch Dienstleistungsunternehmen, Krankenhäuser oder Behörden gemeint.

Tabelle 2.1: Grundbegriffe des Geschäftsprozessmanagements, basierend auf [Obe96a, Sch12, Ham94, Dav93, Wes19, Dum18].

Begriff	Beschreibung
Geschäftsprozess	Eine Menge von Aktivitäten, die in einem Unternehmen nach bestimmten Regeln auf ein bestimmtes Ziel hin ausgeführt werden.
Ziel	Angestrebtes Ergebnis eines Geschäftsprozesses.
Kontrollfluss	Kausale Beziehungen zwischen Aktivitäten.
Datenfluss	Austausch von Daten zwischen Aktivitäten.
Ressource	Ein Objekt, das für die Ausführung eines Geschäftsprozesses benötigt wird.
Prozesstyp	Allgemeiner Rahmen zur Beschreibung von Aktivitäten und ihrer Zusammenhänge.
Prozessinstanz	Ausprägung eines Prozesstyps im Sinne einer Ausführung der entsprechenden Aktivitäten.
Aktivitätstyp	Allgemeine Beschreibung einer Aktivität, von der es unterschiedliche Ausprägungen geben kann.
Aktivitätsinstanz	Ausprägung eines Aktivitätstyps im Sinne einer Ausführung einer Aktivität.
Leistungsindikator	Misst die Leistung eines Geschäftsprozesses.

Aus der Definition eines Geschäftsprozesses ist zu entnehmen, dass die betrachteten Aktivitäten „auf ein bestimmtes Ziel hin ausgeführt werden“. Ähnliche Formulierungen finden sich auch bei [Wes19, Ham94, Dav93, Dum18]. [Dum18] spricht auch von dem angestrebten Ergebnis eines Geschäftsprozesses, das für eine Kundin oder einen Kunden von Wert ist. Dabei sind sowohl interne als auch externe Kundinnen und Kunden gemeint. Ein Ziel eines Geschäftsprozesses könnte sein, dass eine Rechnung bezahlt wird oder dass ein Leihgerät bereitgestellt wird [Dum18].

Bei der Betrachtung der Zusammenhänge zwischen Aktivitäten eines Geschäftsprozesses über „Personen, Maschinen, Dokumente, Betriebsmittel u.ä.“ [Obe96a, S. 14] kann zwischen zwei Aspekten unterschieden werden. Zum Einen werden im Geschäftsprozessmanagement kausale Zusammenhänge zwischen Aktivitäten betrachtet [Obe96a, S. 18]: Aktivitäten können sich „kausal bedingen“, sich „gegenseitig ausschließen“ oder auch „unabhängig voneinander [...] stattfinden“.[1] Zum Anderen fließen zwischen den Aktivitäten Daten und Informationen in Form von zum Beispiel Belegen, Formularen und Betriebsmitteln oder es werden Eigenschaften von Objekten (zum Beispiel Formulare, Maschinen, Personen) geändert [Obe96a]. Einige Quellen unterscheiden hier explizit zwischen einem Kontrollfluss und einem Datenfluss, wobei anstelle von Datenfluss auch die Begriffe Nachrichten- oder Objektfluss

[1] Für die Unabhängigkeit von Aktivitäten ist auch der Begriff „nebenläufig“ gebräuchlich [Obe96a, S. 18].

gebräuchlich sind [Dum18, Wes19]. Die Beispiele zeigen, dass auch der Fluss von Materialien und (physischen) Gütern als Fluss von Daten und Informationen interpretiert, beziehungsweise als Datenfluss in Prozessmodellen abgebildet werden kann [Wes19, Dum18]. Auf die Unterscheidung zwischen Kontroll- und Datenfluss wird in Unterabschnitt 2.2.2 näher eingegangen.

Die erwähnten „Personen, Maschinen, Dokumente, Betriebsmittel u.ä." [Obe96a, S. 14] werden auch als Ressourcen bezeichnet [Sch12, Obe96a]. Eine Abgrenzung zu dem eben beschriebenen Datenfluss ist hier nicht in jedem Fall möglich: [Fra03] definiert Ressourcen allgemein als Objekte, die zur Ausführung eines Geschäftsprozesses benötigt werden. Bei der Betrachtung von Ressourcen wird allerdings meist auf Menschen und Maschinen fokussiert, die Aktivitäten ausführen [Wes19, Dum18, Sch12]. In diesem Fall handelt es sich um Objekte, die in Geschäftsprozessen benötigt beziehungsweise benutzt werden und nicht um Objekte, die in Geschäftsprozessen verbraucht werden (wie zum Beispiel Rohstoffe oder Verbrauchsmaterial) [Fra03].

Nach der eingangs zitierten Definition wird ein Geschäftsprozess *in einem Unternehmen* ausgeführt. Demnach ist der Betrachtungsrahmen der Modellierung im Geschäftsprozessmanagement typischerweise ein Unternehmen. Zielgruppe des Geschäftsprozessmanagements sind üblicherweise unternehmensinterne Entscheiderinnen und Entscheider [Wes19]. Auch nach der Definition bei [Wes19] wird ein Geschäftsprozess innerhalb eines Unternehmens ausgeführt, wobei damit die Betrachtung von Schnittstellen zu den Geschäftsprozessen anderer Unternehmen nicht ausgeschlossen ist.

Die Modellierung eines Geschäftsprozesses kann auf unterschiedlichen Abstraktionsstufen beziehungsweise mit unterschiedlichen Detaillierungsgraden stattfinden. So ist zunächst zwischen Prozesstypen und Prozessinstanzen zu unterscheiden [Obe96a, Wes19]. Nach [Obe96a, S. 34] handelt es sich bei einem Prozesstyp um einen „allgemeinen Rahmen" zur Beschreibung von Aktivitäten und ihrer Zusammenhänge, zu dem es zu einem bestimmten Zeitpunkt mehrere konkrete Ausprägungen beziehungsweise Instanzen geben kann. Eine Prozessinstanz entspricht einem konkreten Geschäftsvorfall [Wes19], beziehungsweise einer Ausführung des Geschäftsprozesses mit eindeutig identifizierbaren Aktivitäten [Obe96a]. Weiterhin können Prozessmodelle schrittweise verfeinert werden. Die Granularitätsstufen reichen üblicherweise von den Geschäftsfunktionen eines Unternehmens (zum Beispiel Marketing und Verkauf) über Geschäftsprozesse (zum Beispiel ein Bestellprozess) und Unterprozesse bis hin zu einzelnen Aktivitäten (zum Beispiel das Erstellen einer Rechnung oder der Versand eines Artikels) [Sch12, Dum18, Wes19].

Neben dem Erstellen von Prozessmodellen ist die Messung der Leistung von Geschäftsprozessen eine weitere Säule des Geschäftsprozessmanagements [Sch12, Dum18]. Die Leistung eines Geschäftsprozesses wird mit Leistungsindikatoren auf Basis der für einen Geschäftsprozess verfügbaren Daten ermittelt [Dum18]. Typische Aspekte der Leistungsmessung sind Zeit, Kosten und Qualität [Wes19, Dum18]. Das Thema Leistungsindikatoren wird in Unterabschnitt 2.2.3 vertieft betrachtet.

Die in diesem Unterabschnitt beschriebenen Begriffe und Konzepte des Geschäftsprozessmanagements wurden im Hinblick auf eine Gegenüberstellung mit Begriffen und Konzepten der Nachhaltigkeitsanalyse zusammengestellt. Weitere Aspekte des Geschäftsprozessmanagements, die in dieser Arbeit nicht vertieft betrachtet werden, betreffen beispielsweise Risiken oder Organisationsstrukturen [Sch12]. Im Folgenden wird die Betrachtung der bereits angesprochenen Aspekte Kontroll- und Datenfluss (Unterabschnitt 2.2.2) sowie Leistungsindikatoren (Unterabschnitt 2.2.3) in Vorbereitung auf die Gegenüberstellung mit vergleichbaren Begriffen und Konzepten der Nachhaltigkeitsanalyse in Kapitel 4 vertieft.

2.2.2 Kontroll- und Datenfluss

Im Geschäftsprozessmanagement werden sowohl kausale Beziehungen zwischen Aktivitäten (Kontrollfluss), als auch der Fluss von Daten und Informationen (Datenfluss) betrachtet [Obe96a, Dum18, Wes19]. Abbildung 2.3 zeigt ein Beispiel für ein in der Prozessmodellierungssprache BPMN erstelltes Prozessmodell mit Kontrollfluss und Datenfluss. In BPMN wird der Kontrollfluss dadurch visuell abgebildet, dass Aktivitätsknoten, Ereignisknoten und Entscheidungsknoten durch Pfeile mit durchgezogenen Linien verbunden werden [OMG13]. Den Ereignisknoten (Kreise) und Aktivitätsknoten (abgerundete Rechtecke) entlang den Pfeilen mit durchgezogenen Linien folgend, lässt sich der Kontrollfluss in der Abbildung nachvollziehen: Nachdem eine Bestellung eingetroffen ist (Ereignis *Auftrag erhalten*), wird ein Produktionsprozess angestoßen (Aktivitäten *Material bei Lieferant anfragen* und *Produkt herstellen*). Danach wird der Auftrag bestätigt und das Produkt geliefert. Mit der Ergänzung eines Entscheidungsknotens könnte im gegebenen Prozessmodell auch der Fall dargestellt werden, dass ein Produkt bereits im Lager vorhanden ist und nicht geliefert werden muss.

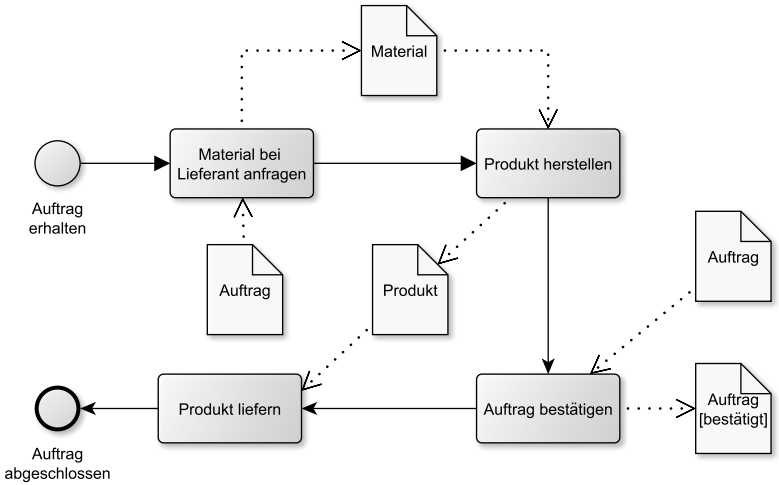


Abbildung 2.3: BPMN-Prozessmodell mit Datenfluss, in Anlehnung an [Dum18, S. 94].

Mit Notizzettelsymbolen und Pfeilen mit gestrichelten Linien wird in BPMN der Fluss von Datenobjekten dargestellt [Dum18]. In Abbildung 2.3 ist ein Auftragsdokument (*Auftrag*) ein Input-Datenobjekt für die Aktivitäten *Material bei Lieferant anfragen* und *Auftrag bestätigen*. Die Aktivität *Material bei Lieferant anfragen* erzeugt das Datenobjekt *Material* als Output, das als Input für die Aktivität *Produkt herstellen* dient.

Prozessmodellierungssprachen wie BPMN und EPK sind darauf ausgelegt, den Kontrollfluss darzustellen [Wes19, Dum18]. Sprachelemente zur Darstellung des Datenflusses sind eher ergänzender Natur [Wes19, Dum18]. Das zeigt, dass im Geschäftsprozessmanagement traditionell ein größeres Interesse am Kontrollfluss als am Datenfluss besteht. So empfiehlt auch [Dum18], den Datenfluss nur in Ausnahmefällen zu modellieren, um Prozessmodelle nicht zu überfrachten. In gleicher Weise adressiert die Prozessdefinition in [Fra03] lediglich den Kontrollfluss.

Auch wenn mit Datenfluss-Konzepten die Darstellung von (greifbaren) Materialien und Gütern möglich ist [Dum18, Wes19], werden die genannten Prozessmodellierungssprachen historisch bedingt eher für die Modellierung von Dienstleistungsprozessen, als für die Modellierung von Herstellungsprozessen greifbarer Produkte verwendet [Dum18, Fra03].

Insofern der Datenfluss in einem Prozessmodell betrachtet wird, können auch Wechselwirkungen zwischen Daten- und Kontrollfluss auftreten. So kann der Fluss von Daten die Ausführungsreihenfolge von Aktivitäten beeinflussen (eine Aktivität kann nur ausgeführt werden, wenn die benötigten Input-Datenobjekte vorhanden sind) [Dum18]. Im Unterschied zu BPMN [OMG13] und BPMN-orientierten Quellen wie [Dum18] und [Wes19] wird bei der Petri-Netz-orientierten Modellierung von Geschäftsprozessen nach [Obe96a] und [Sch12] nicht explizit zwischen Sprachelementen für die Abbildung des Kontrollflusses (wie Ereignis- und Entscheidungsknoten) und Sprachelementen für die Abbildung des Datenflusses (wie Datenobjekte) unterschieden. Auf die Ausdrucksmöglichkeiten von Petri-Netz-orientierten Ansätzen zur Modellierung von Geschäftsprozessen wird in Abschnitt 2.3 näher eingegangen.

2.2.3 Leistungsindikatoren

Neben dem Erstellen von Prozessmodellen ist die Messung der Leistung von Geschäftsprozessen eine weitere Säule des Geschäftsprozessmanagements [Dum18]. Die Leistung eines Geschäftsprozesses wird mit Leistungsindikatoren auf Basis der für einen Geschäftsprozess verfügbaren Daten ermittelt [Dum18]. Dadurch soll das Erreichen von Zielen messbar gemacht werden [Sch12]. Typische Aspekte der Leistungsmessung sind Zeit, Kosten und Qualität [Wes19, Dum18]. Zur Bewertung dieser Aspekte werden zum Beispiel Durchlauf- und Wartezeiten, Arbeitskosten sowie die Zufriedenheit der Kundinnen und Kunden gemessen [Dum18].

Für jeden Leistungsindikator können Ziel-, Minimum-, Maximum-, und Schwellwerte sowie Toleranzen festgelegt werden [Sch12, Wes19]. Weiterhin können Leistungsindikatoren hierarchisch strukturiert sein [Sch12, Dum18]. Die Kosten einer Aktivität lassen sich zum Beispiel in Produktionskosten, Lieferkosten und Lohnkosten auffächern und gegebenenfalls auch noch weiter verfeinern [Dum18]. [Dum18] erwähnt auch die Möglichkeit, die Leistungsindikatoren aller Themenbereiche auf einen Wert für die Gesamtleistung des Prozesses zu aggregieren.

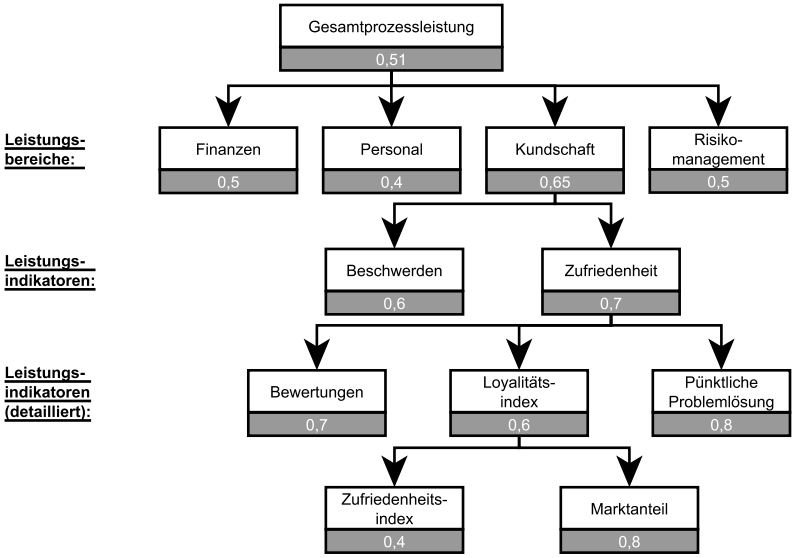


Abbildung 2.4: Aggregation von Leistungsindikatoren, in Anlehnung an [Dum18, S. 63].

Abbildung 2.4 zeigt ein Beispiel für eine hierarchische Indikatorstruktur, mit der die Leistung eines Prozesses gemessen wird. Auf der untersten Ebene werden zum Beispiel Werte für einen Zufriedenheitsindex (*0,4*) und zum Marktanteil erfasst (*0,8*). Die Werte sind in diesem Beispiel auf eine Skala von 0 bis 1 normiert und werden eine Ebene höher zu einem Loyalitätsindex (*0,6*) zusammengefasst. Im gegebenen Beispiel werden alle Werte für die Aggregation gleichgewichtet. Diese Gleichgewichtung und Aggregation zieht sich bis zur obersten Ebene, dem Maß für die Gesamtleistung des Prozesses (*0,51*), durch.

2.3 Prozessmodellierungssprachen

In diesem Abschnitt wird ein Überblick über gängige Prozessmodellierungssprachen gegeben, wobei der Fokus auf Petri-Netz-orientierten Sprachen liegt. Es werden grundlegende Petri-Netz-Konzepte erläutert und verschiedene Varianten höherer Petri-Netze vorgestellt und eingeordnet. Teile der folgenden Übersichten zu Prozessmodellierungssprachen wurden aus [Fri23b] übernommen und adaptiert.

2.3.1 Überblick

Modelle bilden reale Phänomene ab und fokussieren dabei, je nach Zweck des Modells, auf bestimmte Aspekte des Phänomens [Sta73]. Modellierungssprachen unterstützen die Erstellung von Modellen in verschiedenen Domänen. Sie stellen dazu Syntax (Sprachelemente, die zur Erstellung von Modellen verwendet werden können, sowie Regeln zu ihrer Verwendung), Semantik (Beschreibung der Bedeutung der Modellelemente) und eine Notation (Vorgaben zur visuellen Darstellung) bereit [Kar02]. Im Kontext des Geschäftsprozessmanagements bilden Prozessmodelle die Grundlage für die Analyse und Verbesserung von existierenden oder geplanten Geschäftsprozessen in Unternehmen [Obe96a].

Prozessmodellierungssprachen wie EPK und BPMN fokussieren auf die Abbildung des Kontrollflusses von Geschäftsprozessen (siehe Unterabschnitt 2.2.2). Die Darstellung des Datenflusses wird in diesen Sprachen nur eingeschränkt unterstützt [Sno21]. Darüber hinaus fehlen Semantik und syntaktische Regeln, um den Datenaustausch zwischen Aktivitäten und verschiedenen Datenquellen präzise darzustellen [Mey11]. Durch die Trennung der Daten- und Kontrollflussperspektive soll die inhärente Komplexität des Entwurfs von Informationssystemen handhabbar gemacht werden [Sno21]. Problematisch daran ist jedoch, dass gewisse Phänomene, bei denen der Datenfluss eine wichtige Rolle spielt, dadurch nur unzureichend darstellbar sind [Ste19]. Eine mangelnde Integration der Daten- und Prozessperspektive wurde auch als Hinderungsgrund für die Verbreitung von prozessorientierten Informationssystemen identifiziert [Ste19]. Wie in Kapitel 4 dargelegt wird, ist die Integration einer Datenperspektive in Prozessmodellen auch für die Betrachtung von Nachhaltigkeitsaspekten relevent.

Es existiert eine Reihe von Ansätzen (datenorientierten Prozessmodellierungssprachen) zur engeren Verbindung der Daten- und Prozessperspektive in Prozessmodellen [Ste19, Rei17, Pol19]. Eine Familie von Ansätzen integriert dazu Petri-Netze mit verschiedenen Datenmodellen. Petri-Netze stellen eine einfache grafisch Darstellung und formale Semantik bereit und unterstützen dadurch vielfältige Analysemethoden für das Geschäftsprozessmanagement [Aal15, Kos18]. Über Erweiterungen des Petri-Netz-Formalismus (sogenannte Höhere Petri-Netze) unterstützen sie auch die Darstellung des Datenflusses in Geschäftsprozessen [Aal15]. Im Folgenden werden zunächst grundlegende Petri-Netz-Konzepte vorgestellt und anschließend Ansätze betrachtet, die Höhere Petri-Netze zur Modellierung von Geschäftsprozessen nutzen.

2.3.2 Petri-Netze

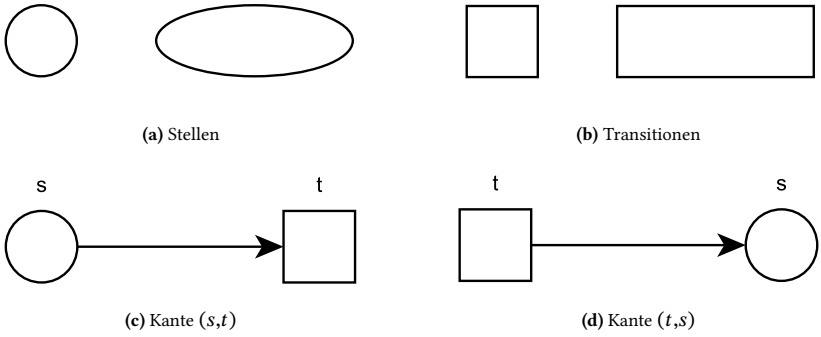
Petri-Netze können als generische Modellierungssprache verstanden werden, da sie nur wenige abstrakte Konzepte bereitstellen, die eine Vielzahl realer Phänomene abbilden können [Rei10]. Sie können dadurch in einer Vielzahl von Anwendungsgebieten zum Einsatz kommen. Die folgenden Ausführungen zu grundlegenden Petri-Netz-Konzepten basieren auf den Definitionen nach [Rei10].

Definition 2.3.1: Netzstruktur [Rei10]

Eine Netzstruktur $N = (S,T,F)$ besteht aus einer Menge von Stellen S, einer Menge von Transitionen T und einer Menge von Kanten F. Kanten sind dabei als Paare, also F als Relation $F \subseteq (S \times T) \cup (T \times S)$ aufzufassen.

In Definition 2.3.1 wird zunächst die Struktur eines Petri-Netzes beschrieben. Sie besteht aus zwei Arten von Elementen, *Stellen* und *Transitionen*. Stellen werden als Kreis oder Ellipse dargestellt (siehe Abbildung 2.5a) und modellieren passive Komponenten, zum Beispiel Lager für Objekte oder Zustandsspeicher. Transitionen werden als Rechteck dargestellt (siehe Abbildung 2.5b) und modellieren aktive Komponenten, die beispielsweise Objekte erzeugen, verbrauchen, transportieren oder verändern können. Bei der Modellierung von Geschäftsprozessen mit Petri-Netzen werden demnach Aktivitäten als Transitionen abgebildet [Obe96a, Sch12].

Stellen und Transitionen können durch gerichtete *Kanten* verbunden werden (siehe Abbildungen 2.5c und 2.5d). Dabei dürfen nie zwei Stellen oder zwei Transitionen miteinander verbunden werden, sondern immer Stellen mit Transitionen oder Transitionen mit Stellen. Eine Kante stellt somit eine Beziehung zwischen Komponenten, beispielsweise einen kausalen Zusammenhang oder räumliche Nähe, dar. Sie wird formal als Paar (s,t) oder (t,s), je nach Flussrichtung, bestehend aus einer Stelle s und einer Transition t beschrieben. Insgesamt wird eine Netzstruktur N also durch ein Tripel $N = (S,T,F)$ beschrieben. Dabei ist S die Menge der Stellen, T die Menge der Transitionen und F die Menge aller Kanten. F wird auch Flussrelation genannt, beschrieben durch $F \subseteq (S \times T) \cup (T \times S)$.

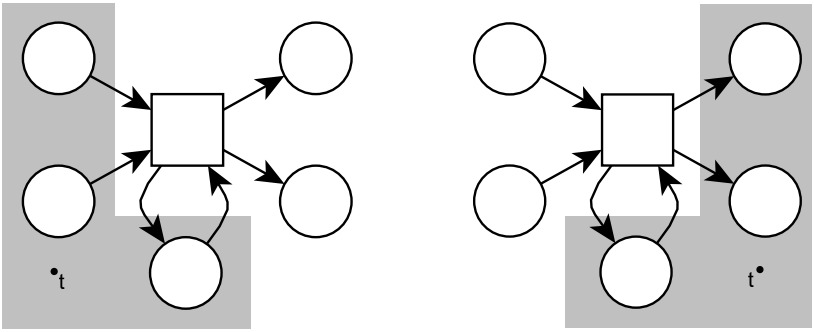


(a) Stellen **(b)** Transitionen

(c) Kante (s,t) **(d)** Kante (t,s)

Abbildung 2.5: Grafische Darstellung der Petri-Netz-Elemente.

Bei Stellen und Transitionen kann zwischen dem Vor- und Nachbereich eines Elements unterschieden werden. Der Vorbereich eines Elements (${}^\bullet x$) ist die Menge der Elemente, die eine ausgehende Kante zum betreffenden Element x haben: ${}^\bullet x = \{y \mid (y,x) \in F\}$. Der Nachbereich eines Elements ($x^\bullet$) umfasst die Elemente, die eine eingehende Kante ausgehend vom betreffenden Element x haben: $x^\bullet = \{y \mid (x,y) \in F\}$. Abbildung 2.6 zeigt ein Beispiel für den Vor- und Nachbereich einer Transition. Wie in der Abbildung gezeigt, kann eine Stelle sowohl zum Vorbereich als auch zum Nachbereich einer Transition zählen (sowie eine Transition zum Vor- und Nachbereich einer Stelle).



(a) Vorbereich ${}^\bullet t$ (grau markierte Stellen) **(b)** Nachbereich $t^\bullet$ (grau markierte Stellen)

Abbildung 2.6: Vor- und Nachbereich einer Transition.

Durch *Markierungen* werden Zustände eines Systems abgebildet. Marken können als Objekte der realen Welt, zum Beispiel eine Münze oder ein Signal, interpretiert werden. Bei der Modellierung von Geschäftsprozessen kann durch die Markierung der Stellen einer Netzstruktur der Zustand einer *Prozessinstanz* abgebildet werden. [Rei10, S. 35] unterscheidet zwischen Petri-Netzen mit unterscheidbaren Marken (Höhere Petri-Netze) und „elementaren" Petri-Netzen. Nach [Rei10] sind elementare Petri-Netze solche, die lediglich abstrakte ununterscheidbare Marken zulassen, die grafisch als schwarze Punkte in den Stellen dargestellt werden. Ein elementares Petri-Netz besteht somit aus einer Netzstruktur $N = (S,T,F)$, sowie einer Markierung $M : S \to \mathbb{N}$, das heißt jede Stelle s trägt $M(s)$ Marken. Die Anfangsmarkierung wird dabei als M_0 gekennzeichnet.

Definition 2.3.2: Schaltregel für elementare Petri-Netze [Rei10]

Eine Markierung M aktiviert eine Transition t, wenn $M(s) \geq 1$ für jede Stelle $s \in {}^\bullet t$. Das bedeutet, eine Transition ist aktiviert, wenn in jeder Stelle ihres Vorbereichs mindestens eine Marke liegt.

Ein Schaltvorgang $M \xrightarrow{t} M'$ ist dann beschrieben durch:

$$M'(s) = \begin{cases} M(s) - 1 & s \in {}^\bullet t \quad \wedge \quad s \notin t^\bullet \\ M(s) + 1 & s \in t^\bullet \quad \wedge \quad s \notin {}^\bullet t \\ M(s) & sonst. \end{cases}$$

Durch die *Schaltregel* werden in markierten Petri-Netzen die Zustandsänderungen von Systemen beschrieben. Sie legt fest, unter welchen Bedingungen eine Transition schalten kann und was beim Schalten einer Transition geschieht. Eine Transition, die schalten kann, heißt auch *aktiviert*. Im Kontext des Geschäftsprozessmanagements wird mit dem Schalten einer Transition die Zustandsänderung einer Prozessinstanz durch die Ausführung einer Aktivität abgebildet [Obe96a]. Definition 2.3.2 gibt die Schaltregel für elementare Petri-Netze an.

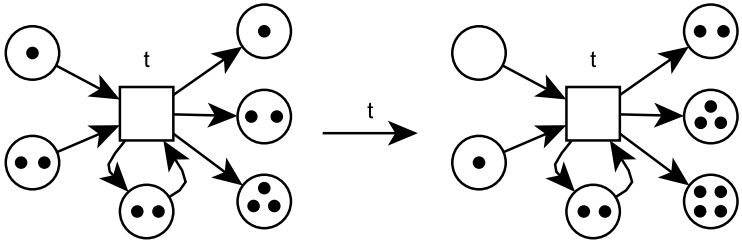


Abbildung 2.7: Schaltvorgang in einem elementaren Petri-Netz.

Wie in Abbildung 2.7 dargestellt, „fließt“ beim Schalten einer aktivierten Transition durch jede Eingangs- oder Ausgangskante der betreffenden Transition eine Marke. Aus jeder Stelle im Vorbereich der betreffenden Transition wird eine Marke entfernt und in jede Stelle im Nachbereich eine Marke hinzugefügt. Zusätzlich können für elementare Petri-Netze Kapazitäten und Kantengewichte angegeben werden. Eine Kapazität gibt für eine Stelle an, wie viele Marken in ihr liegen dürfen. Ein Kantengewicht gibt an, wie viele Marken beim Schalten einer Transition verbraucht oder erzeugt werden.

2.3.3 Höhere Petri-Netze

Petri-Netze mit unterscheidbaren Marken, beziehungsweise Marken, die Daten tragen, werden Höhere Petri-Netze genannt [Gen81, Rei10]. Sie eignen sich insbesondere für die Abbildung des Datenflusses in Geschäftsprozessen [Aal15]. Abbildung 2.8 illustriert die Ausdrucksmächtigkeit Höherer Petri-Netze. Im Vergleich zu elementaren Petri-Netzen können nicht nur anonyme beziehungsweise ununterscheidbare Marken dargestellt werden, sondern auch Marken unterschiedlicher Typen. Zum Beispiel werden in der Abbildung in der Stelle s_1 Aufträge gespeichert, die von den in der Stelle s_3 gespeicherten Lieferungen unterschieden werden können. Auch die Marken innerhalb einer Stelle können voneinander unterschieden werden (unterschiedliche Aufträge mit unterschiedlichen Positionen, unterschiedliche Lieferungen, ...). In gefärbten Petri-Netzen, als eine Variante Höherer Petri-Netze [Rei10, Jen81], werden Stellen sogenannte Farben zugewiesen, beziehungsweise mit Farben typisiert. Diese Farben repräsentieren verschiedene Daten- oder Objekt-Typen wie Ressourcen, Güter oder Menschen im modellierten System. Kanten können mit Farben beschriftet werden, um darzustellen, dass nur Marken einer bestimmten Farbe von ihnen erzeugt oder produziert werden können.

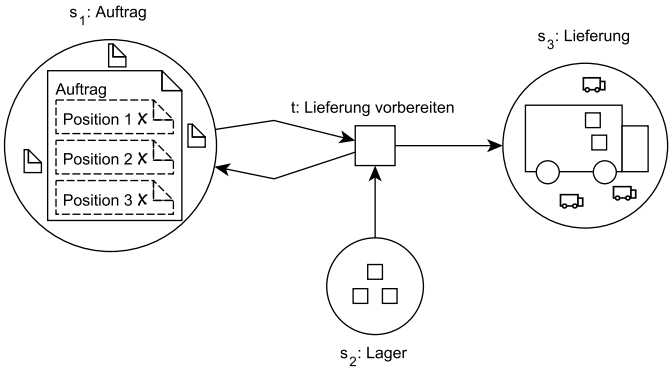


Abbildung 2.8: Illustration der Ausdrucksmächtigkeit Höherer Petri-Netze.

Weitere Varianten Höherer Petri-Netze verwenden zur Typisierung von Stellen Datenmodelle, die auch in Datenbanksystemen eingesetzt werden. In dieser Familie von Höheren Petri-Netzen können für Kanten Filter-Operationen angegeben werden. Diese Filter-Operationen werden verwendet, um Daten aus Input-Stellen einer Transition auszuwählen und Daten in Output-Stellen einzufügen, vergleichbar mit dem Auswählen und Einfügen von Daten in Datenbanken. Die Verknüpfung des Petri-Netz-Formalismus mit Datenbank-Datenmodellen hat im Kontext des Geschäftsprozessmanagements den Vorteil, dass dadurch eine engere Verzahnung der Prozess- und der Datenperspektive in den Informationssystemen der Unternehmen erreicht werden kann [Len03]. Im Idealfall können Daten und Informationen in der Form, in der sie im Unternehmen vorliegen, auch in das Prozessmodell eingebunden werden. In Exkurs 2.3.1 werden Hintergründe und die historische Entwicklung aktueller Datenmodelle für Datenbanksysteme zusammengefasst.

Exkurs 2.3.1: Datenmodelle für Datenbanksysteme

Lange Zeit war bei Datenbank-Datenmodellen das relationale Datenmodell vorherrschend, bei dem Daten in Relationen (Tabellen) gespeichert werden [Abi97]. Neben dem relationalen Datenmodell gibt es auch sogenannte semi-strukturierte Datenmodelle, bei denen Daten keinem strengen (relationalen) Schema unterliegen [Bun97]. Zu diesen semi-strukturierten Datenmodellen zählt Extensible Markup Lanuguage (XML)

[W3C12]. XML-Datenbanksysteme können als frühe Vertreter heutiger NoSQL-Datenbanksysteme[a] betrachtet werden [Mei19]. NoSQL ist dabei ein Sammelbegriff für Ansätze, in denen Daten nicht nur in relationalen Strukturen gespeichert und verarbeitet werden, sondern unter anderem auch semi-strukturiert [Sah18]. JavaScript Object Notation (JSON) stellt ein alternatives Datenmodell zu XML dar, das sich bei der Weitergabe von Daten im Internet zum de-facto Standard entwickelt hat [Nur09, Bou20].

[a] NoSQL steht für *not only SQL* [Mei19]. Structured Query Language (SQL) ist die Standard-Anfragesprache für relationale Datenbanksysteme.

Als früher Vertreter dieser Familie von Höheren Petri-Netzen, die den Petri-Netz-Formalismus mit Datenbank-Datenmodellen kombiniert, sind zunächst Prädikat/Transitionsnetze zu nennen [Gen81, Obe96a]. Dabei werden Stellen Relationstypen zugewiesen und Markierungen als Relationen des entsprechenden Relationstyps dargestellt. Beim Schalten einer Transition werden in Prädikat/Transitionsnetzen also Daten in Relationen (Tabellen) verarbeitet. In Relationen werden Daten „normalisiert“, was unter anderem bedeutet, dass mehrwertige Attribute in Relationen nicht vorgesehen sind [Saa18]. Dadurch kann in Prädikat/Transitionsnetzen ein komplex strukturiertes Datenobjekt, wie beispielsweise ein Auftrag mit Auftragspositionen, nicht unmittelbar abgebildet werden [Obe96a]. Als Erweiterung von Prädikat/Transitionsnetzen unterstützen Nested-Relation/Transitionsnetze[1] die Abbildung komplex strukturierter Datenobjekt und die Durchführung von Operationen auf Substrukturen dieser komplex strukturierten Datenobjekte [Obe96a, Obe96c, Obe96b]. Die Transition aus Abbildung 2.8 könnte, als Nested-Relation/Transitionsnetz umgesetzt, zum Beispiel eine einzelne Auftragsposition verbrauchen beziehungsweise erzeugen. Nested-Relation/Transitionsnetze bilden so gewissermaßen eine Brücke zwischen den Prädikat/Transitionsnetzen, die mit dem relationalen Datenmodell arbeiten, und späteren Varianten Höherer Petri-Netze, die semi-strukturierte Datenmodelle verwenden. Diese Idee, Petri-Netzen mit semi-strukturierten Daten zu kombinieren, wird zum Beispiel mit SGML-Netzen umgesetzt [Wei98] (Standard Generalized Markup Language (SGML) ist eine Obermenge und Vorläufer des im Exkurs erwähnten XML-Datenmodells [W3C12]). In SGML-Netzen werden Stellen mit der Schemabeschreibungssprache Document Type Definition (DTD) typisiert, das heißt die Struktur der

[1] „Nested Relation“ steht für „verschachtelte Relation“ [Obe96c].

in der Stelle gespeicherten SGML-Dokumente vorgegeben. Sogenannte Dokumenten-Muster beschreiben, welche Dokument-Instanzen beim Schalten einer Transition gelesen und eingefügt werden. Darauf aufbauend werden in XML-Netzen XML-Schemas zur Typisierung von Stellen sowie als Kantenbeschriftungen zur Auswahl und zum Einfügen von XML-Daten eingesetzt [Len03].

In der Literatur finden sich weitere Varianten Höherer Petri-Netze, die ähnliche Eigenschaften aufweisen (siehe zum Beispiel auch [Sib85, Bad15]). Grundsätzlich ist die Ausdrucksmächtigkeit aller Varianten Höherer Petri-Netze vergleichbar [Rei10] und sie bieten im Geschäftsprozessmanagement im Vergleich zu Sprachen wie BPMN und EPK Vorteile in Bezug auf ihre Analysierbarkeit [Aal15]. Nested-Relation/Transitionsnetze und Höhere Petri-Netze, die mit semi-strukturierten Daten arbeiten, verfügen über die weiterführende Eigenschaft, dass sie auch Operationen auf Substrukturen einer Marke zulassen [Obe96a]. Diese Eigenschaft kann anhand der Abbildung 2.8 nachvollzogen werden. In gefärbten Petri-Netzen wird beim Schalten einer Transition immer eine ganze Marke (im Beispiel: ein ganzer Auftrag) verarbeitet. In Varianten Höherer Petri-Netze, die auch den Zugriff auf Substrukturen erlauben, kann zum Beispiel auch unmittelbar das Verarbeiten einer einzelnen Auftragsposition beim Schalten der Transition t modelliert werden.Die beschriebene Familie Höherer Petri-Netze, die Datenbank-Datenmodelle mit Petri-Netzen verknüpft, bietet darüber hinaus Vorteile bei der Integration der Prozess- und Datenperspektive in Informationssystemen von Unternehmen. Die Kombinationen von verschiedenen Datenmodellen mit Petri-Netzen eröffnet weiterhin die Möglichkeit, dass die Modellierungssprache von den Eigenschaften des jeweiligen Datenmodells profitieren kann. Bisher existieren noch keine Ansätze für die Kombination von Petri-Netzen mit dem im Exkurs erwähnten JSON-Datenmodell, das inzwischen zum de-facto Standard für den Datenaustausch im Internet geworden ist [Bou20]. Dafür wird in dieser Arbeit ein Ansatz erarbeitet (siehe Kapitel 8).

3 Grundlagen der Nachhaltigkeitsanalyse

In diesem Kapitel werden die für die vorliegende Arbeit wichtigen Grundlagen der Nachhaltigkeitsanalyse vorgestellt. Dazu wird zunächst in Abschnitt 3.1 das Konzept der Nachhaltigen Entwicklung eingeführt. Es folgt eine Vorstellung des Lebenszyklusgedankens und der Lebenszyklusanalyse als grundlegende Nachhaltigkeitsanalysemethode (Abschnitt 3.2). Zur Vorbereitung der Gegenüberstellung von Konzepten der Lebenszyklusanalyse und des Geschäftsprozessmanagements (siehe Kapitel 4) erfolgt dann eine vertiefte Betrachtung der Begriffe und Konzepte der Lebenszyklusanalyse in Abschnitt 3.3. Ergänzend dazu wird in Abschnitt 3.4 ein Überblick zu existierenden Modellierungsansätzen und Software-Werkzeugen gegeben.

3.1 Nachhaltige Entwicklung

Der Begriff Nachhaltigkeit bezeichnet dem Duden nach zunächst eine längere Zeit anhaltende Wirkung [Dud24]. Heutzutage wird er allerdings oft als Synonym für *Nachhaltige Entwicklung* verwendet. In ihrem Bericht an die Vereinten Nationen prägte die Weltkommission für Umwelt und Entwicklung diesen Begriff als Leitbild für globalen Wandel [Uni87, Gar04]. Demnach soll sich die Welt und sollen sich Regionen der Erde *nachhaltig* entwickeln. Das Ziel ist dabei, die Bedürfnisse der gegenwärtigen Generationen zu befriedigen, ohne die Möglichkeiten zukünftiger Generationen, ihre Bedürfnisse zu befriedigen, zu gefährden. Beim Entwurf dieses Leitbildes reflektiert die Kommission den Einfluss, den das erste Bild der Erde vom Weltraum aus auf die Menschheit hatte. Diese neue Perspektive machte die Zerbrechlichkeit des Planeten Erde und den Einfluss, den die Menschheit auf seine natürlichen Systeme hat, in besonderer Weise sichtbar. Um zu bestehen, muss die Menschheit die Wechselwirkungen dieser Beziehung verstehen und gestalten. Als wesentlicher Beitrag des im Brundtland-Bericht entworfenen Nachhaltigkeitskonzepts wird die gemeinsame Betrachtung von sozialen, ökologischen und ökonomischen Faktoren gesehen [Gar04, Pur19].

Das beschriebene Verständnis von Nachhaltigkeit lässt einigen Interpretationsspielraum, was sowohl als Stärke als auch als Schwäche angesehen werden kann [Lél91, Con07]. Die Stärke des breiten Interpretationsspielraums besteht darin, dass der Begriff ein Werkzeug sein kann, um zwischen verschiedenen gesellschaftlichen Akteuren mit unterschiedlichen Interessen einen Konsens zu schaffen [Lél91]. Jedoch besteht dadurch auch die Gefahr, dass der Begriff lediglich als „rhetorischer Deckmantel" (engl.: „rhetoric cloak") für Praktiken verwendet wird, die sich bei genauerem Hinsehen als ökologisch oder sozial nachteilig erweisen [Con07, S. 259]. So hält [Con07] fest, dass heutzutage die meisten Unternehmen von sich *behaupten*, nachhaltig zu sein. Eine „zynische" Verwendung des Begriffs kann so als Deckmantel für umweltschädliches Verhalten dienen [Con07, S. 273] – ein verbreitetes Phänomen, für das sich der Begriff *Greenwashing* etabliert hat [Del11].

[Con07] schlägt in diesem Zusammenhang eine Darstellung vor, die dabei helfen kann, sich über verschiedene Standpunkte oder Interpretationen des Nachhaltigkeitsbegriffs zu verständigen (siehe Abbildung 3.1). Unterschiedliche Interpretationen Nachhaltiger Entwicklung können entsprechend ihrer relativen Betonung von Umweltschutz (A), Wirtschaftswachstum (B) und sozialer Gerechtigkeit (C) angeordnet werden. Jede Ecke repräsentiert dabei extreme Standpunkte, bei denen die anderen Aspekte keine Rolle spielen. Die Bedeutung des Begriffs Nachhaltige Entwicklung wird als Bereich im Zentrum des Dreiecks dargestellt. Dadurch wird verdeutlicht, dass alle drei Aspekte gemeinsam betrachtet werden sollten, eine unterschiedliche Gewichtung innerhalb des Nachhaltigkeitsdiskurses aber durchaus möglich ist. So lässt sich zum Beispiel die von [Mye96] vorgenommene Unterscheidung zwischen schwacher Nachhaltigkeit und starker Nachhaltigkeit mit dieser Darstellung visualisieren. Unter schwacher Nachhaltigkeit versteht [Mye96] Positionen, die das gegenwärtige Wirtschaftswachstum beibehalten wollen und positive Wechselwirkungen zwischen Wirtschaftswachstum und Umweltschutz betonen. Bei Positionen, die [Mye96] als starke Nachhaltigkeit bezeichnet, stehen hingegen die natürlichen Grenzen des Wachstums im Vordergrund. Das Spektrum möglicher Sichtweisen zwischen schwacher Nachhaltigkeit und starker Nachhaltigkeit ergibt eine Verbindungslinie zwischen Punkt B und der Kante AC [Con07]. Auch wenn sich die „wahre" Bedeutung Nachhaltiger Entwicklung innerhalb des Dreiecks nicht scharf abgrenzen lässt, kann die Darstellung nach [Con07] dabei helfen, Beliebigkeit bei der Verwendung des Nachhaltigkeitsbegriffs entgegenzuwirken. Nachhaltigkeitsberichte von Unternehmen, die Greenwashing betreiben, oder Positionen wie die von [Mor02], die allein auf Wirtschaftswachstum und freie Märkte setzen, wären außerhalb des Bereichs im Zentrum des Dreiecks darzustellen.

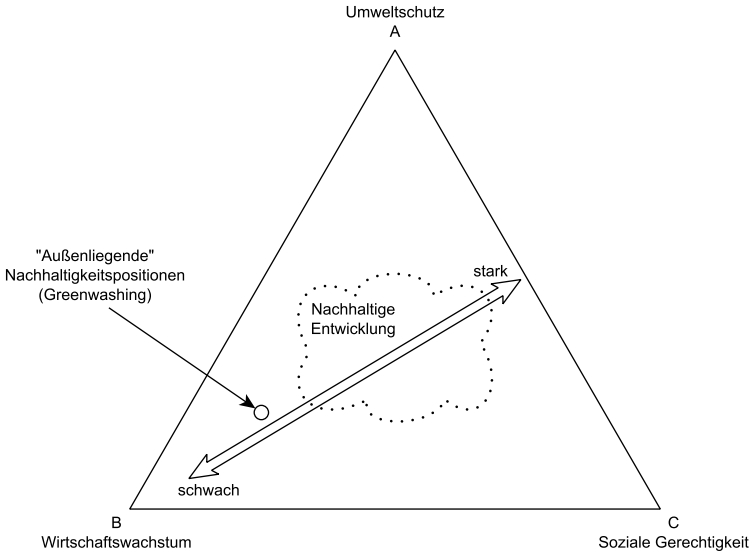


Abbildung 3.1: Illustration unterschiedlicher Nachhaltigkeitspositionen, in Anlehnung an [Con07, S. 271].

Die von [Con07] beschriebenen Pole oder Ecken des Dreiecks, Umweltschutz, Wirtschaftswachstum und soziale Gerechtigkeit entsprechen der weit verbreiteten Unterscheidung der ökologischen, ökonomischen und sozialen Nachhaltigkeitsdimension [Pur19]. Im weiteren Verlauf der Arbeit wird diese Unterscheidung verwendet, um die Diskussion von verschiedenen Aspekten Nachhaltiger Entwicklung zu strukturieren. Es wird deutlich, dass Nachhaltige Entwicklung ein gesellschaftliches und politisches Ziel ist, zu dem Unternehmen einen Beitrag leisten sollen [Gar04]. Das zeigt sich zum Beispiel in den Nachhaltigkeitszielen, die von den Vereinten Nationen in der 2030-Agenda für Nachhaltige Entwicklung ausgegeben wurden [UNE15]. So werden zum Beispiel menschenwürdige Arbeit für alle (Ziel 8) und verantwortungsvolle Konsum- und Produktionsmuster (Ziel 12) gefordert. Von diesem Standpunkt aus sollte die Leistung eines Unternehmens also nicht nur nach finanziellen Gesichtspunkten beurteilt werden, sondern auch entsprechend der sozialen und ökologischen Auswirkungen seiner Aktivitäten [Elk97].

3.2 Lebenszyklusgedanke und Lebenszyklusanalyse

Die Analyse der sozialen und ökologischen Auswirkungen von Unternehmensaktivitäten stellen eine komplexe Herausforderung dar. So muss eine Vielzahl an unterschiedlichen Nachhaltigkeitsaspekten (unterschiedliche ökologische und soziale Problemfelder) in eine Analyse miteinbezogen werden. Dabei ist zu bedenken, dass Nachhaltigkeitsauswirkungen einer Aktivität möglicherweise auch außerhalb der Unternehmensgrenzen zu verorten sind. Ein wichtiges Konzept ist in diesem Zusammenhang der Lebenszyklusgedanke [Hei02b, Mel10]. Der Lebenszyklusgedanke beschreibt, dass bei der Nachhaltigkeitsanalyse eines Phänomens, zum Beispiel eines Produkts oder eines Unternehmens, eine systemische Perspektive eingenommen wird. Diese systemische Perspektive umfasst auch die Beziehungen und Wechselwirkungen zwischen dem betrachteten Produkt oder Unternehmen und der (natürlichen) Umwelt sowie anderen Anspruchsgruppen. Das Ziel ist die Identifikation von Verbesserungsmöglichkeiten für das gesamte System [Hei00]. Die Anwendung des Lebenszyklusgedankens wird als wesentlich für die Untersuchung der Nachhaltigkeit von Unternehmen gesehen [Mel10, Elk94].

Die Lebenszyklusanalyse[1] (engl. „Life Cycle Assessment“) ist eine standardisierte Methode, eine solche Untersuchung durchzuführen [ISO06a]. Die Standardisierung dient dazu, Bias in Analysen und ungerechtfertigten Nachhaltigkeitsbehauptungen entgegenzuwirken [Fin14a]. Die (Produkt-)Lebenszyklusanalyse wurde zunächst zur Analyse der ökologischen Auswirkungen eines Produkts entwickelt [Gui11, Bjø18]. Eine Weiterentwicklung der Methode erweitert die Perspektive zur Analyse der ökologischen Auswirkungen eines ganzes Unternehmens [ISO14]. Im Folgenden wird in dieser Arbeit daher auch zwischen Produkt-Lebenszyklusanalysen und Unternehmens-Lebenszyklusanalysen unterschieden. Weiterhin existieren inzwischen sowohl auf Produkt- als auch auf Unternehmensebene Ansätze, die auch die soziale Nachhaltigkeitsdimension berücksichtigen [Gui11, Mar15, UNE20].

Der Begriff Lebenszyklus muss dabei vom in Kapitel 2 beschriebenen Geschäftsprozesslebenszyklus abgegrenzt werden. Im Fall der Lebenszyklusanalyse werden unter einem (Produkt-)Lebenszyklus insbesondere die Abschnitte einer Wertschöpfungskette verstanden. Sie reichen dabei von der Extraktion der für das Produkt benötigten

[1] Im Deutschen ist auch die Bezeichnung *Ökobilanzierung* gebräuchlich [Klö09]. Da es inzwischen auch Varianten der Methode für die soziale Nachhaltigkeitsdimension gibt, wird hier in Anlehnung an den im Englischen gebräuchlichen Namen der Methode (*Life Cycle Assessment*) der Begriff *Lebenszyklusanalyse* verwendet.

Rohstoffe über die Erstellung des Produkts und die Benutzung bis hin zur Entsorgung beziehungsweise dem Recycling des Produkts [ISO06a]. Bei einem Unternehmens-Lebenszyklus wird diese Betrachtung auf die Inputs und Outputs eines ganzen Unternehmens ausgeweitet [ISO14].[1]

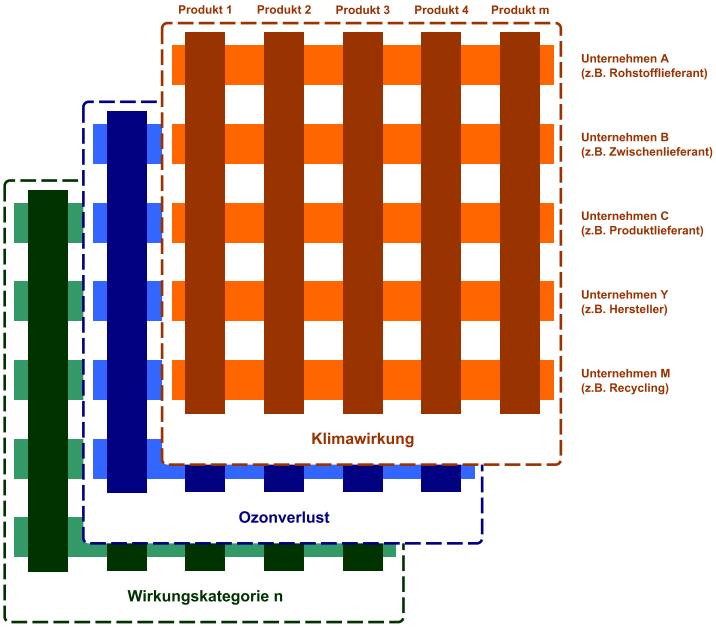


Abbildung 3.2: Vollständige Unternehmens-Lebenszyklusanalyse, in Anlehnung an [UNE15, S. 86].

Abbildung 3.2 stellt den Umfang einer vollständigen Unternehmens-Lebenszyklusanalyse nach [UNE15] dar. Sie umfasst eine Vielzahl an sogenannten Wirkungskategorien, die in der Abbildung als mit gestrichelten Linien umrandete Flächen dargestellt werden. Mit Wirkungskategorien sind verschiedene Nachhaltigkeitsaspekte gemeint (im Beispiel: Klimawirkung oder Ozonverlust), auf welche die Aktivitäten des Unternehmens Auswirkungen haben [UNE15, Mar15].[2] Für die Lebenszyklusanalyse eines

[1] Im Folgenden ist mit dem Begriff *Lebenszyklus* das hier beschriebene Konzept aus der Nachhaltigkeitsanalyse gemeint. Wenn es um Konzepte des Geschäftsprozessmanagements geht, wird der Begriff *Geschäftsprozesslebenszyklus* verwendet.

[2] In Unterabschnitt 3.3.3 wird näher auf das Thema Wirkungskategorien eingegangen.

Unternehmens (in der Abbildung zum Beispiel Unternehmen Y) sind dann sämtliche Produkte, beziehungsweise Inputs und Outputs des Unternehmens entlang ihrer Lebenszyklen, dargestellt als vertikale Balken, zu betrachten.

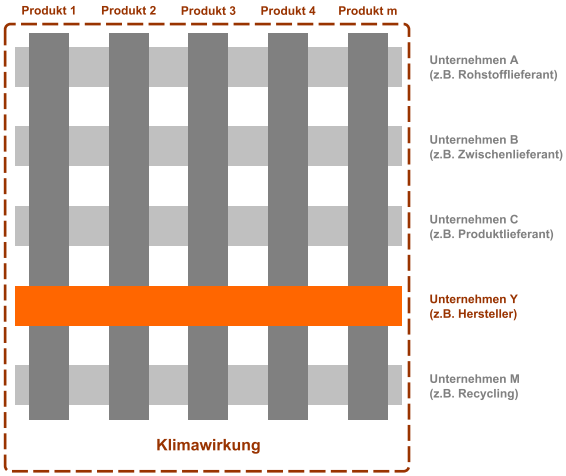


Abbildung 3.3: Scope 1-Analyse nach dem Treibhausgasprotokoll, in Anlehnung an [UNE15, S. 86].

Es wird deutlich, dass das Durchführen einer Lebenszyklusanalyse (sei es für ein einzelnes Produkt oder ein ganzes Unternehmen) komplex und umfangreich werden kann. Die Methode kann jedoch auf verschiedene Situationen zugeschnitten werden. So ist es möglich, die Analyse auf einen Ausschnitt des beschriebenen vollständigen Umfangs zu beschränken. Dieser Ausschnitt muss jedoch klar und präzise definiert werden [UNE15]. Das Treibhausgasprotokoll ist beispielsweise ein weit verbreiteter Ansatz zur Analyse der Auswirkungen der Aktivitäten eines Unternehmens auf die Globale Erwärmung [WBC04]. Er orientiert sich an den Konzepten der Lebenszyklusanalyse und kann somit als Variante der Lebenszyklusanalyse verstanden werden, bei der nur eine Wirkungskategorie (Klimawirkung) betrachtet wird. Weiterhin definiert das Treibhausgasprotokoll unterschiedliche *Scopes*, also Umfänge einer Analyse. In *Scope 1* werden lediglich direkte Treibhausgasemissionen auf dem Betriebsgelände eines Unternehmens untersucht. Abbildung 3.3 veranschaulicht das dadurch vorgenommene Zuschneiden einer Unternehmens-Lebenszyklusanalyse. In Abbildung 3.3

wird im Vergleich zu Abbildung 3.2 nur eine Wirkungskategorie betrachtet und weitere potentiell zu betrachtenden Lebenszyklus-Abschnitte außerhalb des betrachteten Unternehmens Y sind ausgegraut. *Scope 2* erweitert den Umfang einer Analyse, sodass auch Treibhausgasemissionen aus dem Bezug von Energie ermittelt werden. In *Scope 3* werden schließlich die Treibhausgasemissionen entlang des gesamten Lebenszyklus eines Produkts oder eines Unternehmens betrachtet. In Abbildung 3.2 entspräche das einem orangenen vertikalen Balken (für ein Produkt) beziehungsweise der gesamten orangenen Fläche (für ein Unternehmen).

Abbildung 3.4 veranschaulicht die Phasen der Durchführung einer Lebenszyklusanalyse nach [ISO14, UNE15, ISO06a, ISO06b]. Die erste Phase ist die Ziel- und Umfangsdefinition. Hier werden die Systemgrenzen, sowie grundlegende Anforderungen und Annahmen festgelegt. Zweitens erfolgt die Inventaranalyse, bei der Daten gesammelt und Berechnungen durchgeführt werden, um relevante Inputs und Outputs zu quantifizieren. Als dritte Phase folgt die Wirkungsabschätzung, bei der die Signifikanz von Nachhaltigkeitsauswirkungen bewertet wird. Als letzte Phase erfolgt die Interpretation der Ergebnisse. Wie in der Abbildung dargestellt, folgen die Phasen nicht notwendigerweise strikt sequentiell aufeinander, sondern bedingen sich gegenseitig. Die Orientierungshilfen empfehlen auch ein iteratives Vorgehen, sodass beispielsweise in einer vorläufigen Analyse die signifikantesten Problemfelder ermittelt werden, für die dann die Anforderungen in Bezug auf Datenqualität für eine vertieften Analyse verschärft werden [UNE15]. Im Folgenden werden die einzelnen Phasen einer Unternehmens-Lebenszyklusanalyse ausführlicher vorgestellt. Die Beschreibung basiert auf dem entsprechenden Standard [ISO14] und der vom Umweltprogramm der Vereinten Nationen herausgegebenen Orientierungshilfe [UNE15]. Das Vorgehen folgt dabei den Phasen und Konzepten der Produkt-Lebenszyklusanalyse, wie sie in den zugehörigen Standards [ISO06a, ISO06b] festgelegt sind.

Phase 1 (Ziel- und Umfangsdefinition): Zunächst werden Ziel und Umfang der Analyse festgelegt. Die Ziele für die Durchführung einer Unternehmens-Lebenszyklusanalyse können vielfältig sein, müssen aber offengelegt werden. Mögliche analytische Ziele können das Gewinnen von Verständnis über die Lieferkette und die Prozesse des Unternehmens oder die Identifikation von Problemfeldern bezüglich der Nachhaltigkeitsauswirkungen der Unternehmensaktivitäten sein. Die gewonnenen Erkenntnisse können dann die Basis etwa für die Berichterstattung und Kommunikation mit externen Anspruchsgruppen oder auch für das Marketing bilden. Bezüglich des festzulegenden Umfangs ist es zulässig, nur einen Ausschnitt des Gesamtlebenszyklus eines Unternehmens zu untersuchen. Dabei muss die Umfangsdefinition jedoch kompatibel mit

den festgelegten Zielen der Analyse sein und transparent dargelegt werden. Zur Umfangsdefinition zählen unter anderem eine adäquate Beschreibung des untersuchten Unternehmens, die Festlegung der Anforderungen an die Datenqualität und die Auswahl der zu untersuchenden Wirkungskategorien.

Phase 2 (Inventaranalyse): In der nächsten Phase, der Inventaranalyse, wird das zu untersuchende System modelliert und die Inputs (wie Energie, Wasser und Materialien) und Outputs (wie Produkte, Abfall, und Emissionen) der im Lebenszyklus durchgeführten Aktivitäten werden quantifiziert. Es wird insbesondere zwischen direkten und indirekten Aktivitäten unterschieden. Direkte Aktivitäten sind solche, die innerhalb des untersuchten Unternehmens durchgeführt werden. Sonstige Aktivitäten, die entlang des Lebenszyklus stattfinden, werden indirekte Aktivitäten genannt (beispielsweise die Produktion von eingekauften Materialien oder die spätere Nutzung von erstellten Produkten).

Phase 3 (Wirkungsabschätzung): Auf die Inventaranalyse folgt die Wirkungsabschätzung, hier werden die zuvor erhobenen Inventardaten in Nachhaltigkeitsauswirkungen übersetzt. Hierbei findet immer eine Klassifizierung und Charakterisierung der Inventardaten statt, womit die Zuordnung von Inventardaten zu Wirkungskategorien

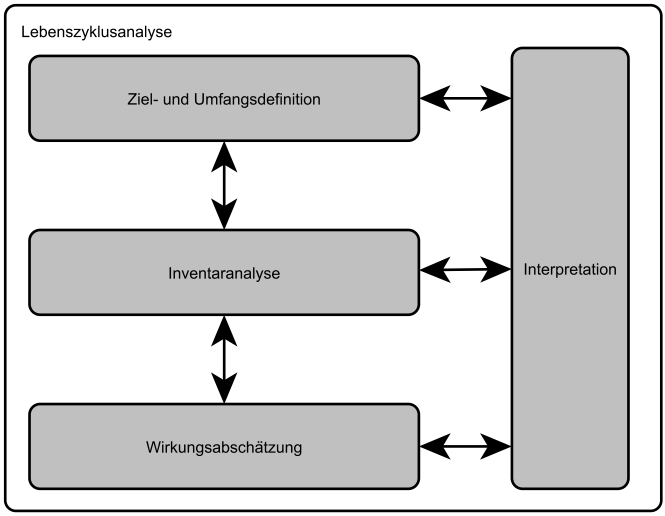


Abbildung 3.4: Phasen einer Lebenszyklusanalyse, in Anlehnung an [ISO06a, S. 17].

(CO2-Emissionen und andere Treibhausgase können zum Beispiel der Wirkungskategorie Klimawirkung zugeordnet werden) und eine Abschätzung der Signifikanz (verschiedene Treibhausgase haben unterschiedliche starke Klimawirkung) gemeint ist. Weiterhin kann in dieser Phase optional eine Normalisierung, Aggregation und Gewichtung der Indikatoren stattfinden.

Phase 4 (Interpretation): In der Phase Interpretation werden schließlich die Ergebnisse der Inventaranalyse und Wirkungsabschätzung gemeinsam betrachtet. Unter Einbeziehung des festgelegten Ziels und des Umfangs der Analyse soll dabei auch die Konsistenz der Ergebnisse bewertet werden. Es werden Schlussfolgerungen gezogen, Grenzen der Analyse aufgezeigt und Handlungsempfehlungen gegeben.

3.3 Begriffe und Konzepte

Zur Vorbereitung einer Gegenüberstellung der Begriffe und Konzepte des Geschäftsprozessmanagements und der Lebenszyklusanalyse in Kapitel 4 wird im Folgenden ausführlicher auf Begriffe und Konzepte der Lebenszyklusanalyse eingegangen. Die Struktur dieses Abschnitts folgt dabei der Struktur des entsprechenden Abschnitts 2.2 zu Geschäftsprozessmanagement in Kapitel 2. So wird zunächst in Unterabschnitt 3.3.1 ein Überblick zu Grundbegriffen gegeben. In den Unterabschnitten 3.3.2 und 3.3.3 wird dann die Betrachtung des Stoffstromkonzepts und der Wirkungsabschätzung vertieft. Die Beschreibungen fokussieren dabei zunächst auf die ökologische Nachhaltigkeitsdimension, da die Lebenszyklusanalyse ursprünglich für ökologische Aspekte konzipiert wurde [Gui11, Bjø18] und die Methode daher für diese Nachhaltigkeitsdimension ausgereifter ist. Bisher ist auch noch keine vollständige konzeptuelle und methodische Integration für die Betrachtung der sozialen und ökologischen Nachhaltigkeitsdimension in einer Lebenszyklusanalyse erfolgt [Gui11, Mar20]. Auf bestehende Ansätze für soziale Lebenszyklusanalysen wird in Unterabschnitt 3.3.3 ergänzend eingegangen.

3.3.1 Grundbegriffe

In diesem Unterabschnitt werden Grundbegriffe der Lebenszyklusanalyse auf Basis der zugrundeliegenden Standards [ISO06a, ISO06b] sowie der ergänzenden Quelle [Klö09] dargelegt. Tabelle 3.1 fasst die Grundbegriffe der Lebenszyklusanalyse in der Reihenfolge ihres Erscheinens in den folgenden Beschreibungen zusammen.

Tabelle 3.1: Grundbegriffe der Lebenszyklusanalyse nach [ISO06a, Klö09].

Begriff	Beschreibung
Produkt	Ware oder Dienstleistung.
Lebenszyklus	Phasen im „Leben“ eines Produkts von der Rohstoffgewinnung oder Rohstofferzeugung bis zur Entsorgung.
Produktsystem	Modell des Lebenszyklus eines Produkts.
Prozess	Menge von in Wechselbeziehung oder Wechselwirkung stehenden Tätigkeiten, die Inputs in Outputs umwandelt.
Prozessmodul	Kleinste Prozesseinheit, für die Inputs und Outputs quantifiziert werden.
Stoffstrom	Quantifizierter Fluss von Materialien, Produkten, Stoffen oder Energie.
Funktionelle Einheit	Quantifizierter Nutzen eines Produktsystems für die Verwendung als Vergleichseinheit.
Wirkungskategorie	Nachhaltigkeitsaspekt, dem Inputs und Outputs eines Produktsystems zugeordnet werden können.
Wirkungsindikator	Quantifizierung einer Wirkungskategorie.

Der Zweck einer (Produkt-)Lebenszyklusanalyse ist die möglichst vollständige Abschätzung der von einer Ware oder Dienstleistung verursachten potentiellen Umweltlasten [ISO06a]. Im Zentrum steht dabei die Betrachtung des Lebenszyklus eines Produkts, von der Rohstoffgewinnung bis zur Entsorgung [ISO06a]. Mit Produkten sind dabei nicht nur materielle Güter gemeint, sondern „jede Ware oder Dienstleistung“, wie auch „die Vermittlung von Informationen“ [ISO06a]. Der Fokus liegt aber mehr auf der Produktion materieller Güter und weniger auf Dienstleistungen oder administrativen Prozessen [Wol19, UNE15].

Das zum Zweck der Analyse erstellte Modell des Produkt-Lebenszyklus wird auch Produktsystem genannt. Es unterteilt den Lebenszyklus in mehrere Prozessschritte [ISO06a]. Abbildung 3.5 zeigt eine Skizze für ein Produktsystem mit verschiedenen Teilprozessen wie Rohstoffgewinnung, Produktion oder Transport.

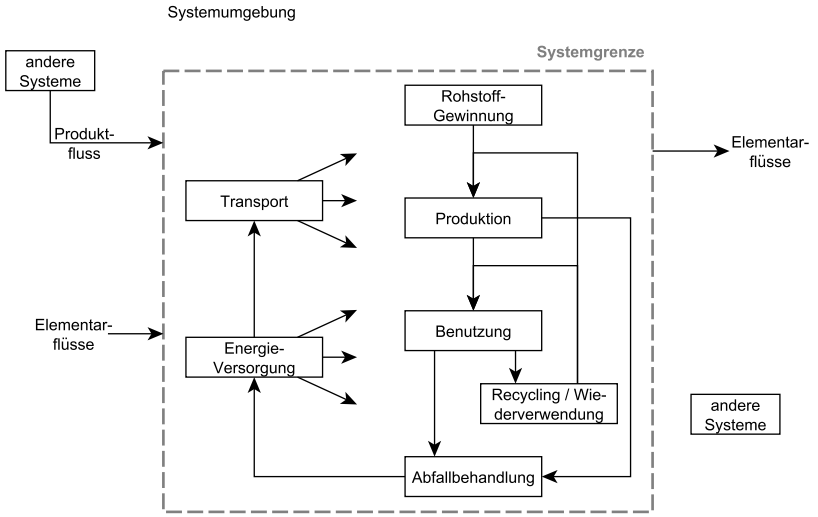


Abbildung 3.5: Generisches Produktsystem mit Stoffströmen, in Anlehnung an [ISO06a, S. 10]).

Nach [ISO06a, S. 11] ist ein Prozess eine Menge „von in Wechselbeziehung oder Wechselwirkung stehenden Tätigkeiten“, die Inputs in Outputs verwandeln. Nach dieser Prozessdefinition ist auch ein Lebenszyklus mit Inputs, die aus der Umwelt fließen, und Outputs, die in die Umwelt fließen, als ein Prozess zu verstehen. Inputs und Outputs beziehungsweise Flüsse können dabei beispielsweise Rohstoffe, Zwischenprodukte, Koppelprodukte, Emissionen oder Energie sein [ISO06a]. In Abbildung 3.5 sind die Flüsse als Pfeile zwischen den Teilprozessen dargestellt. Die kleinste Prozesseinheit, für die Inputs und Outputs quantifiziert werden, heißt Prozessmodul [ISO06a].

Flüsse zwischen der Technosphäre (dem, was Menschen kontrollieren) und der Umwelt (alles, was nicht Technosphäre ist) heißen „Elementarflüsse“ [ISO06a, S. 19, Klö09, S.30]. [ISO06a] gibt Beispiele für die jeweiligen Flüsse in der Skizze: Elementarflüsse können Erdöl oder Sonneneinstrahlung als Input beziehungsweise Emissionen in die Luft oder Bodenverunreinigungen als Output sein. Zwischen den Teilprozessen können Materialien oder Baugruppen als Zwischenprodukte fließen (sogenannte „Zwischenproduktflüsse“ [ISO06a, S. 12]). Bauteile oder recycelte Materialien können von oder zu anderen Produktsystemen als Produktflüsse abgegeben werden. Allgemein

wird bei all diesen unterschiedlichen Arten von Flüssen auch von „Stoffströmen“ gesprochen [Klö09, S. 149]. In Unterabschnitt 3.3.2 wird näher auf das Konzept der Stoffströme eingegangen.

Um die ökologische Nachhaltigkeit eines Produktsystems zu bewerten, werden die Inputs und Outputs verschiedenen Wirkungskategorien zugeordnet (zum Beispiel CO2-Emissionen zur Wirkungskategorie Klimawirkung) und über entsprechende Umrechnungsfaktoren kann ein quantifizierter Wirkungsindikator für jede Wirkungskategorie angegeben werden [Klö09]. Wirkungskategorien und -indikatoren werden in Unterabschnitt 3.3.3 ausführlicher besprochen.

Ein weiteres wichtiges Konzept der Lebenszyklusanalyse ist die funktionelle Einheit als ein „quantifizierter Nutzen eines Produktsystems“ [ISO06a, S. 12]. Die Inputs und Outputs des Produktsystems werden anhand dieser quantifizierten Größe normiert [ISO06b]. Erst über eine eindeutige Definition des Nutzens eines Produktsystems werden Vergleiche zwischen Produkten, beziehungsweise auch zwischen materiellen Produkten und Dienstleistungen möglich [Klö09, ISO06b]. Für einen Vergleich zwischen einem Lufttrocknungssystem und Papierhandtüchern könnte beispielsweise eine Anzahl getrockneter Handpaare als funktionelle Einheit festgelegt werden [ISO06a]. Ein anderes Beispiel wäre die Bereitstellung einer bestimmten Menge Flüssigkeit als funktionelle Einheit, um die Nachhaltigkeitsauswirkungen von Mehrwegflaschen mit denen von Einwegflaschen zu vergleichen [Klö09].

Wie in Abbildung 3.5 skizziert, können Produktsysteme auch grafisch dargestellt beziehungsweise modelliert werden, wobei jedes Kästchen ein Prozessmodul darstellt [Klö09]. Im Rahmen der Durchführung einer Lebenszyklusanalyse wird zunächst ein grobes Systemfließbild erstellt und in einem iterativen Prozess Stück für Stück verfeinert [Klö09]. Je nach Datenlage und Zielsetzung der Lebenszyklusanalyse unterscheidet sich der Detaillierungsgrad der Prozessmodule. So kann ein Prozessmodul in einem Fall einen Prozessschritt wie eine Abfüllung oder eine Metallverformung darstellen und in einem anderen Fall eine ganze Produktionsstätte [Klö09]. Auf verschiedene existierende Ansätze zur Modellierung von Produktsystemen wird in Abschnitt 3.4 näher eingegangen.

Bei [Klö09] wird weiterhin beschrieben, dass innerhalb eines modellierten Produktsystems auch Datensätze unterschiedlicher Spezifizität zum Einsatz kommen können. Damit ist gemeint, dass zur Modellierung verwendete Datensätze spezifisch, also bei einem bestimmten Unternehmen für einen bestimmten Prozess erhoben, oder generisch sein können. Bei generischen Datensätzen handelt es sich um „Mittelwerte oder

repräsentative Einzelwerte“ [Klö09, S. 133], wie zum Beispiel ein Prozessmodul zur Erzeugung elektrischer Energie in Deutschland. Im Allgemeinen wird die Verwendung von spezifischen Datensätzen angestrebt. Der Einsatz von generischen Daten erfolgt dann, wenn keine spezifischen Daten vorliegen. Er kann aber auch fachlich begründet sein, etwa dann, wenn nicht ein spezielles Produkt sondern eine Gruppe ähnlicher Produkte analysiert werden sollen [Klö09].

Die hier beschriebenen Begriffe und Konzepte der Lebenszyklusanalyse wurden im Hinblick auf eine Gegenüberstellung mit dem Geschäftsprozessmanagement zusammengestellt. Weitere Aspekte der Lebenszyklusanalyse, die in dieser Arbeit nicht vertieft behandelt werden, betreffen zum Beispiel die Umwandlung unterschiedlicher Energieformen (wie Elektrischer Strom, Wärme und Kernenergie) und die Behandlung unterschiedliche Wirkungsgrade der Elektrizitätserzeugung [Klö09]. Im Folgenden wird die Betrachtung des Stoffstromkonzepts (Unterabschnitt 3.3.2) sowie von Wirkungsindikatoren (Unterabschnitt 3.3.3) vertieft.

3.3.2 Stoffströme

Die Zusammenstellung und Quantifizierung der Inputs und Outputs je Prozessmodul beziehungsweise Stoffströme geschieht in der Inventaranalyse-Phase einer Lebenszyklusanalyse [ISO06a]. Bei [Klö09, S. 11] wird die Inventaranalyse auch als „Stoff- und Energieanalyse des untersuchten Systems von der Wiege bis zur Bahre“ eines Produkts beschrieben. Das Ergebnis dieser Zusammenstellung wird auch Sachbilanz genannt [Klö09]. Die Stoffströme eines Produktsystems sollten dabei bis hin zu Elementarflüssen, also dem stofflichen oder energetischen Austausch mit der Umwelt zurückverfolgt werden [ISO06b]. So werden auch bei der Analyse von Dienstleistungen die Stoffströme entlang der betroffenen Wertschöpfungsketten untersucht [Klö09]. Beispiele für Inputs sind Mineralien aus Erzen, oder Hilfsstoffe wie Dünger oder Schmierstoffe [ISO06b]. Unter Outputs werden neben erstellten Produkten zum Beispiel auch Emissionen in die Luft wie Kohlenmonoxid, Kohlendioxid oder Stickstoffoxide erfasst [ISO06b]. [ISO06b] nennt als Kategorien von Stoffströmen zum Beispiel Energieinputs, Rohstoffinputs, Produkte und Emissionen in Luft, Wasser und Boden. Abbildung 3.6 zeigt die Skizze eines Prozessmoduls unter anderem mit Materialien, Energie, Wasser und Hilfsstoffen als Input und Produkten, Abwärme und Emissionen als Output. Der

Pfad in einem Produktsystem entlang der Vorprodukte bis hin zu Herstellung und Gebrauch des betrachteten Produkts wird auch als Hauptkette bezeichnet und alle anderen Inputs und Outputs als Nebenketten (wobei die Unterscheidung zwischen Haupt- und Nebenketten nicht immer eindeutig ist) [Klö09].

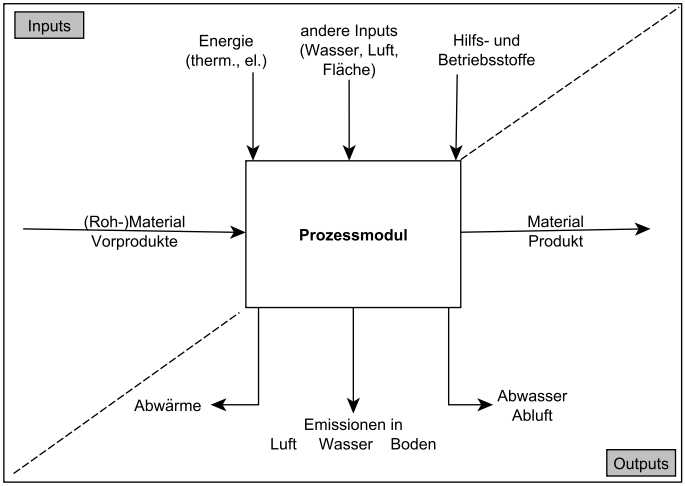


Abbildung 3.6: Verschiedene Inputs und Outputs eines Prozessmoduls, in Anlehnung an [Klö09, S. 67].

Die Sachbilanz hat den Anspruch, möglichst alle umweltrelevanten Inputs und Outputs zu quantifizieren [ISO06a]. Aus diesem Anspruch folgt, dass Naturgesetze (Erhaltung der Masse, Erhaltung der Energie, ...) einen Rahmen für die Bestimmung der Inputs und Outputs bilden [Klö09, ISO06b]. In praktischer Hinsicht bedeutet dies, dass Inputs und Outputs auf Basis von entsprechenden Gesetzmäßigkeiten abgeschätzt werden können [Klö09].

Unter gewissen Voraussetzungen können bestimmte Inputs und Output abgeschnitten werden, um die Inventaranalyse praktikabel zu gestalten. Die Entscheidung ist zum Beispiel dann gerechtfertigt, wenn ersichtlich ist, dass die betreffenden Stoffströme für die betrachteten Wirkungskategorien nicht relevant sind oder nur sehr wenig Masse beziehungsweise Energie zum gesamten Produktsystem beitragen [Klö09]. Grundsätzlich ist das Weglassen von Stoffströmen „nur zulässig, wenn damit die allgemeinen Schlussfolgerungen [...] [der Analyse] nicht wesentlich verändert werden“

[ISO06b, S. 18]. Es gibt entsprechende Konventionen für Abschneidekriterien [ISO06a], zum Beispiel, dass ein abzuschneidender Stoffstrom nicht mehr als 1% der Masse des Gesamtsystems ausmachen darf [Klö09]. Derartige Entscheidungen müssen insbesondere bezüglich der Einbeziehung von Infrastruktur und Investitionsgütern getroffen werden [Klö09]. So beträgt zum Beispiel die zur Errichtung eines Kraftwerks benötigte Energie nur einen Bruchteil dessen, was ein Kraftwerk über seine Lebensdauer erzeugt. Aufgrund dieser Überlegung kann begründet werden, dass dieser Pfad (der Lebensweg des Kraftwerks) bei der Ermittlung relevanter Stoffströme, die mit der Erzeugung der für ein Produkt benötigten Energie zusammenhängen, in einer Lebenszyklusanalyse nicht weiterverfolgt wird. Die Produktion von Infrastruktur und Investitionsgütern (zum Beispiel Maschinen, die für die Herstellung eines Produkts benötigt werden) kann aber auch relevante Stoffströme beitragen, sodass derartige Entscheidungen in jedem Fall geprüft werden sollten [Fri07, Kas14, Zen14].

Das Auftreten von „Verzweigungen“ zwischen Prozessmodulen stellt eine weitere Herausforderung dar [Klö09, S. 69]. Mit Verzweigungen sind solche Fälle gemeint, bei denen ein Prozessmodul mehrere Inputs (gegebenenfalls aus unterschiedlichen Prozessmodulen) erhält oder mehrere nutzbare Outputs beziehungsweise Koppelprodukte (zum Beispiel Getreide und Stroh als Output eines landwirtschaftlichen Prozesses) erzeugt. Prozessmodule mit Verzweigungen werden auch Multi-Input- oder Multi-Output-Prozesse genannt [Klö09]. Bei Multi-Output-Prozessen besteht das Problem darin, unterschiedlichen Produkten die Umweltlasten eines Prozessmoduls angemessen zuzuordnen [Klö09]. [Klö09] schlägt für die Behandlung von derartigen Verzweigungen ein abgestuftes Vorgehen vor. Die sieben Schritte des Vorgehens werden im Folgenden in gekürzter Form wiedergegeben:

1 Falls ein Koppelprodukt keinen ökonomischen Nutzen hat, ist es Abfall und es werden ihm keine Umweltlasten zugeordnet.

2 Als „wissenschaftlichste Lösung“ des Problems wird die Systemerweiterung genannt. Dabei wird das Koppelprodukt Teil des modellierten Produktsystems. In diesem Fall findet keine Aufteilung der Umweltlasten statt, allerdings ergeben sich so schnell unübersichtlich große Systeme, die nicht mehr mit vertretbarem Aufwand zu untersuchen sind.

3 Aus wissenschaftlicher Sicht ist neben der Systemerweiterung auch die Systemverkleinerung gegenüber einer (subjektiven) Aufteilung der Umweltlasten zu präferieren. Zu groß gewählte Prozessmodule können möglicherweise so eingeengt werden, dass nur ein Produkt als Output erzeugt wird (beispielsweise wird nur eine Produktstraße untersucht statt einer ganzen Organisation).

4 Eine Zuordnung nach „physikalischer Verursachung“ ist dann möglich, wenn „naturwissenschaftlich-technische Begründungen für die Zuordnung von Umweltlasten“ gegeben sind [Klö09, S. 102]. Ein Beispiel wäre eine Müllverbrennungsanlage, bei der gewisse bei der Verbrennung auftretende Emissionen eindeutig bestimmten verbrannten Abfällen zuzuordnen sind.

5 In den Fällen, in denen die Strategien 1 bis 4 nicht angewendet werden können, kann eine Allokation nach Masse durchgeführt werden. Die Umweltlasten werden auf die Koppelprodukte nach ihrem jeweiligen Masseanteil verteilt. Dieses Vorgehen ist als einfache Allokationsregel in der Praxis weit verbreitet. Sie sollte aber insbesondere dann nicht angewendet werden, wenn zwischen zwei Koppelprodukten ein großer Unterschied in ihrem ökonomischen Wert besteht. Das „Diamanten-Paradoxon“ [Klö09, S. 103] illustriert dieses Problem. Beim Abbau von Diamanten entstehen große Mengen an Gestein, die für den Straßenbau genutzt werden können. Sie sind also nicht als Abfall zu zählen (siehe Strategie 1) – aber bei einer Allokation nach Masse würde dem nur als Nebenprodukt entstehenden Gestein der Großteil der Umweltlasten zugeordnet werden.

6 Die Massen-Allokation kann über den ökonomischen Wert der Produkte gewichtet werden, um das unter Punkt 5 beschriebene Problem abzufangen.

7 Für Spezialanwendungen können auch alternative Allokationen, zum Beispiel nach Molmasse und Brennwert, durchgeführt werden.

Es wird deutlich, dass bei der Allokation oftmals subjektive Entscheidungen getroffen werden müssen. Umso wichtiger ist dabei ein transparentes und gut dokumentiertes Vorgehen [Klö09, ISO06b].

3.3.3 Wirkungsindikatoren

Die ermittelten Inputs und Outputs der Sachbilanz müssen in Bezug auf ihre Umweltwirkung interpretiert werden. In der Wirkungsabschätzung erfolgt daher eine ökologische Analyse und Gewichtung der Sachbilanzdaten [Klö09]. Die Wirkungsabschätzung ist der „Bestandteil der Ökobilanz, der dem Erkennen und der Beurteilung der Größe und Bedeutung von [...] [ökologischen Auswirkungen] eines Produktsystems im Verlauf des [...] [Lebenszyklus] des Produktes dient“ [ISO06b, S. 11]. In der Wirkungsabschätzung erfolgt dazu zunächst eine Zuordnung der Sachbilanzergebnisse zu

Wirkungskategorien (Klassifizierung) und dann eine Berechnung der Wirkungsindikatorwerte (Charakterisierung). Optional kann anschließend eine Normierung, Ordnung und Gewichtung stattfinden [Klö09].

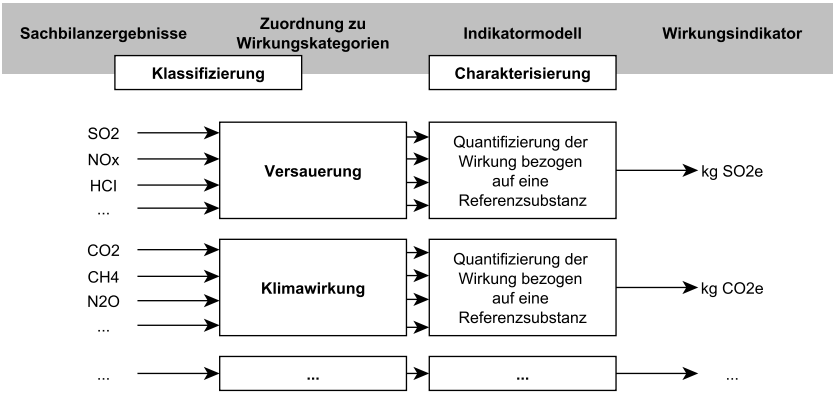


Abbildung 3.7: Klassifizierung und Charakterisierung von Wirkungsindikatoren, in Anlehnung an [Klö09, S. 206].

Das Vorgehen bei der Klassifizierung und Charakterisierung ist in Abbildung 3.7 skizziert und lässt sich anhand der Wirkungskategorie Klimawirkung verdeutlichen [Klö09]: In der Sachbilanz sind Werte für verschiedene Treibhausgase wie Methan und CO2 angegeben. Moleküle dieser Gase erhöhen die Strahlungsabsorption in der Atmosphäre, was zu einer Erhöhung der mittleren Temperatur der Troposphäre führt [Klö09]. Diese Sachbilanzwerte werden daher der Wirkungskategorie Klimawirkung zugeordnet (klassifiziert). Über einen Charakterisierungsfaktor kann die Klimawirkung für jedes Treibhausgas in CO2-Äquivalenten (CO2e) berechnet werden. Durch die Addition der Werte ergibt sich ein Wirkungsindikator für die Wirkungskategorie Klimawirkung. Es gibt eine Reihe von Listen von Wirkungskategorien für die Lebenszyklusanalyse, die sich nur geringfügig unterscheiden [Klö09]. Grundsätzlich ist festzuhalten, dass eine solche Liste von Wirkungskategorien nie vollständig ist, und sich mit dem wissenschaftlichen Fortschritt und öffentlichem Bewusstsein für Umweltprobleme ändern kann [Klö09]. [Klö09] listet neben der Klimawirkung die Umweltproblemfelder beziehungsweise Wirkungskategorien Ressourcenverbrauch, Stratosphärischer Ozonabbau, Humantoxizität, Ökotoxizität, Sommersmog, Versauerung Eutrophierung (Überdüngung), Belästigungen (Lärm, Geruch), Harte

Strahlung (Radioaktivität) und Abfall. Zu all diesen Wirkungskategorien müssen für eine sachgerechte Klassifizierung und Charakterisierung der Sachbilanzwerte Umweltwirkungsmechanismen beschrieben werden, was in vielen Fällen auch noch weiterer Forschung bedarf [Klö09, Fin14b].

Obgleich vergleichbare Ansätze zur Betrachtung der sozialen Nachhaltigkeitsdimension (soziale Lebenszyklusanalysen) im Vergleich zu den beschriebenen ökologischen Lebenszyklusanalysen weniger standardisiert und ausgereift sind, folgen sie den gleichen Prinzipien [Mar15, UNE20]. Auswirkungen der Unternehmensaktivitäten auf Aspekte sozialer Gerechtigkeit sollen möglichst umfangreich erfasst werden. Dazu wird genauso wie bei ökologischen Lebenszyklusanalysen der Lebenszyklus eines Produkts beziehungsweise der Inputs und Outputs eines gesamten Unternehmens betrachtet. Nur müssen die identifizierten Stoffströme dann, im Zuge einer Wirkungsabschätzung, sozialen Wirkungskategorien zugeordnet werden. Existierende Richtlinien orientieren sich dabei an internationalen Normen und Abkommen, die soziale Problemfelder beschreiben [UNE21]. Beispiele für soziale Wirkungskategorien sind demnach Aspekte sozialer Gerechtigkeit wie Kinderarbeit soziale Absicherung von Arbeitenden oder die Rechte indigener Gemeinschaften [UNE21].

3.4 Modellierungsansätze und Software-Werkzeuge

Zum Abschluss der Behandlung der Grundlagen zu Nachhaltigkeitsanalysen werden in diesem Abschnitt bestehende Modellierungsansätze und Software-Werkzeuge der Lebenszyklusanalyse vorgestellt. Das ermöglicht anschließend, in Kapitel 4, die Ansätze von Geschäftsprozessmanagement und Lebenszyklusanalyse auch aus technischer Sicht gegenüberzustellen und Schlussfolgerungen für Nachhaltiges Geschäftsprozessmanagement daraus zu ziehen. In diesem Abschnitt wird zunächst in Unterabschnitt 3.4.1 ein kurzer Überblick zu bestehenden Modellierungsansätzen und Software-Werkzeugen für Lebenszyklusanalysen gegeben. In Unterabschnitt 3.4.2 folgt eine vertiefte Beschreibung eines speziellen Ansatzes, den sogenannten Stoffstromnetzen, die mit den bereits vorgestellten Petri-Netzen (siehe Unterabschnitt 2.3.2) verwandt sind.

3.4.1 Überblick

Als etablierte Software-Werkzeuge für Lebenszyklusanalysen werden in der Literatur insbesondere SimaPro [PRé24], GaBi [Sph24], openLCA [Gre24a] und Umberto [iPo24] genannt [Fri20a, Bac19]. Sie werden zum Teil schon seit den 90er Jahren entwickelt und können für eine große Bandbreite von Fragestellungen in der Lebenszyklusanalyse eingesetzt werden [Bac19, Ung04, Mül97]. Die in der Lebenszyklusanalyse verfolgten Modellierungsansätze sollen Aussagen über die Nachhaltigkeitsauswirkungen eines Prozesses ermöglichen. Die erstellten Modelle eignen sich jedoch (im Vergleich zu anderen Modellierungsansätzen) weniger für die Steuerung von Prozessen in Unternehmen [Löf11]. Die Operationalisierung und Einbindung der Analyseergebnisse in die betrieblichen Informationssysteme wird in der Literatur zwar diskutiert, steht aber nicht im Vordergrund [UNE15, Löf11]. Weiterhin stehen Software-Werkzeuge zur Unterstützung von Lebenszyklusanalysen vor dem Zielkonflikt, einerseits möglichst generisch und flexibel einsetzbar zu sein und andererseits möglichst einfach bedient werden zu können [Ung04]. Es gibt daher neben den erwähnten Software-Werkzeugen, die eher generische Werkzeuge für Expertinnen und Experten sind, noch Bedarf an spezialisierten Werkzeugen, die auf die Anforderungen bestimmter Industriedomänen zugeschnitten sind [Reb04].

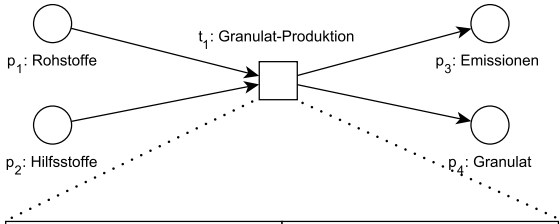
Die genannten Software-Werkzeuge verfolgen unterschiedliche Modellierungs- und Berechnungsansätze. Sie unterstützen jedoch alle, in unterschiedlicher Form, die (grafische) Modellierung von Prozessen beziehungsweise (Produkt-)Systemen und der dazugehörenden Stoffströme [Ung04, Mül97]. Dazu werden oftmals generische Datensätze integriert, oder das Einbinden von Datenbanken mit Standarddatensätzen unterstützt [Ung04, Fri20a, Häu95]. Bei der Modellierung der Prozesse werden Gleichungssysteme aufgestellt und üblicherweise lineare Zusammenhänge zwischen den Inputs und Outputs der einzelnen Prozessmodule beschrieben [Ung04, Hei02a, Möl97].[1] Insbesondere die Behandlung von speziellen Modellkonstellationen wie Prozessschleifen oder Multi-Input- und Multi-Output-Prozesse (siehe Ausführungen zum Thema Allokation in Unterabschnitt 3.3.2) stellen technische Herausforderungen dar und werden unterschiedlich gelöst [Hei02a].

[1] In der Lebenszyklusanalyse ist die Annahme von linearen Zusammenhängen in vielen Fällen als Vereinfachung akzeptiert, die Beschreibung nicht-linearer Zusammenhänge ist aber möglich [Möl95].

Im Kontext der vorliegenden Arbeit ist der Ansatz der Stoffstromnetze, der im Software-Werkzeug Umberto [iPo24] verfolgt wird, besonders relevant, da er Petri-Netz-Prinzipien für die Prozessmodellierung in der Lebenszyklusanalyse adaptiert. Wie in Abschnitt 2.3.2 erwähnt, werden Petri-Netze auch zur Prozessmodellierung im Geschäftsprozessmanagement eingesetzt. Eine nähere Betrachtung der Stoffstromnetze erlaubt es im weiteren Verlauf dieser Arbeit daher, Gemeinsamkeiten und Unterschiede der Modellierungsansätze von Lebenszyklusanalyse und Geschäftsprozessmanagement herauszuarbeiten.

3.4.2 Stoffstromnetze

Beim Ansatz der Stoffstromnetze werden Petri-Netz-Prinzipien für die Prozessmodellierung in der Lebenszyklusanalyse adaptiert [Häu95, Sch97a]. Dadurch werden sowohl flexible Modellierungs- und Berechnungsmöglichkeiten bereitgestellt, als auch Mechanismen, um die inhärente Komplexität der Prozessmodellierung handhabbar zu machen. Dazu zählen unter anderem Möglichkeiten zur Visualisierung, Hierarchisierung und Modularisierung von Prozessmodellen [Häu95].



Input:			Output:		
Stelle	Material	Menge	Stelle	Material	Menge
p_1	Erdöl (kg)	413,48	p_4	Granulat (kg)	100
p_2	Wasserstoff (kg)	2.424	p_3	Kohlendioxid (kg)	560,6
p_2	Hilfsstoffe (kg)	0,763	p_3	NOx (kg)	1,27

Abbildung 3.8: Beispiel für ein Stoffstromnetz, in Anlehnung an [Sch97b, S. 30].

Abbildung 3.8 zeigt ein Beispiel für ein Stoffstromnetz, das einen Granulatherstellungsprozess darstellt. Wie zu erkennen ist, wird für ein Stoffstromnetz zunächst eine Netzstruktur, bestehend aus Stellen, Transitionen und Kanten (siehe auch Definition 2.3.1

in Unterabschnitt 2.3.2), modelliert [Möl95]. Transitionen stehen dabei für Vorgänge, bei denen eine Stoffumwandlung stattfindet [Sch97b], und Stellen dienen als „Lager“ für das Halten und die Verteilung von Stoffbeständen [Sch97a]. Das gegebene Stoffstromnetz besteht aus einer Transition t_1, die über Kanten mit zwei Input-Stellen für Rohstoffe (p_1) und Hilfsstoffe (p_2) sowie zwei Output-Stellen für Emissionen (p_3) und Granulat (p_4) verbunden ist. An den Kanten wird angegeben, welche Stoffe in welcher Menge zwischen zwei Elementen fließen [Häu97]. Für t_1 ist in der Abbildung eine Transitions-Spezifikation angegeben, welche die Stoffströme für die jeweiligen Input- und Outputkanten festlegt. So wird zum Beispiel für die Herstellung von 100 kg Granulat als Output 413,48 kg Erdöl als Input benötigt.

Es ist zu beachten, dass sich Stoffstromnetze zwar an der Struktur von Petri-Netzen orientieren, es aber keine Schaltregel (siehe die Definition 2.3.2 der Schaltregel für elementare Petri-Netze in Unterabschnitt 2.3.2) für Stoffstromnetze gibt [Woh00]. Vielmehr stellen Stoffstromnetze eine grafische Möglichkeit zur Aufstellung von Gleichungssystemen dar [Hei02a]. Anhand dieser Gleichungssysteme können dann die Stoffströme eines Produktsystems ermittelt werden [Hei02a]. Die im Beispiel in Abbildung 3.8 genannten Mengen können also als Koeffizienten zur Beschreibung von linearen Zusammenhängen zwischen den Inputs und Outputs interpretiert werden [Sch97b]. Das ermöglicht es, anhand der Koeffizienten im Vor- und Nachbereich einer Transition, in einem Rechenschritt bisher unbekannte Stoffströme zu ermitteln [Häu97, Möl95]. Die „Rechenrichtung“ ist dabei beliebig und kann sowohl „vorwärts“ als auch „rückwärts“ entlang des mit Kanten modellierten Stoffstroms erfolgen [Sch97a, S. 22]. Das Lokalitätsprinzip der Petri-Netze, das heißt, dass ein Rechenvorgang nur anhand des Vor- und Nachbereichs einer Transition durchgeführt wird, bleibt also erhalten [Häu97, Möl97]. Es findet aber kein Schalten einer Transition im Sinne eines Erzeugens oder Verbrauchens von Marken statt.

4 Nachhaltiges Geschäftsprozessmanagement

Die Ausführungen in den vorangegangenen Kapiteln 2 und 3 zeigen, dass es einige Überschneidungen in den Konzepten des Geschäftsprozessmanagements und der Lebenszyklusanalyse gibt – zumal der Prozessbegriff in beiden Disziplinen eine zentrale Rolle spielt. Weiterhin wird in Kapitel 3 deutlich, dass mit der Lebenszyklusanalyse bereits eine umfangreiche Methode für die Analyse der Nachhaltigkeit von Unternehmen bereitsteht. Vor diesem Hintergrund ist zu klären, welchen Beitrag Methoden des Nachhaltigen Geschäftsprozessmanagements als Alternativen oder Ergänzungen zur Lebenszyklusanalyse leisten können. Dazu werden in diesem Kapitel zunächst Gemeinsamkeiten und Unterschiede der Begriffe und Konzepte beider Disziplinen gegenübergestellt (siehe Abschnitt 4.1). In Abschnitt 4.2 wird auf Basis der beschriebenen Gemeinsamkeiten und Unterschiede eine Definition für Nachhaltiges Geschäftsprozessmanagement erarbeitet. In diesem Zusammenhang wird diskutiert, wie sich Methoden des Nachhaltigen Geschäftsprozessmanagements und der Lebenszyklusanalyse gegenseitig ergänzen können. Insbesondere können bewährte Ansätze der Analyse von Nachhaltigkeitsauswirkungen von der Lebenszyklusanalyse auf das Geschäftsprozessmanagement übertragen werden. Dazu werden in Abschnitt 4.3 Nachhaltigkeitsmuster für Geschäftsprozesse definiert. Sie bilden die Grundlage für die Untersuchung verwandter Arbeiten in Kapitel 5 und die Definition von Anforderungen an eine Modellierungsmethode für Nachhaltiges Geschäftsprozessmanagement in Kapitel 6.

4.1 Geschäftsprozessmanagement und Lebenszyklusanalyse

Zur Vorbereitung der Diskussion des Beitrags, den Methoden des Nachhaltigen Geschäftsprozessmanagements als Alternative oder Ergänzung zur Lebenszyklusanalyse leisten können, werden in diesem Abschnitt zunächst Gemeinsamkeiten (Unterabschnitt 4.1.1) und Unterschiede (Unterabschnitt 4.1.2) von Begriffen und Konzepten aus beiden Disziplinen gegenübergestellt.

4.1.1 Gemeinsamkeiten

Tabelle 4.1 zeigt vergleichbare Begriffe aus dem Geschäftsprozessmanagement und der Lebenszyklusanalyse. Zu den Begriffen sind jeweils Beispiele angegeben. In der Tabelle sind die Begriffe in der Reihenfolge ihres Erscheinens in den folgenden Erläuterungen sortiert. Zur besseren Unterscheidbarkeit werden Begriffe, die explizit aus einer der beiden Disziplinen stammen, mit einem tiefergestellten $_{GPM}$ für Geschäftsprozessmanagement beziehungsweise $_{LCA}$ für Lebenszyklusanalyse gekennzeichnet.

Tabelle 4.1: Übersicht zu vergleichbaren Begriffen.

Geschäftsprozessmanagement		Lebenszyklusanalyse	
Begriff	**Beispiele**	**Begriff**	**Beispiele**
Geschäftsprozess	Fertigung eines Lufttrocknungssystems, Bearbeitung einer Bestellung	Lebenszyklus	Phasen entlang der Wertschöpfungskette eines Lufttrocknungssystems
Aktivität	Werkstück bearbeiten, Bestellung bestätigen, Rechnung verschicken	Prozessmodul	Werkstück bearbeiten, Energieerzeugung in Deutschland
Ziel	Produkt ist erstellt, Antrag ist bearbeitet, Dienstleistung ist erbracht	Funktionelle Einheit	Anzahl getrockneter Handpaare, Bereitstellung einer best. Menge Flüssigkeit
Datenfluss	Daten, Rohstoffe, Güter	Stoffstrom	Daten, Rohstoffe, Güter, CO2, Abwärme, Wasser
Ressource	Computer, Stanzmaschine	Kapitalgut	Computer, Stanzmaschine
Leistungsindikator	Kosten in Euro, Durchlaufzeit, Wartezeit	Wirkungsindikator	Klimawirkung in CO2e, Versauerungspot. in SO2e

Würde dasselbe reale Phänomen einmal aus Sicht der Lebenszyklusanalyse und einmal aus Sicht des Geschäftsprozessmanagements modelliert werden, wären in den resultierenden Modellen einige Parallelen identifizierbar. Als Beispiel für ein zu modellierendes Phänomen kann das in Unterabschnitt 3.3.1 erwähnte Lufttrocknungssystem herangezogen werden. Sowohl im Modell eines *Geschäftsprozesses*$_{GPM}$, das die Fertigung eines Lufttrocknungssystems abbildet, als auch im Modell des *Lebenszyklus*$_{LCA}$ eines Lufttrocknungssystems könnte zum Beispiel die *Aktivität*$_{GPM}$ beziehungsweise das *Prozessmodul*$_{LCA}$ *Werkstück bearbeiten* vorhanden sein. Auch die Bearbeitung einer Bestellung könnte sowohl als Geschäftsprozess$_{GPM}$, als auch als Teil eines Lebenszyklus$_{LCA}$ aufgefasst werden. Allerdings läge der Fokus im Geschäftsprozessmanagement auf den Aktivitäten innerhalb des betreffenden Unternehmens (mit Aktivitäten$_{GPM}$ wie *Bestellung bestätigen*, *Rechnung verschicken*, ...) [Wes19]. In der Lebenszyklusanalyse wären hingegeben die Stoffströme entlang der vor- und

nachgelagerten Wertschöpfungskette zu untersuchen [ISO06a]. Dazu zählt zum Beispiel die für die Durchführung einer Aktivität benötigte Energie. Zur Bestimmung der Nachhaltigkeitsauswirkungen der Energieerzeugung kann ein Prozessmodul$_{LCA}$ *Energieerzeugung in Deutschland* im Lebenszyklusmodell$_{LCA}$ ergänzt werden.

Die Begriffe *Ziel*$_{GPM}$ und *funktionelle Einheit*$_{LCA}$ weisen ebenfalls Gemeinsamkeiten auf. Im Geschäftsprozessmanagement ist mit dem Ziel$_{GPM}$ eines Geschäftsprozesses das angestrebte Ergebnis gemeint, das einen Wert für eine Kundin oder einen Kunden haben soll [Dum18]. Im Fall eines Fertigungsprozesses wäre zum Beispiel das Ziel$_{GPM}$, dass das Produkt (ein Lufttrocknungssystem) erstellt ist. Vergleichbar dazu könnte eine funktionelle Einheit$_{LCA}$ zum Beispiel als *Produktion von 1000 Lufttrocknungssystemen* quantifiziert werden. Allerdings wird in der Lebenszyklusanalyse bei der Festlegung einer funktionellen Einheit$_{LCA}$ üblicherweise weniger das konkrete Ergebnis eines Fertigungsprozesses betrachtet, sondern eher der spätere Nutzen. Im betrachteten Beispiel des Lufttrocknungssystems könnten zum Beispiel eine Anzahl getrockneter Handpaare als funktionelle Einheit$_{LCA}$ festgelegt werden [Klö09]. So wird ein systematischer Vergleich zwischen unterschiedlichen Produkten mit ähnlicher Funktion, zum Beispiel ein Vergleich zwischen Papierhandtüchern und Lufttrocknungssystemen, ermöglicht [Klö09, ISO06a]. Es lassen sich aber auch Beispiele finden, bei denen die Auffassungen eines Ziels$_{GPM}$ und einer funktionellen Einheit$_{LCA}$ beziehungsweise das Ergebnis und der Nutzen des Ergebnisses eines Prozesses näher beieinander liegen. Zum Beispiel ist ein Geschäftsprozess vorstellbar, der die Dienstleistung *Hände trocknen* erbringt. Hier entspräche das Ziel$_{GPM}$ direkt einer sinnvoll zu spezifizierenden funktionellen Einheit$_{LCA}$ für die betrachtete Dienstleistung.

Das Konzept eines *Datenflusses*$_{GPM}$ im Geschäftsprozessmanagement ist vergleichbar mit der Betrachtung von *Stoffströmen*$_{LCA}$ in der Lebenszyklusanalyse. In Unterabschnitt 2.2.2 wird dargelegt, dass jede Form von Austausch zwischen Aktivitäten, seien es *Daten, Rohstoffe, Güter* oder allgemein Objekte, als Datenfluss$_{GPM}$ modelliert werden kann [Dum18, Wes19]. Typische Stoffströme$_{LCA}$ wie zum Beispiel *CO2, Wasser oder Abwärme* [Klö09] ließen sich also auch als Datenfluss$_{GPM}$ in einem Geschäftsprozessmodell abbilden. Auch das Konzept eines Stoffstroms$_{LCA}$ fasst nicht nur materielle Güter, sondern auch Informationen und Daten [ISO06a], sodass beide Konzepte aufeinander übertragbar sind.

Der Begriff *Ressource*$_{GPM}$ lässt sich, wie in Unterabschnitt 2.2.1 diskutiert, nicht eindeutig von sonstigen Objekt-Inputs in einem Geschäftsprozess$_{GPM}$ trennen. Typischerweise sind mit diesem Begriff aber Objekte wie *Maschinen*, die für die Ausführung einer Aktivität$_{GPM}$ verantwortlich sind, beziehungsweise für die Ausführung

genutzt werden, gemeint [Fra03]. Aus Sicht der Lebenszyklusanalyse würde eine solche Maschine typischerweise als ein für die Herstellung eines Produkts benötigtes Kapitalgut$_{LCA}$ betrachtet werden [Klö09]. Aus Sicht der Lebenszyklusanalyse sind für Kapitalgüter, die an der Herstellung eines Produkts beteiligt sind, Stoffströme entlang ihrer Wertschöpfungskette zu ermitteln [Fri07, Kas14, Zen14]. Diese Stoffströme, beziehungsweise die mit ihnen zusammenhängenden Nachhaltigkeitsauswirkungen, können dann anteilig dem hergestellten Produkt angerechnet werden (siehe die Ausführungen zum Thema Allokation in Unterabschnitt 3.3.2).

Schließlich gibt es auch Überschneidungen zwischen *Leistungsindikatoren*$_{GPM}$ und *Wirkungsindikatoren*$_{LCA}$. Leistungsindikatoren$_{GPM}$ werden zwar typischerweise für Zeit, Kosten und Qualität definiert [Wes19, Dum18], können aber auch für diverse weitere Themenbereiche zum Einsatz kommen. Es ist also denkbar, zusätzlich zu den klassischen Leistungsindikatoren$_{GPM}$ weitere Indikatoren zu definieren, die auf Wirkungsindikatoren$_{LCA}$ wie *Klimawirkung in CO2e* oder *Versauerungspotential in SO2-Äquivalenten (SO2e)* beruhen. Aus Sicht der Lebenszyklusanalyse müssten dafür allerdings zuvor die relevanten Stoffströme$_{LCA}$ ermittelt werden.

4.1.2 Unterschiede

Im vorangegangenen Abschnitt wurde gezeigt, dass Begriffe aus den beiden betrachteten Disziplinen jeweils übertragbar sind. In der Gegenüberstellung wurde jedoch auch deutlich, dass es auch Unterschiede in der Handhabung der Begriffe und Konzepte gibt. Das deutet bereits darauf hin, dass sich die beiden Disziplinen in einigen Aspekten unterscheiden. Diese Unterschiede werden in diesem Abschnitt näher beleuchtet. Tabelle 4.2 gibt dazu einen Überblick zu den verschiedenen in diesem Abschnitt besprochenen Aspekten, in denen sich Geschäftsprozessmanagement und Lebenszyklusanalyse unterscheiden. In der Tabelle sind die verschiedenen Aspekte in der Reihenfolge ihres Erscheinens in den folgenden Ausführungen sortiert. Für eine bessere Unterscheidbarkeit werden, wie im vorangegangenen Abschnitt, Begriffe, die explizit auf ein Konzept aus einer der Disziplinen verweisen, mit einem tiefergestellten $_{GPM}$ beziehungsweise $_{LCA}$ gekennzeichnet.

Bezüglich der *Anwendungsdomäne* werden im Geschäftsprozessmanagement üblicherweise Dienstleistungen betrachtet [Dum18, Fra03] und in der Lebenszyklusanalyse üblicherweise Produktionsprozesse [Wol19, UNE15]. In beiden Fällen ist das aber mehr

eine empirische Beobachtung und keine in der Theorie der jeweiligen Disziplin begründete Einschränkung auf eine bestimmte Domäne [Dum18, Wol19]. Der Vergleich zeigt jedoch auch, dass beide Disziplinen unterschiedliche Zielsetzungen haben.

Der *Zweck der Modellierung* in der Lebenszyklusanalyse ist die möglichst vollständige Abschätzung der von einem Produkt oder einem Unternehmen verursachten ökologischen und sozialen Auswirkungen [ISO06a, UNE20]. Daraus folgt auch der Anspruch, ein Produktsystem$_{LCA}$ so zu modellieren, dass Stoffströme$_{LCA}$ bis hin zur natürlichen Umwelt zurückverfolgt werden [ISO06a]. Dahingegen dienen im Geschäftsprozessmanagement Prozessmodelle üblicherweise etwa als Anleitung für die Gestaltung realer Abläufe in einer Organisation, als Grundlage für Wirtschaftlichkeitsanalysen oder für die Umsetzung und Unterstützung der Prozessausführung in Informationssystemen [Obe96a, Fra03]. Dementsprechend liegen auch den bei der Modellierung zu treffenden Entscheidungen, zum Beispiel welche Details eines realen Phänomens im Modell abzubilden sind, jeweils andere Erwägungen zugrunde. Modellierungsentscheidungen in der Lebenszyklusanalyse basieren idealerweise auf naturwissenschaftlichen Erkenntnissen [ISO06a]. Im Geschäftsprozessmanagement hingegen werden für solche Entscheidungen üblicherweise betriebswirtschaftliche Erwägungen herangezogen

Tabelle 4.2: Übersicht zu Unterschieden.

Aspekt	Geschäftsprozessmanagement	Lebenszyklusanalyse
Domäne	Dienstleistungen öfter betrachtet als Produktionsprozesse.	Produktionsprozesse öfter betrachtet als Dienstleistungen.
Modellierungszweck	Anleitung, Wirtschaftlichkeitsanalysen, Umsetzung in Informationssystemen, ...	Möglichst vollständige Abschätzung von ökologischen und sozialen Auswirkungen.
Prozessfluss	Fokus auf Kontrollfluss, materielle Flüsse seltener betrachtet.	Modellierung von materiellen Flüssen (Stoffströmen).
Systemgrenze	Prozesse in einem Unternehmen, Schnittstellen zu anderen Unternehmen.	Nachverfolgung von Flüssen bis zur Grenze zwischen Technosphäre und Umwelt.
Granularität	Granularitätsstufen von Geschäftsfunktionen eines Unternehmens bis zu Aktivitäten.	Ein Prozessmodul kann für eine Aktivität stehen, aber auch für einen Industriezweig.
Abstraktion	Unterscheidung zwischen Prozesstyp und Prozessinstanz.	Normierung von Stoffstromdaten.
Leistungsmessung	Definition von Leistungsindikatoren in Bezug auf Zeit, Kosten und Qualität. Gewichtung und Aggregation üblich.	Ableitung von Wirkungsindikatoren aus Stoffströmen. Gewichtung und Aggregation mit Einschränkungen.
Zielgruppe	Für interne Entscheidungsfindung.	Auch für externe Kommunikation.

[Sch12]. Diese unterschiedlichen Zielsetzungen führen auch dazu, dass im Geschäftsprozessmanagement bei der Betrachtung des *Prozessflusses* mehr auf kausale Beziehungen von Aktivitäten (Kontrollflüsse) fokussiert wird, während in der Lebenszyklusanalyse der Fokus auf materiellen Flüssen, beziehungsweise Stoffströmen$_{LCA}$, liegt.

Im Geschäftsprozessmanagement werden meist Prozesse innerhalb eines Unternehmens betrachtet [Wes19]. Die Betrachtung von Schnittstellen zwischen Prozessen unterschiedlicher Unternehmen ist möglich [Wes19], aber üblicherweise entsprechen die Grenzen eines Unternehmens auch der *Systemgrenze* eines Modells. Die Lebenszyklusanalyse hat hingegen den Anspruch, Prozessflüsse bis zur Grenze zwischen Technosphäre und Umwelt nachzuverfolgen [ISO06a]. Die unterschiedlichen Systemgrenzen beeinflussen auch die *Granularität* der Modellierung. So wird im Geschäftsprozessmanagement zwischen den Granularitätsstufen Geschäftsfunktionen$_{GPM}$ eines Unternehmens, Geschäftsprozessen$_{GPM}$ (zum Beispiel Marketing, Vertrieb, Produktion, ...), Unterprozessen$_{GPM}$ und Aktivitäten$_{GPM}$ unterschieden [Sch12, Dum18, Wes19]. In der Lebenszyklusanalyse kann ein Prozessmodul$_{LCA}$ auch äquivalent zu einer Aktivität$_{GPM}$ im Geschäftsprozessmanagement spezifiziert werden (zum Beispiel das Bearbeiten eines Werkstücks). Sie kann aber auch, je nach Datenverfügbarkeit und Anforderungen der Studie, für einen Industriezweig (zum Beispiel Energieerzeugung in Deutschland) stehen [Klö09].

Die beiden Disziplinen unterscheiden sich auch in den vorgenommenen *Abstraktionen* bei der Modellierung von Prozessen. Im Geschäftsprozessmanagement wird zwischen Prozesstypen$_{GPM}$ und Prozessinstanzen$_{GPM}$ unterschieden [Wes19, Obe96a]. Auf Typebene werden alternative Ausführungsmöglichkeiten eines Prozesses gemeinsam betrachtet, während eine Prozessinstanz$_{GPM}$ eine konkreten Ausführung realisiert [Sch12]. Diese Unterscheidung findet in der Lebenszyklusanalyse nicht statt. Hier bildet eine quantifizierte funktionelle Einheit$_{LCA}$ eine Referenz für die Normierung der Stoffstromdaten aller modellierten Prozessmodule$_{LCA}$ [ISO06a]. Der Unterschied zwischen den in beiden Disziplinen bei der Modellierung vorgenommenen Abstraktionen kann anhand des in Unterabschnitt 4.1.1 beschriebenen Beispiels der Fertigung eines Lufttrocknungssystems erläutert werden. Im Geschäftsprozessmanagement kann in einem Geschäftsprozessmodell auf Typebene die Fertigung eines Lufttrocknungssystems modelliert werden. Eine Prozessinstanz$_{GPM}$ würde dann die Ausführung eines Geschäftsprozesses, beziehungsweise die Herstellung eines spezifischen Lufttrocknungssystems darstellen. In der Lebenszyklusanalyse würde ausgehend von einer definierten funktionellen Einheit$_{LCA}$, welche den Nutzen des Lufttrocknungssystems quantifiziert (zum Beispiel eine Anzahl getrockneter Handpaare), die Anzahl

der dafür benötigten Lufttrocknungssysteme erfasst werden. Davon ausgehend kann das Prozessmodell dann um weitere Prozessmodule, welche die Anzahl der benötigten Werkstücke oder die zur Herstellung benötigte Energie quantifizieren, ergänzt werden. Dabei können sowohl generische als auch spezifische Daten zum Einsatz kommen [Klö09] (siehe auch Unterabschnitt 3.3.1). Es ist vorstellbar, dass spezifische Daten für ein Prozessmodul$_{LCA}$ der Ausführung einer Aktivität$_{GPM}$, also einer Aktivitätsinstanz im Verständnis des Geschäftsprozessmanagements, entsprechen. Diese Daten würden dann aber anhand der quantifizierten funktionellen Einheit$_{LCA}$ skaliert werden. In diesem Sinn wird in der Lebenszyklusanalyse eher auf der Typ-Ebene modelliert, aber insgesamt ist die Unterscheidung zwischen Typen und Instanzen nicht unmittelbar auf die Lebenszyklusanalyse übertragbar.

Im Geschäftsprozessmanagement können für die *Leistungsmessung* eine Vielzahl von Leistungsindikatoren$_{GPM}$ für eine Aktivität$_{GPM}$ oder einen Geschäftsprozess$_{GPM}$ angegeben werden [Wes19, Dum18]. Üblicherweise werden Leistungsindikatoren für Aspekte wie Zeit, Kosten und Qualität aufgestellt [Wes19, Dum18]. Wie im vorangegangenen Abschnitt bereits erwähnt, ist es denkbar, Wirkungsindikatoren$_{LCA}$ aus der Lebenszyklusanalyse als ergänzende Leistungsindikatoren$_{GPM}$ zur Messung von ökologischen und sozialen Auswirkungen eines Geschäftsprozesses (zum Beispiel Wasserverbrauch, Treibhausgasemissionen oder Arbeitsbedingungen) heranzuziehen. Zu beachten ist dabei jedoch, dass dabei auch die in der Lebenszyklusanalyse aufgestellten Regeln zur Ableitung der Indikatoren aus den modellierten Stoffströmen beachtet werden müssen [Klö09]. In der Lebenszyklusanalyse wird auf Probleme bei der Aggregation und Gewichtung von Indikatoren hingewiesen. Diese sollten bei (für die Öffentlichkeit bestimmten) vergleichenden Aussagen vermieden werden, um Subjektivität und Informationsverlust zu vermeiden [ISO06a]. In den betrachteten Quellen zum Geschäftsprozessmanagement werden diese potentiellen Nachteile von Aggregationen und Gewichtungen nach Wissen des Autors nicht diskutiert.

Der unterschiedliche Umgang mit der Leistungsmessung von Prozessen hängt auch damit zusammen, dass das Geschäftsprozessmanagement als *Zielgruppe* im Vergleich zur Lebenszyklusanalyse weniger externe Anspruchsgruppen und mehr interne Entscheiderinnen und Entscheider adressiert [Wes19, Klö09].

4.2 Synthese und Definition

Bei der Reflexion der beschriebenen Gemeinsamkeiten und Unterschiede von Geschäftsprozessmanagement und Lebenszyklusanalyse ist zunächst festzuhalten, dass in beiden Disziplinen Prozesse modelliert und analysiert werden Auch in technischer Hinsicht sind die verfolgten Modellierungsansätze vergleichbar und miteinander verwandt (siehe die Ausführungen zu Petri-Netzen in Abschnitt 2.3 und Stoffstromnetzen in Abschnitt 3.4). Die bestehenden konzeptuellen und technischen Unterschiede hängen insbesondere mit den unterschiedlichen Zielsetzungen zusammen. Im Geschäftsprozessmanagement werden Prozessmodelle zum Beispiel als Anleitung für die Gestaltung realer Abläufe in einer Organisation, als Grundlage für Wirtschaftlichkeitsanalysen oder für die Umsetzung und Unterstützung der Prozessausführung in Informationssystemen eingesetzt [Obe96a, Fra03]. Der Zweck einer Lebenszyklusanalyse ist dahingegen die möglichst vollständige Abschätzung der von einem Produkt oder Unternehmen (oder auch allgemein von einem Prozess) verursachten ökologischen und sozialen Auswirkungen [ISO06a, UNE20].

Anhand der beschriebenen unterschiedlichen Zwecke kann jedoch auch auf mögliche Synergien geschlossen werden. Damit ist gemeint, dass sich die Methoden des Geschäftsprozessmanagements und der Lebenszyklusanalyse gegenseitig ergänzen können. Abbildung 4.1 illustriert mögliche Synergien anhand einer Gegenüberstellung der Phasen einer Lebenszyklusanalyse [ISO06a] mit den Phasen eines Geschäftsprozesslebenszyklus [Dum18]. Mit den grauen Schattierungen in der Abbildung wird dargestellt, welche Phasen jeweils ähnliche Ziele oder Ergebnisse haben. Eine Beobachtung ist zunächst, dass die Prozessmodellierung in beiden Disziplinen eine zentrale Aufgabe darstellt. In den Phasen *Ziel- und Umfangsdefinition* und *Inventaranalyse* einer Lebenszyklusanalyse werden, vergleichbar mit der *Modellierungs*-Phase im Geschäftsprozesslebenszyklus, Prozessmodelle erstellt und Daten zu Inputs und Outputs gesammelt [Klö09, Wes19]. Die im Geschäftsprozessmanagement durchgeführte *Analyse* der Leistung eines Geschäftsprozesses entspricht der *Wirkungsabschätzung* einer Lebenszyklusanalyse, in der die Nachhaltigkeitsauswirkungen eines Prozesses ermittelt werden [Dum18, Klö09]. Die letzte Phase einer Lebenszyklusanalyse, *Interpretation*, in der Prozessveränderungen vorgeschlagen werden, ist vergleichbar mit der Phase *Verbesserung* im Geschäftsprozesslebenszyklus [Klö09, Dum18]. Herausforderungen bei der *Umsetzung* der Prozessverbesserungen werden zwar in vorhandener Literatur zu Lebenszyklusanalysen diskutiert [Hei02b], in den zugrundeliegenden ISO-Standards wird die Umsetzung aber nicht explizit als Phase einer Lebenszyklusanalyse genannt.

Die Umsetzung von Prozessverbesserungen und damit auch die Steuerung von Prozessen in Unternehmen liegen demnach, anders als im Geschäftsprozessmanagement, nicht im Fokus der Lebenszyklusanalyse.

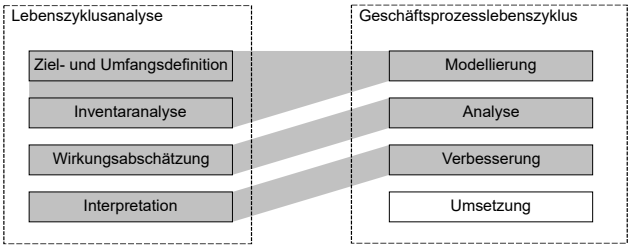


Abbildung 4.1: Phasen der Lebenszyklusanalyse [ISO06a] und des Geschäftsprozesslebenszyklus [Dum18].

Das mögliche Zusammenspiel von Lebenszyklusanalyse und Geschäftsprozessmanagement kann aus zwei Perspektiven betrachtet werden. Aus der ersten Perspektive lässt sich die Frage stellen, welchen Beitrag das Geschäftsprozessmanagement zu Fragestellungen der Lebenszyklusanalyse leisten kann. Die Lebenszyklusanalyse ist ein bewährter Ansatz, Nachhaltigkeitsprobleme zu strukturieren und zu verstehen. So können Unternehmen sensibilisiert und befähigt werden, diese Probleme zu lösen [Hei02b]. Jedoch sollten die notwendigen Veränderungen auch in den Unternehmen umgesetzt werden [Hei02b]. Dazu müssen die Nachhaltigkeitsdaten aus Lebenszyklusanalysen operationalisiert (in die steuernden Informationssysteme eines Unternehmens eingebunden) werden [UNE15]. Konventionelle Modellierungsansätze der Lebenszyklusanalyse eignen sich allerdings nur bedingt für die Darstellung einer Geschäfts- und Steuerungssicht auf Unternehmensaktivitäten [Woh06]. Die Konzepte und Methoden des Geschäftsprozessmanagements sind dagegen darauf ausgelegt, Prozessverbesserungen und die Steuerung von Prozessen in Unternehmen umzusetzen [Dum18]. Nachhaltiges Geschäftsprozessmanagement könnte daher die aus Nachhaltigkeitssicht notwendige Operationalisierung von Nachhaltigkeitsdaten unterstützen.

Die zweite Perspektive betrachtet das mögliche Zusammenspiel aus der entgegengesetzten Richtung. Dabei stellt sich die Frage, welchen Beitrag die Lebenszyklusanalyse zu Fragestellungen des Geschäftsprozessmanagements stellen kann. Unter dem Begriff *Nachhaltiges Geschäftsprozessmanagement* beginnen Forschende in ihren

Geschäftsprozessmanagement-Konzepten und -Methoden Nachhaltigkeitsaspekte miteinzubeziehen (siehe dazu auch Kapitel 5). Dieses Nachhaltige Geschäftsprozessmanagement benötigt belastbare und konsistente Nachhaltigkeitsdaten – und eine zentrale Stärke der Lebenszyklusanalyse ist die Möglichkeit, Nachhaltigkeitsauswirkungen zu quantifizieren [Klö09]. Die Übertragung dieser Quantifizierungsmöglichkeiten könnten somit die Grundlage für Nachhaltigkeitsindikatoren im Geschäftsprozessmanagement legen. Als Stärke der Lebenszyklusanalyse ist dabei zu nennen, dass sie auf internationalen Standards beruht, da diese Standardisierung der Gefahr der Beliebigkeit von Nachhaltigkeitsbehauptungen entgegenwirken kann [Fin14a]. Weiterhin unterstützt die Standardisierung die Vergleichbarkeit und die Qualität von Nachhaltigkeitsanalysen [Fin14a].

Nachhaltiges Geschäftsprozessmanagement kann auf Grundlage der vorangegangenen Ausführungen auch als Synthese der Ansätze von Lebenszyklusanalyse und Geschäftsprozessmanagement aufgefasst werden. Definition 4.2.1 adaptiert dazu die Geschäftsprozessmanagement-Definition nach [Wes19] und die Geschäftsprozesslebenszyklus-Definition nach [Dum18] und ergänzt sie um den Aspekt der ökologischen und sozialen Auswirkungen von Geschäftsprozessen.

Definition 4.2.1: ***Nachhaltiges Geschäftsprozessmanagement***

Nachhaltiges Geschäftsprozessmanagement umfasst Konzepte, Methoden und Werkzeuge zur Unterstützung der Modellierung, Analyse, Verbesserung, Umsetzung und Verwaltung von Geschäftsprozessen unter Einbeziehung ihrer ökologischen und sozialen Auswirkungen.

Die gegebene Definition 4.2.1 orientiert sich an der Definition für Geschäftsprozessmanagement nach [Wes19] und umfasst sowohl mögliche technische Ansätze (zur Modellierung, Analyse, Verbesserung und Umsetzung von Geschäftsprozessen) als auch Management-orientierte Ansätze (zur Verwaltung von Geschäftsprozessen).[1] Die Herausforderung für die Bereitstellung von Konzepten, Methoden und Werkzeugen für Nachhaltiges Geschäftsprozessmanagement besteht nun darin, die Zwecke des (konventionellen) Geschäftsprozessmanagements weiterhin zu unterstützen, und dabei zusätzlich eine integrierte Betrachtung von Nachhaltigkeitsaspekten zu ermöglichen. Damit könnte Nachhaltiges Geschäftsprozessmanagement einen wichtigen Beitrag im

[1] Zur Unterscheidung von technischen und Management-orientierten Ansätzen des Geschäftsprozessmanagement siehe Abschnitt 2.1.

Vergleich zu konventionellem Geschäftsprozessmanagement einerseits und ökologischen und sozialen Lebenszyklusanalysen andererseits leisten: Nachhaltigkeitsaspekte würden nicht mehr isoliert von den sonstigen Geschäftszielen eines Unternehmens betrachtet, sondern in einen unmittelbaren Zusammenhang mit der Steuerung des Unternehmens gebracht.

Nachhaltiges Geschäftsprozessmanagement kann nach Definition 4.2.1 auch als Geschäftsprozessmanagement verstanden werden, das von der Lebenszyklusanalyse die Bewertung der Nachhaltigkeit von Prozessen lernt. Bewährte Konzepte können übernommen, oder auch bestehende Daten aus Lebenszyklusdatenbanken für die Bewertung der Nachhaltigkeit eines Geschäftsprozesses herangezogen werden. Grundlegend ist dabei die Anwendung des Lebenszyklusgedankens: ein Geschäftsprozess ist nicht nur für seine unmittelbaren Nachhaltigkeitsauswirkungen verantwortlich, sondern zum Beispiel auch für die Nachhaltigkeitsauswirkungen, die mit der Produktion der von ihm genutzten Ressourcen verursacht werden.

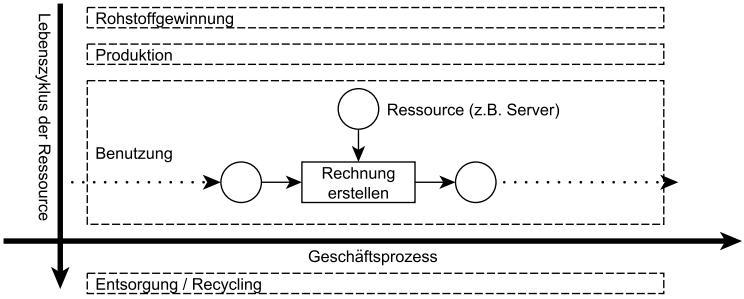


Abbildung 4.2: Zusammenhang von Geschäftsprozess und Lebenszyklus.

Abbildung 4.2 verdeutlicht diesen Zusammenhang beispielhaft. Der Geschäftsprozess verläuft dabei orthogonal zum Lebenszyklus der genutzten Ressourcen [Poh19]. Auf der einen Seite liegt der Geschäftsprozess in der Benutzungs-Phase des Lebenszyklus der betrachteten Ressource. Und auf der anderen Seite wird eine Ressource in einem Geschäftsprozess genutzt und demnach wären die durch diese Ressource verursachten Nachhaltigkeitsauswirkungen dem Geschäftsprozess anzurechnen Nachhaltiges Geschäftsprozessmanagement sollte also den Anspruch haben, auch den Lebenszyklus von genutzten Ressourcen zu betrachten. Dieser Anspruch folgt aus einer zentralen

Anforderung Nachhaltiger Entwicklung, dass (negative) ökologische oder soziale Auswirkungen nicht „verschoben“[1] werden, sei es hin zu zukünftigen Generationen, zwischen Unternehmen oder sonstigen Anspruchsgruppen [Uni87, Klö09, UNE20]. Dieser Aspekt stellt eine besondere Herausforderung für die Entwicklung von Ansätzen für Nachhaltiges Geschäftsprozessmanagement dar. Im konventionellen Geschäftsprozessmanagement werden oft nur unternehmensinterne Zusammenhänge untersucht [Wes19]. Wenn diese Perspektive unbedacht auf Nachhaltigkeitsanalysen angewandt wird, besteht die Gefahr, dass mögliche Verschiebungen von Nachhaltigkeitsauswirkungen nicht beachtet werden.

Das Problem der Verschiebung von Nachhaltigkeitsauswirkungen kann anhand eines Beispiels basierend auf [Klö09] verdeutlicht werden. Angenommen, es werden die Geschäftsprozesse eines Gastronomie-Unternehmens untersucht, das unter anderem Kartoffelsalat erstellt. Die Kartoffeln werden dazu gewaschen, geschält und schließlich geschnitten. Nach einer Umstrukturierung der Geschäftsprozesse werden nun allerdings von einer neuen Lieferfirma bereits geschälte und gewaschene Kartoffeln bezogen. Aus finanzieller Sicht wären die damit einhergehenden Zeitersparnisse als Verbesserung für das Unternehmen zu betrachten. Zusätzlich kann beobachtet werden, dass der Salatherstellungsprozess des Unternehmens mit dieser Umstrukturierung weniger Wasser verbraucht. Dies wäre aber keine Verbesserung im Sinne der Nachhaltigkeit, da der Wasserverbrauch lediglich entlang der Lieferkette verschoben wurde – zu dem Unternehmen, das die Kartoffeln zuvor wäscht. In ähnlicher Weise müssen auch Wechselwirkungen zwischen verschiedenen Nachhaltigkeitsaspekten betrachtet werden. Vielleicht kann das Unternehmen, das die gewaschenen Kartoffeln liefert, die Kartoffeln effizienter waschen, sodass auch bei einer erweiterten Betrachtung der Gesamtwasserverbrauch durch die Umstrukturierung sinkt. Möglicherweise werden die Kartoffeln nun jedoch in Plastik eingewickelt, um sie während des Transports sauber zu halten, was zu mehr Müll führt. So würde die Verbesserung in einem Nachhaltigkeitsaspekt durch eine Verschlechterung in einem anderen Nachhaltigkeitsaspekt „erkauft“ werden. Überlegungen dieser Art sind ein wichtiger Bestandteil der Methoden und Ansätze der Lebenszyklusanalyse und sollten auch bei einer Integration von Nachhaltigkeitsaspekten in das Geschäftsprozessmanagement beachtet werden.

[1] Im Englischen gebräuchlich ist hierfür der Begriff „burden shifting“ [Klö14].

4.3 Nachhaltigkeitsmuster für Geschäftsprozesse

Auf Basis der Ausführungen in diesem Kapitel lassen sich die folgenden Nachhaltigkeitsmuster für Geschäftsprozesse formulieren. Sie beschreiben unterschiedliche nachhaltigkeitsrelevante Aspekte von Geschäftsprozessen und adaptieren dazu Konzepte der Lebenszyklusanalyse für das Geschäftsprozessmanagment. Ihre Umsetzung in Ansätzen für Nachhaltiges Geschäftsprozessmanagement ermöglicht es, in der Lebenszyklusanalyse bewährte Konzepte der Analyse von Nachhaltigkeitsauswirkungen auf das Geschäftsprozessmanagement zu übertragen.

Für die vier Nachhaltigkeitsmuster *NMG 1 Zuordnung von nachhaltigkeitsrelevanten Inputs und Outputs*, *NMG 2 Zuordnung von Nachhaltigkeitsauswirkungen*, *NMG 3 Spezifikation von Nachhaltigkeitsauswirkungen* und *NMG 4 Allokation von Nachhaltigkeitsauswirkungen* werden jeweils eine Beschreibung, eine Herleitung und gegebenenfalls Varianten beschrieben. Die Formulierung und Beschreibung der Nachhaltigkeitsmuster orientiert sich an Geschäftsprozessmustern, die in der Literatur für verschiedene zu modellierende Geschäftsprozessphänomene (zum Beispiel Kontrollflüsse [Rus06], Datenflüsse [Rus04a] und Ressourcen [Rus04b]) formuliert wurden. Sie sind allgemein formuliert, sodass sie auf unterschiedliche Modellierungsansätze des Geschäftsprozessmanagements, unabhängig von spezifischen Prozessmodellierungssprachen oder -werkzeugen angewandt werden können.

(NMG 1) Zuordnung von nachhaltigkeitsrelevanten Inputs und Outputs: Einer Prozesskomponente[1] können nachhaltigkeitsrelevante Inputs und Outputs zugeordnet werden. Zum Beispiel benötigt eine Aktivität bei ihrer Ausführung eine bestimmte Menge Energie als Input oder produziert eine bestimmte Menge Müll als Output.

Herleitung: Die Erfassung von Inputs und Outputs von Prozesskomponenten entspricht der Inventaranalyse einer Lebenszyklusanalyse (siehe Unterabschnitt 3.3.2). Nur bei einer möglichst vollständigen Erfassung der (materiellen) Inputs und Outputs eines Geschäftsprozesses lässt sich die Signifikanz seiner Nachhaltigkeitsauswirkungen ermitteln.

(NMG 2) Zuordnung von Nachhaltigkeitsauswirkungen: Einer Prozesskomponente können Nachhaltigkeitsauswirkungen zugeordnet werden. Dabei können unterschiedliche ökologische und soziale Auswirkungen unterschieden werden (siehe

[1] Eine Prozesskomponente kann beispielsweise eine Aktivität oder eine Ressource auf Typ oder Instanzebene meinen [Rus06, Rus04a, Rus04b].

die folgenden Beschreibungen der Varianten von NMG 2). Die Nachhaltigkeitsauswirkungen können von den einer Prozesskomponente zugeordneten Inputs und Outputs (NMG 1) abgeleitet werden.

Herleitung: Die Zuordnung von Nachhaltigkeitsauswirkungen zu einer Prozesskomponente entspricht der Wirkungsabschätzung einer Lebenszyklusanalyse (siehe Unterabschnitt 3.3.3). Die mit den Inputs und Outputs einer Prozesskomponente verbundenen Nachhaltigkeitsauswirkungen sollten möglichst umfassend ermittelt werden. Nur bei einer möglichst vollständigen Betrachtung können potentielle Lastenverschiebungen zwischen verschiedenen Nachhaltigkeitsauswirkungen (Wirkungskategorien) erkannt werden.

Variante NMG 2-Öko: Einer Prozesskomponente können ökologische Auswirkungen zugeordnet werden. Zum Beispiel verursacht die Ausführung einer Aktivität eine Klimawirkung in Höhe von 5 kg CO2e oder trägt zum stratosphärischen Ozonabbau mit 3 kg CFC-Äquivalente (CFCe) bei.

Variante NMG 2-Klima: Einer Prozesskomponente kann eine Klimawirkung zugeordnet werden. Zum Beispiel verursacht die Ausführung einer Aktivität eine Klimawirkung in Höhe von 5 kg CO2e. *NMG 2-Klima* wird als Spezialfall von *NMG 2-Öko* definiert, da es auch im Kontext der Nachhaltigkeitsanalyse von besonderer Relevanz ist und spezielle Analysestandards wie das Treibhausgasprotokoll [WBC04] dafür existieren (siehe Abschnitt 3.2).

Variante NMG 2-Sozial: Einer Prozesskomponente können soziale Auswirkungen zugeordnet werden. Zum Beispiel verursacht die Ausführung einer Aktivität durchschnittlich eine bestimmte Anzahl Unfälle.

(NMG 3) Spezifikation von Nachhaltigkeitsauswirkungen: Einer Prozesskomponente können Nachhaltigkeitsauswirkungen für unterschiedliche Systemgrenzen zugeordnet werden. Diese Systemgrenzendefinition kann für unterschiedliche betrachtete Nachhaltigkeitsauswirkungen erfolgen (siehe folgende Varianten von NMG 3).

Herleitung: Entsprechend dem der Lebenszyklusanalyse zugrundeliegenden Lebenszyklusgedanken sollten Ansätze für Nachhaltiges Geschäftsprozessmanagement es ermöglichen, potentielle Verschiebungen von Nachhaltigkeitsauswirkungen entlang von Wertschöpfungsketten zu beschreiben (siehe Abschnitt 3.2). Da dies jedoch auch in der Lebenszyklusanalyse, in Abhängigkeit etwa von der jeweiligen Datenverfügbarkeit, nicht immer vollständig umgesetzt werden kann [Klö09], sollte eine transparente Definition der betrachteten Systemgrenzen unterstützt werden.

Variante NMG 3-Öko: Einer Prozesskomponente können ökologische Auswirkungen für unterschiedliche Systemgrenzen zugeordnet werden. Zum Beisipiel trägt die Ausführung einer Aktivität unter Einbeziehung der direkten Emissionen mit 3 kg CFCe zum stratosphärischen Ozonabbau bei. Dieselbe Aktivitätsausführung trägt unter Einbeziehung der Emissionen, die in der vorgelagerten Wertschöpfungskette entstehen, 15 kg CFCe zum stratosphärischen Ozonabbau bei.

Variante NMG 3-Klima: Einer Prozesskomponente können Klimawirkungen für unterschiedliche Systemgrenzen zugeordnet werden. Die Definition der Systemgrenzen kann sich zum Beispiel an Scope 1-3 des Treibhausgasprotokolls [WBC04] orientieren (siehe auch Abschnitt 3.2). Zum Beispiel verursacht die Ausführung einer Aktivität eine Scope 1-Klimawirkung in Höhe von 5 kg CO2e. Dieselbe Aktivitätsausführung versursacht eine Scope 3-Klimawirkung in Höhe von 30 kg CO2e.

Variante NMG 3-Sozial: Einer Prozesskomponente können soziale Auswirkungen für unterschiedliche Systemgrenzen zugeordnet werden. Zum Beispiel verursachen Ausführungen einer Aktivität unter Einbeziehung der Unfälle, die innerhalb des betrachteten Unternehmens geschehen, durchschnittlich 0,00001 Unfälle. Unter Einbeziehung von Unfällen, die während der Produktion der genutzten Ressourcen geschehen, verursachen Ausführungen derselben Aktivität durchschnittlich 0,00002 Unfälle.

(NMG 4) Allokation von Nachhaltigkeitsauswirkungen: Nachhaltigkeitsauswirkungen einer Prozesskomponente können auf andere Prozesskomponenten (anteilig) aufgeteilt werden. Zum Beispiel können die Treibhausgasemissionen, die bei der Herstellung einer Ressource entstanden sind, anteilig einzelnen Aktivitätsausführungen, in denen die Ressource genutzt wird, angerechnet werden.

Herleitung: Herausforderungen bei der adäquaten Anrechnung von Stoffströmen beziehungsweise quantifizierten Nachhaltigkeitsauswirkungen bei Multi–Input- und Multi–Output–Prozessen werden in der Literatur zu Lebenszyklusanalysen umfassend behandelt (siehe Unterabschnitt 3.3.2). Ähnliche Herausforderungen sind im Nachhaltigen Geschäftsprozessmanagement zu lösen, wenn zum Beispiel eine Prozessinstanz mehrere Outputs hat, oder eine Ressource von mehreren Prozessinstanzen genutzt wird.

Die hier formulierten Nachhaltigkeitsmuster für Geschäftsprozesse werden im weiteren Verlauf dieser Arbeit für die Untersuchung existierender Ansätze für Nachhaltiges Geschäftsprozessmanagement in Kapitel 5 herangezogen. Sie gehen auch in die Ermittlung von Anforderungen an die in dieser Arbeit entworfene Modellierungsmethode für Nachhaltiges Geschäftsprozessmanagement in Kapitel 6 ein.

5 Verwandte Arbeiten

In diesem Kapitel werden der Stand der Forschung und aktuelle Forschungsbedarfe im Bereich Nachhaltiges Geschäftsprozessmanagement beschrieben. Dazu wird in Abschnitt 5.1 das Vorgehen zur Identifikation verwandter Arbeiten basierend auf der tertiären Literaturstudie [Fri22] vorgestellt. In Abschnitt 5.2 erfolgt zunächst eine Vorstellung des Stands der Forschung im Bereich Nachhaltiges Geschäftsprozessmanagement unter Einbeziehung sowohl technischer als auch Management-orientierter Ansätze.[1] Daraufhin werden zur in dieser Arbeit beschriebenen Modellierungsmethode vergleichbare Ansätze untersucht. Die in Abschnitt 4.3 erarbeiteten Nachhaltigkeitsmuster für Geschäftsprozesse dienen dabei als Bewertungskriterien. Abschließend werden in Abschnitt 5.3 auf Basis der untersuchten Arbeiten identifizierte Forschungsbedarfe vorgestellt.

5.1 Vorgehen zur Identifikation verwandter Arbeiten

Wenn in einem Forschungsfeld bereits eine Reihe von systematischen Literaturrecherchen vorliegt, kann eine tertiäre Studie durchgeführt werden, um den Stand der Forschung zu ermitteln [Kit07]. Zur Identifikation verwandter Arbeiten wurden daher in der tertiären Studie [Fri22] existierende Literaturrecherchen zum Thema Nachhaltiges Geschäftsprozessmanagement untersucht. In der vorliegenden Arbeit werden die in [Fri22] ermittelten Ergebnisse in gekürzter Form wiedergegeben und um eine Untersuchung von in Primärarbeiten beschriebenen Modellierungsansätzen ergänzt. Das Vorgehen bei der Suche und Auswahl von Literatur sowie bei der Extraktion von Ergebnissen und Primärarbeiten wird im Folgenden beschrieben.

[1] Zur Unterscheidung zwischen technischen und Management-orientierten Ansätzen des Geschäftsprozessmanagements siehe Abschnitt 2.1.

5.1.1 Literatursuche

Zur Beschreibung des Stands der Forschung (Forschungsfrage F1) und der Forschungsbedarfe (Forschungsfrage F2) im Bereich Nachhaltiges Geschäftsprozessmanagement sollen die Ergebnisse bestehender Literaturrecherchen konsolidiert werden.

F1 Was ist der aktuelle Stand der Forschung im Nachhaltigen Geschäftsprozessmanagement?

F2 Welche Forschungsbedarfe wurden für Nachhaltiges Geschäftsprozessmanagement identifiziert?

Weiterhin sollen bestehende, zur in dieser Arbeit beschriebenen Modellierungsmethode verwandte Ansätze ermittelt und untersucht werden (Forschungsfrage F3).

F3 Welche Modellierungsansätze für Nachhaltiges Geschäftsprozessmanagement existieren?

Die Suche nach Literatur wurde im Januar 2021 bei den Datenbanken Google Scholar, ACM, IEEE Explore, Science Direct, Scopus, Springer Link und Web of Science durchgeführt. Der unten beschriebene Suchstring wurde zunächst auf die Titel der Arbeiten angewandt. Insofern eine der Datenbanken eine detailliertere Auswahl an Suchfeldern bot, wurde die Suche auf die Kurzzusammenfassung und die Schlüsselwörter ausgedehnt, um mehr potentiell relevante Arbeiten zu finden. Eine Suche im Volltext wurde nicht durchgeführt, da sie zu viele irrelevante Treffer ergeben hätte – im Fall von Google Scholar allein mehrere zehntausend. Der Suchstring wurde wie folgt zusammengesetzt:

- Der Begriff „business process“ sollte vorhanden sein, um Arbeiten aus dem Bereich Geschäftsprozessmanagement zu identifizieren. „BPM“ wurde als Abkürzung für die englische Bezeichnung *Business Process Management* ergänzt.
- Die Begriffe „sustainable“ oder „sustainability“ wurden vorausgesetzt, um Arbeiten zu Nachhaltigkeitsaspekten zu identifizieren. Als verbreitetes Synonym für die ökologische Nachhaltigkeitsdimension wurde der Begriff „green“ ergänzt.
- Die Begriffe „review“ oder „survey“ beziehungsweise die verbreitete Abkürzung „SLR“ für die englische Bezeichnung von systematischen Literaturrecherchen

(Systematic Literature Review) wurden vorausgesetzt, um systematische Literaturrecherchen zu identifizieren. Eine Variante von systematischen Literaturrecherchen sind sogenannte „mapping studies“ [Kit07], daher wurde alternativ auch der Begriff „mapping“ im Suchstring verwendet.

Insgesamt ergab sich dadurch der Suchstring *(green OR sustainable OR sustainability) AND (BPM OR „business process“) AND (review OR mapping OR survey OR SLR)*, der entsprechend auf die Datenbanken angewandt wurde.

5.1.2 Literaturauswahl

Wie in Abbildung 5.1 dargestellt, ergab die Suche in den Datenbanken 487 Treffer. Nach der Entfernung von Duplikaten blieben 414 Arbeiten übrig. Die folgenden Auswahlkriterien wurden definiert, um relevante systematische Literaturrecherchen zu identifizieren:

- Die Arbeit beschreibt eine systematische Literaturrecherche zum Thema Nachhaltiges Geschäftsprozessmanagement mit definierten Forschungsfragen, einer Beschreibung von Suchprozess und Datenextraktion sowie Ergebnispräsentation.
- Die Arbeit ist eine „mapping study“ [Kit07] (das heißt eine Variante einer systematischen Literaturrecherchen) zum Thema Nachhaltiges Geschäftsprozessmanagement.

Basierend auf den Forschungsfragen und um die Qualität der ausgewählten Arbeiten sicherzustellen, wurden folgende Ausschlusskriterien definiert:

- Die Arbeit behandelt nicht explizit Nachhaltige Entwicklung im Zusammenhang mit Geschäftsprozessmanagement.
- Die Arbeit hat keinen Peer-Review-Prozess durchlaufen.
- Die Arbeit ist nicht auf englisch verfasst.
- Die Arbeit ist nicht online verfügbar.
- Die Arbeit beschreibt keine systematische Literaturrecherche.

Nach Anwendung der Auswahl- und Ausschlusskriterien auf den Titel und die Kurzzusammenfassung blieben 30 Arbeiten übrig. Danach wurden die Kriterien auf den gesamten Text angewandt, sodass neun Arbeiten übrigblieben. Schließlich wurde eine

Rückwärts-Schneeballsuche [Woh14] durchgeführt und somit insgesamt elf Arbeiten aus den Jahren 2012 bis 2020 identifiziert (siehe Tabelle 5.1). Für die Schneeballsuche wurden Kandidaten aus den Referenzen der bisher schon ausgewählten Arbeiten extrahiert. Für jede potentiell auszuwählende Arbeit aus der Schneeballsuche wurde überprüft, ob zuvor schon eine Auswahl- oder Ausschlussentscheidung getroffen wurde. Falls bisher noch keine Entscheidung getroffen wurde, wurden die Auswahl- und Ausschlusskriterien angewandt. Um Verzerrungen bei der Auswahl vorzubeugen wurden Auswahlentscheidungen von zwei Forschenden unabhängig voneinander getroffen. Falls unterschiedliche Entscheidungen vorlagen, wurden diese diskutiert bis ein Konsens gefunden wurde.[1]

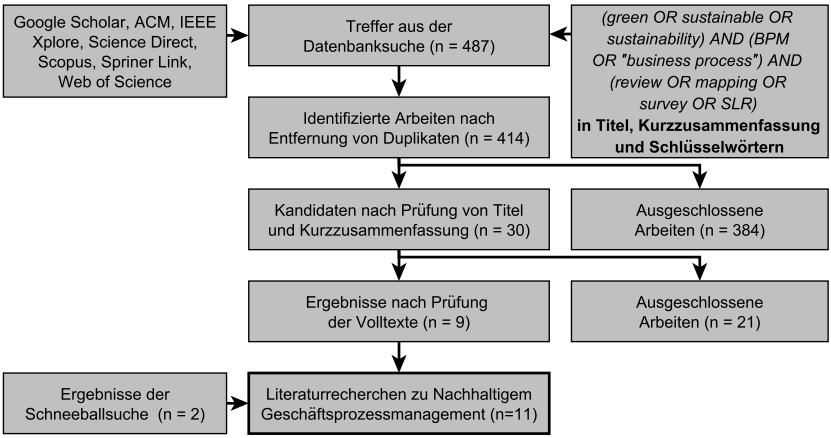


Abbildung 5.1: Auswahlprozess der tertiären Studie.

5.1.3 Extraktion der Ergebnisse und Primärarbeiten

Tabelle 5.1 listet die im Rahmen des Such- und Auswahlprozesses identifizierten Literaturrecherchen. Zu jeder Studie ist die Jahreszahl der Veröffentlichung, eine Referenz zur in der jeweiligen Arbeit verwendeten Methode für systematische Literaturrecherchen, die Referenz zur Arbeit und die jeweils in der betreffenden Literaturrecherche identifizierte Gesamtzahl an Primärarbeiten gegeben. Zur besseren Unterscheidbarkeit

[1] An der Suche, Auswahl und Extraktion von Ergebnissen aus den Literaturstudien waren neben dem Autor der vorliegenden Arbeit Koautorinnen und Koautoren der Veröffentlichung [Fri22] beteiligt.

der Literaturrecherchen von Primärarbeiten und sonstigen Referenzen wird im Folgenden auf die Literaturrecherchen unter Verwendung der in der Tabelle vergebenen ID verwiesen. Sechs der Literaturrecherchen orientierten sich bei ihrem Vorgehen an der in [Web02] beschriebenen Methode für systematische Literaturrecherchen. Daneben wurde einige ähnliche Vorgehensweisen [Bro09, Lev06, Gen14, Kit10, Pet08, Pet15, Boe14, Ban15] verfolgt. In der Literaturrecherche SGOA wurde ein systematisches Vorgehen mit Suchstring und Auswahlkriterien beschrieben, aber keine Referenz auf eine zugrundeliegende Methode angegeben. Für die Literaturrecherchen wurde eine Qualitätsbewertung durchgeführt und es wurden Ergebnisse zum Stand der Forschung (F1) und Forschungsbedarfen (F2) extrahiert. Die extrahierten Ergebnisse werden in Unterabschnitt 5.2.1 und Abschnitt 5.3 basierend auf [Fri22] zusammengefasst wiedergegeben. Zur Beantwortung von F3 wurden darüber hinaus die in den Sekundärstudien identifizierten Primärarbeiten extrahiert. Wie in Tabelle 5.1 gelistet, wurden in den Literaturrecherchen insgesamt 93 individuelle Primärarbeiten zum Thema Nachhaltiges Geschäftsprozessmanagement identifiziert. 26 der Primärarbeiten beschreiben zur vorliegenden Arbeit verwandte Modellierungsansätze und werden in Unterabschnitt 5.2.2 im Rahmen der Beantwortung von F3 näher untersucht. Die verbleibenden Arbeiten beschreiben beispielsweise Management-orientierte Ansätze oder reine Forschungsagenden ohne vorliegende Ergebnisse. Diese Arbeiten fließen in die aus den Literaturrecherchen extrahierten Aussagen zum Stand der Forschung und Forschungsbedarfen ein, werden aber nicht detailliert behandelt.

Tabelle 5.1: Übersicht zu identifizierten Literaturrecherchen.

ID	Jahr	Methode	Referenz	# Primärarbeiten
CST	2012	[Bro09, Web02]	[Sto12]	34
NOPA	2014	[Lev06, Web02]	[Opi14b]	26
NOPB	2014	[Lev06, Web02]	[Opi14a]	11
SGOA	2015	-	[Goh15]	36
JCM	2017	[Web02]	[Mac17]	42
TSCA	2017	[Bro09, Web02]	[Sch17]	48
AHG	2019	[Gen14]	[Her19]	56
DCOA	2019	[Kit10, Pet08, Pet15]	[Cou19b]	60
DCOB	2019	[Boe14]	[Cou19a]	60
TSCB	2019	[Web02]	[Sch19]	12
SGOB	2020	[Ban15]	[Goh20]	49
		Anzahl Primärarbeiten insgesamt (ohne Duplikate):		93
		Davon verwandte Modellierungsansätze:		26

5.2 Stand der Forschung

In diesem Abschnitt wird der Stand der Forschung im Bereich Nachhaltiges Geschäftsprozessmanagement allgemein (Beantwortung von F1 in Unterabschnitt 5.2.1) und spezifisch in Bezug auf existierende Modellierungsansätze (Beantwortung von F3 in Unterabschnitt 5.2.2) vorgestellt. Abschließend wird ein Überblick zu ergänzenden Ansätzen in Unterabschnitt 5.2.3 gegeben.

5.2.1 Ansätze für Nachhaltiges Geschäftsprozessmanagement

Seit 2007 werden Arbeiten zum Thema Nachhaltiges Geschäftsprozessmanagement veröffentlicht, ein Aufwärtstrend in der jährlichen Anzahl an Veröffentlichungen ist aber nicht zu beobachten (CST, SGOA, AHG, DCOA, DCOB). An Nachhaltigem Geschäftsprozessmanagement wurde bisher primär von wenigen Forschungsgruppen in Europa geforscht (CST, DCOB). Insgesamt ist das Forschungsfeld damit noch relativ jung, was sich auch darin zeigt, dass es bisher nur vereinzelte Veröffentlichungen in Journals gibt; die meisten Arbeiten wurden in Konferenzbänden und teilweise auch in Buchkapiteln veröffentlicht (CST, SGOA, AHG, DCOA, DCOB). Ein Großteil der veröffentlichten Arbeiten beschreibt technische Ansätze (siehe auch die in den nächsten beiden Unterabschnitten vorgestellten Arbeiten). In den Literaturrecherchen werden darüber hinaus noch einzelne eher Management-orientierte Ansätze identifiziert, wie zum Beispiel Rollenkonzepte für Nachhaltiges Geschäftsprozessmanagement, oder Ansätze für Trainingsprogramme zur Förderung des Nachhaltigkeitsbewusstseins der Belegschaft (NOPB, JCM und DCOB). Die Arbeiten mit technischem Fokus sind meist noch in einem konzeptionellen Stadium, da nur wenige angewandte Arbeiten und Implementierungen beschrieben werden (JCM, AHG).

In Bezug auf adressierte ökologische Aspekte wurden bisher meistens Treibhausgasemissionen und Energiebedarfe betrachtet. Diese Aspekte wurden in den Literaturrecherchen unterschiedlich bezeichnet: „CO2 Footprint“ und „Energy Consumption“ bei SGOA, „Reduce carbon footprint“ und „Reduce consumption“ bei TSCA, „Emissions“ und „Energy Efficiency“ bei AHG und „Carbon emissions“ und „Energy consumption“ bei DCOA. Daneben ergibt sich keine eindeutige Kategorisierung von ökologischen Aspekten aus den Ergebnissen der Sekundärstudien. Eine eindeutige Zuordnung verschiedener Begriffe für Nachhaltigkeitsaspekte in den Sekundärstudien war nicht möglich, da sich in den Studien keine konkreten Beschreibungen oder Erklärungen

finden. Dies kann auch darauf zurückzuführen sein, dass auch die Primärarbeiten inkonsistente und unpräzise Terminologie für Nachhaltigkeitsaspekte verwenden. Weiterhin wiesen die identifizierten Nachhaltigkeitsaspekte qualitative Unterschiede auf, wie etwa die Aspekte „Emissions“ und „Biodiversity“ bei AHG: aus Sicht der Lebenszyklusanalyse wäre ersteres ein Stoffstrom, der erst noch in einen Wirkungskategorie übersetzt werden müsste. Letzteres wäre eine Wirkungskategorie, für die zu klären wäre, welche Stoffströme sich potentiell negativ auf die Biodiversität auswirken [Klö09, Fin14b].

Die soziale und die ökonomische Nachhaltigkeitsdimension sind im Nachhaltigen Geschäftsprozessmanagement bisher unterrepräsentiert. DCOA identifiziert einige Arbeiten, die Nachhaltigkeit von einer breiteren Perspektive betrachten, aber keine, die konkret auf soziale oder ökonomische Aspekte eingehen. TSCA identifiziert einige wenige ökonomische und soziale Aspekte, die in den Primärarbeiten erwähnt werden. In der ökonomischen Dimension sind das konventionelle Leistungsindikatoren wie Kosten, Zeit oder Profit. In der sozialen Nachhaltigkeitsdimension unterscheidet TSCA zwischen Indikatoren, die das Personal adressieren (zum Beispiel „wages“ oder „health and safety“) und solchen, die eine Gesellschaft als ganzes adressieren (zum Beispiel „contribute to social equity“ und „track social effects“). Insgesamt haben nur zwei der Studien (TSCA und TSCB) soziale oder ökonomische Nachhaltigkeitsaspekte identifizieren können.

Aus den vorangegangenen Ausführungen folgt, dass Nachhaltiges Geschäftsprozessmanagement bisher auf technische Ansätze fokussiert ist. Bei diesen Ansätzen mit technischem Fokus liegen aber nur wenige Implementierungen und angewandte Arbeiten vor. Die überwiegende Anzahl an Primärarbeiten betrachtet in Bezug auf Nachhaltigkeitsaspekte bisher lediglich Treibhausgasemissionen und Energiebedarfe. Da in den Literaturrecherchen für Nachhaltigkeitsaspekte qualitativ unterschiedliche Begriffe verwendet werden (zum Beispiel „Carbon emissions“ und „avoid unnecessary information“ bei TSCA), zeigt sich, dass im Nachhaltigen Geschäftsprozessmanagement bisher ein einheitliches Nachhaltigkeitskonzept fehlt. Bestehende Konzepte der Nachhaltigkeitsanalyse, wie zum Beispiel die in Unterabschnitt 3.3.3 beschriebene Unterscheidung zwischen Stoffströmen und Wirkungsindikatoren in der Lebenszyklusanalyse, werden nicht verwendet.

5.2.2 Bestehende Modellierungsansätze

Für die Untersuchung bestehender Modellierungsansätze wurden aus den 93 identifizierten Primärarbeiten zunächst Arbeiten mit einem Modellierungsbezug extrahiert (insgesamt 26). Verschiedene in den Primärarbeiten beschriebene Ansätze wurden von diversen Gruppen von Autorinnen und Autoren über mehrere Arbeiten hinweg weiterentwickelt. Für die folgenden Beschreibungen und Bewertungen wurden die Primärarbeiten daher anhand des jeweils beschriebenen Ansatzes und den beteiligten Autorinnen und Autoren gruppiert. Die Bewertung richtet sich dabei jeweils nach dem aktuellsten Stand der Veröffentlichungen zu einem Ansatz. Wenn zum Beispiel frühere Arbeiten nur grundlegende Ideen und Prinzipien beschreiben, aber in späteren Arbeiten dann Konzepte und Beispiele für die Anwendung formuliert werden, wurde der Ansatz nach dem Stand in den späteren Arbeiten bewertet.

Die betrachteten Modellierungsansätze wurden anhand des Kriteriums *Reife* sowie der in Abschnitt 4.3 beschriebenen Nachhaltigkeitsmuster für Geschäftsprozesse untersucht. Die Kriterien und ihre Bewertung werden in der folgenden Liste erläutert.

Reife Mit der *Reife* eines Ansatzes ist der Detaillierungsgrad von beschriebenen Konzepten und die praktische Anwendbarkeit gemeint. Die Bewertungsstufen orientieren sich an den in [Man95] vorgeschlagenen Reifegraden für Technologien.

- \- Grundlegende Prinzipien und Ideen werden beschrieben.
- o Konzepte und Beispiele für die Anwendung werden formuliert.
- \+ Implementierung des Ansatzes in einem Softwarewerkzeug, das die Modellierung von Nachhaltigkeitsaspekten unterstützt.

NMG 1 In wie weit unterstützt der Ansatz die *Zuordnung von nachhaltigkeitsrelevanten Inputs und Outputs* zu Prozesskomponenten?

- \- Der Ansatz bietet keine Modellierungsunterstützung für NMG 1.
- \+ Der Ansatz bietet Modellierungsunterstützung für NMG 1.

NMG 2 In wie weit unterstützt der Ansatz die *Zuordnung von Nachhaltigkeitsauswirkungen* zu Prozesskomponenten? Dabei wird unterschieden, ob Klimawirkungen betrachtet werden (NMG 2-Klima), ökologische Auswirkungen inklusive Klimawirkungen (NMG 2-Öko) oder auch soziale Auswirkungen (NMG 2-Sozial).

- \- Der Ansatz bietet keine Modellierungsunterstützung für die Zuordnung von Nachhaltigkeitsauswirkungen.

o Der Ansatz bietet Modellierungsunterstützung für die Zuordnung von Nachhaltigkeitsauswirkungen. Er unterscheidet nicht explizit zwischen nachhaltigkeitsrelevanten Inputs/Outputs und Nachhaltigkeitsauswirkungen.

\+ Der Ansatz bietet Modellierungsunterstützung für die Zuordnung von Nachhaltigkeitsauswirkungen. Er unterscheidet explizit zwischen nachhaltigkeitsrelevanten Inputs/Outputs und Nachhaltigkeitsauswirkungen.

NMG 3 In wie weit unterstützt der Ansatz die *Spezifikation von Systemgrenzen* bei der Zuordnung von Nachhaltigkeitsauswirkungen zu Prozesskomponenten? Dabei wird unterschieden, ob Klimawirkungen betrachtet werden (NMG 3-Klima), ökologische Auswirkungen inklusive Klimawirkungen (NMG 3-Öko) oder auch soziale Auswirkungen (NMG 3-Sozial).

\- Die Definition von Systemgrenzen wird nicht betrachtet oder die Betrachtung wird explizit auf die Unternehmensgrenzen beschränkt.

o Der Ansatz bietet Modellierungsunterstützung für die Zuordnung von Nachhaltigkeitsauswirkungen für unterschiedliche Systemgrenzen. Er unterscheidet nicht explizit zwischen nachhaltigkeitsrelevanten Inputs/Outputs und Nachhaltigkeitsauswirkungen.

\+ Der Ansatz bietet Modellierungsunterstützung für die Zuordnung von Nachhaltigkeitsauswirkungen für unterschiedliche Systemgrenzen. Er unterscheidet explizit zwischen nachhaltigkeitsrelevanten Inputs/Outputs und Nachhaltigkeitsauswirkungen.

NMG 4 In wie weit unterstützt der Ansatz die *Allokation von Nachhaltigkeitsauswirkungen* zwischen unterschiedlichen Prozesskomponenten?

\- Der Ansatz stellt keine Modellierungsunterstützung für die anteilige Aufteilung von Nachhaltigkeitsauswirkungen bereit.

\+ Der Ansatz stellt Modellierungsunterstützung für die anteilige Aufteilung von Nachhaltigkeitsauswirkungen bereit.

Die Tabelle 5.2 zeigt einen Überblick über bestehende Modellierungsansätze für Nachhaltiges Geschäftsprozessmanagement, die jeweils verwendeten Modellierungssprachen und die Bewertung des Ansatzes nach den beschriebenen Kriterien. Die Ansätze sind jeweils nach dem am häufigsten auftauchenden Primärautor beziehungsweise der Primärautorin der zugeordneten Veröffentlichungen benannt.

Tabelle 5.2: Verwandte Modellierungsansätze.

Ansatz	Reife	NMG 1	NMG 2			NMG 3			NMG 4
			Kl.	Ök.	So.	Kl.	Ök.	So.	
Houy	-	+	-	-	-	-	-	-	-
Hoesch-Klohe	o	+	+	+	-	-	-	-	+
Recker	o	+	+	-	-	+	-	-	+
Wesumperuma	o	+	+	-	-	+	-	-	+
Zhu	o	+	-	-	-	-	-	-	-
Betz	-	-	o	o	o	o	o	o	-

Houy et al. beschreiben in den Arbeiten [Hou11, Hou12] die Idee, Aktivitäten in EPK-Modellen mit Nachhaltigkeitsindikatoren wie Ressourcenverbrauch oder Energiebedarf zu annotieren (siehe Abbildung 5.2). Auf Basis dieser Annotationen soll die Nachhaltigkeit eines Geschäftsprozesses bewertet werden können. Die Arbeiten beschreiben dabei ein erstes Konzept, bei dem tiefergehende Analysefähigkeiten noch fehlen (- für *Reife*). So findet beispielsweise bei der Nachhaltigkeitsbewertung keine Unterscheidung zwischen unterschiedlichen Prozessinstanzen statt. Es wird keine Modellierungsunterstützung für Allokation oder die Definition von Systemgrenzen bereitgestellt (- für *NMG 4* und *NMG 3*). Verschiedene nachhaltigkeitsrelevante Inputs und Outputs wie Wasserverbrauch, Energiebedarf und CO2-Emissionen werden als Möglichkeiten für eine Bewertung genannt (+ für *NMG 1*), aber es wird dabei nicht auf die Erfassung von Nachhaltigkeitsauswirkungen eingegangen (- für *NMG 2*).

Auch in den Arbeiten von *Hoesch-Klohe et al.* werden Aktivitäten, in diesem Fall in BPMN-Prozessmodellen, mit Nachhaltigkeitsindikatoren verknüpft. Über mehrere Veröffentlichungen hinweg [Gho10, Hoe10c, Hoe10a, Hoe10b, Hoe12a, Hoe12b] wird der Ansatz von der Beschreibung von anfänglichen Ideen und Prinzipien bis hin zur detaillierteren Beschreibung von Modellierungs- und Analysemöglichkeiten ausgearbeitet (o für *Reife*). Im Unterschied zu *Houy et al.* wird auch beschrieben, wie Nachhaltigkeitsindikatoren über verschiedene Prozessinstanzen hinweg akkumuliert werden können. In Bezug auf das Allokationsproblem wird in der Arbeit [Gho10] vorgeschlagen, einzelne Aktivitäten in BPMN-Geschäftsprozessmodellen mit der Menge an Treibhausgasemissionen zu annotieren, die bei der Ausführung dieser Aktivitäten anfallen. Dazu werden die in den Aktivitäten verwendeten Ressourcen (wie zum Beispiel Schweißgeräte oder Lastwägen) und ihre Energiebedarfe beziehungsweise Treibstoffverbräuche modelliert. Mit diesen Annotationen können dann die Gesamtemissionen für verschiedene Ausführungen eines Prozesses berechnet und Gestaltungsvarianten mit geringeren Verbräuchen vorgeschlagen und untersucht werden (siehe dazu die Beschreibung des Ansatzes mit Fragmentbibliotheken im

nächsten Abschnitt). In [Hoe10c] wird der Gedanke der Ressourcenmodellierung erweitert, indem nun Beziehungen zwischen verschiedenen Ressourcen in Netzen (gerichteten, azyklischen Graphen) modelliert werden (+ für *NMG 4*). Ein Beispiel ist die Ressource Drucker, die in einer Aktivität verwendet wird und selbst wieder die Ressourcen Papier und Elektrizität benötigt. Auf Basis dieser Netze können dann wieder die Energiebedarfe und Treibhausgasemissionen, die bei der Ausführung einer Aktivität anfallen, berechnet werden. Weiterhin wird in [Hoe10a] die Unterscheidung zwischen verschiedenen Beziehungs-Typen von Ressourcen (wie Spezialisierung oder Aggregation) und die Unterscheidung von Ressourcen auf Typ- oder Instanzebene eingeführt. Bei den betrachteten Indikatoren wird zwischen nachhaltigkeitsrelevanten Inputs und Outputs (wie Energiebedarfen) und Klimawirkungen (gemessen in CO2e) unterschieden (+ für *NMG 1* und *NMG 2-Klima*). In der Arbeit [Hoe10b] werden Beispiele für andere ökologische Auswirkungen und ihre Verknüpfung mit Ressourcenmodellen genannt (+ für *NMG 2-Öko*). Die Betrachtung von Klimawirkungen, die außerhalb der Unternehmensgrenzen liegen, wird (mit Ausnahme der mit eingekaufter Energie verbundenen Treibhausgasemissionen) explizit ausgeschlossen (- für *NMG 3-Klima*). Für andere Ökologische Auswirkungen werden unterschiedliche Systemgrenzen nicht thematisiert (- für *NMG 3-Öko*). Auch soziale Auswirkungen werden nicht betrachtet (- für *NMG 2-Sozial* und *NMG 3-Sozial*).

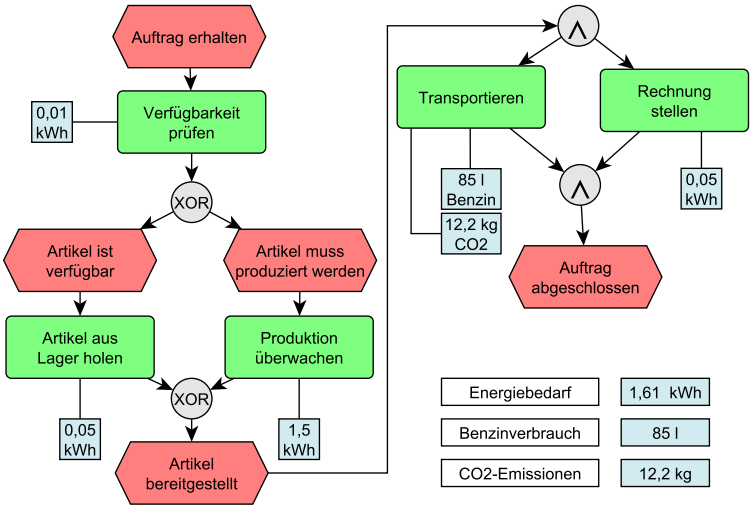


Abbildung 5.2: EPK-Prozessmodell mit Nachhaltigkeitsindikatoren, in Anlehnung an [Hou11].

In den Arbeiten von *Recker et al.* [Rec11, Rec12] wird eine BPMN-Erweiterung für Nachhaltigkeitsaspekte vorgeschlagen und ein Vorgehen zur Bewertung mit einem Beispiel ausgearbeitet (o für *Reife*). Mit diesen neuen Notationselementen können Aktivitäten zum Beispiel als papierverbrauchend oder treibstoffverbrauchend markiert werden (+ für *NMG 1*). Darüber hinaus werden die resultierende Klimawirkung für verschiedene Gruppen von Aktivitäten in einem speziellen Notationselement farblich kodiert dargestellt (+ für *NMG 2-Klima*). Zur Ermittlung der Emissionswerte werden Aktivitätsdaten (zum Beispiel Gewicht des Papiers oder Länge der zurückgelegten Distanz bei einer Geschäftsreise) und Emissionsfaktoren zusammengetragen. Zur Bewertung der Nachhaltigkeitsauswirkungen eines Geschäftsprozesses werden diese ermittelten Werte dann aufsummiert. Auf das Allokationsproblem wird eingegangen, indem ein mehrstufiges Verfahren vorgeschlagen wird, bei dem zunächst je Aktivität alle benötigten Ressourcen ermittelt und dann aufgeteilt werden (+ für *NMG 4*). Dies geschieht aber im Unterschied zum Ansatz von *Hoesch-Klohe et al.* nicht modellgestützt. Die zur Ermittlung der von den verwendeten Ressourcen verursachten Treibhausgasemissionen berücksichtigen auch die Herstellung dieser Ressourcen. Die Systemgrenzen der Analyse von Klimawirkungen gehen also über die Unternehmensgrenzen hinaus (+ für *NMG 3-Klima*). Konzepte für die Betrachtung weiterer Wirkungskategorien neben der Klimawirkung werden nicht erarbeitet (- für *NMG 2-Öko*, *NMG 2-Sozial*, *NMG 3-Öko* und *NMG 3-Sozial*).

Nah verwandt zum Ansatz von *Recker et al.* sind die Arbeiten von *Wesumperuma et al.* [Wes11, Wes13, Wes15], bei denen ebenfalls eine Erweiterung von BPMN um Notationselemente für die Erfassung von Klimawirkungen vorgeschlagen und beispielhaft ausgearbeitet wird (o für *Reife*). Es wird ein Vorgehen beschrieben, bei dem die Klimawirkung für einzelne Aktivitäten, Unterprozesse und Prozesse aufsummiert werden. Weiterhin wird auch ein Berechnungsverfahren vorgeschlagen, mit der die von Ressourcen verursachten Nachhaltigkeitsauswirkungen auf einzelne Aktivitäten umgeschlagen werden können (+ für *NMG 4* und *NMG 1*). Wie bei *Recker et al.* beschränkt sich der Ansatz auf die Ermittlung von Klimawirkungen (- für *NMG 2-Öko* und *NMG 2-Sozial*), berücksichtigt dabei aber auch Klimawirkungen entlang der Wertschöpfungskette (+ für *NMG 3-Klima*, aber - für *NMG 3-Öko* und *NMG 3-Sozial*).

Der Ansatz von *Zhu et al.* [Zhu15] zeigt, wie ein BPMN-Prozessmodell mit nachhaltigkeitsrelevanten Kontext-Daten (wie Ressourcen, Ländern, Personen) angereichert und dann in ein gefärbtes Petri-Netz übersetzt werden kann (o für *Reife* und + für *NMG 1*).

In Bezug auf Nachhaltigkeitsaspekte bleibt der Ansatz aber unspezifisch. Es wird keine konkrete Modellierungsunterstützung für die anderen Nachhaltigkeitsmuster beschrieben (- für *NMG 2-4*).

Bei *Betz* [Bet14] findet sich ebenfalls ein Ansatz, der mit Höheren Petri-Netzen arbeitet. Es wird vorgeschlagen, Nachhaltigkeitsaspekte in Geschäftsprozessen mithilfe von XML-Netzen abzubilden. Es wird erwähnt, dass verschiedene Indikatoren, die ökologische und soziale Auswirkungen messen, in ein Prozessmodell integriert werden können. Dabei wird zwischen direkten, indirekten und sozio-ökonomischen Auswirkungen unterschieden. Der Zusammenhang zwischen nachhaltigkeitsrelevanten Inputs/-Outputs und Nachhaltigkeitsauswirkungen wird dabei allerdings nicht betrachtet (- für *NMG 1*, o für *NMG 2* und o für *NMG 3*). Möglichkeiten der Allokation von Nachhaltigkeitsauswirkungen sowie weiterführende Untersuchungen oder Implementierungen des Vorschlags werden nicht beschrieben (- für *NMG 4* und *Reife*).

Eine Reihe von weiteren Ansätzen (*Cappiello et al.* [Per08, Ard08, Cap11a, Cap11b, Cap13, Cap14], *Nowak et al.* [Now13a, Now14b, Now11b] und *Reiter et al.* [Rei14, Lub15]) beschränken sich in ihren Ansätzen auf die Energieeffizienz der in Prozessen genutzten IT-Infrastruktur. Da sich die erarbeiteten Konzepte aber auf IT-Infrastruktur beschränken und nicht auf andere Domänen übertragbar sind, werden diese Ansätze hier nicht näher beschrieben.

In [Sta19] wurde untersucht, in wie weit bereits existierende konventionelle Prozessmodellierungswerkzeuge für Nachhaltigkeitsbetrachtungen angewendet oder adaptiert werden können. Die Arbeit kommt zu dem Ergebnis, dass in erweiterbaren Modellierungswerkzeugen, oder solchen, in denen individuelle Risiko- oder Ressourcenmodelle erstellt werden können, potentiell auch Nachhaltigkeitsaspekte abbildbar sind. Existierende Werkzeuge unterstützen Anwendende aber nicht dabei, soziale oder ökologische Analysen durchzuführen. [Sta19] fordert daher unter anderem die Bereitstellung von Nachhaltigkeitsindikatoren und entsprechenden Notationselementen in Modellierungssprachen und -werkzeugen. Darüber hinaus sollten nach [Sta19] Funktionalitäten für Vergleiche zwischen verschiedenen Gestaltungsalternativen von Prozessen und für die Simulation von Prozessmodellen unter Beachtung von Nachhaltigkeitsaspekten in bestehende Modellierungswerkzeuge integriert werden.

Die Untersuchung zeigt, dass keiner der vorhandenen Ansätze bisher soweit ausgearbeitet wurde, dass eine Werkzeugunterstützung vorhanden ist. In Bezug auf Analysefähigkeiten beschränken sich die meisten Ansätze auf die Bereitstellung von statischen Berechnungsverfahren ohne Simulationsfähigkeit. Zwei Ansätze (*Zhu et al.*

und *Betz*) erlauben grundsätzlich unter Ausnutzung der Eigenschaften Höherer Petri-Netzen eine Simulation der Modelle. In beiden Fällen wird jedoch nur eingeschränkte Modellierungsunterstützung für die in Abschnitt 4.3 beschriebenen Nachhaltigkeitsmuster bereitgestellt. In Bezug auf die Modellierungsunterstützung für die Allokation von Nachhaltigkeitsauswirkungen finden sich in drei Ansätzen (*Hoesch-Klohe et al.*, *Recker et al.* und *Wesumeruma et al.*) Konzepte, welche die Aufteilung von Klimawirkungen ermöglichen. In Bezug auf die Erfassung von nachhaltigkeitsrelevanten Inputs und Outputs sowie Nachhaltigkeitsauswirkungen beschränken sich die Arbeiten im Wesentlichen auf Energiebedarfe und Treibhausgasemissionen beziehungsweise Klimawirkungen. Eine Einbindung eines Indikatorkonzepts wie in der Lebenszyklusanalyse mit systematischer Unterscheidung zwischen Stoffströmen und Wirkungsindikatoren findet nicht statt. Lediglich zwei Ansätze (*Recker et al.* und *Wesumperuma et al.*) gehen in Anlehnung an das Treibhausgasprotokoll [WBC04] (siehe auch Abschnitt 3.2) davon aus, dass für die Prozessbewertung Daten zur Verfügung stehen, die auch während der Herstellung von genutzten Ressourcen anfallende Treibhausgasemission mit einbeziehen. Sie setzen für die Analyse damit implizit eine Systemgrenze, die über die Unternehmensgrenzen hinausgeht (Scope 3 des Treibhausgasprotokolls). Es werden aber keine darüber hinausgehenden Konzepte für eine transparente und konsistente Definition von Systemgrenzen auch für andere Nachhaltigkeitsaspekte bereitgestellt.

5.2.3 Ergänzende Ansätze

Einige weitere Arbeiten im Nachhaltigen Geschäftsprozessmanagement behandeln technische Teilprobleme und ergänzen die beschriebenen Modellierungsansätze. Sie werden im Folgenden für einen umfassenden Überblick beschrieben. Einen Schwerpunkt bilden dabei Ansätze zur Definition von Geschäftsprozessmustern. Im Unterschied zu den in dieser Arbeit in Abschnitt 4.3 beschriebenen Nachhaltigkeitsmustern für Geschäftsprozesse beschreiben diese keine grundlegenden Konzepte für die Modellierung der Nachhaltigkeitsauswirkungen von Geschäftsprozessen. Die Autorinnen und Autoren gehen stattdessen davon aus, dass eine Nachhaltigkeitsbewertung der Geschäftsprozesse stattgefunden hat und beschreiben in strukturierter Form, wie Verbesserungsmöglichkeiten erkannt und umgesetzt werden können. So untersucht [Lüb16] Geschäftsprozesse in der öffentlichen Verwaltung, um eine Liste von Schwachstellen-Mustern zu generieren. Ein Beispiel für ein identifiziertes Schwachstellen-Muster ist das Ausdrucken und postalische Versenden von Dokumenten, die auch ressourcensparsamer als Download angeboten werden könnten. In

[Lub16] wird der Ansatz um eine strukturiertere (textbasierte) Beschreibungsform erweitert und um einige weitere Muster (zum Beispiel Nutzung eines energiesparsamen Browsers) ergänzt.

Ein ähnlicher Ansatz wird in den Veröffentlichungen [Now11a, Now12, Now13c, Now13b, Now13d, Now14a] ausgearbeitet. Die hier beschriebenen Muster zeigen aber keine Schwachstellen auf, sondern geben Hinweise zu Möglichkeiten für Prozessverbesserungen. Es werden teilweise allgemeine Muster beschrieben, die für verschiedene Domänen anwendbar sind. Das Muster „Resource Change“ beschreibt zum Beispiel, dass ein Prozess verbessert werden kann, indem die verwendeten Ressourcen (etwa Strom aus erneuerbaren Energien anstelle von konventionellem Strom) ausgetauscht werden [Now11a]. Teilweise werden auch konkretere Muster mit einem Fokus auf den Energiebedarf der IT-Infrastruktur formuliert. Zum Beispiel beschreibt das Muster „Green Data Transfer Object“, dass die Anzahl von Anfragen (etwa an ein Datenbanksystem) gebündelt werden sollten [Now13b].

Bei [Sch17, Sch19] finden sich Vorschläge zu ähnlichen Mustern, welche die soziale Nachhaltigkeitsdimension adressieren. Parallel zum eben beschriebenen Muster „Resource Change“ nennt [Sch17] zum Beispiel „Social Resource Change“. Es beschreibt, dass bei der Ausführung von Aktivitäten Ressourcen (wie ergonomische Tische) zum Einsatz kommen, die sich positiv auf das menschliche Wohlergehen auswirken.

Eine Reihe weiterer Arbeiten adressieren spezifische technische Teilprobleme im Nachhaltigen Geschäftsprozessmanagement. So wird in [Hoe10a, Hou12, Hoe12a] ein Vorgehen zur Prozessverbesserung beschrieben. Als Grundlage dienen Prozessmodelle, die mit Nachhaltigkeitsinformationen wie im oben beschriebenen Ansatz von *Hoesch-Klohe et al.* annotiert sind. Für ein bestehendes Prozessmodell werden dann aus einer Bibliothek mit Prozessmodellfragmenten Gestaltungsalternativen generiert. Aus den Gestaltungsalternativen kann schließlich ein Prozessmodell mit einer vorteilhafteren Nachhaltigkeitsbewertung ausgewählt werden.

Die Arbeit [Hoe10b] adressiert das Problem der Aggregation von Nachhaltigkeitsindikatoren. So können für die Bewertung von Aktivitäten sowohl quantitative als auch qualitative Daten vorliegen, was das Vergleichen verschiedener Prozesse erschwert. Mithilfe einer in der Arbeit vorgestellten mathematischen Struktur wären Vergleiche und Aggregationen dennoch möglich.

[Grä13] beschreibt einen Ansatz zur Abbildung von Sprachelementen verschiedener Modellierungssprachen auf eine gemeinsame Ontologie. Damit wird der Vergleich von Nachhaltigkeitsauswirkungen verschiedener Geschäftsprozesse unterstützt.

Eine praxisorientierte Methode zur Datenerhebung, genannt Process Mapping, findet sich bei [Whi14]. Bei einem gegebenen Prozessmodell werden vor Ort, bei der Prozessausführung, für jede Aktivität strukturiert Daten erfasst. Die Methode zielt ursprünglich darauf ab, Prozesslaufzeiten zu dokumentieren und (Zeit-)Verschwendungen aufzudecken. Sie wird in der genannten Arbeit erweitert, um auch ökologische Indikatoren zu erheben.

5.3 Forschungsbedarfe

In Bezug auf Forschungsbedarfe fordern die Autorinnen und Autoren der untersuchten Literaturrecherchen mehr internationale, interdisziplinäre und praxisorientierte Zusammenarbeit (CST, DCOA, DCOB). DCOA hält diesbezüglich fest, dass Wissen über Nachhaltigkeitsthemen im Bereich der Lebenszyklusanalyse bereits vorhanden ist, und Forschende hiervon lernen können. Weiterhin sollten zukünftige Arbeiten mehr auf die Implementierung, Anwendung und Evaluation von Ansätzen fokussieren (CST, NOPA, SGOA, JCM, TSCA, AHG, DCOB). CST, JCM und SGOB sehen den konkreten Bedarf nach Implementierung von Modellierungswerkzeugen und CST, SGOA, SGOB, JCM und AHG fordern fundierte Rahmenkonzepte für Nachhaltiges Geschäftsprozessmanagement.

Von den Literaturrecherchen, die in den Primärarbeiten behandelte Nachhaltigkeitsaspekte detailliert untersucht haben, stellen DCOA, SGOA und SGOB fest, dass der Umfang der betrachteten ökologischen Auswirkungen erweitert werden sollten (SGOB bezieht diese Forderung auch auf soziale Auswirkungen). Alle drei Studien stellen fest, dass der Fokus bisher auf Klimawirkungen und Energiebedarfen lag, und andere Aspekte ökologischer Nachhaltigkeit nur in wenigen Fällen betrachtet wurde. Die drei Literaturrecherchen, die in ihrer Untersuchung auch die soziale Nachhaltigkeitsdimension miteinbezogen haben (DCOA, TSCA und TSCB) stellen ebenfalls fest, dass soziale Aspekte mehr betrachtet werden sollten.

Diese Feststellungen aus den Literaturrecherchen zeigen sich auch in den Ergebnissen aus der näheren Untersuchung von Modellierungsansätzen in Unterabschnitt 5.2.2. So existiert bisher noch kein Ansatz, der so weit ausgereift ist, dass er in einem Modellierungswerkzeug umgesetzt wurde. In Bezug auf behandelte Nachhaltigkeitsaspekte beschränken sich die Ansätze im Wesentlichen auf Klimawirkungen und Energiebedarfe und bieten nur eingeschränkt Konzepte zur Unterstützung der in Abschnitt 4.3 herausgearbeiteten Nachhaltigkeitsmuster für Geschäftsprozesse.

Vorstellung der Ergebnisse

6 Anforderungen an die Modellierungsmethode

In diesem Kapitel wird basierend auf den zuvor erarbeiteten Grundlagen die Zielsetzung für die in dieser Arbeit entworfene Modellierungsmethode für Nachhaltiges Geschäftsprozessmanagement (Abschnitt 6.1) formuliert. Aus der Zielsetzung werden in Abschnitt 6.2 Anforderungen an die entworfene Modellierungsmethode abgeleitet.

6.1 Zielsetzung für die Modellierungsmethode

Das Ziel der vorliegenden Arbeit ist die Bereitstellung einer Modellierungsmethode [Kar02] für Nachhaltiges Geschäftsprozessmanagement. Diese Modellierungsmethode soll Unternehmen bei der Verbesserung der Nachhaltigkeitsauswirkungen ihrer Geschäftsprozesse unterstützen. Sie stellt dazu grundlegende Konzepte des Geschäftsprozessmanagements bereit und ergänzt diese um Konzepte der Nachhaltigkeitsanalyse. So kann sie auch als Vorbild für die Erweiterung bestehender Modellierungsmethoden für (konventionelles) Geschäftsprozessmanagement (zum Beispiel Horus [Sch12]) um Nachhaltigkeitsaspekte dienen.

Nach [Kar02] besteht eine Modellierungsmethode aus drei Komponenten: einer Modellierungssprache, einem Vorgehensmodell und einem Modellierungswerkzeug. Die Modellierungssprache stellt Sprachelemente zur Erstellung von Modellen bereit. Das Vorgehensmodell beschreibt, wie die Modellierungssprache genutzt werden kann.[1] Ein Modellierungswerkzeug stellt schließlich Funktionen und unterstützende Mechanismen zur Anwendung der Modellierungssprache bereit. Im Rahmen der vorliegenden Arbeit wird ein *Vorgehensmodell für Nachhaltiges Geschäftsprozessmanagement* bereitgestellt, das allgemein und unabhängig von konkreten Modellierungssprachen oder -werkzeugen beschreibt, wie Unternehmen ihre Geschäftsprozesse und deren

[1] In [Kar02] wird lediglich ein Vorgehen (engl. „modeling procedure“) als Komponente einer Modellierungsmethode gesehen. Dieses Vorgehen soll Schritte zur Erstellung eines Modells beschreiben. In der vorliegenden Arbeit wird diese Komponente umfassender als Vorgehensmodell [Bal11, Fis98] aufgefasst, das insbesondere auch Rollen beschreibt.

soziale und ökologische Auswirkungen, modellieren, analysieren und verbessern können (siehe Kapitel 7). In Kapitel 8 werden *JSON-Netze als Prozessmodellierungssprache* vorgestellt. JSON-Netze unterstützen sowohl die Darstellung der betrachteten Phänomene (Geschäftsprozesse), als auch die Integration und den Austausch von (Nachhaltigkeits-)Daten. Sie eignen sich damit für Modellierung und Analyse der Nachhaltigkeitsauswirkungen von Geschäftsprozessen, können aber auch in anderen Domänen eingesetzt werden. Der in Kapitel 9 vorgestellte *JSON-Netz-Editor* stellt als *Modellierungswerkzeug* Funktionen und unterstützende Mechanismen zur Anwendung der JSON-Netze und des Vorgehensmodells bereit.

6.2 Anforderungsermittlung

Für die Anforderungsermittlung wurden in der Literatur beschriebene Anforderungslisten an Modellierungsmethoden für (konventionelles) Geschäftsprozessmanagement [Fra03, Obe96a] herangezogen. Zu beachten ist, dass die in der Literatur aufgestellten Anforderungen zum Teil in Konflikt miteinander stehen [Obe96a]. Das bedeutet, dass die Anforderungen anhand der Zielsetzung einer zu entwerfenden Modellierungsmethode abzuwägen sind. Das übergeordnete Ziel der hier entworfenen Modellierungsmethode ist die Unterstützung der Verbesserung der Nachhaltigkeitsauswirkungen von Geschäftsprozessen. Eine Reihe von in der Literatur beschriebenen Anforderungen,wie die hierarchische Verfeinerung von Modellen [Obe96a] oder die Darstellung unterschiedlicher Sichten [Fra03, Obe96a], stehen daher nicht im Fokus. Sie bieten jedoch Anknüpfungspunkte für Weiterentwicklungen, auf die in Kapitel 12 eingegangen wird. Im Folgenden werden die Anforderungen an die Modellierungsmethode beschrieben. Zur Begründung werden die Ergebnisse der Kapitel 2 bis 5 herangezogen.

(A 1) Ausdrucksmächtigkeit: Die Modellierungssprache soll das zu untersuchende Phänomen (Geschäftsprozesse) darstellen können [Fra03]. Dazu zählt die Darstellbarkeit des Kontroll- und Datenflusses von Geschäftsprozessen.

Begründung: Die Möglichkeit, Kontroll- und Datenflüsse darzustellen, ist eine zentrale Anforderung an Prozessmodellierungssprachen [Fra03, Obe96a]. Die Modellierung von Datenflüssen ist vergleichbar mit der Modellierung von Stoffströmen in der Lebenszyklusanalyse (siehe Kapitel 4). Sie stellt damit eine Voraussetzung für die Umsetzbarkeit der in Abschnitt 4.3 beschriebenen Nachhaltigkeitsmuster dar (siehe auch Anforderung A 2).

Einschränkung: Mit der Darstellbarkeit von Kontroll- und Datenflüssen sind zwei wesentliche Anforderungen an die Ausdrucksmächtigkeit einer Prozessmodellierungssprache erfüllt [Obe96a, Fra03]. Weitere Modellierungsaspekte im Geschäftsprozessmanagement wie zum Beispiel die Modellierung von Rollen und Organisationsstrukturen [Obe96a] werden im Rahmen des Methodenentwurfs nicht ausgearbeitet, da der Fokus auf der Darstellbarkeit von Nachhaltigkeitsauswirkungen liegt.

(A 2) Analysierbarkeit: Mithilfe der Modellierungsmethode sollen, basierend auf den in Abschnitt 4.3 definierten Nachhaltigkeitsmustern, Aussagen zu den Nachhaltigkeitsauswirkungen eines Geschäftsprozesses ermöglicht werden [Obe96a, Fra03]. Dies bedeutet, dass Aussagen zu verschiedenen Inputs und Outputs eines Geschäftsprozesses (NMG 1), zu seinen sozialen und ökologischen Auswirkungen bezüglich unterschiedlicher Systemgrenzen (NMG 2 und NMG 3) sowie zur Allokation der Nachhaltigkeitsauswirkungen (NMG 4) gemacht werden können.

Begründung: Wie die Untersuchung verwandter Arbeiten in Kapitel 5 zeigt, ist die Analysierbarkeit für bestehende Ansätze im Nachhaltigen Geschäftsprozessmanagement bisher nur eingeschränkt gegeben. Wird diese gewährleistet, stellt das einen wesentlichen Beitrag der vorgestellten Methode dar.

(A 3) Benutzbarkeit: Die Modellierungssprache und das Modellierungswerkzeug sollen für potentielle Anwendende leicht benutzbar sein [Fra03].

Begründung: Eine einfache Benutzbarkeit verringert die Wahrscheinlichkeit, dass bei der Anwendung der Modellierungssprache Fehler gemacht werden [Fra03]. Sie ist auch förderlich für die Verbreitung der Modellierungsmethode. Zu beachten ist, dass in vielen Fällen eine Modellierungssprache erst über die Bereitstellung eines Modellierungswerkzeugs effektiv und effizient nutzbar wird [Fra03]. Die Benutzbarkeit des Modellierungswerkzeugs spielt daher für die Erfüllung dieser Anforderung eine wichtige Rolle.

Einschränkung: Es wird zwischen einfacher Benutzbarkeit für verschiedene Gruppen von Anwendenden unterschieden (siehe auch die in Kapitel 7 beschriebenen Rollen). Das angestrebte Niveau der Modellierungsunterstützung des JSON-Netz-Editors zielt auf technisch versierte Anwendende (Werkzeugexpertinnen und -experten), die mit dem JSON-Datenmodell vertraut sind. Es soll aber anhand der Modellierungsmethode auch für nicht-technisch versierte Anwendende möglich sein, Geschäftsprozesse zu modellieren und Nachhaltigkeitsanalysen durchzuführen.

(A 4) Anpassbarkeit Die Modellierungssprache soll für individuelle Anwendungsdomänen (spezielle Unternehmen, Branchen und Nachhaltigkeitsaspekte) anpassbar sein [Fra03].

Begründung: Eine domänenspezifische Anpassbarkeit der Modellierungssprache unterstützt ihre Benutzbarkeit für Anwendende aus einer bestimmten Domäne [Fra13]. Auch im Kontext existierender Softwareunterstützung für Nachhaltigkeitsanalysen gibt es Bedarf an spezialisierten Software-Werkzeugen (siehe Unterabschnitt 3.4.1).

(A 5) Operationalisierbarkeit: Die Modellierungssprache soll die Entwicklung von Informationssystemen unterstützen [Fra03]. Daraus folgt, dass anhand der Spezifikation der Modellierungssprache Software-Werkzeuge umgesetzt werden können. Diese Software-Werkzeuge sollen auch den Datenaustausch mit anderen Informationssystemen unterstützen.

Begründung: Wird die Entwicklung von Informationssystemen unterstützt, kann künftig auch die Steuerung von Geschäftsprozessen in Unternehmen (mit prozessorientierten Informationssystemen) anhand von Nachhaltigkeitskriterien erfolgen. Wie in Kapitel 4 herausgearbeitet wurde, stellt das einen wichtigen potentiellen Beitrag des Nachhaltigen Geschäftsprozessmanagements (gegenüber bestehender Nachhaltigkeitsanalysemethoden) dar.

7 Vorgehensmodell für Nachhaltiges Geschäftsprozessmanagement

Als erste der drei Komponenten der in dieser Arbeit beschriebenen Modellierungsmethode, bestehend aus Vorgehensmodell, Modellierungssprache und Modellierungswerkzeug, wird in diesem Kapitel ein Vorgehensmodell für Nachhaltiges Geschäftsprozessmanagement vorgestellt. Es verbindet Konzepte des Geschäftsprozessmanagements mit Konzepten der Nachhaltigkeitsanalyse, mit dem Ziel, Orientierung bei der Modellierung, Analyse und Verbesserung der Nachhaltigkeit von Geschäftsprozessen zu bieten. Das Vorgehensmodell ist dabei generisch formuliert, sodass es für unterschiedliche Domänen und Nachhaltigkeitsaspekte sowie unabhängig von konkreten Modellierungssprachen und -werkzeugen anwendbar ist. In Abschnitt 7.1 wird zunächst eine Übersicht über das Vorgehensmodell gegeben und zentrale Begriffe in einem Glossar vorgestellt. Die einzelnen Phasen des Vorgehensmodells werden schließlich in Abschnitt 7.2 näher beschrieben.[1]

7.1 Herleitung und Übersicht

Im Geschäftsprozessmanagement bieten Vorgehensmodelle Orientierung und Hilfestellungen beim Modellieren, Analysieren und Verbessern von Geschäftsprozessen [Bal11]. Sie stellen dazu insbesondere eine Beschreibung von *Phasen* mit zu erstellenden *Ergebnissen* und zuständigen *Rollen* bereit [Fis98]. Ergänzend dazu werden im Folgenden *Verbesserungsmerkmale* definiert. Sie sind den einzelnen Phasen des Vorgehensmodells zugeordnet und beschreiben, wie sich die Validität und Effektivität des Nachhaltigen Geschäftsprozessmanagements iterativ verbessern lässt. Das hier vorgestellte Vorgehensmodell integriert Konzepte des Geschäftsprozessmanagement und der Lebenszyklusanalyse, um die Modellierung, Analyse und Verbesserung der Nachhaltigkeitsauswirkungen von Geschäftsprozessen zu unterstützen. Es orientiert

[1] Eine erste Version des Vorgehensmodells findet sich in der Veröffentlichung [Fri23a]. Für die hier vorliegenden Arbeit wurde das Vorgehensmodell erweitert und überarbeitet.

sich sowohl an den Phasen des Geschäftsprozesslebenszyklus [Wes19, Dum18] und einem Vorgehensmodell für (konventionelles) Geschäftsprozessmanagement [Wes19], als auch an den Phasen einer Lebenszyklusanalyse [ISO06a].

Abbildung 7.1 zeigt eine Übersicht der *Phasen* des Vorgehensmodells: (1) Spezifikation des Indikatormodells, (2) Geschäftsprozessidentifikation, (3) Datenerhebung und -integration, (4) Analyse und Verbesserung und (5) Umsetzung. Für jede *Phase* ist in einem grauen Kasten jeweils das *Ergebnis* (zum Beispiel ein Geschäftsprozessmodell als Ergebnis des Schritts Geschäftsprozessidentifikation) angegeben. Die einzelnen Phasen sind dabei nicht streng sequentiell durchzuführen. So können sich in der praktischen Anwendung auch mehrere Phasen in ihrer zeitlichen Folge überschneiden. Entsprechend zeigen die rückführenden Pfeile an den einzelnen Phasen in der Abbildung, dass Ergebnisse einer Phase auch Überarbeitungen der Ergebnisse einer vorangegangenen Phase erfordern können. Der große rückführende Pfeil zeigt, dass die Phasen iterativ durchgeführt werden können. So kann zunächst, in einer ersten Iteration, nur ein eng abgegrenzter Anwendungsfall unterstützt werden. In weiteren Iterationen kann die Validität und Effektivität des Nachhaltigen Geschäftsprozessmanagements dann schrittweise verbessert werden. Entlang des rückführenden Pfeils sind den einzelnen Phasen *Verbesserungsmerkmale* wie Prozessabdeckung und Granularität zugeordnet. Sie beschreiben für die jeweilige Phase, welche Aspekte innerhalb einer Iteration verbessert werden können.

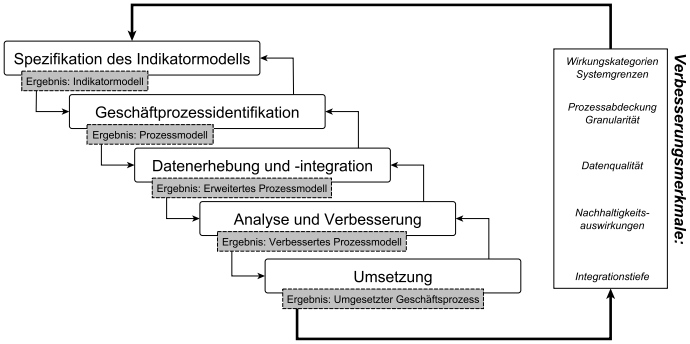


Abbildung 7.1: Vorgehensmodell für Nachhaltiges Geschäftsprozessmanagement.

Zentrale Begriffe des Vorgehensmodells werden in der folgenden Liste vorgestellt. Es werden dabei sowohl Begriffe und Definitionen aus dem Geschäftsprozessmanagement (siehe Abschnitt 2.2) und aus der Lebenszyklusanalyse (siehe Abschnitt 3.3) verwendet und adaptiert. Für jeden Begriff werden neben der gegebenen Definition die Quellen angegeben. Zu beachten ist, dass die Definitionen aus den Quellen nicht wörtlich übernommen werden, sondern angepasst wurden, um im Kontext dieser Arbeit ein konsistentes Begriffsverständnis zu schaffen.

Geschäftsprozess: Eine Menge von Aktivitäten, die in einem Unternehmen nach bestimmten Regeln auf ein bestimmtes Ziel hin ausgeführt werden.

Quellen: [Obe96a].

Prozesstyp: Allgemeiner Rahmen zur Beschreibung von Aktivitäten und ihrer Zusammenhänge.

Quellen: [Obe96a, Wes19].

Prozessinstanz: Ausprägung eines Prozesstyps mit eindeutig identifizierbaren Aktivitäten im Sinne einer Ausführung der entsprechenden Aktivitäten.

Quellen: [Obe96a, Wes19].

Aktivitätstyp: Allgemeine Beschreibung einer Aktivität, von der es unterschiedliche Ausprägungen geben kann.

Quellen: [Obe96a, Wes19].

Aktivitätsinstanz: Ausprägung eines Aktivitätstyps im Sinne einer Ausführung einer Aktivität.

Quellen: [Obe96a, Wes19].

Prozessmodell: (Grafische) Repräsentation eines Geschäftsprozesses. Kann sowohl einen Prozesstyp als auch den Zustand einer oder mehrerer Prozessinstanzen darstellen.

Quellen: [Wes19].

Wirkungskategorie: Ein sozialer oder ökologischer Nachhaltigkeitsaspekt, für den die Auswirkungen eines Geschäftsprozesses gemessen werden sollen.

Quellen: [ISO06a, UNE20]

Wirkungsindikator: Misst die Nachhaltigkeitsauswirkungen eines Geschäftsprozesses bezüglich einer Wirkungskategorie.

Quellen: [ISO06a, UNE20]

Inventarindikator: Misst nachhaltigkeitsrelevante Inputs und Outputs einer Aktivität. Inventarindikatoren erfassen zum Beispiel verarbeitete Materialien, Energiebedarfe oder verwendete Ressourcen wie Maschinen oder menschliche Arbeitszeit.

Quellen: [Klö09, UNE20].

Wirkungsfaktor: Faktor zur Umrechnung eines Inventarindikators in einen Wirkungsindikator.

Quellen: [WBC04, Klö09].

Allokationsfaktor: Faktor zur Skalierung eines Inventarindikators oder Wirkungsindikators. Wird benötigt, um Nachhaltigkeitsauswirkungen zwischen Prozesskomponenten aufzuteilen.

Quellen: [ISO06a, Klö09].

Systemgrenze: Definition von Kriterien, die den Umfang der zu ermittelnden Inventarindikatoren und die Interpretation der Wirkungsfaktoren bestimmen.

Quellen: [ISO06a].

Beteiligte *Rollen* bei der Anwendung des Vorgehensmodells sind Personen mit *Nachhaltigkeitsexpertise*, *Branchenexpertise*, *Werkzeugexpertise* und *Anwendende*. Personen mit Nachhaltigkeitsexpertise verfügen über Fachwissen zu einem Nachhaltigkeitsaspekt. Personen mit Branchenexpertise verfügen über Fachwissen bezüglich einer Branche (wie Elektronik, Landwirtschaft, Bankenwesen, ...). Personen mit Werkzeugexpertise verfügen über technisches Wissen zur Anpassung eines Modellierungswerkzeugs. Anwendende nutzen ein Modellierungswerkzeugs zur Modellierung, Analyse und Verbesserung der Nachhaltigkeit eines oder mehrerer Geschäftsprozesse. Die einzelnen Rollen haben in den verschiedenen Phasen des Vorgehensmodells unterschiedliche Aufgaben, die im nächsten Abschnitt beschrieben werden.

7.2 Phasen des Vorgehensmodells

Im Folgenden werden die Phasen des Vorgehensmodells im Detail beschrieben. Für jede Phase wird beschrieben, welche Aufgaben die unterschiedlichen Rollen haben und welche Ergebnisse erwartet werden. Weiterhin werden Verbesserungsmerkmale beschrieben, die aufzeigen, wie die Validität[1] und Effektivität des Nachhaltigen

[1] Mit Validität ist die Belastbarkeit der Nachhaltigkeitsanalysen gemeint, also eine möglichst präzise und umfassende Identifikation von Nachhaltigkeitsauswirkungen.

Geschäftsprozessmanagements im Rahmen des Vorgehensmodells verbessert werden können. Unterschiedliche Ausgestaltungsmöglichkeiten der jeweiligen Phasen werden anhand von Beispielen erläutert.

7.2.1 Spezifikation des Indikatormodells

In der ersten Phase des Vorgehensmodells wird ein Indikatormodell spezifiziert und gegebenenfalls ein Modellierungswerkzeug entsprechend angepasst. Das Indikatormodell wird von Personen mit Nachhaltigkeits- und Branchenexpertise gemeinsam erarbeitet. Es beschreibt, wie Nachhaltigkeitsauswirkungen anhand entsprechender Wirkungsindikatoren gemessen werden können. Für jeden Wirkungsindikator wird beschrieben, welche Inventarindikatoren zu seiner Ermittlung erfasst werden sollen. Weiterhin ist festzulegen wie die Inventarindikatoren (anhand von Wirkungsfaktoren) zu Wirkungsindikatoren umgerechnet werden können. In dieser Phase sind weiterhin Systemgrenzen festzulegen. Dazu werden Kriterien definiert, die bestimmen, welche Inventarindikatoren zur Wirkungsmessung herangezogen werden. Die Definition des Indikatormodells kann zunächst natürlichsprachlich erfolgen. Eine Person mit Werkzeugexpertise kann auf Basis dieser Beschreibungen ein Modellierungswerkzeug anpassen, um Anwendende bei der Erfassung von Inventarindikatoren und Wirkungsfaktoren zu unterstützen.

Ergebnis: Natürlichsprachliche Beschreibung eines *Indikatormodells*. *Angepasstes Modellierungswerkzeug* mit umgesetztem Indikatormodell.

Verbesserungsmerkmale: Die Auswahl der betrachteten *Wirkungskategorien* kann schrittweise ausgeweitet werden. Weiterhin können zunächst engere *Systemgrenzen* gezogen werden, um diese in späteren Iterationen zu erweitern.

Beispiel: Für die Analyse der Nachhaltigkeitsauswirkungen von Geschäftsprozessen in der Elektronikindustrie wird zunächst die Wirkungskategorie Klimaänderung ausgewählt. Als Wirkungsindikator sind daher Treibhausgasemissionen in kg CO2e zu erfassen. Dem Treibhausgasprotokoll [WBC04] folgend können die Systemgrenzen Scope 1, Scope 2 und Scope 3 unterschieden werden.[1] In einer ersten Iteration des Vorgehensmodells kann die Analyse zunächst auf die Systemgrenze Scope 1 eingeschränkt werden. Für diese Systemgrenze wären als Inventarindikatoren Mengen von (direkten) Treibhausgasemissionen zu ermitteln. Wirkungsfaktoren quantifizieren die Wirkung

[1] Zur Unterscheidung der Scopes siehe die Ausführungen zum Treibhausgasprotokoll in Abschnitt 3.2.

von unterschiedlichen Treibhausgasen relativ zu CO2. In einer späteren Iteration wäre die Betrachtung auf Scope 2 auszuweiten (Verbesserungsmerkmal Systemgrenzen). Für Scope 2 sind dann Energiebedarfe als Inventarindikatoren zu erfassen. Ein weiteres Verbesserungsmerkmal betrifft die betrachteten Wirkungskategorien. Hierzu kann das Indikatormodell um weitere ökologische Aspekte wie Wasserverbrauch und Biodiversität sowie soziale Aspekte wie Arbeitsbedingungen erweitert werden. Dazu wären jeweils entsprechende Systemgrenzendefinitionen zu ergänzen.

7.2.2 Geschäftsprozessidentifikation

Mit Hilfe des angepassten Modellierungswerkzeugs können Anwendende die zu analysierenden Geschäftsprozesse modellieren. Bei der Auswahl eines Geschäftsprozess kann zunächst eine Abschätzung der Nachhaltigkeitsauswirkungen der Geschäftsprozesse eines Unternehmens erfolgen. Die Geschäftsprozesse mit hohen zu erwartenden Nachhaltigkeitsauswirkungen sind geeignete Kandidaten für eine erste Iteration. Alternativ kann die Auswahl auch pragmatisch erfolgen, wenn beispielsweise schon eine ausführliche Dokumentation eines Geschäftsprozesses vorhanden ist.

Ergebnis: Ein *Prozessmodell* oder mehrere *Prozessmodelle*.

Verbesserungsmerkmale: Je nach Anwendungsfall kann die Betrachtung der Geschäftsprozesse in unterschiedlichem Umfang (*Prozessabdeckung*) und auf unterschiedlichen Granularitätsstufen (*Granularität*) erfolgen. Für ein effektives Nachhaltiges Geschäftsprozessmanagement sollte eine hohe Abdeckung (also eine Modellierung möglichst aller Geschäftsprozesse des Unternehmens) und hohe Granularität angestrebt werden. Es kann aber sinnvoll sein, zunächst mit geringer Prozessabdeckung oder geringer Granularität zu beginnen und die Betrachtung später zu erweitern beziehungsweise zu verfeinern.

Beispiel: Ein produzierendes Unternehmen aus der Elektronikindustrie kann zunächst einen Kernprozess wie die Leiterplattenbestückung als Kandidaten auswählen. Dieser Geschäftsprozess kann detailliert unter Beachtung diverser Teilschritte (zum Beispiel Bestücken und Löten) betrachtet werden. In diesem Fall würde die Untersuchung mit einer hohen Granularität, aber einer geringen Prozessabdeckung stattfinden, da nur ein Geschäftsprozess detailliert betrachtet wird. Ein anderer Ansatz wäre, das ganze Unternehmen in den Blick zu nehmen und Daten zu Inputs und Outputs auf Basis von Einkaufsdaten und Verkaufszahlen zu erheben. So können (in den folgenden Schritten

des Vorgehensmodells) zum Beispiel die Treibhausgasemissionen für das gesamte Unternehmen oder für Teilbereiche abgeschätzt werden. In diesem Fall würde zunächst keine detaillierte Prozessmodellierung stattfinden und damit eine Betrachtung mit geringer Granularität, aber hoher Prozessabdeckung erfolgen.

7.2.3 Datenerhebung und -integration

Nachdem ein Geschäftsprozess modelliert wurde, können Anwendende entsprechend dem in der ersten Phase spezifizierten Indikatormodell Inventarindikatoren und Wirkungsfaktoren zum Geschäftsprozess ermitteln. Insbesondere für Ressourcen, die von mehreren Aktivitäten genutzt werden, ist zu klären, wie die damit zusammenhängenden Nachhaltigkeitsauswirkungen den jeweiligen Aktivitäten angerechnet werden können. Dafür sind geeignete Allokationsfaktoren zu definieren und anzuwenden. Auch für Aktivitäten, die mehrere Outputs generieren, ist zu klären ob und wie die Nachhaltigkeitsauswirkungen der Aktivität den einzelnen Outputs angerechnet werden.

Ergebnis: Erweitertes Prozessmodell mit zugeordneten Inventarindikatoren, Wirkungsfaktoren und Allokationsfaktoren.

Verbesserungsmerkmale: Je nach Anwendungsfall und Datenverfügbarkeit können *Datenquellen unterschiedlicher Qualität* akzeptiert werden. In der Literatur zu Nachhaltigkeitsanalysen wird zwischen spezifischen und generischen Daten unterschieden [Klö09] (siehe auch Unterabschnitt 3.3.1). Spezifische Daten umfassen direkte Messungen an den betrachteten Prozessen. Generische Daten zur Messung von Nachhaltigkeitsauswirkungen stammen zum Beispiel aus kommerziellen Datenbanken, die herangezogen werden können, wenn spezifische Daten fehlen. Soweit vorhanden, sollten spezifische Daten verwendet werden, bis hin zur Integration von Echtzeitdaten zu einzelnen Ausführungen eines Geschäftsprozesses. Wenn keine spezifischen Daten verfügbar sind, kann zunächst auch mit der Integration von generischen Daten begonnen werden. Diese können in weiteren Iterationen durch spezifische Daten ersetzt werden.

Beispiel: Als Inventarindikator ist zum Beispiel die Energie zu erfassen, die für die verschiedenen Aktivitäten der Leiterplattenbestückung benötigt wird (falls als Systemgrenze Scope 2 des Treibhausgasprotokolls [WBC04] im Indikatormodell festgelegt wurde). Ein Wirkungsfaktor kann in diesem Fall in Abhängigkeit vom verwendeten Strommix angegeben werden. Als generische Daten können branchenübliche

Durchschnittsdaten für den Energieverbrauch angenommen werden. Um die Datenqualität zu verbessern, wäre der Energieverbrauch bei der Aktivitätsausführung im Unternehmen zu messen. Zur Bestimmung der Scope 3-Emissionen wären verbaute Elektronikkomponenten als Inventarindikatoren zu erfassen. Generische Daten für entsprechende Wirkungsfaktoren können aus kommerziellen Lebenszyklusanalysedatenbanken herangezogen werden [Klö09]. Spezifische Daten zu Scope 3-Emissionen wären in diesem Fall direkt bei den liefernden Unternehmen zu erfragen.

7.2.4 Analyse und Verbesserung

In der nächsten Phase, *Analyse und Verbesserung*, können Anwendende anhand des erweiterten Prozessmodells (softwaregestützte) Analysen durchführen und so Wirkungsindikatoren zum Geschäftsprozess sowie einzelnen Prozesskomponenten ermitteln. Basierend auf den durchgeführten Analysen können Verbesserungspotentiale erkannt und die zugehörigen Prozessmodelle gegebenenfalls entsprechend angepasst werden. Die Ergebnisse können gegebenenfalls zusammen mit Personen mit Branchen- und Nachhaltigkeitsexpertise evaluiert werden.

Ergebnis: Verbessertes Prozessmodell mit geringeren (negativen) Nachhaltigkeitsauswirkungen.

Verbesserungsmerkmale: In jeder Iteration kann das Prozessmodell mit dem Ziel verbessert werden, dass geringere (negative) *Nachhaltigkeitsauswirkungen* anfallen. Dabei sollte das Verschieben von negativen ökologischen oder sozialen Auswirkungen zu anderen Unternehmen und Anspruchsgruppen oder zwischen verschiedenen Wirkungskategorien vermieden werden. Sollte das Verschieben von Nachhaltigkeitsauswirkungen nicht vermeidbar sein, ist eine begründete Priorisierung von Wirkungskategorien vorzunehmen.

Beispiel: Ein Elektronikunternehmen kann bei seinen Löt-Aktivitäten auf Zinn aus recyceltem Material umstellen, der in der Herstellung geringere Umweltlasten verursacht und damit die (Scope 3-)Nachhaltigkeitsbewertung verbessert. Diese mögliche Verbesserung kann zunächst in einem Prozessmodell abgebildet und untersucht werden.

7.2.5 Umsetzung

In der letzten Phase wird das verbesserte Prozessmodell von Anwendenden im Unternehmen *umgesetzt*, um eine Verbesserung hinsichtlich der tatsächlichen Nachhaltigkeitsauswirkungen des Unternehmens zu erreichen. Die notwendigen Anpassungen im Unternehmen werden veranlasst, sodass die realen Geschäftsprozesse den verbesserten Prozessmodellen aus der vorangegangenen Phase entsprechen.

Ergebnis: Ein *umgesetzter Geschäftsprozess*.

Verbesserungsmerkmale: Bei der Umsetzung sollte eine möglichst hohe *Integrationstiefe* bezüglich der Informationssysteme des Unternehmens angestrebt werden. Eine hohe Integrationstiefe bedeutet in diesem Zusammenhang, dass Nachhaltigkeitsdaten unmittelbar in die steuernden und entscheidungsunterstützenden Informationssysteme des Unternehmens eingebunden werden.

Beispiel: Zur Umsetzung des verbesserten Prozessmodells aus dem vorangegangenen Schritt werden die notwendigen Anpassungen, wie zum Beispiel der Einkauf von Zinn aus recyceltem Material, umgesetzt. Wenn das Unternehmen lediglich jährlich einen Nachhaltigkeitsbericht erstellt und die dazu notwendigen Daten manuell erhoben werden [Elk18], kann das als Nachhaltiges Geschäftsprozessmanagement mit geringer Integrationstiefe angesehen werden. Ein Beispiel für eine hohe Integrationstiefe wäre die (automatisierte) Steuerung von Aktivitäten in Abhängigkeit vom verfügbaren Strommix. So kann die Ausführung von energieintensiven Aktivitäten verzögert werden, bis ein großer Anteil Strom aus erneuerbaren Energien verfügbar ist [Heh24].

8 Entwicklung einer datenorientierten Prozessmodellierungssprache

In diesem Kapitel werden JSON-Netze als eine Variante Höherer Petri-Netze zur datenorientierten Modellierung von Geschäftsprozessen vorgestellt. Es werden die Struktur, die Syntax der Transitions-, Kanten- und Stellenbeschriftungen, sowie die Schaltregel für JSON-Netze beschrieben. Die in dieser Arbeit beschriebene Spezifikation von JSON-Netzen baut auf der Veröffentlichung [Fri23b] auf. Grundlagen zum JSON-Datenmodell (siehe Abschnitt 8.3) und zu JSON-Technologien (siehe Abschnitt 8.2) wurden in Teilen aus [Fri23b] übernommen. Die Spezifikation der JSON-Netze selbst wurde wesentlich überarbeitet und erweitert. Insbesondere verfügt die hier vorliegende Version der JSON-Netze im Vergleich zu [Fri23b] über die Fähigkeit, nicht nur JSON-Dokumente als Ganzes zu verarbeiten, sondern auch Substrukturen in JSON-Dokumenten zu manipulieren.

8.1 Zielsetzung für die Prozessmodellierungssprache

JSON-Netze verbinden einen etablierten und gut erforschten Formalismus zur Prozessmodellierung (Petri-Netze) [Aal15] mit einem modernen Datenmodell (JSON) [IET17]. Das erlaubt es, die Stärken von Petri-Netzen (insbesondere Simulationsfähigkeit [Obe96a, Aal15]) einerseits und die Stärken von JSON (weite Verbreitung, einfache Darstellung und Verarbeitung von Datenobjekten, Programmiersprachenunabhängigkeit [Dro22, ECM17, Bou20]) für die Modellierung von Geschäftsprozessen zu nutzen. Als Inspiration dienen vergleichbare Ansätze, die Petri-Netze mit semi-strukturierten Daten [Abi97] verknüpfen [Len01, Bad15].

In den Abbildungen 8.1 und 8.2 werden die im Folgenden beschriebenen Konzepte und Ausdrucksmöglichkeiten von JSON-Netzen anhand eines Beispiels illustriert. Das abgebildete Petri-Netz stellt einen einfachen Geschäftsprozess dar, der aus einer Aktivität, der Vorbereitung einer Lieferung (t), besteht. Im Prozess werden anhand von Aufträgen (gespeichert in der Stelle s_1) und Produkten im Lager (s_2) Lieferungen (s_3)

zusammengestellt. In JSON-Netzen werden Stellen mit *Daten im JSON-Datenmodell* (JSON-Dokumenten) markiert, um individuelle Marken (im Beispiel unterschiedliche Aufträge, Produkte, Lieferungen) darzustellen. Diese Marken können auch komplex strukturiert sein, das heißt ihre Attribute oder Eigenschaften können selbst Attribute oder Eigenschaften besitzen. In der Abbildung wird das dadurch illustriert, dass ein Auftrag aus mehreren Unterpositionen besteht, von denen manche erfüllt sind (markiert mit einem Haken) und andere nicht (markiert mit einem Kreuz). Ein anderes Beispiel für komplex strukturierte Marken in der Abbildung sind Lieferungen, die als Lastwagen-Symbole dargestellt werden und verschiedene Produkte (dargestellt als Quadrate) enthalten. Mit *Schemas* können Stellen typisiert werden, um so Regeln für die Struktur und Werte der enthaltenen Marken festzulegen. So kann zum Beispiel festgelegt werden, welche Eigenschaften für Auftragspositionen oder Lieferungen erfasst werden müssen. Mit *Filtern* können Marken (Aufträge, Produkte, Lieferungen) oder Substrukturen der Marken (Auftragspositionen, Lieferpositionen) zur Bearbeitung ausgewählt werden. Transitionen werden mit *Logischen Ausdrücken* beschriftet, die Zusammenhänge zwischen Marken sowie Manipulationen an den Daten in den Marken beschreiben. Im gegebenen Beispiel könnte ein logischer Ausdruck beschreiben, dass ein bestimmter Auftrag immer zu einer bestimmten Lieferung gehört, und

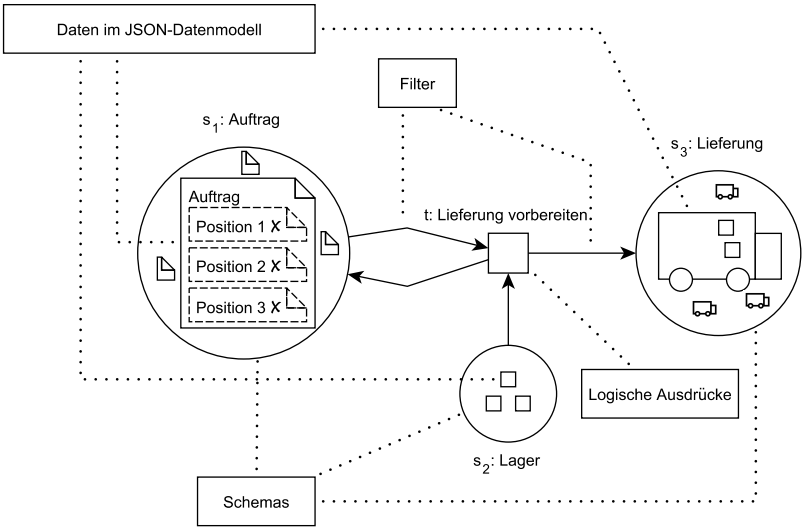


Abbildung 8.1: Konzepte und Ausdrucksmöglichkeiten von JSON-Netzen.

dass eine Auftragsposition, nachdem das entsprechende Produkt zur Lieferung hinzugefügt wurde, als erledigt zu markieren ist. Das dynamische Verhalten von JSON-Netzen wird im Zusammenspiel der Abbildungen 8.1 und 8.2 deutlich. Nach dem Schalten der Transition wurde ein Produkt dem Lager entnommen, der Lieferung hinzugefügt und die entsprechende Auftragsposition aktualisiert. Zusammengefasst werden in JSON-Netzen Daten im JSON-Datenmodell verarbeitet, indem *Werte in JSON-Dokumenten ausgewählt und gelöscht*, neue *Werte erzeugt und in JSON-Dokumente eingefügt* sowie *Datenstrukturen und -werte überprüft* werden.

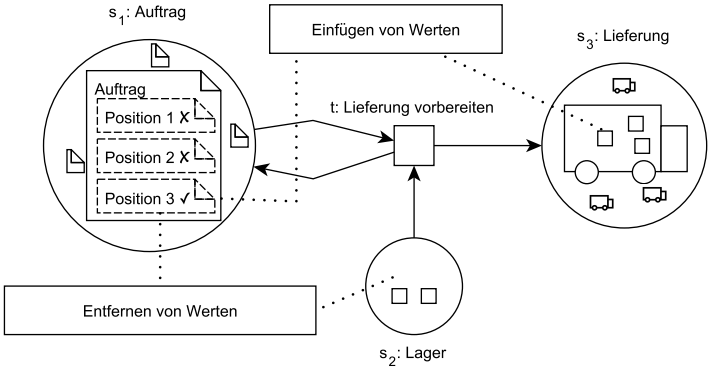


Abbildung 8.2: Darstellung des dynamischen Verhaltens von JSON-Netzen.

Zur Umsetzung der erwähnten Konzepte wie Filter, Schemas und Logischen Ausdrücke sind unterschiedliche Technologien denkbar. Mit Technologien sind an dieser Stelle formalisierte (Computer-)Sprachen gemeint, die JSON-Dokumente verarbeiten oder beschreiben können. Bei der Spezifikation und Implementierung soll jedoch sowohl eine gewisse Technologieneutralität, als auch eine einfache Benutzbarkeit von JSON-Netzen als Modellierungssprache erreicht werden. Mit Technologieneutralität ist hier zum Einen gemeint, dass JSON-Netze in unterschiedlichen technischen Kontexten, etwa unter Verwendung unterschiedlicher Programmiersprachen, implementierbar sein sollen. Zum Anderen sollen JSON-Netze auch offen und anpassbar für Weiterentwicklungen und Verbesserungen von bestehenden Technologien sein. Neben der Modellierungsunterstützung in einem JSON-Netz-Werkzeug (siehe Kapitel 9), zählt zum Ziel der einfachen Benutzbarkeit, dass die zur Implementierung von JSON-Netzen ausgewählten Technologien für eine spezifische Aufgabe (zum Beispiel als Filter oder zur Beschreibung von Logischen Ausdrücken) gut geeignet sein sollen. Die Technologie sollte

aber gleichzeitig keinen zu großen Funktionsumfang haben, um unnötige Komplexität für die Nutzerinnen und Nutzer zu vermeiden [Ray03]. Dazu werden JSON-Netze einerseits formal, unter Verwendung abstrakter Konzepte, definiert. Es wird andererseits aber auch eine konkrete Umsetzung der abstrakten Konzepte unter Verwendung aktueller Technologien vorgeschlagen. So wird erreicht, dass die abstrakte Spezifikation gültig bleibt, auch wenn sich die ausgewählten Technologien in der Zukunft weiterentwickeln sollten, oder neue, in Bezug auf Mächtigkeit oder Benutzbarkeit „bessere" Technologien entwickelt werden. Die Auswahl konkreter, etablierter Technologien für die Umsetzung der Spezifikation stellt weiterhin die einfache Benutzbarkeit der JSON-Netze sicher und erfüllt in den folgenden Beschreibungen auch den Zweck, die abstrakten Konzepte der Spezifikation verständlich und nachvollziehbar zu machen. Bei der Auswahl der Technologien wird darauf geachtet, dass sie selbst möglichst technologieneutral sind, in dem Sinn, dass sie möglichst auf programmiersprachenunabhängigen Standards beruhen sollen.

8.2 Übersicht und Bewertung von JSON-Technologien

Aus der Zielsetzung für die Spezifikation von JSON-Netzen geht hervor, dass in JSON-Netzen Möglichkeiten (1) zur Beschreibung von Strukturen und Werten von JSON-Dokumenten (*Schemas*), (2) zur Auswahl von Werten in einem JSON-Dokument (*Filter*) und (3) zur Beschreibung von logischen Zusammenhängen zwischen JSON-Dokumenten und Regeln zur Manipulation von JSON-Dokumenten (*logische Ausdrücke*) bereitgestellt werden. Aufgrund seiner einfachen Syntax ist JSON als Datenmodell insbesondere für den Datenaustausch im Internet und bei NoSQL-Datenbanken im Einsatz [Bou20]. In der Praxis existiert daher eine Vielzahl von Technologien, die mit Daten im JSON-Datenmodell arbeiten und für die Spezifikation von JSON-Netzen in Betracht gezogen werden können. Im Unterschied zu XML-Technologien wurden JSON-Technologien bisher jedoch weniger formal untersucht [Bou20], was den Vergleich der Fähigkeiten verschiedener JSON-Technologien erschwert. In Anlehnung an [Bou17, Bou20] können JSON-Technologien in *Pfadsprachen*, *Anfragesprachen*, *Schemasprachen* und *Programmiersprachen* unterteilt werden. Als ein Spezialfall wird hier auch eine *Templatesprache* betrachtet. In der folgenden Liste wird jede Sprachkategorie beschrieben und es werden jeweils Beispiele genannt. Weiterhin wird jede Kategorie nach ihren Fähigkeiten in Bezug auf die drei benötigen Funktionalitäten (*Schemas*, *Filter* und *logische Ausdrücke*) bewertet. Die Bewertung der Fähigkeiten wird in Tabelle 8.1 zusammengefasst. Aus Platzgründen sind die drei genannten

Funktionalitäten in der Tabelle gekürzt wiedergegeben (*Sch* für *Schemas*, *Filt* für *Filter* und *Log* für *logische Ausdrücke*). Bei der Bewertung drückt ein leerer Kreis ○ aus, dass mit den Technologien der betreffenden Kategorie eine Funktionalität nicht umsetzbar ist. Mit einem halb ausgefüllten Kreis ◑ wird angegeben, dass die Fähigkeit für eine Funktionalität gegeben ist. Mit einem voll ausgefüllten Kreis ● wird schließlich ausgedrückt, dass die Technologien der betreffenden Kategorie besonders gut für eine Aufgabe geeignet sind. Aufgrund der oben bereits erwähnten fehlenden formalen Untersuchung von JSON-Technologien ist eine objektive Untersuchung der Mächtigkeit der Sprachen nur begrenzt möglich. Die Unterscheidung zwischen einem halb ausgefüllten Kreis und einem voll ausgefüllten Kreis ist daher zu einem gewissen Grad subjektiv und basiert darauf, ob sich die betreffenden Sprachen in ihrer Spezifikation oder Dokumentation selbst so beschreiben. Die folgende Liste soll einen qualitativen Überblick über verfügbare JSON-Technologien geben und einen Rahmen für die Diskussion der unterschiedlichen Fähigkeiten bieten. Sie umfasst viele populäre JSON-Technologien, ist aber nicht als vollständige Auflistung anzusehen.

Pfadsprachen Mit Pfadsprachen sind Technologien gemeint, die lediglich eine „deterministische“ Auswahl [Bou20] von Werten in einem JSON-Dokument erlauben. Das bedeutet, dass anhand eines Ausdrucks in einer Pfadsprache nur genau ein Wert in einem JSON-Dokument ausgewählt werden kann. Komplexere Regeln, mit denen innerhalb eines JSON-Dokuments auch mehrere Werte ausgewählt werden können, sind nicht formulierbar (◑ für *Filt*). Pfadausdrücke sind Bestandteil vieler JSON-Technologien, werden dort aber um mächtigere Ausdrucksmöglichkeiten erweitert [Bou20]. Für die Formulierung von Pfadausdrücken existiert mit JSON Pointer [IET13] ein programmiersprachenunabhängiger Standard. Mit Pfadausdrücken sind Schemabeschreibungen nur sehr eingeschränkt möglich (◑ für *Sch*). Zusammenhänge zwischen JSON-Dokumenten und Manipulationen können mit Pfadausdrücken nicht beschrieben werden (○ für *Log*).

Anfragesprachen werden verwendet, um Anfragen auf Datenbanken von JSON-Dokumenten zu formulieren und Knoten in komplex strukturierten JSON-Dokumenten auszuwählen [W3C17a, W3C17b]. Eine Familie in dieser Sprachkategorie orientiert sich am XPath-Standard für XML [W3C17a]. Sprachen dieser Familie stellen mächtige Funktionen bereit, um in JSON-Dokumenten zu navigieren [Bou20] (● für *Filt*). JSONPath [Gös07] und JMESPath [Sar15] gehören zu dieser Familie, genauso wie XPath selbst, das in der aktuellen Version des Standards die Fähigkeit bekommen hat, JSON-Dokumente zu verarbeiten

[W3C17a]. Andere Beispiele für Anfragesprachen sind XQuery [W3C17b] (eine Erweiterung von XPath, die ebenfalls in der aktuellen Version sowohl mit JSON- als auch mit XML-Daten arbeiten kann), JSoniq [Rob22] (eine JSON-spezifische Anfragesprache, die eine ähnliche Syntax wie XQuery hat), SQL++ [Ong14] (hier wird SQL-artige Syntax von relationalen Datenbanken auf JSON übertragen [Bou17]) und Anfragesprachen für JSON-Datenbanksysteme wie MongoDB [Mon24, The23]. Viele der Anfragesprachen haben auch Fähigkeiten, JSON-Dokumente zu manipulieren (◑ für *Log*). Manche bieten hier nur eingeschränkte Ausdrucksmöglichkeiten, wie die Anfragesprache für MongoDB [Bou17]. Andere Anfragesprachen wie zum Beispiel XQuery bieten aber auch mächtigere Möglichkeiten zur Manipulation von JSON-Dokumenten [Bou17, W3C17b]. Wie [Bou17] zeigt, sind Anfragesprachen wie JSONPath in Bezug auf ihre Ausdrucksmächtigkeit vergleichbar mit Schemasprachen wie JSON Schema. Anfragesprachen können daher, auch wenn sie darauf nicht ausgelegt sind, prinzipiell auch für Schemabeschreibungen von JSON-Dokumenten genutzt werden (◑ für *Sch*).

Schemasprachen dienen dazu, die Struktur von JSON-Dokumenten zu spezifizieren [Bou17, Pez16] (● für *Sch*). Mit JSON Schema liegt ein Entwurf einer standardisierten Schemasprache für JSON-Dokumente vor [Wri22a]. Die Fähigkeiten der Schemasprache sind vergleichbar mit dem XML-Standard XML Schema [W3C12]. So erlaubt JSON Schema zum Beispiel die Definition von Wertebereichen oder die Angabe, ob ein Wert optional oder erforderlich ist. Eine leichtgewichtigere Alternative zur Schemadefinition ist JSON TypeDef [JSO23]. Im Allgemeinen ist es nicht möglich, JSON-Dokumente mit einer Schemasprache zu manipulieren, zumindest solange sie nicht mit Fähigkeiten anderer Technologien kombiniert wird (○ für *Log*). Wie oben bereits erwähnt, sind Schema- und Anfragesprachen vergleichbar in Bezug auf ihre Ausdrucksmächtigkeit [Bou17]. Schemasprachen können prinzipiell also auch für Auswahlaufgaben verwendet werden (◑ für *Filt*).

Programmiersprachen Für die von JSON spezifizierten Datentypen gibt es in gängigen Programmiersprachen Äquivalente [Dro22]. Daher können Programmiersprachen, auch wenn sie keine spezialisierten JSON-Technologien sind, eingesetzt werden, um JSON-Dokumente einzulesen und dann zum Beispiel zu filtern oder zu manipulieren (◑ für *Sch*; ◑ für *Filt*; ◑ für *Log*). Besonders einfach ist das in Skriptsprachen wie JavaScript oder Python umzusetzen, da hier JSON-Dokumente unmittelbar als in den Sprachen bereits vorhandene Objekt- oder Listenstrukturen interpretiert werden können.

Templatesprachen Einen Spezialfall in der hier vorgestellten Liste von JSON-Technologien bildet die Templatesprache Jsonnet [Goo24c]. In der Sprache Jsonnet können Ausdrücke, die JSON-Dokumente erzeugen, und Vergleiche zwischen JSON-Dokumenten formuliert werden (● für *Log*). Jsonnet wurde als Erweiterung von JSON spezifiziert, sodass jedes JSON-Dokument selbst schon ein valider Jsonnet-Ausdruck ist. Darüber hinaus wurden Fähigkeiten aus Programmiersprachen hinzugefügt, wie Variablendefinitionen, Funktionsdefinitionen und Objektorientierung. Damit ist die Sprache nicht nur fähig, Regeln für die Manipulation von JSON-Dokumenten zu beschreiben (was die Primärfunktion ist), sondern kann auch für Schemas und zur Auswahl von Werten in JSON-Dokumenten verwendet werden (◑ für *Sch*; ◑ für *Filt*).

Tabelle 8.1: Bewertung von JSON-Technologien.

Sprachkategorie	Beispiele	Sch	Filt	Log
Pfadsprachen	JSON Pointer	◑	◑	○
Anfragesprachen	JSONPath, MongoDB, XQuery	◑	●	◑
Schemasprachen	JSON Schema, JSON TypeDef	●	◑	○
Programmiersprachen	JavaScript, Python	◑	◑	◑
Templatesprachen	Jsonnet	◑	◑	●

JSON-Netze können mit einer nahezu beliebigen Kombination der oben beschriebenen JSON-Technologien umgesetzt werden. Für die in dieser Arbeit beschriebene Spezifikation von JSON-Netzen wird eine begründete Auswahl getroffen. Andere Kombinationsmöglichkeiten können gegebenenfalls in anknüpfenden Arbeiten untersucht werden. Wie in der Zielsetzung zur Spezifikation der JSON-Netze (siehe Unterabschnitt 8.1) beschrieben, soll bei der Auswahl der zur Implementierung verwendeten JSON-Technologien besonders darauf geachtet werden, dass diese möglichst programmiersprachenunabhängig sind. Weiterhin soll unnötige Komplexität für Anwendende vermieden werden, indem geeignete, aber in Bezug auf ihren Funktionsumfang möglichst einfache JSON-Technologien ausgewählt werden [Ray03]. Auf Basis dieser Kriterien werden die folgenden Technologien verwendet: (1) *JSON Pointer* [IET13] als programmiersprachenunabhängiger Standard für Pfadausdrücke, für die Fälle in denen lediglich eine deterministische Auswahl von Werten erforderlich ist. (2) *JSONPath* [Gös07] für die Formulierung von nicht-deterministischen *Filterausdrücken* zur Auswahl von Werten. JSONPath stellt dazu mächtige Fähigkeiten bereit, es fehlen aber Fähigkeiten zum Erzeugen und Manipulieren von JSON-Dokumenten (wie sie zum Beispiel bei XQuery oder in der Anfragesprache für die JSON-Datenbank MongoDB vorhanden sind) [Bou17]. Im Sinn der Komplexitätsreduktion ist JSONPath

daher den anderen Anfragesprachen vorzuziehen. Für JSONPath existieren Implementierungen in verschiedenen Programmiersprachen [Bur24] und ein Standard-Entwurf, der bisher aber noch nicht finalisiert wurde [IET24]. (3) Für *JSON Schema* [Wri22a, Wri22b] existiert ebenfalls ein Entwurf für einen programmiersprachenunabhängigen Standard. Die Schemasprache stellt umfangreiche Ausdrucksmöglichkeiten für *Schemas* zur Verfügung, ohne jedoch für das Manipulieren von JSON-Dokumenten geeignet zu sein, und ist daher ein guter Kandidat für den Einsatz in JSON-Netzen. (4) Die Templatesprache *Jsonnet* [Goo24c] kommt in JSON-Netzen dort zum Einsatz, wo JSON-Dokumente manipuliert und Zusammenhänge zwischen JSON-Dokumenten beschrieben werden (*logische Ausdrücke*). Jsonnet ist in Bezug auf die Ausdrucksmächtigkeit vergleichbar mit Programmiersprachen, ist dabei aber speziell auf das Manipulieren von JSON-Dokumenten ausgelegt. Weiterhin kann Jsonnet in verschiedenen Programmiersprachen-Kontexten eingesetzt werden [Goo24c].

8.3 JSON-Datenmodell

In diesem Abschnitt wird zunächst das JSON-Datenmodell mit seinen Datentypen vorgestellt (Unterabschnitt 8.3.1) und dann die Baumstruktur von JSON-Dokumenten erläutert (Unterabschnitt 8.3.2). Dabei wird auch die Bedeutung der Begriffe *JSON-Dokument* und *Wert* im Kontext von JSON-Netzen präzisiert.

8.3.1 JSON-Dokumente

In einem JSON-Netz werden JSON-Dokumente zur Markierung einer Netzstruktur verwendet. In diesem Unterabschnitt werden daher die Eigenschaften des Datenmodells JavaScript Object Notation[1] (JSON) nach den zugrundeliegenden Standards [IET17, ECM17] erläutert. Die genannten Standards beschreiben JSON als leichtgewichtiges, textbasiertes Datenmodell, das insbesondere zum Austausch von Daten zwischen verschiedenen Systemen geeignet ist. Die Leichtgewichtigkeit wird dadurch erreicht, dass nur wenige Formatierungsregeln für die Repräsentation von Daten festgelegt werden. Der Austausch von Daten zwischen verschiedenen Systemen wird dadurch erleichtert, dass sich für die von JSON vorgegebenen Datentypen in

[1] Der Name rührt daher, dass die Notation von der Objekt-Syntax der Programmiersprache JavaScript abgeleitet wurde [ECM17, ECM22].

vielen gängigen Programmiersprachen Äquivalente finden [Dro22]. So können JSON-Dokumente in unterschiedlichen Systemen einfach erzeugt und interpretiert werden können. Im JSON-Datenmodell gibt es Werte von einfachem Datentyp (String, Number, Boolean und der Wert Null) und Werte von komplexem Datentyp (Object und Array). Zur Abgrenzung von allgemeinen Konzepten in dieser Arbeit, wie Objekte, Zeichenketten oder Listen, werden im Folgenden, wenn explizit JSON-Datentypen gemeint sind, die hier vorgestellten englischen Begriffe verwendet.

Einfache Datentypen: Zu den einfachen Datentypen zählen *String*, *Number*, *Boolean* und der Wert *Null*. Ein *String* beginnt und endet mit Anführungsstrichen und kann eine beliebige Zeichenfolge enthalten. Zahlenwerte werden mit dem Datentyp *Number* dargestellt. Ein Number-Wert enthält immer eine Ganzzahl ohne führende Nullen. Optional kann ein Minuszeichen vorangestellt werden, ein Punkt und eine Zahlenfolge folgen, oder ein Exponent (ein `e` oder `E` gefolgt von einer Zahlenfolge mit optionalem Plus- oder Minuszeichen) folgen. Boolesche Werte, beziehungsweise der Wert Null werden über die Zeichenfolgen `true`, `false` (für den Datentyp *Boolean*) oder die Zeichenfolge `null` dargestellt.

Komplexe Datentypen: Objects und *Arrays* sind die zwei komplexen Datentypen im JSON-Datenmodell. Ein *Object* wird durch ein Paar geschweifter Klammern `{}` dargestellt, das beliebig viele Schlüssel/Wert-Paare umschließt.[1] Schlüssel und Wert werden von einem Doppelpunkt getrennt, wobei der Schlüssel immer vom Datentyp String ist. Schlüssel/Wert-Paare werden voneinander mit Kommas getrennt. Innerhalb eines Objects müssen Schlüssel immer eindeutig sein. Ein *Array* wird durch ein Paar eckiger Klammern `[]` dargestellt und enthält beliebig viele, mit Kommas getrennte Werte. Werte in Arrays oder Objects müssen nicht von gleichem Typ sein.

Der Begriff *JSON-Dokument* meint im Folgenden ein serialisiertes Datenobjekt, das gültig zu den Syntaxregeln des JSON-Datenmodells ist und zwischen Systemen über JSON-fähige Schnittstellen ausgetauscht werden kann. Das kann ein Wert eines komplexen Datentyps sein, also ein Object oder Array. Da jeder Wert in einem Array oder Object selbst wieder ein Array oder Object sein kann, können JSON-Dokumenten tief verschachtelt sein. Ein JSON-Dokument kann aber auch ein einfacher Wert sein, wie die Zeichenketten `true`, `42` oder `"Hello world!"`.[2]

[1] Im Folgenden wird für Schlüssel/Wert-Paare auch der Begriff *Feld* verwendet.

[2] In früheren Versionen des JSON-Standards war vorgeschrieben, dass ein serialisiertes Datenobjekt immer ein Object oder ein Array sein muss. In der aktuellen Spezifikation gilt diese Anforderung nicht mehr [IET17].

```
1  {
2    "name": {
3      "type": "PC Mouse",
4      "model": "Fast Clicker X3000"
5    },
6    "price": 10.99,
7    "colors": ["blue", "green"]
8  }
```

Auflistung 8.1: Beispiel für ein JSON-Dokument.

Auflistung 8.1 zeigt ein Beispiel für ein JSON-Dokument, das eine Produktbeschreibung für eine Computer-Maus darstellt. Das JSON-Dokument ist ein Object, welches über drei Felder mit den Schlüsseln `"name"`, `"price"` und `"colors"` verfügt. Der Wert zu `"name"` ist selbst ein Object. Dieses Object speichert zwei Strings: im Feld mit dem Schlüssel `"type"` die Typbezeichnung und im Feld mit dem Schlüssel `"model"` die Modellbezeichnung des Produkts. Das Feld mit dem Schlüssel `"price"` enthält den Preis der Computer-Maus als Number und unter dem Schlüssel `"colors"` werden verfügbare Farben als Strings in einem Array angegeben.

8.3.2 JSON-Bäume und Werte

Für die folgende Definition von Operationen auf JSON-Dokumenten ist es hilfreich JSON-Dokumente als Baum, das bedeutet als eine Menge von Knoten in einer Baumstruktur [Bou20], zu interpretieren. In Anlehnung an [Bou20] wird für ein JSON-Dokument d mit $j = tree(d)$ der zugehörige JSON-Baum notiert.[1] Die Knoten des Baums bilden dabei die Werte eines JSON-Dokuments und die Kanten entsprechen den Schlüsseln der Schlüssel/Wert-Paare eines Objects, beziehungsweise den Indizes eines Arrays. Daraus ergibt sich, dass die Blätter des Baums immer von einem Wert eines einfachen Datentyps oder von einem leeren Object beziehungsweise Array gebildet werden. Alle anderen Knoten des Baums werden von Object- oder Array-Werten gebildet. Mit einem *Wert* ist im Folgenden ein Teilbaum eines JSON-Baums gemeint (das kann auch der komplette JSON-Baum beziehungsweise das JSON-Dokument selbst sein). Jede Wert in einem JSON-Dokument, sei es ein Wert eines einfachen Datentyps oder eines komplexen Datentyps, kann also selbst als JSON-Dokument interpretiert

[1] [Bou20] stellt eine formale Definition für JSON-Bäume bereit. Für die folgenden Ausführungen genügt die Feststellung, dass ein JSON-Dokument als Baum dargestellt werden kann.

werden. Für einen Knoten $n \in V(j)$ wird daher mit $json(n)$ der zugehörige Wert beziehungsweise das zugehörige JSON-Dokument notiert.[1] Der Wurzelknoten eines JSON-Baums j wird mit $root(j)$ notiert. Da eine Umwandlung von einem JSON-Dokument in einen JSON-Baum (und umgekehrt) immer möglich ist,[2] wird im Folgenden nicht streng zwischen JSON-Dokumenten und JSON-Bäumen unterschieden.

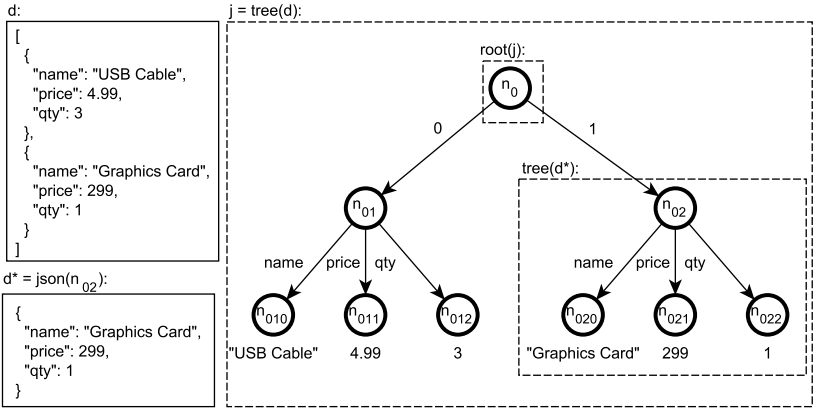


Abbildung 8.3: Abbildung von JSON-Dokumenten auf JSON-Bäume.

Abbildung 8.3 zeigt ein JSON-Dokument d, das aus einem Array mit zwei Objects besteht. Die Objects stellen in diesem Fall Auftragspositionen in einer Bestellung dar. Jedes der Objects besteht aus drei Feldern mit den Schlüsseln `"name"` für den Namen des bestellten Produkts, `"price"` für die Angabe des Preises und `"qty"` für die jeweilige Bestellmenge. Rechts daneben ist der zugehörige JSON-Baum $j = tree(d)$ abgebildet. Der Wurzelknoten des Baums $root(j) = n_0$ entspricht dem umschließenden Array des JSON-Dokuments. Die zwei Kindknoten von n_0, n_{01} und n_{02}, entsprechen den Object-Elementen des Arrays und die verbindenden Kanten sind mit den jeweiligen Array-Indizes 0 beziehungsweise 1 beschriftet. Die Kindknoten von n_{01} und n_{02} entsprechen den Werten der Schlüssel/Wert-Paare der jeweiligen Objekte, und die verbindenden Kanten sind mit den zugehörigen Schlüsseln beschriftet. Diese Werte sind jeweils von

[1] $V(j)$ ist die Menge der Knoten des Baums j.
[2] Es gilt für ein JSON-Dokument d und einen JSON-Baum $j = tree(d)$: $d = json(root(j))$.

einfachem Datentyp und bilden somit die Blätter des JSON-Baums.[1] Die Abbildung zeigt weiterhin das JSON-Dokument $d^* = json(n_{02})$. Das JSON-Dokument d^* ist ein Wert in d, nämlich das Object an Position 1 des umschließenden Arrays. Es entspricht dem Teilbaum von j mit Wurzel in n_{02}.

8.4 Beschreibung der Teilkonzepte

In diesem Abschnitt werden zunächst die in der Zielsetzung zum Sprachentwurf in Abschnitt 8.1 eingeführten Teilkonzepte von JSON-Netzen spezifiziert (siehe Unterabschnitt 8.4.1 für Filter, Unterabschnitt 8.4.2 für Logische Ausdrücke und Unterabschnitt 8.4.3 für Schemas). Für jedes Teilkonzept wird eine formale Beschreibung gegeben und dann jeweils in einem Exkurs gezeigt, wie es sich mit einer entsprechenden JSON-Technologie umsetzen lässt. Abschließend werden in Unterabschnitt 8.4.4 die zum Löschen und Einfügen von Werten in JSON-Dokumenten benötigten Operationen vorgestellt.

8.4.1 Filter (JSON Pointer und JSONPath)

Filter erfüllen in JSON-Netzen die Funktion, Werte in JSON-Dokumenten zur Bearbeitung auszuwählen. Dazu wird in diesem Unterabschnitt zunächst der einfache Fall, das Auswählen genau eines Werts in einem JSON-Dokument mit JSON Pointer-Pfadausdrücken, vorgestellt. Dabei wird auch auf die Partitionierung und Konkatenation, sowie auf die Konfliktfreiheit und Ordnung von Pfadausdrücken eingegangen. Diese Konzepte werden in Unterabschnitt 8.4.4 zur Definition von Operationen auf JSON-Dokumenten und in Abschnitt 8.5 zur Definition der Schaltregel benötigt. Schließlich werden mächtigere Ausdrucksmöglichkeiten zur Auswahl mehrerer Werte in einem JSON-Dokument mit JSONPath-Filterausdrücken vorgestellt.

Pfadausdrücke mit JSON Pointer

Eine Basisfunktion für alle JSON-Technologien ist die Möglichkeit, in einem verschachtelten JSON-Dokument iterativ ausgehend von einem Object-Knoten den Wert unter einem bestimmten Schlüssel, beziehungsweise ausgehend von einem

[1] Neben Werten von einfachem Datentyp können auch leere Objects und leere Arrays die Blätter eines JSON-Baums bilden.

Array-Knoten den Wert unter einem bestimmten Index anzusprechen [Bou20]. Die Aneinanderreihung von Schlüsseln und Indizes bis zu einem bestimmten Wert in einem verschachtelten JSON-Dokument beschreibt einen Pfad im zugehörigen JSON-Baum. Ausdrücke, die eindeutig einen Wert in einem JSON-Dokument (einen Knoten in einem JSON-Baum) identifizieren, werden daher im Folgenden *Pfadausdrücke* genannt. In unterschiedlichen JSON-Technologien ist für Pfadausdrücke die Notation mit eckigen Klammern `[]`, Schrägstrichen `/` oder Punkten `.` üblich [Bou20, IET13]. Mit JSON Pointer existiert ein programmiersprachenunabhängiger Standard für derartige Pfadausdrücke, der im Folgenden verwendet wird. Die Syntax von JSON Pointer ist im Exkurs 8.4.1 genauer beschrieben. Allgemein wird mit $w(d)$ der vom Pfadausdruck w ausgewählte Wert im JSON-Dokument d, beziehungsweise der Knoten und Teilbaum unter dem entsprechenden Pfad notiert. Für das Beispiel in Abbildung 8.3 wählt also der JSON Pointer-Pfadausdruck $w_1 =$ `"/0/name"` den Wert `"USB Cable"` aus. Weiterhin wird für ein gegebenes JSON-Dokument d und einen Pfadausdruck w mit $d \vDash w$ notiert, dass das JSON-Dokument zum Pfadausdruck gültig ist, was bedeutet, dass der entsprechende Pfad im JSON-Baum $tree(d)$ vorhanden ist. Für das gegebene Beispiel gilt also $d \vDash w_1$. Für einen Pfadausdruck $w_2 =$ `"/2/name"` gilt jedoch $d \nvDash w_2$, da der entsprechende Pfad im zugehörigen JSON-Baum nicht vorhanden ist.

Exkurs 8.4.1: JSON Pointer

Die Pfadsprache JSON Pointer wird im Standard [IET13] spezifiziert. Dabei werden unter anderem auch Konventionen und Regeln für Ausnahmefälle und Sonderzeichen beschrieben. An dieser Stelle erden lediglich die für das Verständnis der in dieser Arbeit vorliegenden Beispiele wichtigen Prinzipien des Standards wiedergegeben.

Ein JSON Pointer-Pfadausdruck ist ein JSON-String, der einen bestimmten Wert in einem JSON-Dokument eindeutig identifiziert. Der Ausdruck besteht aus einer beliebigen (endlichen) Anzahl von Teil-Zeichenketten, denen jeweils das Trennzeichen `/` vorangestellt wird. Die Auswertung eines JSON Pointer-Pfadausdruck erfolgt sequentiell für jede Teil-Zeichenkette, wobei zunächst die Wurzel eines auszuwertenden JSON-Dokuments referenziert wird. In der folgenden Liste werden die Regeln für die Auswertung beschrieben und Beispiele unter Verwendung des JSON-Dokuments d aus Abbildung 8.3 gegeben.

Falls es sich beim aktuell referenzierten Wert um ein Array handelt, entspricht die Zeichenkette dem Index des auszuwählenden Array-Elements.

Beispiel: Die Anwendung des JSON-Pointer-Ausdrucks `"/1"` auf d gibt das JSON-Dokument d^* (das Object, das die Grafikkarte repräsentiert) aus.

Falls es sich beim aktuell referenzierten Wert um ein Array handelt, und die Zeichenkette nicht als Number interpretiert werden kann, oder der Index nicht im Array vorhanden ist, kann kein Wert ausgewählt werden.

Beispiel: Der Pfadausdruck `"/name"` kann für d keinen Wert auswählen.

Falls es sich beim aktuell referenzierten Wert um ein Object handelt, entspricht die Zeichenkette dem Schlüssel des auszuwählenden Schlüssel/Wert-Paares.

Beispiel: `"/1/name"` wählt für d zunächst das Object unter Index `1` aus und dann den Wert unter dem Schlüssel `"name"`. Das Ergebnis ist der String `"Graphics Card"`.

Falls es sich beim aktuell referenzierten Wert um ein Object handelt, und kein Schlüssel/Wert-Paar mit einem entsprechenden Schlüssel vorhanden ist, kann kein Wert ausgewählt werden.

Beispiel: Für d kann `"/1/type"` keinen Wert auswählen.

Einen Sonderfall bildet das Zeichen -. Falls es sich beim aktuell referenzierten Wert um ein Array handelt, wird mit diesem Zeichen das (bisher nicht existierende) Element nach dem letzten Array-Element ausgewählt. Diese Ausdrucksmöglichkeit kann verwendet werden, um ein Element an das Ende eines Arrays anzuhängen.

Beispiel: Mit der Referenz `"/-"` könnte ein weiterer Wert an das Ende des Arrays in d angehängt werden, ohne dass die Länge des Arrays bekannt sein muss.

Falls der JSON Pointer-Ausdruck leer ist (`""`), wird das ganze JSON-Dokument ausgegeben.

Zu beachten ist, dass eine als Number interpretierbare Zeichenkette (zum Beispiel `"/1"`) sowohl ein Element mit Index `1` in einem Array als auch ein Schlüssel/Wert-Paare in einem Object mit Schlüssel `"1"` auswählen kann. Die Entscheidung wird erst mit der Anwendung des JSON Pointer-Ausdrucks auf ein JSON-Dokument getroffen. Auflistung 8.2 zeigt ein Beispiel für ein syntaktisch korrektes JSON-Dokument, das alternativ zum JSON-Dokument d in

Abbildung 8.3 Auftragspositionen in einem Object statt in einem Array speichert. Der JSON Pointer-Ausdruck `/1/name` erzeugt für beide JSON-Dokumente die Ausgabe `"Graphics Card"`.

```
{
  "1": {
    "name": "Graphics Card",
    "price": 299,
    "qty": 1
  },
  "0": {
    "name": "USB Cable",
    "price": 4.99,
    "qty": 3
  }
}
```

Auflistung 8.2: Object mit als Number interpretierbaren Schlüsseln.

Partitionierung und Konkatenation von Pfadausdrücken

Für die in Abschnitt 8.5 beschriebene Schaltregel von JSON-Netzen ist es hilfreich, für einen von einem Pfadausdruck ausgewählten Wert das zugehörige Eltern-Objekt oder -Array und den zugehörigen Schlüssel beziehungsweise den zugehörigen Array-Index zu ermitteln. Weiterhin sollen Kinder eines Wurzel-Objekts oder Wurzel-Arrays auswählbar sein. Die genannten Funktionalitäten können über die Partitionierung von Pfadausdrücken erreicht werden. Für einen Pfadausdruck w und ein JSON-Dokument d wird mit $\mathit{parent}(w)$ der Ausdruck notiert, der den Elternknoten von w (d) auswählt. Das entspricht dem Abschneiden der letzten Teil-Zeichenkette von w. Mit $\mathit{key}(w)$ wird die letzte Teil-Zeichenkette notiert, um den Schlüssel oder Array-Index eines vom Pfadausdruck w ausgewählten Werts zu ermitteln. Für den JSON Pointer-Pfadausdruck $w =$ `"/0/price"` ergibt sich also $\mathit{parent}(w) =$ `"/0"` und $\mathit{key}(w) =$ `"price"`. Zur Ermittlung des umschließenden Objects oder Arrays eines ausgewählten JSON-Werts wird

mit $env(w)$ die erste Teil-Zeichenkette eines Pfadausdrucks ausgewählt. Da der Pfadausdruck w nur aus zwei Teil-Zeichenketten besteht, gilt hier $env(w) = parent(w) =$ `"/0"`.[1]

Wie in Exkurs 8.4.1 beschrieben, kann die aus der Operation $key()$ resultierende Zeichenkette, beziehungsweise der Name einer Kante im JSON-Baum, als String und gegebenenfalls auch als Number interpretiert werden. Die Interpretation ist abhängig vom JSON-Dokument, auf den der Pfadausdruck ausgeführt wird. Das bedeutet, falls $parent(w)(d)$ ein Object ist, entspricht $key(w)$ dem zugehörigen Schlüssel eines Schlüssel/Wert-Paares und falls $parent(w)(d)$ ein Array ist und $key(w)$ als ganzzahlige Number interpretiert werden kann, dem Index eines Elements.

Falls ein Pfadausdruck nur aus einem Kantennamen besteht, wählt $parent(w)$ die Wurzel eines JSON-Dokuments aus, das bedeutet für einen JSON Pointer-Pfadausdruck $w =$ `"/0"` ergäbe $parent(w)$ den leeren JSON Pointer-Pfadausdruck $parent(w) =$ `""`. $env(w)$ wählt in diesem Fall weiterhin die erste Kante des Pfadausdrucks aus und es gilt $env(w) = w =$ `"/0"`. Für die Partitionierung des leeren JSON Pointer-Pfadausdrucks $w =$ `""` ergibt $parent(w)$, $key(w)$ und $env(w)$ den leeren Pfadausdruck: $parent(w) = key(w) = env(w) =$ `""`.

Da Pfadausdrücke lediglich Zeichenketten sind, können sie zusammengefügt werden, indem die Zeichenketten konkateniert werden. Im Folgenden wird die Konkatenation von zwei Pfadausdrücken w und v mit der Punktnotation `w.v` notiert. Bei der Konkatenation von JSON Pointer-Pfadausdrücken wird gegebenenfalls der führende Schrägstrich für eine Teil-Zeichenkette ergänzt. Es gilt also immer $w = parent(w).key(w)$.

Konfliktfreiheit und Ordnung von Pfadausdrücken

In Fällen, in denen auf einem JSON-Dokument mehrere Operationen gleichzeitig stattfinden (siehe dazu die Definition der Schaltregel für JSON-Netze in Unterabschnitt 8.5), können Konflikte auftreten, wenn diese auf bestimmten Werten (Teilbäumen) eines JSON-Dokuments stattfinden. Die Konzepte Konfliktfreiheit (Definition 8.4.1) und Ordnung (Definition 8.4.2) von Pfadausdrücken werden eingeführt, um diese Konflikte aufzulösen. Von der Konfliktfreiheit zwischen zwei Pfadausdrücken ist abhängig, ob zwei Operationen beim Schalten einer Transition ausgeführt werden können.

[1] Mit der Operation $env(w)$ sollen Objects und Arrays ausgewählt werden, die gegebenenfalls tiefer liegende Werte umschließen (siehe Abschnitt 8.5). Daher die Notation mit „env“ für englisch „environment“.

Nach Definition 8.4.1 wären die JSON Pointer-Pfadausdrücke w_1 = `"/items/0"`, w_2 = `"/items/1"` und w_3 = `"/name/model"` jeweils paarweise konfliktfrei. Die Pfadausdrücke w_1 und w_2 stünden aber mit einem Pfadausdruck v = `"/items"` in Konflikt.

Definition 8.4.1: Konfliktfreiheit

Zwischen zwei Pfadausdrücken v und w liegt ein Konflikt vor, wenn ein Pfadausdruck einen Teilpfad des anderen darstellt. Ein Konflikt liegt also dann vor, wenn eine der beiden folgenden Bedingungen erfüllt ist:

- $\exists x$, *sodass* $v.x = w$
- $\exists x$, *sodass* $w.x = v$

Andernfalls heißen v und w *konfliktfrei.*

Die Ordnung zweier Pfadausdrücke (siehe Definition 8.4.2) entscheidet in JSON-Netzen über die Ausführungsreihenfolge von Operationen auf den von den Pfadausdrücken ausgewählten Werten. Wie oben bereits erwähnt, ist aus einem Pfadausdruck allein nicht in jedem Fall erkennbar, ob ein Wert in einem Array oder in einem Object ausgewählt wird. Daher ist die Ordnung von Pfadausdrücken auch abhängig vom JSON-Dokument, auf das die Pfadausdrücke angewendet werden. Nach Definition 8.4.2 entspricht die Tiefe eines JSON Pointer-Pfadausdrucks der Anzahl der mit Schrägstrichen / getrennten Teil-Zeichenketten. Für die genannten Pfadausdrücke w_1, w_2 und w_3 gilt also $depth(w_1) = depth(w_2) = depth(w_3) = 2$. Für zwei Dokumente d_1 (entsprechend dem Beispiel in Abbildung 8.3) und d_2 (entsprechend dem Beispiel in Auflistung 8.2) und die Pfadausdrücke w_1 = `"/0/name"`, w_2 = `"/1/name"`, v = `"/0"`, gilt also $w_2 \succ_{d_1} w_1$ und $w_2 \sim_{d_2} w_1$. In jedem Fall gilt $w_1 \succ v$ und $w_2 \succ v$.

Definition 8.4.2: Ordnung

Die Ordnung von Pfadausdrücken ist durch ihre Tiefe, sowie gegebenenfalls durch die Position eines ausgewählten Werts in seinem Eltern-Array bestimmt. Die Tiefe eines Pfadausdrucks w, notiert mit $depth(w)$, entspricht der Anzahl seiner Teil-Zeichenketten (und damit auch der Anzahl der Kanten des zugehörigen Pfads in einem JSON-Baum). Die Ordnung von Pfadausdrücken wird wie folgt bestimmt:

(1) Falls zwei Pfadausdrücke w und v gleich sind, haben sie auch die gleiche Ordnung, notiert mit $w \sim v$: falls $v = w$, gilt $v \sim w$

(2) Ein Ausdruck w mit höherer Tiefe als ein Ausdruck v ist auch von höherer Ordnung, notiert mit $w \succ v$: falls $depth(w) > depth(v)$, gilt $w \succ v$.

(3) Falls zwei Ausdrücke w und v die gleiche Tiefe haben, aber Knoten mit unterschiedlichen Eltern ausgewählt werden, sind sie von gleicher Ordnung: falls $depth(w) = depth(v)$ und $parent(w) \neq parent(v)$ gilt $w \sim v$.

(4) Falls zwei Ausdrücke Knoten mit dem gleichen Elternknoten auswählen, aber nicht gleich sind, ist ihre Ordnung abhängig vom JSON-Dokument d, auf das sie angewendet werden. Diese Abhängigkeit wird durch eine entsprechende Indizierung an der Ordnungsbeziehung ($\succ_d$ und $\sim_d$) gekennzeichnet.

(4a) Falls $parent(w)(d) = parent(v)(d)$ und $parent(w)(d)$ einen JSON-Wert vom Typ Object auswählt, gilt $w \sim_d v$.

(4b) Falls $parent(w)(d) = parent(v)(d)$ und $parent(w)(d)$ einen Array auswählt, ist die Ordnung abhängig von der Position des ausgewählten Elements im Array. Aus einem höheren Index im Array folgt dann eine höhere Ordnung des entsprechenden Pfadausdrucks: $key(w) > key(v) \Rightarrow w \succ_d v$ (und umgekehrt).

Filterausdrücke mit JSONPath

Mit Hilfe von in Anfragesprachen formulierten Ausdrücken können Werte in einem JSON-Dokument (Knoten eines JSON-Baums) ausgewählt werden [Bou20]. Derartige Formulierungen werden im Folgenden *Filterausdrücke* genannt. Während ein Pfadausdruck (siehe oben) keinen, oder genau einen Knoten eines JSON-Baums identifiziert (nämlich dann, wenn der Pfad im JSON-Baum vorhanden ist), kann ein Filterausdruck, je nachdem wie das JSON-Dokument auf den er angewandt wird strukturiert ist, keinen, einen oder mehrere Knoten auswählen. Allgemein formuliert ist ein Filterausdruck eine Funktion p: $\mathcal{D} \rightarrow \mathcal{P}(\mathcal{W})$, die ein JSON-Dokument $d \in \mathcal{D}$ auf eine Menge von Pfadausdrücken $W \in \mathcal{P}(\mathcal{W})$ abbildet. $\mathcal{D}$ ist dabei die Menge aller JSON-Dokumente und $\mathcal{P}(\mathcal{W})$ die Potenzmenge der Menge aller Pfadausdrücke $\mathcal{W}$. Die Menge der Pfadausdrücke, die ein Filterausdruck p für ein JSON-Dokument d generiert, wird mit $p(d)$ notiert. Weiterhin wird für einen Filterausdruck p und ein JSON-Dokument d mit $d \vDash p$ notiert, dass das JSON-Dokument zum Filterausdruck gültig ist (das heißt der Filterausdruck für das JSON-Dokument mindestens einen Pfadausdruck generiert). Es wird angenommen, dass ein JSON-Dokument gültig zu allen Pfadausdrücken ist, die ein Filterausdruck dafür generiert, $\forall w \in p(d)$: $d \vDash w$.

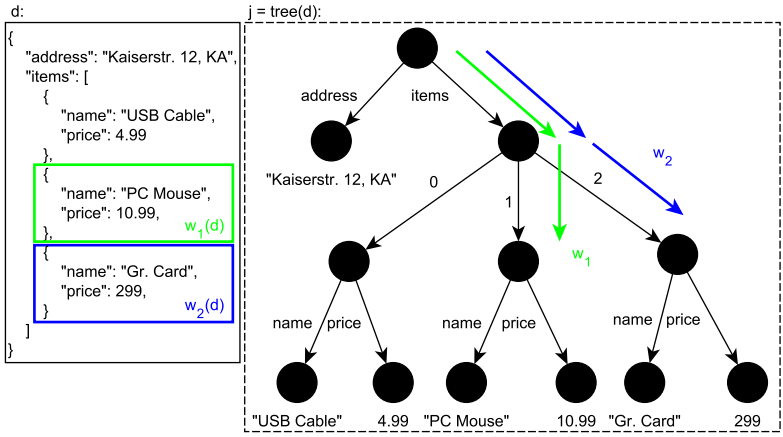


Abbildung 8.4: Anwendung eines Filterausdrucks auf ein JSON-Dokument.

Zur Umsetzung von Filterausdrücken wird in dieser Arbeit die Anfragesprache JSONPath verwendet. Abbildung 8.4 zeigt ein Beispiel für die Anwendung eines JSONPath-Filterausdrucks $p =$ `"$.items[?(@.price > 5)]"` auf ein JSON-Dokument d, beziehungsweise den zugehörigen JSON-Baum $j = tree(d)$. Die Anwendung des Filterausdrucks auf das JSON-Dokument $p(d) = \{w_1, w_2\}$ generiert die beiden Pfadausdrücke $w_1 =$ `"/items/1"` und $w_2 =$ `"/items/2"`, welche die unter dem `"items"`-Array gespeicherten Objects mit einem Wert für den Schlüssel `"price"` größer fünf auswählen. Die Syntax von JSONPath wird in Exkurs 8.4.2 ausführlicher beschrieben.

Exkurs 8.4.2: JSONPath

JSONPath wurde in Anlehnung an den XML-Standard XPath [W3C17a] als leichtgewichtige und effiziente Anfragesprache für das JSON-Datenmodell vorgeschlagen [Gös07]. Inzwischen existieren eine Vielzahl von Implementierungen der vorgeschlagenen Syntax [Bur24] und es gibt Bestrebungen, die Sprache zu standardisieren [IET24]. An dieser Stelle werden auf Basis von [Gös07] die für das Verständnis der Beispiele in dieser Arbeit benötigten syntaktischen Regeln und Ausdrucksmöglichkeiten erläutert.

Ein JSONPath-Ausdruck besteht, vergleichbar mit JSON Pointer-Ausdrücken, aus einer Aneinanderreihung von Teil-Zeichenketten. Im Unterschied zu JSON Pointer-Ausdrücken werden in JSONPath die Teil-Zeichenketten nicht

mit Schrägstrichen, sondern unter Verwendung der Punkt- oder Klammernotation verknüpft. Das Zeichen `$` referenziert dabei die Wurzel eines JSON-Dokuments.

Beispiel: Der JSONPath-Ausdruck `$.hobbies.0` entspricht dem JSON Pointer-Ausdruck `/hobbies/0`.

Das Platzhalter-Zeichen `*` wählt Werte unabhängig von ihrem Namen im Object oder ihrer Position im Array aus.

Beispiel: Für das JSON-Dokument in Abbildung 8.4 wählt der Filterausdruck `$.items.*` alle drei Bestellpositionen (alle drei Objects im Array unter dem Schlüssel `"items"`) aus.

Über die Verknüpfung einer Teil-Zeichenkette mit einer Filteroperation `[?(<expr>)]` kann die Menge der ausgewählten Kindknoten eingeschränkt werden. Innerhalb von `<expr>` wird der aktuelle Kontextknoten mit dem Zeichen `@` angesprochen.

Beispiel: Für das JSON-Dokument in Abbildung 8.4 wählt der Filterausdruck `$.items[?(@.price > 100)]` lediglich die Grafikkarte aus, da diese einen Preis größer 100 hat.

8.4.2 Logische Ausdrücke (Jsonnet)

Mit einem logischen Ausdruck ist eine Funktion gemeint, die einen booleschen Wert, `true` oder `false`, liefert [Gen81]. Im Kontext von JSON-Netzen werden logische Ausdrücke für Transitionsbeschriftungen (siehe Unterabschnitt 8.5) verwendet. Diese Transitionsbeschriftungen beschreiben Zusammenhänge zwischen verschiedenen JSON-Dokumenten und sollen auch die Formulierung komplexer Regeln zur Manipulation von JSON-Dokumenten ermöglichen. Im Folgenden wird zur Formulierung von logischen Ausdrücken daher die Templatesprache Jsonnet verwendet. Auflistung 8.3 gibt ein Beispiel für einen in Jsonnet formulierten logischen Ausdruck. Im Beispiel werden zunächst zwei Variablen `a` und `b` mit Werten belegt. Schließlich findet ein Vergleich zwischen dem Number-Wert der Variablen `b` und dem Wert des Object-Feldes `"myValue"` des in der Variablen `a` gespeicherten Objects statt. Das Ergebnis des Vergleichs, und damit die Ausgabe des Jsonnet-Ausdrucks, ist der boolesche Wert `true`. In Exkurs 8.4.3 wird die Syntax von Jsonnet ausführlicher beschrieben.

```
1  local a = { "myValue": 100 };
2  local b = 50;
3
4  b < a.myValue
```

Auflistung 8.3: Logischer Ausdruck in der Templatesprache Jsonnet.

Exkurs 8.4.3: Jsonnet

Jsonnet ist eine Templatesprache, die dazu dient Regeln zur Erzeugung von JSON-Dokumenten zu beschreiben. Sie wurde als Erweiterung von JSON spezifiziert, bietet unter Anderem Möglichkeiten zur Definition von Variablen und Funktionen und erlaubt die Verwendung von arithmetischen und logischen Operatoren. In diesem Exkurs werden auf Basis der Dokumentation [Goo24c] die für das Verständnis der Beispiele in dieser Arbeit wichtigen Sprachkonzepte erläutert.

Jedes JSON-Dokument ist ein gültiger Jsonnet-Ausdruck. In Jsonnet können JSON-Dokumente darüber hinaus um arithmetische oder logische Operationen erweitert werden.

Beispiel: Auflistung 8.4 zeigt ein Beispiel für einen Jsonnet-Ausdruck, der arithmetische und logische Operatoren verwendet. Bei der Auswertung des Jsonnet-Ausdrucks werden die entsprechenden Operationen durchgeführt und daraus ein JSON-Dokument erzeugt. Im Beispiel ergibt die Auswertung der Addition `3 + 5` unter dem Feld `"computedNumber"` den Wert `8` und die Auswertung der logischen Oder-Verknüpfung `true || false` den Wert `true`.

```
1  // Jsonnet-Ausdruck:
2  { "computedNumber": 3 + 5, "computedBoolean": true || false }
3  // Auswertung:
4  { "computedNumber": 8, "computedBoolean": true }
```

Auflistung 8.4: Jsonnet-Ausdruck.

Weiterhin können in Jsonnet Werte in Variablen gespeichert werden und auf Werte in Objects und Arrays zugegriffen werden.

Beispiel: In Auflistung 8.5 werden mit dem Schlüsselwort `local` zwei Variablen deklariert und in der einen Variablen ein Object und in der anderen Variablen ein Array gespeichert. Schließlich wird in einem Object-Konstruktor ein Wert mit einer arithmetischen Operation erzeugt und dabei auf in den Variablen gespeicherte Werte zugegriffen. Die Auswertung ergibt ein Object mit einem Feld `"b"` . Der Wert des Feldes ergibt sich aus der Addition der Werte `5` (unter dem Schlüssel `"a"` in der Object-Variablen `objectVar`) und `3` (unter Index `0` der Array-Variablen `arrayVar`).

```
1  // Jsonnet-Ausdruck:
2  local objectVar = { "a": 5 };
3  local arrayVar = [3,8,4];
4  { "b": objectVar.a + arrayVar[0] }
5  // Auswertung:
6  { "b": 8 }
```

Auflistung 8.5: Jsonnet-Ausdruck mit Variablendeklaration.

Ein weiteres Sprachkonstrukt in Jsonnet sind Funktionen, die ebenfalls mit dem Schlüsselwort `local` deklariert werden.

Beispiel: Auflistung 8.6 zeigt ein Beispiel für einen Jsonnet-Ausdruck, in dem zunächst eine Funktion deklariert wird, die eine logische Oder-Operation mit zwei Parametern durchführt. Die Funktion wird dann in der zweiten Zeile aufgerufen und dabei zwei boolesche Werte übergeben. Die Auswertung ergibt in diesem Fall den einfachen Wert `true`.

```
1  // Jsonnet-Ausdruck:
2  local myFunction(x, y) = x || y;
3  myFunction(true, false)
4  // Auswertung:
5  true
```

Auflistung 8.6: Jsonnet-Ausdruck mit Funktionsdeklaration.

```
// JSON Schema p:
{
  "type": "object",
  "properties": {
    "studentId": { "type": "string" },
    "level": { "type": "string",  "enum": ["Bachelor", "Master"] },
    "studyProgram": { "type": "string" },
    "email": { "type": "string",  "format": "email" }
  }
}
// gültige Instanz d:
{
  "studentId": "student-1",
  "level": "Master",
  "studyProgram": "Information Systems",
  "email": "student@uni.edu"
}
```

Auflistung 8.7: Beispiel für ein JSON Schema und eine gültige Instanz.

8.4.3 Schemas (JSON Schema)

Mit Schemas können Regeln für die Werte und für die Struktur von JSON-Dokumenten beschrieben werden. Sie können damit zur Typisierung von JSON-Dokumenten eingesetzt werden. Für ein JSON-Dokument d und ein Schema p notiert $d \vDash p$, dass das JSON-Dokument zum Schema gültig ist (die Regeln des Schemas erfüllt). Mit JSON Schema [Wri22a, Wri22b] existiert ein Standard zur Erstellung von Schemas für das JSON-Datenmodell, der im Folgenden verwendet wird. JSON Schema-Regeln für die Struktur von JSON-Dokumenten werden selbst in einem JSON-Dokument gespeichert.

Auflistung 8.7 gibt ein Beispiel für ein Schema p und ein gültiges Dokument d. Der JSON Schema-Ausdruck beschreibt Regeln für ein JSON-Dokument vom Typ Object mit den Feldern `"studentId"`, `"level"`, `"studyProgram"` und `"email"`. Die Werte der einzelnen Felder müssen vom Typ String sein. Darüber hinaus wird für das Feld `"level"` festgelegt wird, dass es nur die Werte `"Bachelor"` oder `"Master"` haben darf. Für das Feld `"email"` muss der Wert eine E-Mail-Adresse sein. Im Beispiel hat das zum Schema gültige Dokument d im Feld `"studentId"` den Wert `"student-1"` und als Studiengang im Feld `"studyProgram"` den Wert `"Information Systems"`. Für das Feld `"level"` wurde mit `"Master"` einer der zugelassenen Werte ausgewählt und in das Feld `"email` wurde mit `"student@uni.edu"` eine gültige E-Mail-Adresse eingetragen. Exkurs 8.4.4 beschreibt die Syntax von JSON Schema näher.

Exkurs 8.4.4: JSON Schema

Im Folgenden werden auf Basis der Quellen [Wri22a, Wri22b, Dro22] die für das Verständnis der in dieser Arbeit verwendeten Beispiele wichtigen Prinzipien der Schemasprache JSON Schema wiedergegeben.

Eine grundlegende Ausdrucksmöglichkeit ist die Festlegung eines Werts auf einen bestimmten Datentyp. Dazu wird im Schema unter dem Feld `"type"` ein Datentyp festgelegt. Außerdem gibt es für die verschiedenen Datentypen unterschiedliche Schlüsselwörter wie `"maximum"` oder `"minimum"`, mit denen zulässige Werte weiter eingeschränkt werden können.

Beispiel: Auflistung 8.8 legt fest, dass ein JSON-Wert vom Typ Number sein muss und nicht größer als der Wert 100 sein darf.

```
1  // JSON Schema:
2  { "type": "number", "maximum": 100 }
3  // gültige instanz:
4  50
```

Auflistung 8.8: Beispiel für einen einfachen JSON Schema-Ausdruck.

Für Object-Werte können unter Verwendung des Schlüsselworts `"properties"` zulässige Schlüssel/Wert-Paare des Objects beschrieben werden.

Beispiel: Das JSON Schema in Auflistung 8.9 legt fest, dass ein JSON-Wert vom Typ Object sein muss, ein Feld mit Schlüssel `"myValue"` und einem Wert nicht größer als 100 haben darf.

```
1  // JSON Schema:
2  {
3    "type": "object",
4    "properties": { "myValue": { "type": "number",  "maximum": 100 } }
5  }
6  // gültige Instanz:
7  { "myValue": 50 }
```

Auflistung 8.9: Beispiel für ein Object-Schema.

Für Arrays können mit dem Schlüsselwort `"items"` gültige Werte für Elemente des Arrays definiert werden

Beispiel: Das JSON Schema in Auflistung 8.10 legt fest, dass nur Number-Werte als Elemente eines Arrays zulässig sind.

```
1  // JSON Schema:
2  { "type": "array", "items": { "type": "number" } }
3  // gültige Instanz:
4  [1, 5, 3, 100]
```

Auflistung 8.10: Beispiel für ein Array-Schema.

Über die Kombination von Array- und Object-Typdefinitionen können verschachtelte JSON Schema-Ausdrücke erstellt werden.

Beispiel: Das JSON-Schema in Auflistung 8.11 zeigt ein JSON Schema für ein JSON-Dokument, das aus einem Array besteht. Elemente des Arrays sollen Objects sein, die ein Feld `"myValue"` haben, entsprechend dem Schema in Auflistung 8.9.

```
1  // JSON Schema:
2  {
3    "type": "array",
4    "items": {
5      "type": "object",
6      "properties": { "myValue": { "type": "number", "maximum": 100 } }
7    }
8  }
9  // gültige Instanz:
10 [{ "myValue": 100 }, { "myValue": 50 }]
```

Auflistung 8.11: Beispiel für eine Kombination von Object und Array-Typen.

8.4.4 Operationen auf JSON-Daten

Nachdem das JSON-Datenmodell sowie grundlegende Konzepte für JSON-Netze (Filter, Logische Ausdrücke und Schemas) erläutert wurden, folgt nun die Vorstellung der Operationen, die beim Schalten in JSON-Netzen an Marken (JSON-Dokumenten) durchgeführt werden. Es wird zunächst der einfache Fall, das Entfernen und Einfügen von Werten in Objects und Arrays, vorgestellt und dann gezeigt, wie diese Operationen auch in verschachtelten JSON-Dokumenten durchgeführt werden können.

Entfernen und Einfügen in Objects und Arrays

Als Zwischenschritt, bevor das Einfügen und Entfernen von Werten in einem verschachtelten JSON-Dokument betrachtet werden, soll zunächst der einfache Fall, das Einfügen und Entfernen eines Werts in einem Object oder Array beschrieben werden. Dabei müssen die Spezifika des JSON-Datenmodells und die Unterschiede zwischen Objects und Arrays beachtet werden. Der wesentliche Unterschied zwischen Arrays und Objects im JSON-Datenmodell ist, dass Werte in Arrays geordnet sind, während Werte in einem Object keiner Ordnung unterliegen [IET17]. Der JSON-Standard selbst definiert keine Operationen auf Objects und Arrays. Es gibt für die beiden komplexen Datentypen aber Entsprechungen in gängigen Programmiersprachen, wie zum Beispiel die Typen `dict` und `list` in Python als Äquivalent von Objects beziehungsweise Arrays [Dro22], oder Objects und Arrays in der Programmiersprache JavaScript [ECM22]. In den Programmiersprachen selbst werden Operationen zum Einfügen und Entfernen von Elementen in diesen Datenstrukturen spezifiziert [ECM22].

In dieser Arbeit werden Operationen zum Einfügen und Entfernen von Werten in beziehungsweise aus Objects und Arrays programmiersprachenunabhängig unter Rückgriff auf grundlegende Datenstrukturen in der Informatik (Listen für Arrays und Dictionaries für Objects) definiert [Ott12]. Die Definition der Operationen trägt der bereits erwähnten Eigenheit vieler JSON-Technologien Rechnung, dass der Typ eines Werts nicht streng festgelegt ist, und ein Number-Wert bei Bedarf auch als String-Wert interpretiert werden kann (und umgekehrt). Zum Beispiel ist in einem JSON Pointer-Pfadausdruck wie `/items/2"` nicht erkennbar, ob das Element mit Index `2` eines Arrays ausgewählt werden soll, oder ein Wert unter dem Namen `"2"` in einem Object. Im ersten Fall wird die zweite Teil-Zeichenkette als Number und Index `2` in einem Array interpretiert und im zweiten Fall als String und Schlüssel `"2"` für ein Schlüssel/Wert-Paar in einem Object. Daher kann der Parameter k der beiden im Folgenden definierten Operationen $remove(k, y)$ und $insert(x, k, y)$ vom Typ String oder Number sein, beziehungsweise je nach Bedarf als String oder Number interpretiert werden. Der Parameter y kann jeweils vom Typ Array oder Object sein.

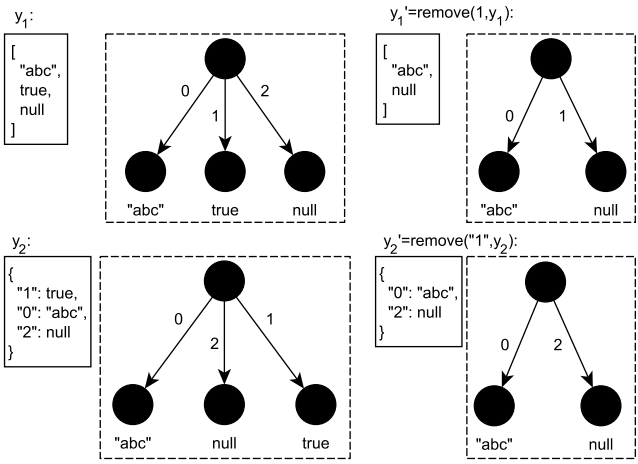


Abbildung 8.5: Entfernen eines Werts aus einem Object oder Array.

Definition 8.4.3 spezifiziert die Operation *remove*(k,y) zur Entfernung eines Werts unter dem Schlüssel oder Index k in einem Object oder Array y. Der Operation kann als Parameter k entweder ein String- oder ein Number-Wert übergeben werden und als Parameter y entweder ein Object oder ein Array. In der Abbildung 8.5 wird jeweils für das Entfernen eines Werts aus einem Object und einem Array ein Beispiel gegeben. Das Beispiel ist so konstruiert, dass das unterschiedliche Verhalten von Objects und Arrays anhand der zugehörigen JSON-Bäume deutlich wird. In den beiden JSON-Dokumenten y_1 und y_2 sind jeweils die gleichen Werte `"abc"`, `true` und `null` gespeichert. Im Array y_1 sind die Werte auch in dieser Reihenfolge an den Index-Positionen `0`, `1` und `2` abgelegt. Für y_2 werden als Schlüssel der Schlüssel/Wert-Paare die entsprechenden Index-Ziffern `"0"`, `"1"` und `"2"` als String-Wert verwendet. Wie in Abbildung 8.5 zu erkennen ist, haben die JSON-Bäume von y_1 und y_2 damit die gleiche Struktur. Mit dem Entfernen des Wertes `true` an Position `1` in y_1, beziehungsweise unter dem Schlüssel `"1"` in y_2 zeigt sich das unterschiedliche Verhalten von Objects und Arrays. Für Arrays verschieben sich potentiell die Positionen der verbliebenen Werte im Array. Im Beispiel ändert sich die Position des Werts `null` von `2` auf `1`. Für Objects bleiben, bis auf das entfernte Schlüssel/Wert-Paar, alle übrigen Schlüssel/Wert-Paare und damit auch die Schlüssel und Kantenbeschriftungen im JSON-Baum unverändert erhalten.

Definition 8.4.3: *remove (k, y)* [Ott12]

Die Operation $remove(k,y)$ entfernt einen Wert aus einem Object oder Array y. Es wird zwischen zwei Fällen unterschieden:

(1) Falls y ein Array ist, wird der Wert mit Index k entfernt. In allen Fällen, in denen k nicht als Index im Array interpretiert werden kann, wird das letzte Element aus dem Array entfernt. Die Operation verändert also ein Array y zu y' wie folgt:

$$y' = \begin{cases} [w_0, ..., w_{k-1}, w_{k+1}, ..., w_n] & \textit{falls} \quad y = [w_0, ..., w_{k-1}, w_k, w_{k+1}, ..., w_n] \quad \wedge \quad 0 \leq k \leq n \\ [w_0, ..., w_{n-1}] & \textit{falls} \quad y = [w_0, ..., w_n] \quad \wedge \quad k \textit{ nicht als Index vorh.} \\ [] & \textit{falls} \quad y = [] \end{cases}$$

(2) Falls y ein Object ist, wird das Schlüssel/Wert-Paar mit dem Schlüssel k (als String interpretiert) aus dem Object entfernt. Die Operation verändert also ein Object y zu y' mit $y' = y \setminus \{k : w\}$.

In Definition 8.4.4 wird die Operation $insert(x,k,y)$ zum Einfügen eines Werts x unter dem Schlüssel beziehungsweise Index k in ein Object oder Array y spezifiziert. Bei *insert*-Operationen kann es in JSON-Netzen zu Situationen kommen, in denen als Parameter y auch ein einfacher JSON-Wert ausgewählt wird. Daher wird dieser Fall in der Definition ebenfalls betrachtet.

Abbildung 8.6 zeigt Beispiele für das Einfügen eines Werts in ein Object oder Array. Die Beispiele sind so konstruiert, dass sie die Umkehrung der *remove*-Operationen in Abbildung 8.5 darstellen. Wieder zeigt sich das unterschiedliche Verhalten von Objects und Arrays anhand der dargestellten JSON-Bäume: im Array y_1 verändert sich der Index eines bereits gespeicherten Werts (und damit die entsprechende Kantenbeschriftungen), und in y_2 wird im JSON-Baum einfach eine weitere Kante hinzugefügt.

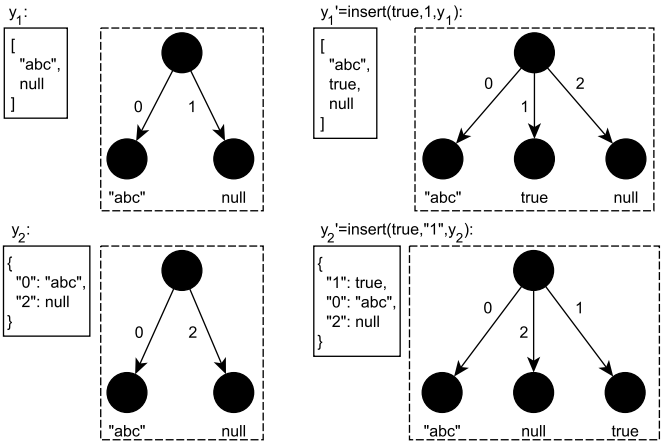


Abbildung 8.6: Einfügen eines Werts in ein Object oder Array.

Definition 8.4.4: *insert (x, k, y)* [Ott12]

Die Operation *insert*(x,k,y) fügt einen Wert x in ein Object oder ein Array y ein. Es wird zwischen drei Fällen unterschieden:

(1) Falls y ein Array ist, wird der Wert x an der Position k eingefügt. In allen Fällen, in denen k nicht als Index im Array interpretiert werden kann, wird x am Ende des Arrays eingefügt. Die Operation verändert also y zu y' wie folgt:

$$y' = \begin{cases} [w_0, ..., w_{k-1}, x, w_k, ..., w_n] & \textit{falls} \quad y = [w_0, ..., w_n] \\ & \quad \wedge \quad 0 \leq k \leq n \\ [w_0, ..., w_n, x] & \textit{falls} \quad y = [w_0, ..., w_n] \\ & \quad \wedge \quad k = n + 1 \\ [x] & \textit{falls} \quad y = [] \\ [w_1, ..., w_n, x] & \textit{sonst.} \end{cases}$$

(2) Falls y ein Object ist, wird das Schlüssel/Wert-Paar $k : x$ in die Menge der Schlüssel/Wert-Paare in y eingefügt. Falls k ein Number-Wert ist, wird er als String-Wert interpretiert und falls bereits ein Schlüssel/Wert-Paar mit Schlüssel k in y vorhanden ist, wird es mit $k : x$ ersetzt. Die Operation verändert also ein Object y zu y' wie folgt:

$$y' = \begin{cases} y \cup \{k : x\} & \textit{falls } k \textit{ nicht als Name in } y \textit{ vorhanden} \\ \textit{remove}(k,y) \cup \{k : x\} & \textit{sonst.} \end{cases}$$

(3) Ist y von einfachem Datentyp, wird er durch den Wert x ersetzt: $y' = x$

Entfernen und Einfügen in verschachtelten Objects und Arrays

Nachdem der einfache Fall, das Einfügen und Entfernen von Werten in ein Object oder Array, betrachtet wurde, sollen nun Werte in Objects oder Arrays eingefügt oder entfernt werden, die beliebig tief in JSON-Dokumenten verschachtelt sein können. Dazu werden die Operationen *deepInsert* und *deepRemove* über JSON-Bäume und Pfadausdrücke definiert. Die Operationen erlauben damit das Einfügen und Entfernen von Werten (Teilbäumen) in JSON-Bäumen. Sie bestehen im Wesentlichen aus dem Einfügen oder Entfernen eines Werts in einem Object oder Array, wobei diese Objects oder Arrays zunächst in komplexen JSON-Dokumenten identifiziert werden müssen.

Die Operation $\textit{deepInsert}(w,j)$ ist in Definition 8.4.5 spezifiziert. Wie in Abbildung 8.7 dargestellt, wird mit dem Pfadausdruck w der zu entfernende Wert, beziehungsweise der zu entfernende Teilbaum von j, ausgewählt. Im gegebenen Beispiel wäre das für den JSON Pointer-Pfadausdruck w = `"/items/1"` das Object an Position `1` im `"items"`-Array. Im gezeigten Fall ist an Position `1` das Object, das eine Auftragsposition für eine Computer-Maus (`"PC Mouse"`) beschreibt. Zur Entfernung des Werts wird am zugehörigen Eltern-Array $\textit{parent}(w)(j)$ die oben beschriebene Operation $\textit{remove}(1, \textit{parent}(w)(j))$ ausgeführt, um das Element mit Index $\textit{key}(w) = 1$ aus dem Array zu entfernen. Das Ergebnis ist der Array $\textit{parent}(w)(j')$ mit nur noch zwei Positionen.

In Definition 8.4.6 wird die Operation $\textit{deepInsert}(v, w, k, j)$ zum Einfügen eines Werts v in einen JSON-Baum j spezifiziert. Wie in Abbildung 8.8 beispielhaft dargestellt, wird mit dem Pfadausdruck w der Knoten $w(j)$ ausgewählt, an den der Wert angehängt werden soll. Das Beispiel ist so konstruiert, dass es die Umkehrung der in Abbildung 8.7 durchgeführten Operation darstellt. So wird für den Parameter w der JSON Pointer-Pfadausdruck w = `"/items"` verwendet und damit das Array unter dem Schlüssel `"items"` ausgewählt. Als Parameter k wird im Beispiel der Wert `1` verwendet, um anzugeben, dass der Wert an Position `1` im Array eingefügt werden soll. Dazu wird die oben beschriebene Operation *insert* auf dem Array $w(j)$ ausgeführt und so das Array $w(j')$ erzeugt, das nun 3 Elemente inklusive dem einzufügenden Wert v hat.

Definition 8.4.5: ***deepRemove (w, j)***

Die Operation $deepRemove(w,j)$ entfernt aus dem JSON-Baum j den durch den Pfadausdruck w gekennzeichneten Teilbaum $w(j)$. Sie verändert j zu j', sodass $parent(w)(j') = remove(key(w), parent(w)(j))$.

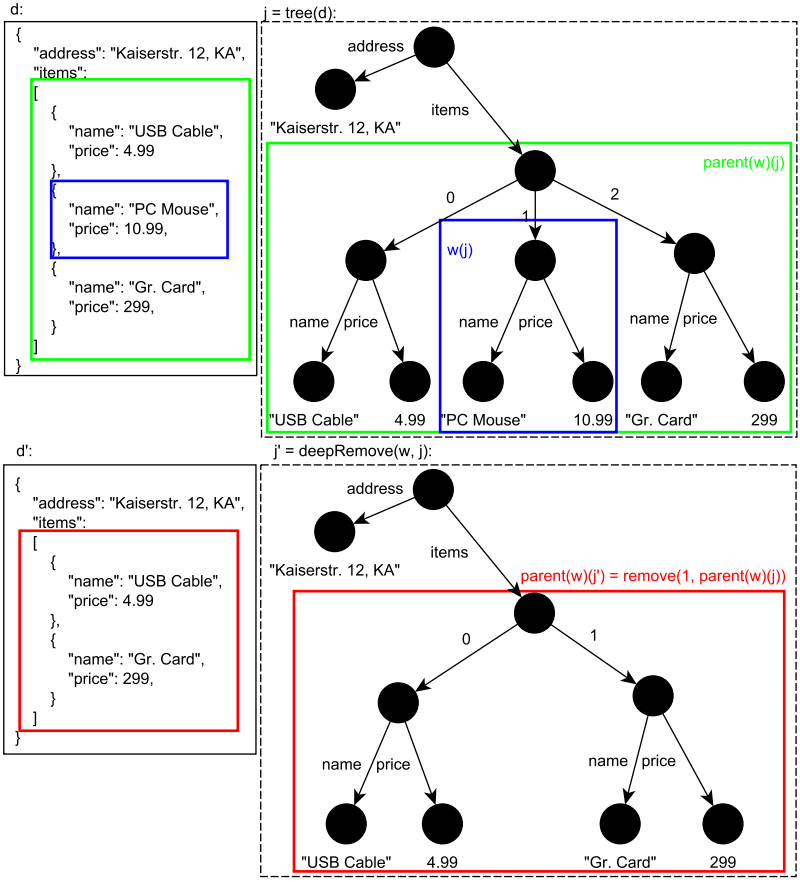


Abbildung 8.7: Entfernen eines Werts aus einem JSON-Dokument.

Definition 8.4.6: *deepInsert (v, w, k, j)*

Die Operation $deepInsert(v, w, k, j)$ fügt in einen JSON-Baum j einen Wert v unter dem Schlüssel oder Index k ein. Der Wert wird am Knoten $w(j)$ angehängt. Die Operation verändert also j zu j', sodass $w(j') = insert(v, k, w(j))$.

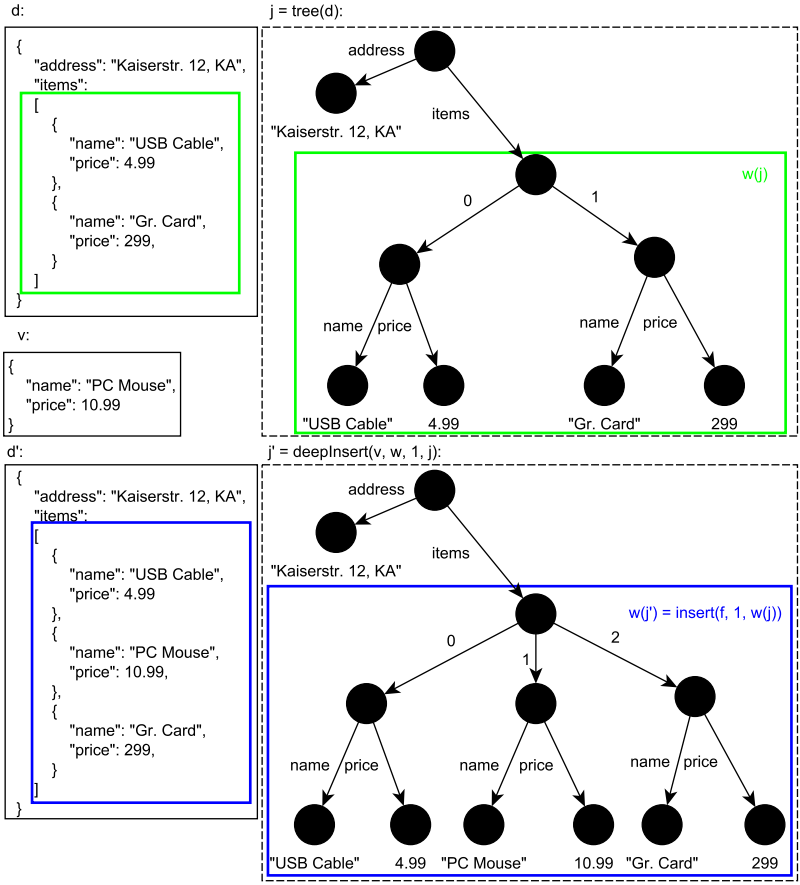


Abbildung 8.8: Einfügen eines Werts in ein JSON-Dokument.

8.5 Struktur und Schaltregel für JSON-Netze

Nachdem die grundlegenden Konzepte und Operationen für JSON-Netze vorgestellt wurden, wird im Folgenden die Struktur und die Schaltregel von JSON-Netzen definiert und anhand von Beispielen erläutert.

8.5.1 Struktur von JSON-Netzen

JSON-Netze sind eine Variante Höherer Petri-Netze, die es erlauben, den Fluss von Daten im JSON-Datenmodell (JSON-Dokumenten) zu modellieren. Vereinfacht zusammengefasst ist ein JSON-Netz ein Höheres Petri-Netz mit folgenden Eigenschaften:

- Marken werden in JSON-Netzen von JSON-Dokumenten dargestellt und Stellen sind Speicher von JSON-Dokumenten.
- Eine Markierung weist einer Stelle einen (auch leeren) Array von Werten (JSON-Dokumenten) zu.[1]
- Jede Stelle wird mit einem Schema typisiert, welches den Typ der JSON-Dokumente in der Markierung der Stelle festlegt. Die Markierung der Stelle muss also zum Schema der Stelle konform sein.
- Kanten werden mit einem Filtertupel, bestehend aus einem Filterausdruck, einer Schlüsselvariablen, einer Wertvariablen und einer Dokumentvariablen beschriftet. Für Kanten im Vorbereich einer Transition beschreiben diese Tupel, welche Werte aus einer Stellenmarkierung zu entfernen sind, und für Kanten im Nachbereich einer Transition beschreiben sie, welche Werte einzufügen sind.
- Filterausdrücke wählen gemeinsam mit Schlüsselvariablen Positionen zum Einfügen und Entfernen von Werten in den Markierungen der inzidenten Stellen aus.
- In Wertvariablen werden die zu entfernenden oder einzufügenden Werte gespeichert.

[1] Markierungen als Arrays von Werten zu definieren hat in JSON-Netzen den Vorteil, dass es die Schaltregel vereinfacht. Beim Entfernen und Einfügen muss auf diese Weise nicht zwischen ganzen JSON-Dokumenten (Marken) und Werten eines JSON-Dokuments (Substrukturen einer Marke) unterschieden werden, weil jede Operation technisch gesehen eine Entfernung oder Einfügung eines Werts in einem JSON-Dokument darstellt. Sowohl Substrukturen der Marken als auch die Marken selbst sind Substrukturen des Wurzel-Arrays der Stellenmarkierung.

- In Dokumentvariablen wird die einen einzufügenden oder zu entfernenden Wert umschließende Marke[1] gespeichert. Dokumentvariablen dienen dazu, bei der Verarbeitung von Werten (Substrukturen von Marken) die Daten der zugehörigen Marke verfügbar zu machen (zum Beispiel sollen wie im einführenden Beispiel in Unterabschnitt 8.1 bei der Verarbeitung einer Auftragsposition auch die Daten des zugehörigen Auftrags berücksichtigt werden können).
- Transitionen werden mit logischen Ausdrücken beschriftet. Sie beschreiben Zusammenhänge zwischen den Variablen der inzidenten Kantenbeschriftungen sowie an ausgewählten Werten vorzunehmende Manipulationen.

Die Struktur eines JSON-Netzes wird in Definition 8.5.1 spezifiziert.

Definition 8.5.1: JSON-Netz

Ein JSON-Netz ist ein Tupel $\mathcal{JSN} = (S,T,F,SB,KB,TB,M_0)$ mit den folgenden Eigenschaften:

(1) S,T sind endliche Mengen mit $S \cap T = \emptyset$, $S \cup T \neq \emptyset$. Dabei werden Stellen $s \in S$ als Speicher von JSON-Dokumenten interpretiert und Transitionen $t \in T$ als Operationen auf JSON-Dokumenten.

(2) $F \subseteq S \times T \cup T \times S$ ist die Flussrelation.

(3) Die Stellenbeschriftung SB weist jeder Stelle $s \in S$ ein Schema p_s zu.

(4) Die Kantenbeschriftung KB weist jeder Kante $(x,y) \in F$ ein Filtertupel $KB(x,y) = (p_{x,y},k_{x,y},v_{x,y},d_{x,y})$ zu. Ein Filtertupel besteht aus einem Filterausdruck p, einer Schlüsselvariablen k, einer Wertvariablen v und einer Dokumentvariablen d.

(5) Die Transitionsbeschriftung TB weist jeder Transition $t \in T$ einen logischen Ausdruck $TB(t)$ unter Verwendung der Variablen der Beschriftungen der inzidenten Kanten zu.

(6) Die Anfangsmarkierung M_0 weist den Stellen $s \in S$ (auch leere) Arrays als Markierung zu. $M(s)$ ist die Markierung der Stelle s unter einer Markierung M, und es muss immer gelten $M(s) \models p_s$.

[1] Kind-Elemente des Wurzel-Arrays der Markierung einer Stelle werden als Marken interpretiert.

Zur Erläuterung der Definition zeigt Abbildung 8.9 beispielhaft die Markierung eines JSON-Netzes. Das JSON-Netz stellt einen einfachen Geschäftsprozess dar, der aus einer Aktivität, der Vorbereitung einer Lieferung (t), besteht. Im Prozess werden anhand von Aufträgen (gespeichert in der Stelle s_1) und Produkten im Lager (s_2) Lieferungen (s_3) zusammengestellt. Im Array der Markierung $M(s_1)$ ist ein Auftrag als Object mit den Feldern `"id"`, `"address"` und `"items"` zu sehen. Weitere Aufträge im Array werden mit ... angedeutet. Im Array der Markierung $M(s_2)$ ist eine vorbereitete Lieferung zu sehen. Sie hat ebenfalls die Felder `"id"` und `"address"`. Um anzuzeigen, dass die Lieferung zum Auftrag gehört, hat die Lieferung für diese Felder die gleichen Werte. Daneben hat sie ein Feld `"package"`, das bisher noch leer ist. Weitere Lieferungen sind mit ... angedeutet. Produkte im Lager werden als einfache Strings im Array der Markierung $M(s_3)$ dargestellt. Aktuell sind zwei Computer-Mäuse (`"PC Mouse"`) und ein USB-Kabel (`"USB Cable"`) im Lager vorhanden.

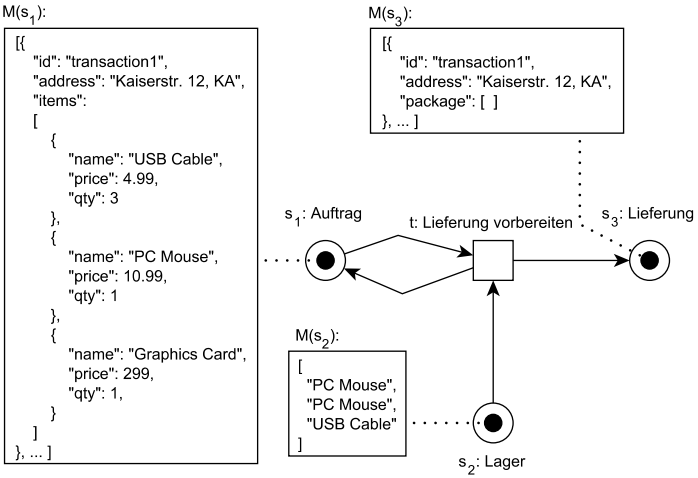


Abbildung 8.9: JSON-Netz mit Markierung.

Die Abbildung 8.10 zeigt die Beschriftungen für das Beispiel-Netz. Für jede Kantenbeschriftung $KB(s_1,t)$, $KB(t,s_1)$, $KB(s_2,t)$ und $KB(t,s_3)$ ist ein Tupel bestehend aus einem Filterausdruck p, einer Schlüsselvariablen k, einer Wertvariablen v und einer Dokumentvariablen d angegeben. Der JSONPath-Filterausdruck an der Kante (s_1,t) wählt mit $p_{s_1,t}$ = `"$.*.items.*"` zu entfernende Positionen in Aufträgen (Elemente des Arrays unter dem Feld `"items"`) aus. Mit dem Filterausdruck p_{t,s_1} = `"$.*.items"`

an der Kante (t,s_1) wird beschrieben, dass in das Array unter dem Feld `"items"` Werte eingefügt werden sollen. Für $k_{s_1,t}$ und k_{t,s_1} wird jeweils die gleiche Schlüsselvariable `orderPos` verwendet, womit ausgedrückt wird, dass beim Schalten der Transition (siehe Schaltregel unten) an derselben Position im Array zunächst ein Wert entfernt und dann wieder eingefügt wird.

In den Wertvariablen $v_{s_1,t}$ = `orderItemIn` und v_{t,s_1} = `orderItemOut` werden der zu entfernende beziehungsweise einzufügende Wert gespeichert. An beiden Kanten wird die gleiche Dokumentvariable $d_{s_1,t} = d_{t,s_1}$ = `order` verwendet, was ausdrückt, dass die Einfüge- und Entfernungsoperation beim Schalten an derselben Marke durchgeführt wird. Im Fall der Kantenbeschriftung $KB(s_2,t)$ wählt der Filterausdruck $p_{s_2,t}$ = `"$.*"` Kindelemente des Wurzel-Arrays aus. Hier ist also der zu entfernende Wert gleich dem zugehörigen umschließenden JSON-Dokument (der Marke). In diesem Fall speichern daher die Wertvariable $v_{s_2,t}$ = `packageItem` und die Dokumentvariable $d_{s_2,t}$ = `storage` den gleichen Wert (siehe auch die Erläuterungen zur Belegung in Unterabschnitt 8.5.2). In der Variablen `storagePos` wird die Array-Position des aus dem Lager entfernten Produkts gespeichert. In der Kantenbeschriftung $KB(t,s_3)$ wählt der JSONPath-Filterausdruck p_{t,s_3} = `"$.*.package"` das Array unter dem Feld `"package"` zum Einfügen eines Werts an der Position k_{t,s_3} = `packagePos` aus. Der einzufügende Wert, ein Produkt für die Lieferung, wird in der Variablen v_{t,s_3} = `packageItem` gespeichert. Die zugehörige Lieferung wird der Variablen d_{t,s_3} = `shipment` zugewiesen. Die Wertvariable `packageItem` findet sich sowohl an der Kante (s_2,t), als auch an der Kante (t,s_3), um auszudrücken, dass dasselbe Produkt, das aus dem Lager entnommen wird, der Lieferung hinzugefügt wird.

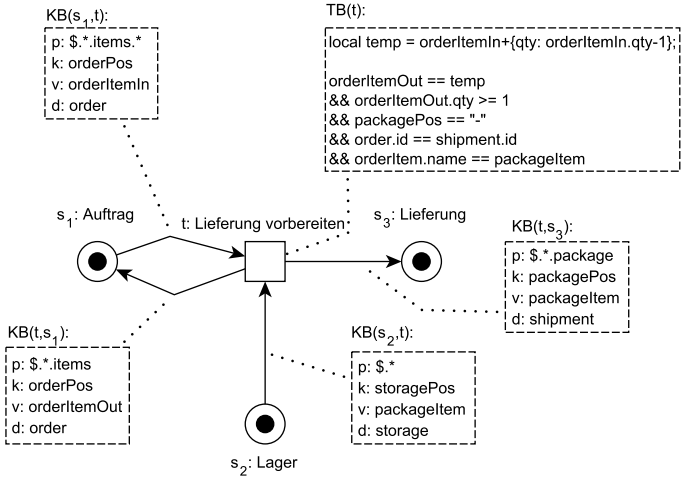


Abbildung 8.10: Beschriftungen eines JSON-Netzes.

Die Transitionsbeschriftung der Aktivität $TB(t)$ beschreibt in Jsonnet-Syntax Regeln für gültige Belegungen der Variablen beim Schalten. In der ersten Zeile wird eine Hilfsvariable `temp` mit dem Wert der eingehenden Bestellposition belegt, wobei die Menge (der Wert des Feldes `"qty"`) um eins reduziert wird. Danach folgen Regeln für gültige Belegungen der Variablen. Der Wert der Variablen `orderItemOut` soll gleich dem Wert der Hilfsvariablen `temp` sein (Zeile 3) und der Auftrag soll für die ausgewählte Auftragsposition noch eine Menge größer eins aufweisen (Zeile 4). Ein neues Produkt in der Lieferung soll immer an das Ende des `"package"`-Arrays angehängt werden (ausgedrückt mit dem Wert `"-"` für die Schlüsselvariable in Zeile 5). Schließlich muss die ID des ausgewählten Auftrags mit der ID der Lieferung übereinstimmen (Zeilen 6) und das ausgewählte Produkt auch noch im Lager vorhanden sein (Zeile 7).

8.5.2 Belegungen für Filtertupel

Bei JSON-Netzen ist die Aktivierung und das Schalten einer Transition abhängig von den Werten, mit denen die Variablen der Filtertupel der inzidenten Kanten belegt werden. Mögliche Belegungen der Variablen sind abhängig von der Auswertung der entsprechenden Filterausdrücke. An dieser Stelle wird der Beschreibung der Aktivierungsregel vorgegriffen, um das Prinzip der Belegungen zu erläutern.

Ein Filterausdruck kann für eine gegebene Stellenmarkierung mehrere Pfadausdrücke erzeugen (mehrere Werte auswählen). Für das Schalten beziehungsweise die Aktivierung der Stelle wird aber nur einer der Pfadausdrücke herangezogen. In Anlehnung an den Begriff der Variablenbelegung wird bei der Auswahl von einem der erzeugten Pfadausdrücke auch von einer *Belegung für den Filterausdruck* gesprochen. Eine Belegung β ist also eine Funktion, die Filterausdrücken Pfadausdrücke zuweist, und Schlüssel-, Wert- und Dokumentvariablen Werte. Die Belegung $\beta(p)$ eines Filterausdrucks p ist also immer ein Pfadausdruck. Für die Belegung von Schlüsselvariablen $\beta(k)$ sind Number- oder String-Werte möglich, da eine Schlüsselvariable immer auf den Schlüssel eines Schlüssel/Wert-Paares in einem Object oder auf den Index eines Arrays verweist. Für Belegungen von Wertvariablen $\beta(v)$ oder Dokumentvariablen $\beta(d)$ sind Werte von beliebigem JSON-Datentyp möglich.

Für ein Filtertupel im Vorbereich einer Transition wird die Wertvariable $v_{s,t}$ immer mit dem vom Pfadausdruck $\beta(p_{s,t})$ in der Markierung $M(s)$ ausgewählten Wert belegt. Die zugehörige Schlüsselvariable $k_{s,t}$ wird mit dem Array-Index oder Schlüssel des ausgewählten Werts belegt. Die Dokumentvariable $d_{s,t}$ wird schließlich mit der zum Wert gehörenden Marke (dem umschließenden JSON-Dokument, das ein direktes Kind des Wurzel-Arrays ist) belegt. Für Nachbereich-Filtertupel wählt der Pfadausdruck $\beta(p_{t,s})$ den Knoten im JSON-Baum der Markierung $M(s)$, an den der Wert $\beta(v_{t,s})$ angehängt wird.

Tabelle 8.2: Filterbelegungen für das Beispiel-JSON-Netz.

Filterausdruck	Ref.	β_1	β_2	β_3
`"$.*.items.*"`	$p_{s_1,t}$	`"/0/items/0"`	`"/0/items/1"`	`"/0/items/1"`
`"$.*.items"`	p_{t,s_1}	`"/0/items"`	`"/0/items"`	`"/0/items"`
`"$.*"`	$p_{s_2,t}$	`"/2"`	`"/0"`	`"/1"`
`"$.*.package"`	p_{t,s_3}	`"/0/package"`	`"/0/package"`	`"/0/package"`

Tabelle 8.2 zeigt drei Belegungen β_1, β_2 und β_3 für die in Abbildung 8.10 gezeigten Filterausdrücke des Beispielnetzes. Für die Belegung β_1 wird in den Auftragspositionen das USB-Kabel (`"USB Cable"`), beziehungsweise der Wert an der Position 0 im `"items"`-Array, ausgewählt. Bei β_2 und β_3 wird jeweils die Computer-Maus (`"PC Mouse"`) ausgewählt. Die beiden Belegungen unterscheiden sich lediglich darin, welche der beiden Computer-Mäuse (welche der beiden String-Werte `"PC Mouse"`) im Lager ausgewählt werden: $\beta_2(p_{s_2,t}) =$ `"/0"` und $\beta_3(p_{s_2,t}) =$ `"/1"`.

Tabelle 8.3: Variablenbelegungen für das Beispiel-JSON-Netz.

Variable	Ref.	β_1	β_2	β_3
orderPos	$k_{s_1,t}$, k_{t,s_1}	0	1	1
storagePos	$k_{s_2,t}$	2	0	1
packagePos	k_{t,s_3}	"-"	"-"	"-"
orderItemIn	$v_{s_1,t}$	`{ "name": "USB Cable", "price": 4.99, "qty": 3 }`	`{ "name": "PC Mouse", "price": 10.99, "qty": 1 }`	`{ "name": "PC Mouse", "price": 10.99, "qty": 1 }`
orderItemOut	v_{t,s_1}	`{ "name": "USB Cable", "price": 4.99, "qty": 2 }`	`{ "name": "PC Mouse", "price": 10.99, "qty": 0 }`	`{ "name": "PC Mouse", "price": 10.99, "qty": 0 }`
packageItem	$v_{s_2,t}$, v_{t,s_3}	"USB Cable"	"PC Mouse"	"PC Mouse"
order	$d_{s_1,t}$, d_{t,s_1}	`{ "id": "transaction1", "address": "K...", "items": [...] }`	`{ "id": "transaction1", "address": "K...", "items": [...] }`	`{ "id": "transaction1", "address": "K...", "items": [...] }`
shipment	d_{t,s_3}	`{ "id": "transaction1", "address": "K...", "package": [...] }`	`{ "id": "transaction1", "address": "K...", "package": [...] }`	`{ "id": "transaction1", "address": "K...", "package": [...] }`
storage	$d_{s_2,t}$	"USB Cable"	"PC Mouse"	"PC Mouse"

In der Tabelle 8.3 sind die Variablenbelegung für β_1, β_2 und β_3 gelistet. Aus Platzgründen sind in der Tabelle String-Werte und die Inhalte von Arrays mit ... abgekürzt. Die Variablenbelegungen sind abhängig von den jeweiligen Filterbelegungen in Tabelle 8.2 und der Transitionsbeschriftung $TB(t)$. Für die Schlüsselvariablen `orderPos` und `storagePos` ergibt sich der Wert direkt aus der zugehörigen Filterausdrucksbelegung, da es sich um Schlüsselvariablen an Kanten im Vorbereich der Transition handelt. Der Wert für `packagePos` ergibt sich aus der Transitionsbeschriftung. Er wird dort auf `"-"` festgelegt, um festzulegen, dass ein neues Produkt an das Ende des Arrays angehängt werden soll (siehe dazu auch die Definition der Einfügeoperation in Arrays in Unterabschnitt 8.4.4). Die Variable `orderItemIn` speichert die ausgewählte Auftragsposition (für β_1 das USB-Kabel-Object und für β_2 und β_3 das Computer-Maus-Object). Aus der Transitionsbeschriftung ergibt sich, dass der Wert von `orderItemOut` dem Wert von `orderItemIn` entspricht, nur dass die Menge `"qty"` um eins reduziert wurde. Der Wert von `packageItem` muss dem `"name"`-Feld der ausgewählten Auftragsposition entsprechen. Und die Dokumentvariablen `order`, `shipment` und `storage` speichern immer die zum Wert gehörende umschließende Marke. Für `order` und `shipment` ist das jeweils der entsprechende Auftrag beziehungsweise die zugehörige Lieferung. Im Fall von `storage` entspricht die Belegung der Dokumentvariablen immer der Belegung der Wertvariablen, da die Stelle nur einfache Werte speichert.

8.5.3 Aktivierung einer Transition

Damit eine Transition schalten kann, muss sie aktiviert sein. Dazu müssen zunächst die Filterausdrücke der Beschriftungen der inzidenten Kanten Werte in den zugehörigen Markierungen auswählen können (die Markierungen zu den Filterausdrücken gültig sein). Falls die Markierungen zu den Filterausdrücken gültig sind, werden die Filtertupel (Pfadausdrücke, Schlüsselvariablen, Wertvariablen und Dokumentvariablen) in Abhängigkeit von der Transitionsbeschriftung belegt (siehe dazu die Besprechung der Belegung von Filtertupeln weiter oben). Darüber hinaus wird überprüft, ob die beim Schalten zu erzeugende Markierung gültig zum Schema der jeweiligen Stellen ist.

Für den Fall, dass eine Stelle sowohl im Nachbereich als auch im Vorbereich derselben Transition liegt, spielt es für die Aktivierung auch eine Rolle, welche Ordnung die jeweiligen Pfadausdrücke haben, und ob sie konfliktfrei sind. Falls sie nicht konfliktfrei sind, ist die Transition nicht aktiviert. Falls sie konfliktfrei sind, richtet sich die Reihenfolge der Einfüge- und Entfernungs-Operationen nach der Ordnung der Pfadausdrücke. Die Sortierung der Operationen nach der Ordnung der zugehörigen

Pfadausdrücke ist darin begründet, dass andernfalls unbeabsichtigte Effekte auftreten könnten. Dadurch, dass sich durch manche Operationen die Position von Elementen in Arrays verschieben kann, bestünde die Gefahr, dass, wenn nicht auf die Ordnung der Pfadausdrücke geachtet wird, die falschen Werte manipuliert werden. Würde im gegebenen Beispiel erst der Wert `orderItemOut` an der Position `0` eingefügt werden und danach die Entfernungsoperation an der Position `0` durchgeführt werden, würde sich im Ergebnis die Markierung $M(s_1)$ nicht ändern, und weiterhin die alte Menge im Feld `"qty"` $= 3$ angezeigt werden.

Definition 8.5.2: Aktivierung im JSON-Netz

Eine Transition t ist für eine Belegung β von Dokument- und Wertvariablen mit JSON-Werten, von Schlüsselvariablen mit String- oder Numberwerten und von Filterausdrücken mit Pfadausdrücken aktiviert, wenn folgende Bedingungen erfüllt sind:

(1) Die Markierungen der adjazenten Stellen sind gültig zu den jeweiligen Filterausdrücke: $\forall s \in {}^\bullet t\colon M(s) \models p_{s,t}$ und $\forall s \in t^\bullet\colon M(s) \models p_{t,s}$, das heißt es kann je Filterausdruck mindestens ein Pfadausdruck erzeugt (ein Knoten im zugehörigen JSON-Baum ausgewählt) werden.

(2) Jeder Filterausdruck ist mit einem Pfadausdruck belegt, der aus der Anwendung des Filterausdrucks auf die Markierung der zur entsprechenden Kante inzidenten Stelle erzeugt wurde: $\forall s \in {}^\bullet t\colon \beta(p_{s,t}) \in p_{s,t}(M(s))$ und $\forall s \in t^\bullet\colon \beta(p_{t,s}) \in p_{t,s}(M(s))$.

(3) Jede Wertvariable im Vorbereich der Transition ist mit dem Wert belegt, der durch den zugehörigen Pfadausdruck ausgewählten wird: $\forall s \in {}^\bullet t\colon \beta(v_{s,t}) = \beta(p_{t,s})(M(s))$.

(4) Jede Schlüsselvariable im Vorbereich der Transition ist mit dem Index, beziehungsweise Schlüssel des ausgewählten Werts belegt: $\forall s \in {}^\bullet t\colon \beta(k_{s,t}) = key(\beta(p_{s,t}))$.

(5) Jede Dokumentvariable ist mit dem umschließenden JSON-Dokument des zugehörigen Werts belegt:

- $\forall s \in {}^\bullet t\colon \beta(d_{s,t}) = env(\beta(p_{s,t}))(M(s))$
- $\forall s \in t^\bullet\colon \beta(d_{t,s}) = env(\beta(p_{t,s}))(M(s))$

(6) Die Transitionsbeschriftung ist für die Belegung der Dokument-, Wert- und Schlüsselvariablen der Filtertupel der inzidenten Kanten wahr.

(7) Die Markierungen der Stellen im Vor- und Nachbereich sind nach dem Einfügen beziehungsweise Entfernen der Werte zu ihrem Schema gültig:

- $\forall s \in {}^\bullet t \setminus t^\bullet$: $deepRemove(\beta(p_{s,t}), M(s)) \vDash p_s$
- $\forall s \in t^\bullet \setminus {}^\bullet t$: $deepInsert(\beta(v_{t,s}), \beta(p_{t,s}), \beta(k_{t,s}), M(s)) \vDash p_s$
- $\forall s \in {}^\bullet t \cap t^\bullet$:
  - Falls $\beta(p_{s,t})$ und $\beta(p_{t,s})$ nicht konfliktfrei sind, dann ist t nicht aktiviert.
  - Falls die Pfadausdrücke konfliktfrei sind, richtet sich die Reihenfolge der durchzuführenden Operationen, und damit auch das Ergebnis der Operationen, notiert mit $M^*(s)$ (siehe Beschreibung der Schaltregel unten), nach der Ordnung der Pfadausdrücke. Damit die Transition aktiviert ist, muss in diesem Fall gelten: $M^*(s) \vDash p_s$.

Wird das in Abbildung 8.9 gezeigte Auftragsdokument mit ID `"transaction1"` ausgewählt, sind β_1, β_2 und β_3 die einzigen möglichen Belegungen. Für weitere im Array der Markierung $M(s_1)$ gespeicherte Aufträge sind aber noch andere Belegungen denkbar, unter denen die Transition aktiviert wäre.

8.5.4 Schaltregel für JSON-Netze

Beim Schalten einer Transition in einem JSON-Netz werden Werte aus den Stellenmarkierungen im Vorbereich entfernt und Werte in die Stellenmarkierungen im Nachbereich eingefügt. Definition 8.5.3 beschreibt dazu die Schaltregel für eine unter einer gegebenen Belegung aktivierte Transition. Aus der Markierung der Stellen im Vorbereich werden Werte entsprechend der Belegung entfernt und im Nachbereich hinzugefügt. Der Fall, dass eine Stelle sowohl im Vorbereich als auch im Nachbereich der Transition liegt, ist mit der Regel für $M^*(s)$ dargestellt. Falls der Pfadausdruck an der ausgehenden Kante $\beta(p_{t,s})$ von höherer Ordnung als der Pfadausdruck an der eingehenden Kante $\beta(p_{s,t})$ ist, wird zuerst die Einfüge- und dann die Entfernungsoperation durchgeführt. Andernfalls wird zuerst die Entfernungsoperation und dann die Einfügeoperation durchgeführt. In Abbildung 8.11 ist das Ergebnis gezeigt, wenn die Transition im Beispiel unter der Belegung β_1 schalten würde. Die Anzahl der USB-Kabel im Auftrag wurde um 1 reduziert, ein USB-Kabel wurde aus dem Lager entfernt und in die Lieferung eingefügt.

Definition 8.5.3: Schaltregel für JSON-Netze

Wenn eine Transition t unter einer Markierung M und einer Belegung β aktiviert ist, dann kann t schalten. Das Schalten von t überführt M in die Folgemarkierung M' mit

$$M'(s) = \begin{cases} deepInsert(\beta(v_{t,s}),\beta(p_{t,s}),\beta(k_{t,s}),M(s)) & s \in t^\bullet \ \wedge \ s \notin {}^\bullet t \\ deepRemove(\beta(p_{s,t}),M(s)) & s \in {}^\bullet t \ \wedge \ s \notin t^\bullet \\ M^*(s) & s \in {}^\bullet t \ \wedge \ s \in t^\bullet \\ M(s) & sonst. \end{cases}$$

Für Stellen, die sowohl im Nachbereich, als auch im Vorbereich der Transition liegen, ist die Reihenfolge der durchzuführenden Operationen abhängig von der Ordnung der Pfadausdrücke der Belegung. Die resultierende Markierung $M^*(s)$ ist wie folgt definiert:

$$M^*(s) = \begin{cases} deepRemove(\beta(p_{s,t}),M(s), & \beta(p_{t,s}) \succ_{M(s)} \beta(p_{s,t}) \\ \quad deepInsert(\beta(v_{t,s}),\beta(p_{t,s}),\beta(k_{t,s}))) & \\ deepInsert(\beta(v_{t,s}),\beta(p_{t,s}),\beta(k_{t,s}), & \beta(p_{s,t}) \succsim_{M(s)} \beta(p_{t,s}) \\ \quad deepRemove(\beta(p_{s,t}),M(s))) & \end{cases}$$

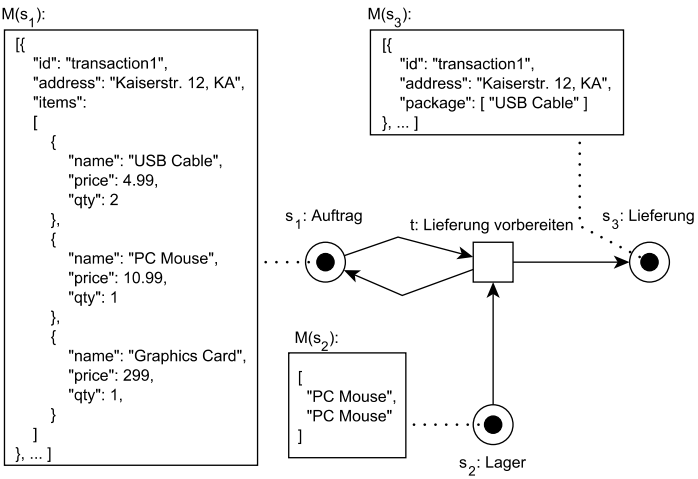


Abbildung 8.11: Das Beispiel-JSON-Netz nach dem Schalten.

9 Entwicklung eines Modellierungswerkzeugs

In diesem Kapitel wird die Entwicklung des JSON-Netz-Editors als Modellierungswerkzeug für JSON-Netze beschrieben. Das Modellierungswerkzeug stellt unterstützende Mechanismen zur Anwendung der in Kapitel 8 vorgestellten Modellierungssprache bereit. In den folgenden Kapiteln 10 und 11 wird er eingesetzt, um anhand des Vorgehensmodells aus Kapitel 7 Nachhaltigkeitsauswirkungen von Geschäftsprozessen zu modellieren und zu analyisieren. In Abschnitt 9.1 werden zunächst die Zielsetzung und die Anforderungen für den JSON-Netz-Editor beschrieben. Anschließend werden in Abschnitt 9.2 die Architektur und technische Aspekte der Implementierung vorgestellt. Abschnitt 9.3 dient der Vorstellung der Konzeption der Benutzungsoberfläche. Anhand von Beispielen aus verwandten Software-Werkzeugen werden Gestaltungsentscheidungen, die bei der Implementierung des Modellierungswerkzeugs getroffen wurden, dargelegt. Die Hauptfunktionen der resultierenden Implementierung werden in Abschnitt 9.4 beschrieben. Abschließend wird in Abschnitt 9.5 ein Überblick zu ergänzenden Funktionen gegeben.

9.1 Zielsetzung und Anforderungen

Das im Rahmen dieser Arbeit entwickelte Modellierungswerkzeug für JSON-Netze soll die Realisierbarkeit der in Kapitel 8 ausgearbeiteten Konzepte zeigen. Daraus folgt, dass Netzstrukturen darstellbar sein sollen, Markierungen und Beschriftungen eines JSON-Netzes vorgenommen sowie Schaltvorgänge ausgeführt werden können (Anforderungen A M1 bis A M3 in der folgenden Liste). Weiterhin sollen Konzepte und Funktionen zur Unterstützung der Benutzbarkeit umgesetzt werden (Anforderung A M4). Insgesamt ergeben sich die folgenden Anforderungen:

(A M1) Grafisches Modellieren von Netzstrukturen: Anwendende sollen Netzstrukturen erstellen können. Dazu zählt die Möglichkeit, Stellen und Transitionen grafisch darzustellen und benennen sowie Stellen und Transitionen mit Kanten verbinden zu können. Stellen, Transitionen und Kanten sollen auch gelöscht werden können.

(A M2) Spezifikation von Markierungen und Beschriftungen: Anwendende sollen Markierungen und Beschriftungen einer Netzstruktur entsprechend der JSON-Netz-Spezifikation vornehmen können. Dazu zählt die Angabe eines Arrays als Markierung für eine Stelle, die Spezifikation eines JSON-Schemas als Typisierung für eine Stelle, die Beschriftung einer Transition mit einem logischen Jsonnet-Ausdruck und die Beschriftung von Kanten mit Filterausdrücken.

(A M3) Ausführen von Schaltvorgängen: Die in Abschnitt 8.5 vorgestellte Schaltregel soll auf ein zuvor erstelltes JSON-Netz anwendbar sein. Das Modellierungswerkzeug soll die Möglichkeit bieten, Belegungen zu finden, unter denen eine Transition aktiviert ist, und aktivierte Transitionen schalten zu lassen. Mithilfe dieser Funktionen sollen Anwendende eine simulationsbasierte Analyse von Geschäftsprozessen vornehmen können.

(A M4) Benutzbarkeitsunterstützung: Anwendende sollen bei der Modellierung unterstützt und angeleitet werden. Die Unterstützung richtet sich dabei sowohl technisch versierte Anwendende, die im Umgang mit JSON-Daten vertraut sind als auch an nicht-technisch versierte Anwendende (siehe auch die Anforderungsspezifikation an die Methode in Kapitel 6).

Die Unterstützung der Benutzbarkeit (Anforderung A M4) bildet aus mehreren Gründen einen Schwerpunkt der Entwicklung des Modellierungswerkzeugs. Erstens ist die effektive und effiziente Nutzung einer Modellierungssprache erst mit der Bereitstellung eines (einfach benutzbaren) Modellierungswerkzeugs möglich [Fra13]. Zweitens stellt die Benutzbarkeit von Modellierungswerkzeugen aufgrund der inhärenten Komplexität von datenorientierten Prozessmodellierungssprachen eine besondere Herausforderung dar [Ste19]. Und drittens wurde mangelnde Benutzbarkeit in der Literatur als Hinderungsgrund für die Akzeptanz von datenorientierten Prozessmodellierungsansätzen identifiziert [Ste19, Rei17].

Um den genannten Herausforderungen zu begegnen, wurden bei der Entwicklung des JSON-Netz-Editors Empfehlungen für den Entwurf benutzungsfreundlicher Software [Gör03] und benutzungsfreundlicher Modellierungswerkzeugen [Moo09] befolgt. Für den Entwurf benutzungsfreundlicher Software empfiehlt [Gör03] die Durchführung einer Wettbewerbsanalyse, um Inspiration und Ideen für die Gestaltung der Benutzungsoberfläche zu generieren. Im Rahmen der Entwicklung des JSON-Netz-Editors wurden daher existierende Software-Werkzeuge, die in unterschiedlicher Form die Arbeit mit Prozessmodellen und Daten unterstützen, untersucht. Geeignete Konzepte zur

Benutzungsführung und Strukturierung der Benutzungsoberfläche wurden identifiziert, bei Bedarf angepasst und übernommen (siehe Abschnitt 3). [Moo09] empfiehlt, sich bei der Entwicklung von Modellierungswerkzeugen[1] an in der Literatur beschriebenen Benutzbarkeitsheuristiken zu orientieren. Die folgende Liste gibt einen Überblick zu in der Literatur beschriebenen Heuristiken für den Entwurf benutzbarer Modellierungswerkzeuge.[2] Es wird jeweils beispielhaft erläutert, wie die genannten Heuristiken bei der Entwicklung des JSON-Netz-Editors umgesetzt wurden. Die Liste sowie die beschriebenen Umsetzungen sind dabei nicht als abgeschlossen zu betrachten. Grundsätzlich kann und muss die Benutzbarkeit einer Software auf Basis von Rückmeldung von Anwendenden auch nach dem Vorliegen eines ausgearbeiteten Gestaltungskonzepts weiter verbessert werden [Gör03]. Die Benutzbarkeit des JSON-Netz-Editors wurde daher im Rahmen der Evaluation der vorliegenden Modellierungsmethode im Einsatz mit Anwendenden untersucht (siehe Kapitel 11). Im Ausblick (siehe Kapitel 12) wird auf mögliche Weiterentwicklungen zur Verbesserung der Benutzbarkeit des JSON-Netz-Editors eingegangen.

Lernen am Beispiel: Konkrete Beispiele unterstützen Anwendende beim Lernen einer Anwendung [Mye86].

Umsetzung (Beispiel): Anwendende erhalten Beispiele für mögliche Transitionsbeschriftungen (siehe Unterabschnitt 9.4.4).

Rückmeldungen: Rückmeldungen unterstützen Anwendende im Modellierungsprozess [Alp15].

Umsetzung (Beispiel): Es werden Rückmeldungen zu gültigen Netzstrukturen, Markierungen, Schemas und Beschriftungen gegeben (siehe Unterabschnitte 9.4.3 und 9.4.2).

Konsistenz: Konsistente Gestaltung der Komponenten eines Modellierungswerkzeugs (Buttons, Leisten, Fenster ...) beschleunigen den Lernprozess [Alp15].

Umsetzung (Beispiel): Für ähnliche Funktionen (Löschen, Editieren, ...) werden gleiche Buttons an Modellelementen angezeigt (siehe Unterabschnitt 9.4.4).

Wiederverwendung bekannter Konzepte: Verwendung von den Anwendenden bekannten Konzepten verringern die wahrgenommene Komplexität [Sno23].

[1] [Moo09] adressiert im Wortlaut den Entwicklung der Notation von Modellierungssprachen, die Empfehlungen lassen sich aber auch auf die Entwicklung von Modellierungswerkzeugen übertragen.

[2] Die genannten Heuristiken zur Unterstützung der Benutzbarkeit von Modellierungswerkzeugen wurden in einer Literaturrecherche im Rahmen einer vom Autor betreuten Masterarbeit [Lag23] identifiziert.

Umsetzung (Beispiel): Die Struktur der Benutzungsoberfläche orientiert sich an existierenden Modellierungswerkzeugen (siehe Unterabschnitte 9.3.1 und 9.4.1).

Low Code: Low Code-Konzepte erhöhen die Produktivität bei der Formulierung von (Code-)Ausdrücken [Sno23].

Umsetzung (Beispiel): Es werden vordefinierte Variablen bereitgestellt, die per Klick-Auswahl in einen Code-Editor übertragen werden können (siehe Unterabschnitt 9.4.4). Markierungen können formularbasiert bearbeitet werden (siehe Unterabschnitt 9.5.1).

Modularisierung: Modularisierung eines Modellierungswerkzeugs reduziert die kognitive Belastung für Anwendende [Moo09].

Umsetzung (Beispiel): Transitionsbeschriftungen werden thematisch in mehrere Abschnitte aufgeteilt (siehe Unterabschnitt 9.4.4).

Verwendung von Icons: Die Verwendung von Icons beschleunigt das Wiedererkennen von Funktionen [Moo09].

Umsetzung (Beispiel): Verwendung von Icons zur Kennzeichnung von Buttons für verschiedene Funktionen wie Löschen und Editieren (siehe Unterabschnitt 9.4.2).

9.2 Architektur und technische Umsetzung

Der JSON-Netz-Editor wurde als Web-Anwendung mit JavaScript-Technologien umgesetzt. Die Wahl von JavaScript begünstigte die Implementierung von JSON-spezifischen Funktionen, da sich Daten in JSON-Datenmodell unmittelbar in JavaScript verarbeiten lassen (siehe Abschnitt 8.3). Die resultierende Web-Anwendung ist unter https://kit-bis.github.io/json-nets/ frei zugänglich. Abbildung 9.1 gibt einen Überblick über die Architektur des Editors. Zu den einzelnen Komponenten der Architektur ist jeweils angegeben, mit welchen unterstützenden Bibliotheken sie umgesetzt wurden. Die zentrale Komponente der Architektur ist die JSON-Netz-Bibliothek. Diese Bibliothek unterstützt die Speicherung eines JSON-Netzes in einer Datenstruktur und implementiert grundlegende Funktionen, zum Beispiel zur Suche nach Belegungen, unter denen eine Transition aktiviert ist, oder zur Anwendung der Schaltregel. Sie bietet Schnittstellen zur Manipulation eines gespeicherten Netzes und kann damit auch unabhängig von den anderen Komponenten der Architektur eingesetzt werden. Diese Gestaltungsentscheidung ermöglicht es, einzelne Komponenten der Anwendung leichter auszutauschen, sollte zum Beispiel in Zukunft eine der genutzten

externen Bibliotheken veralten und nicht mehr unterstützt werden. Weiterhin ermöglicht die eigenständige JavaScript-Bibliothek, JSON-Netze auch unabhängig vom JSON-Netz-Editor in anderen Anwendungen einzubinden. Beispielsweise könnten im JSON-Netz-Editor erstellte Modelle exportiert und dann in einer anderen Anwendung, die ebenfalls die JSON-Netz-Bibliothek integriert, ausgeführt werden. Die JSON-Netz-Bibliothek integriert mehrere JSON-Werkzeuge, womit Bibliotheken gemeint sind, welche die in der Spezifikation genannten Sprachen wie JSONPath [Che24], JSON Pointer [Sto24], JSON Schema [Pob21] und Jsonnet [Goo24c] unterstützen.

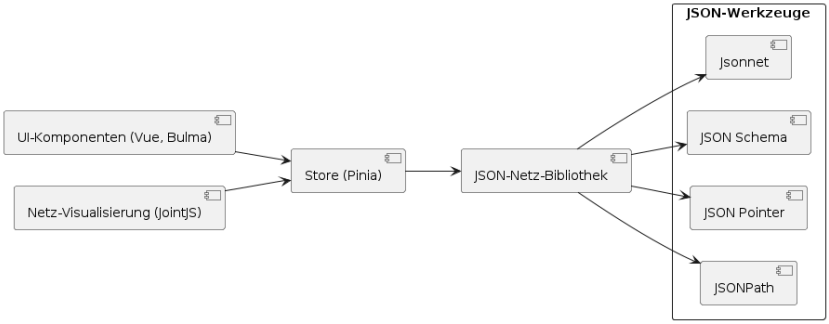


Abbildung 9.1: Architektur des JSON-Netz-Editors.

Die Struktur und Steuerungslogik der Komponenten der Benutzungsoberfläche (in der Abbildung verkürzt UI-Komponenten genannt) wie Eingabemasken und Buttons wurden mit Hilfe der JavaScript-Bibliothek Vue [You24] implementiert. Zur Gestaltung der UI-Komponenten wurde dabei die Bibliothek Bulma [Jer24] genutzt. Bulma unterstützt die benutzungsfreundliche Darstellung einer Web-Anwendung mit Vorgaben unter anderem für Farben und Schriftgrößen für einzelne Elementen der Benutzungsoberfläche. Die JavaScript-Bibliothek JointJS [cli24] unterstützt die grafische Darstellung und Manipulation von Graphen und bildet die Grundlage für die bereitgestellte Funktion zur Modellierung von Netzstrukturen. Der sogenannte Store, der mit der JavaScript-Bibliothek Pinia [Mor24] umgesetzt wurde, verbindet die Benutzungsoberfläche mit der JSON-Netz-Bibliothek. Der Store speichert und verwaltet den Zustand der Webanwendung. So verwaltet er zum Beispiel Informationen dazu, welches Fenster geöffnet wurde oder welcher Button zuletzt angeklickt wurde. Weiterhin leitet er die Eingaben der Anwendenden an die JSON-Netz-Bibliothek weiter und erhält den neuen Zustand des JSON-Netzes zurück. Wird beispielsweise ein Button geklickt, um eine bestimmte Transition schalten zu lassen, wird diese Information vom Store an

die JSON-Netz-Bibliothek weitergegeben. Als Rückgabewert würde die JSON-Netz-Bibliothek die resultierende Belegung der adjazenten Stellen an den Store übergeben. Basierend auf dem neuen zurückgegebenen Zustand wird die Darstellung in der Netz-Visualisierung und den UI-Komponenten angepasst.

9.3 Konzeption der Benutzungsoberfläche

Im Folgenden werden die Ergebnisse der nach der Empfehlung von [Gör03] durchgeführten Wettbewerbsanalyse vorgestellt. Die bei der Entwicklung des JSON-Netz-Editors getroffenen Gestaltungsentscheidungen werden anhand von Beispielen aus der Untersuchung existierender verwandter Software-Werkzeuge vorgestellt.

9.3.1 Struktur der Benutzungsoberfläche

Die Struktur der Benutzungsoberfläche des JSON-Netz-Editors orientiert sich an existierenden Modellierungswerkzeugen wie Horus [Hor24], Signavio [SAP24] oder Camunda [Cam24]. In diesen Modellierungswerkzeugen ist die Unterscheidung zwischen einer Werkzeugleiste und einem Modellierungsbereich üblich.

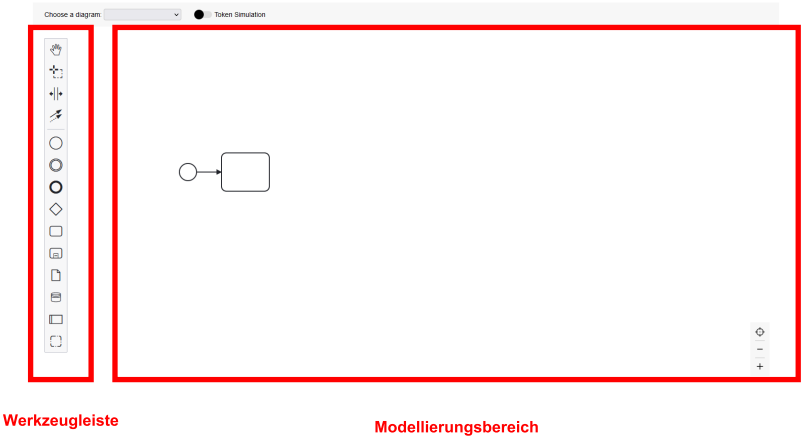


Abbildung 9.2: Werkzeugleiste und Modellierungsbereich von Camunda. Bildschirmfoto von [Cam24] mit eigenen Markierungen.

Abbildung 9.2 zeigt beispielhaft das BPMN-Modellierungswerkzeug Camunda. Auf der linken Seite ist rot die Werkzeugleiste markiert. Sie beinhaltet mehrere Buttons, mit denen Anwendende verschiedene Modi und Modellierungselemente auswählen können. Auf der rechten Seite findet sich der Modellierungsbereich, hier kann das Prozessmodell erstellt und editiert werden.

9.3.2 Eingabeunterstützung für JSON-Daten

Im JSON-Netz-Editor soll die Eingabe von JSON-Daten (für Markierungen) beziehungsweise JSON Schema-Regeln (für Stellenschemas) unterstützt werden. Für die Eingabeunterstützung von JSON-Daten und JSON Schema-Regeln gibt es Beispiele bei Code-Editoren wie Visual Studio Code [Mic24]. So gibt der Code-Editor bei der Eingabe von JSON-Daten visuelle Hinweise, falls syntaktische Fehler erkannt werden. Im Beispiel der Abbildung 9.3a fehlt bei der Eingabe eines JSON-Objects der Doppelpunkt zwischen Schlüssel und Wert. Über eine rote Unterstreichung und eine Fehlermeldung werden Anwendende darauf hingewiesen. Weiterhin kann der Code-Editor auf Basis von hinterlegten JSON Schema-Regeln Vervollständigungsvorschläge für mögliche Eingaben, zum Beispiel Namen für Schlüssel eines JSON-Objects, geben (siehe Abbildung 9.3b).

(a) Beispiel für Syntax-Hinweise.

(b) Beispiel für Vervollständigungsvorschläge.

Abbildung 9.3: Syntax-Hinweise und Vorschläge in Visual Studio Code. Bildschirmfotos von [Mic24].

9.3.3 Eingabeunterstützung für Ausdrücke

Wie die Beispiele in der Spezifikation von JSON-Netzen zeigen (siehe Abschnitt 8.5), eröffnet das Zusammenspiel von Transitionsbeschriftungen mit unterschiedlichen Kantenbeschriftungen, Filterbelegungen und Variablendefinitionen vielfältige Ausdrucksmöglichkeiten. Die Konzeption einer Benutzungsoberfläche, welche diese vielfältigen Ausdrucksmöglichkeiten für Anwendende handhabbar macht, stellt daher für die Implementierung des JSON-Netz-Editors eine Herausforderung dar. Eine mögliche Lösung für derartige Herausforderungen findet sich in der Benutzungsoberfläche

des Automatisierungswerkzeugs n8n [n8n24]. n8n ermöglicht es, Automatisierungsregeln als „Workflows“ zu definieren, und damit verschiedene Webdienste wie zum Beispiel Google Docs [Goo24a] oder Github [Git24] miteinander zu verbinden. In diese Workflows können unter anderem sogenannte „Set“-Knoten eingebunden werden. In Set-Knoten werden, vergleichbar zu Transitionsinschriften in JSON-Netzen, Transformationsregeln für die Erzeugung von Ausgabedaten aus verschiedenen Eingabedaten festgelegt. Die Benutzungsoberfläche für die Eingabe von Transformationsregeln in Set-Knoten wird daher in ähnlicher Form in der Benutzungsoberfläche des JSON-Netz-Editors für die Eingabe von Transitionsbeschriftungen umgesetzt. Abbildung 9.4 zeigt die Benutzungsoberfläche für die Eingabe von Transformationsregeln eines Set-Knotens in n8n. Anwendende können im entsprechenden Fenster auf der linken Seite verschiedene Quellen für Eingabedaten auswählen und bekommen vorhandene Daten (soweit vorhanden) angezeigt. In der Mitte können Transformationsregeln für verschiedene zu erzeugende Werte festgelegt werden und auf der rechten Seite werden die anhand der Transformationsregeln erzeugten Ausgabedaten angezeigt.

Die in Abbildung 9.4 gezeigte Ansicht erlaubt zwar eine übersichtliche Darstellung der verschiedenen Eingabe- und Ausgabedaten, sie stellt aber nur geringen Platz für die Spezifikation von Transformationsregeln zu Verfügung. Diese können in n8n, ähnlich wie Transitionsbeschriftungen für JSON-Netze, auch so umfangreich werden, dass der zur Verfügung stehende Platz nicht ausreicht, um sie übersichtlich anzuzeigen. Daher

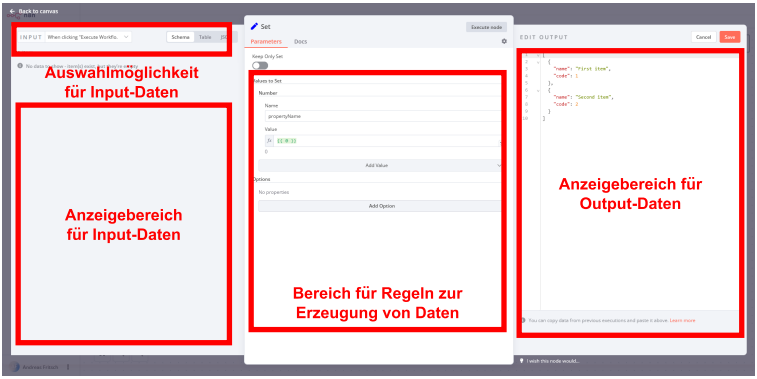


Abbildung 9.4: Set-Knoten des Automatisierungswerkzeugs n8n. Bildschirmfoto von [n8n24] mit eigenen Markierungen.

gibt es in n8n die Möglichkeit, eine Transformationsregel in einem speziellen Bearbeitungsfenster zu öffnen. Wie Abbildung 9.5 zeigt, stellt diese Ansicht einen größeren Bearbeitungsbereich für Transformationsregeln bereit. Sie unterstützt die Formulierung von Transformationsregeln mit einer Liste zur Verfügung stehender Variablen auf der linken Seite. Weiterhin erhalten Anwendende in einer Ergebnis-Vorschau unterhalb des Bearbeitungsbereichs unmittelbar Rückmeldung zum erwarteten Ergebnis einer Transformationsregel.

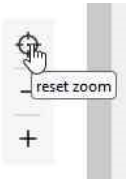
Abbildung 9.5: Bearbeitungsfenster für Transformationsregeln in n8n. Bildschirmfoto von [n8n24] mit eigenen Markierungen.

9.3.4 Gestaltungsdetails zur Benutzungsunterstützung

Zur Unterstützung der Anwendenden eines Software-Werkzeugs sind neben größeren Gestaltungsentscheidungen wie der Struktur der Benutzungsoberfläche auch Details der Gestaltung zu betrachten [Gör03]. In aktuellen Software-Werkzeugen finden sich dafür eine Vielzahl von Beispielen. Auch in Bibliotheken wie Bulma werden Entwicklerinnen und Entwickler dabei unterstützt, solche Details umzusetzen. Im Folgenden werden Beispiele für Gestaltungsdetails zur Benutzungsunterstützung, die in die Umsetzung des JSON-Netz-Editors eingeflossen sind, vorgestellt. Dazu zählt die Möglichkeit, Anwendenden über einen veränderten Mauszeiger Rückmeldung zu Interaktionsmöglichkeiten zu geben. Im BPMN-Modellierungswerkzeug Camunda zeigen zum Beispiel gekreuzte Pfeil an, dass ein Modellelement bewegt werden kann (siehe Abbildung 9.6a). Ergänzend können Hilfstexte am Mauszeiger angezeigt werden, um Interaktionsmöglichkeiten zu erläutern (siehe Abbildung 9.6b). Die bei der Entwicklung

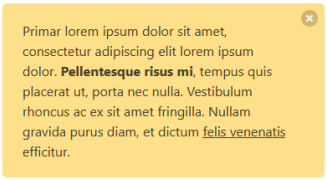
des JSON-Netz-Editors genutzte Bibliothek Bulma bietet selbst auch schon einige vorgefertigte Beispiele für Gestaltungsdetails zur Benutzungsunterstützung. Dazu zählen zum Beispiel Fehlermeldungen bei Formulareingaben (siehe Abbildung 9.6c) und farblich hervorgehobene Hinweistexte (siehe Abbildung 9.6d).

(a) Veränderter Mauszeiger in Camunda.



(b) Hilfstext an Mauszeigern in Camunda.

(c) Fehlermeldung in einem Bulma-Formular.



(d) Beispiel für einen hervorgehobenen Hinweistext mit Bulma.

Abbildung 9.6: Beispiele für Gestaltungsdetails zur Benutzungsunterstützung. Bildschirmfotos von [Cam24] und [Jer24].

9.4 Umsetzung der Anforderungen

Mit den oben beschriebenen Beispielen wurden wichtige Inspirationsquellen für den Entwurf der Benutzungsoberfläche des JSON-Netz-Editors beschrieben. Wie auch die letzten Beispiel gezeigt haben, wurden einige Benutzungskonzepte aus existierenden Software-Werkzeugen im JSON-Netz-Editor übernommen, andere flossen durch die Nutzung von Bibliotheken wie Bulma oder JointJS in die Entwicklung ein. Die im JSON-Netz-Editor umgesetzten Funktionen und Konzepte zur Benutzungsunterstützung werden im Folgenden anhand einer „Story Map“ vorgestellt. Eine Story Map stellt die Aktivitäten, die Anwendende mit einem Software-Werkzeug durchführen können,

beispielhaft chronologisch sortiert vor, sodass eine Art Geschichte über das Software-Werkzeug erzählt wird [Pat15].[1] Abbildung 9.7 zeigt die im Folgenden vorgestellten Aktivitäten der Story Map. Sie lassen sich in die drei Phasen *Kennenlernen*, *Modellieren* und *Ausführen* einteilen. In der Kennenlernen-Phase wird der Editor geöffnet und ein Anwender oder eine Anwenderin macht sich mit der Anwendung vertraut. Die Modellieren-Phase besteht aus den drei Aktivitäten Netzstruktur erstellen, Stellen editieren und Transitionen editieren. Wurde ein JSON-Netz vollständig modelliert, können schließlich in der Ausführen-Phase Transitionen geschaltet werden.

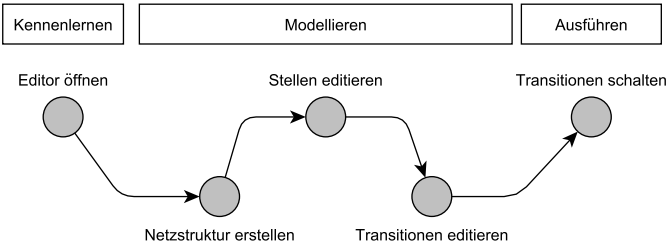


Abbildung 9.7: Story Map zur Vorstellung des JSON-Netz-Editors.

9.4.1 Editor öffnen

Wird die Webanwendung des JSON-Netz-Editors geöffnet, zeigt sich zunächst die Hauptansicht, wie sie in Abbildung 9.8 abgebildet ist. Die Hauptansicht ist wie bei den in Abschnitt 9.3 vorgestellten Inspirationsquellen in eine Werkzeugleiste und einen Modellierungsbereich geteilt. Die Buttons in der Werkzeugleiste stellen Anwendenden verschiedene Funktionen des Editors zur Verfügung. In der Reihenfolge ihres Erscheinens von links nach rechts sind das:

Stellen: Modus zum Hinzufügen neuer Stellen.

Transitionen: Modus zum Hinzufügen neuer Transitionen.

Kanten: Modus zum Erstellen von Kanten zwischen Stellen und Transitionen.

[1] Story Mapping ist ursprünglich eine Methode aus der agilen Softwareentwicklung. Sie dient dazu, im Entwicklungsprozess ein gemeinsames Verständnis der Software mit beteiligten Stakeholdern zu erarbeiten [Pat15]. Die Struktur einer Story Map eignet sich daher auch für die Vorstellung der Funktionen einer bestehenden Software.

Auto-Layout: Automatische Positionierung von Stellen und Transitionen.

Editieren: In diesem Modus können Modellelemente gelöscht und editiert werden.

Simulation: Auführen des Simulations-Algorithmus (siehe Unterabschnitt 9.4.5).

Export: Speichern des JSON-Netzes mit seiner aktuellen Markierung in einer JSON-Datei.

Import: Laden eines JSON-Netzes aus einer JSON-Datei.

Hilfe: Anzeigen eines Fensters mit Erläuterungen der Funktionen des Editors.

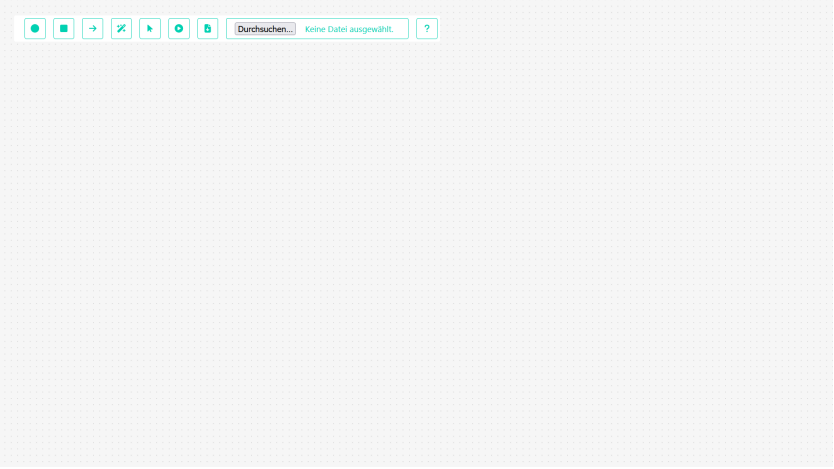
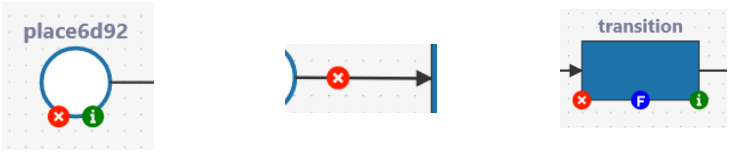


Abbildung 9.8: Werkzeugleiste und Modellierungsbereich im JSON-Netz-Editor.

Die ersten vier Modi (Stellen, Transitionen, Kanten und Auto-Layout) unterstützen die Erstellung einer Netzstruktur und werden im Unterabschnitt 9.4.2 näher vorgestellt. Im Editieren-Modus zeigen Stellen, Kanten und Transitionen beim Überfahren mit der Maus kleine Buttons, die weitere Interaktonsmöglichkeiten bereitstellen. Über den roten X-Button können Modellelemente gelöscht werden (siehe Unterabschnitt 9.4.2). Der grüne i-Button öffnet spezielle Bearbeitungsfenster, mit denen die Eigenschaften eines Modellelements geändert werden können (siehe Unterabschnitte 9.4.3 und 9.4.4). Der blaue F-Button erlaubt das Schalten individueller Transitionen (siehe Unterabschnitt 9.4.5).



(a) Interaktionsbuttons an Stellen. **(b)** Interaktionsbutton an Kanten. **(c)** Interaktionsbuttons an Transitionen.

Abbildung 9.9: Interaktionsmöglichkeiten im Bearbeiten-Modus.

9.4.2 Netzstruktur erstellen

Zum Erstellen einer Netzstruktur können Anwendende nach Auswahl des Stellen- oder Transitionsmodus mit Klick auf die Modellierungsoberfläche entsprechende Modellelemente erstellen. Abbildung 9.10 zeigt, wie Anwendende im Kanten-Modus Kanten zwischen Transitionen und Stellen erstellen können. Ausgehend von einem Modellelement, im Beispiel eine Transition, kann mit der Maus an einem kleinen Pfeil-Button gezogen werden. Der JSON-Netz-Editor gibt über eine orangene Umrandung Rückmeldung, dass eine Verbindung mit einem zweiten Element zulässig ist. Sobald die gezogene Kante über einem zulässigen Element losgelassen wird, wird der Netzstruktur eine entsprechende Kante hinzugefügt.

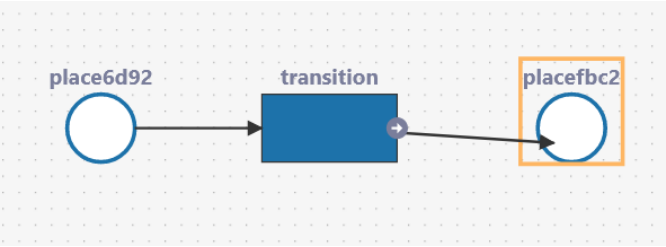
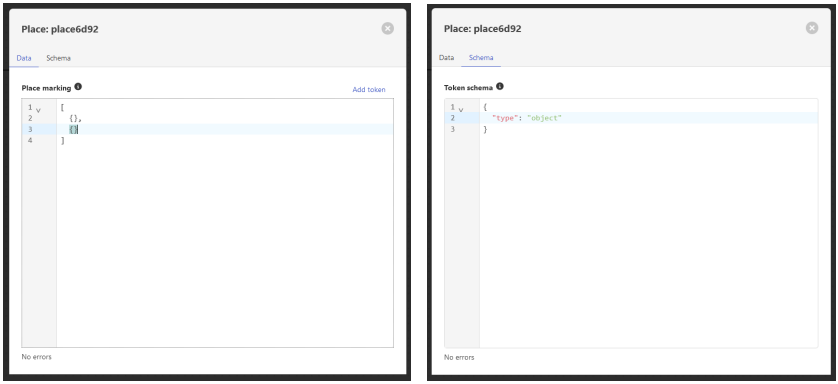


Abbildung 9.10: Beispiel für das Erstellen einer Kante.

9.4.3 Stellen editieren

Wird bei einer Stelle im Editieren-Modus der grüne i-Button angeklickt, öffnet sich ein Fenster, in dem die Eigenschaften der entsprechenden Stelle bearbeitet werden können. Über Klick auf den Namen der Stelle kann dieser in einem Formularfeld geändert werden. Wie Abbildung 9.11 zeigt, können im Fenster weiterhin zwei unterschiedliche

Tabs ausgewählt werden. Der erste, mit der Überschrift *Data*, dient der Bearbeitung der Stellenmarkierung. Im zweiten Tab, *Schema*, kann das Schema der Stelle bearbeitet werden. In beiden Fällen gibt der jeweilige Texteditor Rückmeldung, ob Syntaxfehler vorhanden sind und ob die Eingaben konform zum entsprechenden (Meta-)Schema sind.[1] Entsprechend dem hinterlegten (Meta-)Schema gibt der Texteditor auch Vorschläge für Eingaben, etwa zu erlaubten Schlüsseln für ein JSON-Object.[2] Im Data-Tab kann über den Button *Add token* dem JSON-Array der Stellen-Markierung automatisch ein Schema-konformes JSON-Object als Marke hinzugefügt werden. Der JSON-Netz-Editor gibt zunächst für jede Stelle eine Markierung mit einem leeren JSON-Array vor. Als mögliche Marken erlaubt das Standard-Schema leere JSON-Objects. Mit diesen Vorgaben ist eine Markierung vergleichbar mit den anonymen Marken elementarer Petri-Netze (siehe Unterabschnitt 2.3.2). Schema und Markierung können von Anwendenden jedoch angepasst werden.[3]



(a) Ansicht zur Bearbeitung einer Stellenmarkierung. **(b)** Ansicht zur Bearbeitung eines Stellenschemas.

Abbildung 9.11: Ansicht des Bearbeitungsfensters für Stellen.

[1] Wie in der JSON-Netz-Spezifikation (siehe Kapitel 8) vorgesehen, müssen Stellenmarkierungen konform zum Schema der Stelle sein. Das Schema einer Stelle wird in der Schemasprache JSON Schema (siehe Exkurs 8.7) beschrieben. Ein JSON Schema-Ausdruck ist selbst ein JSON-Dokument, das zu einem enstprechenden Meta-Schema konform sein muss.

[2] Die Funktionalitäten für Syntax- und Schemaüberprüfungen wurden mithilfe der Bibliotheken Codemirror [Hav24] Ajv [Pob21] umgesetzt.

[3] Eingaben von Markierungen oder Schemas, die syntaktisch nicht korrekt sind, werden nicht gespeichert.

9.4.4 Transitionen editieren

Das Bearbeitungsfenster für Transitionen ist in drei Bereiche unterteilt. Im linken Bereich können über ein Dropdown-Menü alle Stellen des Vorbereichs der Transition ausgewählt werden. Für die jeweils ausgewählte Stelle wird darunter die entsprechende Markierung angezeigt und eine Bearbeitungsmöglichkeit für den Filterausdruck der zugehörigen Kantenbeschriftung gegeben. Am angezeigten Array einer Stellenmarkierung werden zur Unterstützung der Auswahl einer gültigen Belegung Auswahlbuttons angezeigt. Diese Auswahlbuttons erscheinen an den Werten der Markierung, die anhand des jeweiligen Filterausdrucks ausgewählt werden dürfen. Mit Klick auf einen dieser Auswahlbuttons können Anwendende eine entsprechende Belegung vornehmen. Zur Unterstützung der Anwendenden gibt der Editor den einfachen Filterausdruck `"$.*"` als JSONPath-Ausdruck für Kantenbeschriftungen vor. Dieser JSONPath-Ausdruck wählt alle Kinder des Wurzelarrays einer Stellenmarkierung aus. So würde beim Schalten eine ganze Marke (ein Kindelement des Wurzelarrays) aus der Stellenmarkierung entfernt, was vergleichbar mit dem Verhalten von elementaren Petri-Netzen ist (siehe Unterabschnitt 2.3.2). Der vorgegebene Filterausdruck kann von Anwendenden über Klick auf das Filtersymbol angepasst werden.

Analog zur Darstellung der Stellenmarkierungen des Vorbereichs auf der linken Seite, wird auf der rechten Seite des Bearbeitungsfensters eine Übersicht über die Stellenmarkierungen des Nachbereichs gegeben. Auch hier können wieder einzelne Stellen ausgewählt, Kantenbeschriftungen bearbeitet und Belegungen ausgewählt werden. Zu beachten ist, dass Filterausdrücke im Nachbereich einer Transition die Werte einer Markierung auswählen, in die beim Schalten der Transition neue Werte eingefügt werden sollen (siehe Abschnitt 8.5). Entsprechend ist bei Kantenbeschriftungen des Nachbereichs, abweichend von der Standard-Beschriftung für Kanten im Vorbereich, der JSONPath-Ausdruck `"$"` voreingestellt. Dieser Filterausdruck wählt immer das Wurzelarray einer Stellenmarkierung aus. Damit orientiert sich die Voreinstellung ebenfalls wieder an dem Verhalten eines elementaren Petri-Netzes: beim Schalten wird eine neue Marke (das heißt ein Kindelement des Wurzelarrays) in den Stellen des Nachbereichs erzeugt. Anwendende können die voreingestellte Kantenbeschriftung anpassen, um die Auswahl einzuschränken oder auch das Einfügen von Werten in tiefer verschachtelten JSON-Werten umzusetzen.

In der Mitte des Bearbeitungsfensters können Anwendende die Transitionsinschrift in der Templatesprache Jsonnet spezifizieren. Zur Unterstützung sind die Eingabemasken in Bereiche für die Definition von Hilfsvariablen und -Funktionen (*Preface*), Regeln

zur Ermittlung der Belegung für Schlüssel- und Wertvariablen des Nachbereichs (*Output Expressions*) und einen Bereich für einen logischen Ausdruck (*Guard Expression*) unterteilt. Für jede Output Expression und die Guard Expression werden Standardeinstellungen vorgegeben. So wird Schlüsselvariablen der Wert `"-"` und Wertvariablen der Wert `{}` zugewiesen. Damit wird vorgegeben, dass im Standardfall beim Schalten der Transition leere JSON-Objects in die Wurzelarrays der Markierungen im Nachbereich eingefügt werden. Der voreingestellte Standardwert der Guard Expression ist `true`. Bei jeder Eingabemöglichkeit gibt der JSON-Netz-Editor an den entsprechenden Abschnitten der Benutzungsoberfläche Rückmeldung, ob eine gültige Eingabe vorgenommen wurde beziehungsweise ob die Transition unter der vorgegebenen Belegung aktiviert ist.

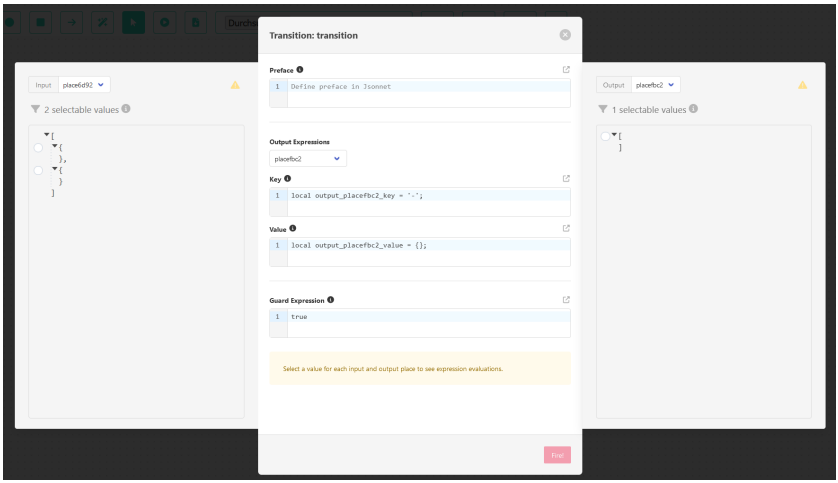


Abbildung 9.12: Ansicht des Bearbeitungsfensters für Transitionen.

Für die verschiedenen genannten Teil-Ausdrücke der Transitionsbeschriftung (Preface, Output Expressions und Guard Expression) steht im Bearbeitungsfenster nur begrenzter Platz zur Verfügung. Daher haben Anwendende die Möglichkeit, jeweils einen Ausdruckseditor in einem separaten Fenster zu öffnen. Dieser Ausdruckseditor gibt Beispiele für Jsonnet-Ausdrücke und stellt einen größeren Bereich für die Formulierung eines Ausdrucks bereit. Unter der Eingabemaske werden Anwendende über

die Gültigkeit und das Ergebnis des eingegebenen Ausdrucks informiert. Zur Unterstützung der Formulierung von Ausdrücken stellt die Anwendung vordefinierte Variablennamen für Schlüssel-, Wert-, und Dokumentvariablen bereit. Diese werden im Ausdruckseditor im Bereich *Variable Explorer* angezeigt und zeigen beim Überfahren mit der Maus ihre aktuelle Belegung an. Die Variablennamen können per Klick in einen Ausdruck eingefügt werden.

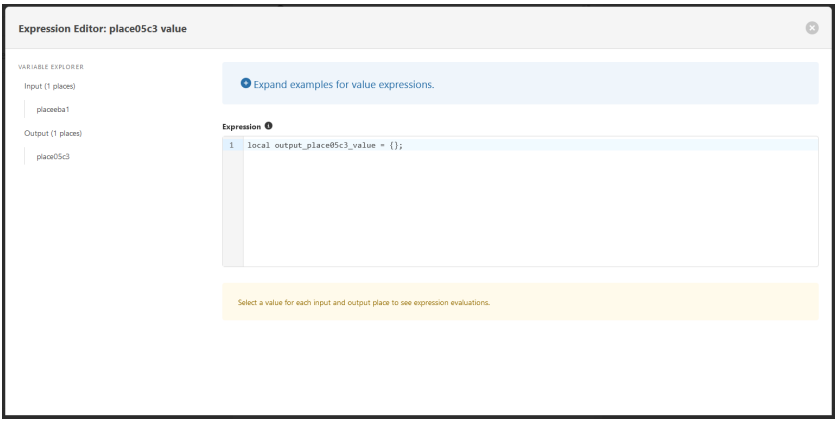


Abbildung 9.13: Ansicht des Ausdruckseditors mit Variablenübersicht.

9.4.5 Transitionen schalten

Bei der Auswahl einer Belegung, unter der die betreffende Transition aktiviert ist, (und korrekter Formulierung einer Transitionsbeschriftung) können Anwendende im Bearbeitungsfenster für Transitionen (siehe Abbildung 9.12) den Button *Fire!* drücken und bekommen unmittelbar die neuen Belegungen der adjazenten Stellen angezeigt. Alternativ können Anwendende den F-Button (siehe Abbildung 9.9c) an einer Transition klicken. In diesem Fall sucht die Anwendung nach einer entsprechenden Belegung und schaltet, falls möglich, automatisch. Der Schaltvorgang wird visuell über die Animation von fließenden Marken entlang der betroffenen Kanten dargestellt. Klicken Anwendende auf den *Ausführen*-Button in der Werkzeugleiste, wird ein einfacher Simulationsalgorithmus ausgeführt. Hierbei wird iterativ für jede Transition geprüft, ob eine Belegung vorhanden ist, unter der die betreffende Transition aktiviert ist. Falls ja, wird die entsprechende Transition geschaltet und der Vorgang wiederholt, bis keine schaltbare Transition mehr gefunden wird.

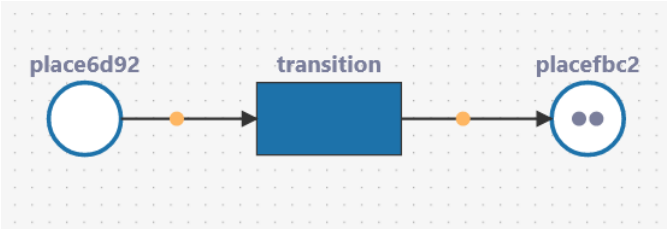


Abbildung 9.14: Animation eines Schaltvorgangs.

9.5 Ergänzende Funktionen

Neben den beschriebenen Hauptfunktionen des JSON-Netz-Editors wurden auf Basis von Rückmeldungen, die in der in Abschnitt 11.3 beschriebenen Fallstudie von Anwendenden eingeholt wurden, ergänzende Funktionen implementiert. Diese ergänzenden Funktionen zielen darauf ab, auch Anwendenden ohne Kenntnis der Syntax von JSON oder der verwendeten JSON-Technologien wie Jsonnet, JSON Schema und JSONPath, die Arbeit mit dem JSON-Netz-Editor zu ermöglichen.

9.5.1 Formulargenerierung

Der JSON-Netz-Editor kann in einen Assistenzmodus umgeschaltet werden (siehe Unterabschnitt 9.5.5), in dem die Stellenmarkierung in einem Formular statt in einem Texteditor bearbeitet werden kann. Diese Funktion wurde mit Hilfe der Bibliothek JSON Forms [Ecl24] umgesetzt. Das Formular wird aus dem jeweiligen Stellenschema generiert und zeigt für unterschiedliche Datentypen entsprechende Formularfelder an.

Abbildung 9.15: Formular zur Bearbeitung einer Stellenmarkierung.

Abbildung 9.15 zeigt beispielhaft ein Formular, das aus dem in Auflistung 9.1 gezeigten Stellenschema generiert wird. Zum Hinzufügen und Entfernen von Kindelementen des Wurzelarrays werden entsprechende +- und x-Buttons angezeigt. Das Schema beschreibt, dass Kindelemente ein Feld `"zahlen"` (vom Typ Number) und ein Feld `"auswahl"` (vom Typ String) haben sollen. Der Typdefinition für `"zahlen"` wird eine Beschreibung (`"description"`) mitgegeben, die im Formular als Hinweistext erscheint. Für das Feld `"auswahl"` sind lediglich die Werte `"a"`, `"b"` oder `"c"` erlaubt. Dementsprechend zeigt das Formular eine Auswahl-Box mit den drei möglichen Werten an.

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "zahlen": { "type": "number", "description": "Eingabefeld für Zahlenwerte." },
      "auswahl": { "type": "string", "enum": ["a","b","c"] }
    }
  }
}
```

Auflistung 9.1: Beispielschema für die Generierung eines Formulars.

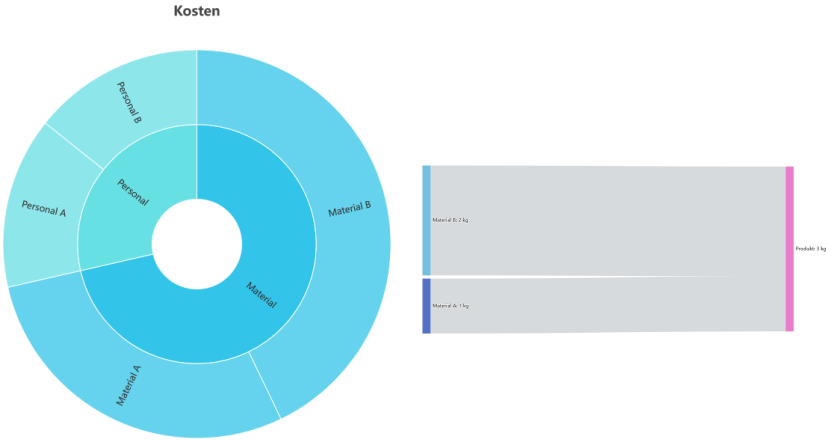
9.5.2 Datenvisualisierung

Eine Funktion zur Generierung von Diagrammen aus Stellenmarkierungen wurde implementiert, um die Interpretation der Daten einer Stellenmarkierung zu unterstützen. Diese Funktion wurde mit Hilfe der Bibliothek ECharts [Apa24] umgesetzt.

```
[{
    "$visualisations": [
      {...},
      {
        "description": "Kosten",
        "type": "sunburst",
        "unit": "Euro",
        "value": [ { "children": [ { "name": "Personal A", "value": 1 }, ... ] } ]
      }
    ]
}]
```

Auflistung 9.2: Beispielmarkierung zur Generierung von Diagrammen.

Auflistung 9.2 zeigt beispielhaft eine (gekürzte) Stellenmarkierung, aus der die in Abbildung 9.16 gezeigten Diagramme generiert wurden. Es wurde die Konvention eingeführt, dass zu visualisierende Daten in einer Marke unter dem Feld `"$visualisations"` gespeichert werden sollen. Im Array unter `"$visualisations"` können mehrere Diagramme definiert werden, wobei für jedes Diagramm der anzuzeigende Diagrammtyp (`"type"`), ein Name (`"name"`), eine Einheit (`"unit"`) und die anzuzeigenden Daten (`"value"`) zu hinterlegen sind. In der in Auflistung 9.2 angegebenen Markierung ist vorgesehen, dass unter anderem ein Sunburst-Diagramm mit dem Namen Kosten zu generieren ist. Dabei soll bei der Anzeige von Zahlenwerten die Einheit Euro hinzugefügt werden. Die im Sunburst-Diagramm zu visualisierenden Daten werden unter dem Feld `value` angegeben. Das generierte Sunburst-Diagramm ist in Abbildung 9.16a zu sehen. Es zeigt verschiedene Beiträge zu einem Gesamtwert (hier Kosten) ähnlich wie in einem Kuchendiagramm an und ordnet dabei die einzelnen Beiträge unterschiedlichen Kategorien zu. In ähnlicher Weise können Daten in einem Sankey-Diagramm angezeigt werden, das sich besonders für die Visualisierung von Stoffströmen eignet (siehe Abbildung 9.16b). ECharts bietet neben Sankey- und Sunburst-Diagrammen eine Vielzahl weiterer Visualisierungsmöglichkeiten, die in Zukunft ebenfalls in den JSON-Netz-Editor integriert werden können.



(a) Beispiel für ein Sunburst-Diagramm.

(b) Beispiel für ein Sankey-Diagramm.

Abbildung 9.16: Datenvisualisierung im JSON-Netz-Editor.

9.5.3 Konfigurationen

Damit Anwendende Stellenschemas, Filter und Transitionsbeschriftungen nicht selbst erstellen müssen, können in einer *Konfiguration* des JSON-Netz-Editors Stellen- und Transitionstypen vordefiniert werden. Anwendende können dann für erstellte Stellen oder Transitionen aus unterschiedlichen, in der jeweiligen Konfiguration vorgegebenen Stellen- und Transitionstypen auswählen. Für Stellen eines bestimmten Stellentyps ist jeweils ein fixes Stellenschema hinterlegt, und nur die Markierung kann (gegebenenfalls formularbasiert, siehe Unterabschnitt 9.5.1) angepasst werden. In ähnlicher Weise geben vordefinierte Transitionstypen eine bestimmte Transitionsbeschriftung und Filter für eingehende und ausgehende Kanten vor. Weiterhin können in einer Konfiguration eine Reihe von Beispiel-Prozessmodellen definiert werden, die Anwendende dann für einen erleichterten Einstieg aus einer Liste im JSON-Netz-Editor aufrufen können. Mit Konfigurationen ist es damit möglich, domänenspezifische Varianten des JSON-Netz-Editors für unterschiedliche Anwendungsfälle zu erstellen. Für das in Kapitel 10 beschriebene Beispielszenario und die in Abschnitt 11.3 beschriebene Fallstudie wurden jeweils eigene Konfigurationen erstellt, die über entsprechende URLs[1] abrufbar sind. Alternative Konfigurationen können als JSON-Datei erstellt und in den JSON-Netz-Editor geladen werden.

9.5.4 Externe Schnittstellen

In einer Konfiguration können Pull- oder Push-Stellentypen zur Interaktion mit externen REST-Schnittstellen[2] definiert werden. Eine Pull-Stelle kann JSON-Daten von einer externen REST-Schnittstelle abrufen. Dabei wird vorausgesetzt, dass die REST-Schnittstelle einen Array zurückgibt und die Kindelemente des Arrays dem vorgegebenen Stellenschema der Pull-Stelle entsprechen. Anwendende können dann aus einer Liste eines der Kindelemente auswählen und als Markierung für die betreffende Stelle festlegen. In ähnlicher Weise stellen Push-Stellen Anwendenden einen „Publish"-Button zur Verfügung. Bei Klick auf diesen Button wird die aktuelle Markierung der

[1] Siehe https://kit-bis.github.io/json-nets/#/scenario/ für das Beispielszenario und https://kit-bis.github.io/json-nets/#/scope3tool/ für die Fallstudie.

[2] Eine Representational State Transfer (REST)-Schnittstelle ist eine Art von Webdienst, der durch einfache Anfragen manipulierbare Daten bereitstellt [Fie00]. Die Daten werden üblicherweise im JSON- oder XML-Datenmodell übermittelt.

Stelle an die hinterlegte URL übertragen. Über Pull- und Push-Stellen können Prozessmodelle im JSON-Netz-Editor mit externen Systemen interagieren – beispielsweise wird in der in Abschnitt 11.3 beschriebenen Fallstudie damit die Weitergabe von berechneten Treibhausgas-Fußabdrücken zwischen Unternehmen einer Elektroniklieferkette umgesetzt.

9.5.5 Assistenzmodus

Die Unterscheidung zwischen einem Assistenz- und einem Fortgeschrittenen-Modus wurde eingeführt, um die Komplexität der Benutzungsoberfläche für Anwendende ohne Kenntnis der Syntax von JSON und der verwendeten JSON-Technologien zu reduzieren. Im Fortgeschrittenen-Modus ist der JSON-Netz-Editor wie in Abschnitt 9.4 beschrieben zu bedienen. Im Assistenzmodus werden die Eingabemöglichkeiten für Filter, Transitionsbeschriftungen und Stellenschemas für Anwendende verborgen. Für einzelne Stellen und Transitionen können dann lediglich die unterschiedlichen (in der jeweiligen Konfiguration vorgegebenen) Stellen- und Transitionstypen ausgewählt werden. Damit können Anwendende unterschiedliche (vorgegebene) Stellenschemas, Transitionsbeschriftungen und Filter festlegen, ohne die jeweilige Syntax zu kennen. Stellenmarkierungen können im Assistenzmodus über die aus den vorgegebenen Schemas generierten Formulare (siehe Unterabschnitt 9.5.1) angepasst werden.

10 Anwendung der Modellierungsmethode

Die Einsatzmöglichkeiten der in dieser Arbeit entwickelten Modellierungsmethode für Nachhaltiges Geschäftsprozessmanagement werden in diesem Kapitel anhand eines Beispielszenarios demonstriert. In Abschnitt 10.1 wird ein beispielhaftes Projekt beschrieben, in dem die Phasen des in Kapitel 7 beschriebenen Vorgehensmodells zur Untersuchung ökologischer Nachhaltigkeitsauswirkungen durchlaufen werden. Eine Untersuchung von sozialen Nachhaltigkeitsauswirkungen wird in Abschnitt 10.2 beschrieben. Möglichkeiten für weitere Iterationen des Vorgehensmodells werden in Abschnitt 10.3 anhand der in Abschnitt 7.2 beschriebenen Verbesserungsmerkmale vorgestellt. Die in diesem Kapitel beschriebenen Prozessmodelle werden in der unter https://kit-bis.github.io/json-nets/#/scenario veröffentlichten Konfiguration des JSON-Netz-Editors als Beispiele bereitgestellt.

10.1 Analyse von ökologischen Nachhaltigkeitsauswirkungen

Im Beispielszenario wird ein fiktives Unternehmen betrachtet, das individuelle 3D-Druck-Artikel produziert und verkauft. Dieses Unternehmen möchte die Nachhaltigkeit seiner Geschäftsprozesse analysieren und verbessern. Es folgt dabei dem Vorgehensmodell aus Kapitel 7 und nutzt zur Modellierung die in Kapitel 8 vorgestellten JSON-Netze mit dem in Kapitel 9 beschriebenen JSON-Netz-Editor als Modellierungswerkzeug. Die folgenden Unterabschnitte entsprechen den einzelnen Phasen des Vorgehensmodells.

10.1.1 Geschäftsprozessidentifikation

Wie in Abschnitt 7.1 beschrieben, sind die Phasen des Vorgehensmodells nicht streng sequentiell zu verstehen. Je nach Anwendungsfall können einzelne Phasen auch wiederholt oder die zeitliche Reihenfolge angepasst werden. Um verschiedene Anwendungsmöglichkeiten des JSON-Netz-Editors zu demonstrieren, wird in der Beschreibung des Beispielszenarios die Phase Geschäftsprozessidentifikation vorgezogen. In der Phase Geschäftsprozessidentifikation modellieren Anwendende einen oder mehrere zu untersuchende Geschäftsprozesse.

Auswahl eines Geschäftsprozesses: Das 3D-Druck-Unternehmen entscheidet sich, zunäct den Prozess der Bearbeitung von Bestellungen zu untersuchen. Damit wird das Ziel verfolgt, gegenüber den eigenen Kundinnen und Kunden aussagefähig bezüglich der Nachhaltigkeit einer Bestellung zu werden. Eine Ausführung des Geschäftsprozesses beginnt damit, dass eine neue Bestellung eingeht. Zunächst wird überprüft, ob der bestellte Artikel im Lager verfügbar ist. Falls der Artikel verfügbar ist, wird er reserviert und aus dem Lager geholt. Falls nicht, wird der Artikel produziert. Wurde der Artikel schließlich aus dem Lager entnommen oder produziert, wird der Versand vorbereitet. Dazu muss sowohl der Artikel transportiert werden, als auch eine Rechnung erstellt und verschickt werden. Sind Transport und Rechnungsversand abgeschlossen, endet der Prozess.[1]

Modellierung des Geschäftsprozesses: Zur Modellierung des Geschäftsprozesses wird zunächt im JSON-Netz-Editor eine entsprechende Netzstruktur erstellt (siehe Abbildung 10.1). In der Standardkonfiguration des JSON-Netz-Editors können damit bereits verschiedene Zustände einer Prozessinstanz dargestellt werden. Im Modell in Abbildung 10.1 enthält die Stelle $p0$ ein leeres Object als Marke, die den Eingang einer neuen Bestellung abbildet. Beim Schalten einer Transition wird dann jeweils eine Marke in den Stellen im Vorbereich einer Transition entfernt und ein leeres Object als Marke in den Stellen im Nachbereich eingefügt. So lassen sich durch schrittweises Schalten der Transitionen bereits unterschiedliche Ausführungsreihenfolgen der Aktivitäten des Geschäftsprozesses darstellen. Mögliche Schaltfolgen für eine Startmarkierung mit einem leeren Object in der Stelle $p0$ wären beispielsweise $t0 \to t1 \to t4 \to t5 \to t6 \to t7 \to t8$ oder $t0 \to t2 \to t3 \to t4 \to t5 \to t6 \to t7 \to t8$. Die erste Schaltfolge stellt den Fall dar, dass für eine Prozessinstanz ein angeforderter Artikel nicht

[1] Die Prozessbeschreibung orientiert ist an ein Referenz-Geschäftsprozessmodell in [Sch94] angelehnt, das in ähnlicher Weise auch in der Arbeit [Hou11] zur Illustration des dort vorgestellten Ansatzes für Nachhaltiges Geschäftsprozessmanagement verwendet wurde.

```
{
  "type": "array",
    "items": {
      "type": "object",
      "properties": { "instanceID": { "type": "string" } }
  }
}
```

Auflistung 10.1: JSON Schema für Prozessinstanzmarken.

verfügbar ist und zunächst produziert wird. Im zweiten Fall ist ein entsprechender Artikel im Lager verfügbar und kann aus dem Lager geholt werden. Das resultierende Prozessmodell mit einfacher Markierung kann unter https://kit-bis.github.io/json-nets/#/scenario/Netzstruktur eingesehen werden.

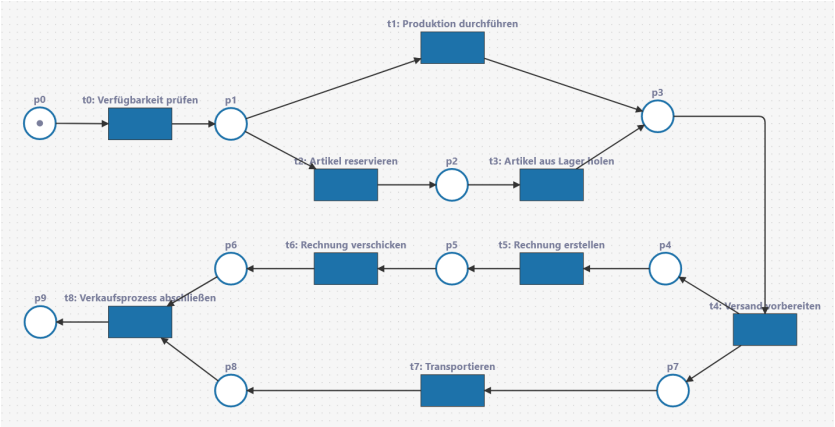


Abbildung 10.1: Prozessmodell der Auftragsbearbeitung.

Es können Objects mit Instanz-IDs als Felder in den Stellenmarkierungen eingeführt werden, um Marken unterschiedlichen Prozessinstanzen zuordnen zu können. Auflistung 10.1 zeigt ein entsprechendes JSON Schema zur Typisierung der Stellen des Prozessmodells. Es schreibt vor, dass Marken in den Stellen nun nicht mehr leer sind, sondern ein Feld `instanceID` haben sollen. Dieses JSON Schema kann im JSON-Netz-Editor jeder Stelle des Prozessmodells zugeordnet werden. Marken, die Instanz-IDs tragen, werden im Folgenden *Prozessinstanzmarken* genannt.

```
1  [ { "instanceID": "i0" }, { "instanceID": "i1" } ]
```

Auflistung 10.2: Prozessinstanzmarken mit unterschiedlichen Instanz-IDs.

Eine mögliche Anfangsmarkierung für die Stelle $p0$ ist in Auflistung 10.2 gegeben. Es zeigt zwei Prozessinstanzmarken mit unterschiedlichen Instanz-IDs. So können zwei unterschiedliche Bestellungen dargestellt werden. Die Anfangsmarkierung kann im JSON-Netz-Editor entsprechend angepasst werden.

Auflistung 10.3 zeigt beispielhaft eine Output Expression[1] für die Transition $t0$. Die Output Expression bewirkt, dass beim Schalten von $t0$ die in $p1$ eingefügte Prozessinstanzmarke der in Stelle $p0$ entfernten Prozessinstanzmarke entspricht. Entsprechende Output Expressions sind bei allen Transitionen für die jeweiligen Stellen im Nachbereich vorzunehmen.

```
1  local output_p1_value = input_p0_value;
```

Auflistung 10.3: Output Expression für Transition $t0$.

An der Transition $t4$ spaltet sich der Prozesspfad in zwei nebenläufige Teilpfade, die in der Transition $t8$ wieder synchronisiert werden. Ohne weitere Anpassungen könnten Belegungen auftreten, bei denen die Transition $t8$ mit Marken unterschiedlicher Prozessinstanzen schaltet. Im Prozessmodell könnte zum Beispiel in der Stelle $p6$ eine Prozessinstanzmarke mit `instanceID = "i0"` und in der Stelle $p8$ eine Prozessinstanzmarke mit `instanceID = "i1"` liegen. Würde Transition $t8$ mit dieser Belegung schalten, entspräche das dem Abschluss eines Verkaufsprozesses, bei dem eine Rechnung verschickt wurde, die nicht zu der transportierten Ware gehört. Auflistung 10.4 zeigt eine Guard Expression für die Transition $t8$, mit der derartige Konstellationen vermieden werden. Sie legt fest, dass die Transition nur schalten kann, wenn die Instanz-IDs der ausgewählten Prozessinstanzmarken im Vorbereich übereinstimmen. So wird sichergestellt, dass eine Prozessinstanz erst dann abgeschlossen wird, wenn sowohl die entsprechende Rechnung verschickt wurde, als auch die zugehörige Ware (und nicht eine beliebige andere Ware) transportiert wurde.

[1] Die Einteilung von Transitionsbeschriftungen in Preface, Output Expressions und Guard Expressions wurde in Unterabschnitt 9.4.4 eingeführt.

```
1  input_p6_value.instanceID == input_p8_value.instanceID
```

Auflistung 10.4: Guard Expression für Transition $t8$.

Nachdem die hier beschriebenen Anpassungen vorgenommen wurden, lassen sich im JSON-Netz-Editor die Zustände unterschiedlicher Prozessinstanzen darstellen und mögliche Abläufe simulieren. Ein Beispiel-Prozessmodell, das mit unterscheidbaren Prozessinstanzmarken arbeitet, kann unter https://kit-bis.github.io/json-nets/#/scenario/Prozessinstanzen eingesehen werden.

10.1.2 Spezifikation des Indikatormodells

Gemeinsam mit Nachhaltigkeits- und Branchenexpertinnen und -experten werden zur Spezifikation des Indikatormodells die zu betrachtenden Wirkungskategorien ausgewählt sowie die zugehörigen Inventarindikatoren und Systemgrenzen beschrieben.

Beschreibung des Indikatormodells: Aufgrund der hohen gesellschaftlichen Relevanz und um eingehende Anfragen von Kundinnen und Kunden zum Treibhausgas-Fußabdruck der verkauften Artikel beantworten zu können, wird zunächst die Wirkungskategorie Klimawirkung ausgewählt. Im Projekt beteiligte Personen mit Nachhaltigkeitsexpertise empfehlen zur Ausarbeitung eines zugehörigen Indikatormodells, sich am Treibhausgasprotokoll [WBC04] zu orientieren. Demnach ist für den Geschäftsprozess ein Wirkungsindikator in der Einheit kg CO2e anzugeben. Als mögliche Systemgrenzen unterscheidet das Treibhausgasprotokoll zwischen Scope 1, Scope 2 und Scope 3 (siehe auch die Erläuterungen zum Treibhausgasprotokoll in Abschnitt 3.2). Die beteiligten Personen mit Nachhaltigkeits- und Branchenexpertise empfehlen, bei der Ausarbeitung des Indikatormodells in der ersten Iteration des Vorgehensmodells zunächst auf Scope 2 und Scope 3 zu fokussieren, da sie für die Geschäftsprozesse des betreffenden Unternehmens nur geringe Scope 1-Emissionen erwarten. Scope 2-Emissionen sind indirekte Treibhausgasemissionen, die aus dem Erzeugen der vom Unternehmen eingekauften Energie stammen. Zur Ermittlung der Scope 2-Emissionen müssen die Energiebedarfe der einzelnen ausgeführten Aktivitäten ermittelt werden. Wenn bekannt ist, aus welchen Quellen die genutzte Energie stammt, kann dann ein Wirkungsfaktor in kg CO2e pro Energieeinheit angegeben werden. Scope 3-Emissionen sind sonstige indirekte Treibhausgasemissionen, die

außerhalb des betreffenden Unternehmens anfallen. Dazu zählen insbesondere Treibhausgasemissionen, die bei der Produktion der in einem Geschäftsprozess genutzten Ressourcen angefallen sind.

Anpassung des Modellierungswerkzeugs: Auf Grundlage der vorliegenden natürlichsprachlichen Beschreibung des Indikatormodells kann eine im Projekt beteiligte Person mit Werkzeugexpertise den JSON-Netz-Editor zur Analyse der Nachhaltigkeitsauswirkungen der Geschäftsprozesse des Unternehmens vorbereiten. Dazu werden die bestehenden Strukturbeschreibungen, Markierungen und Transitionsbeschriftungen im JSON-Netz-Editor angepasst. In den zuvor eingeführten Prozessinstanzmarken kann zusätzlich zu den Instanz-IDs ein Feld `impact` ergänzt werden, das als Wirkungsindikator Auskunft über die ermittelten Treibhausgasemissionen einer zugehörigen Prozessinstanz geben soll. Auflistung 10.5 zeigt, wie die angepasste Startmarkierung für die Stelle $p0$ aussehen könnte, bei der zunächst für jede Prozessinstanz der Wirkungsindikator-Wert 0 angegeben wird. Die Stellenschemas der Stellen $p0$ bis $p9$ sind entsprechend anzupassen.

```
1  [ { "instanceID": "i0", "impact": 0 }, { "instanceID": "i1", "impact": 0 } ]
```

Auflistung 10.5: Prozessinstanzmarken mit Wirkungsindikatorwerten.

Zur Erfassung der einer Aktivität zuzuordnenden Treibhausgasemissionen wird in einer Konfiguration des JSON-Netz-Editors ein spezieller Stellentyp (siehe Unterabschnitt 9.5.3) vorbereitet. Entsprechende Stellen, die Daten zu Nachhaltigkeitsauswirkungen einer Aktivität erfassen, werden im Folgenden *Inventarstellen* genannt. Auflistung 10.6 zeigt ein Stellenschema für eine Inventarstelle, die Daten zu Treibhausgasemissionen erfasst. Das Stellenschema legt fest, dass eine Mengenangabe als Inventarindikator (`amount`), ein Wert für einen Wirkungsfaktor (`impactFactor`) und ein Wert für einen Allokationsfaktor (`allocationFactor`) zu erfassen sind. Das Feld `amount` dient als Inventarindikator dazu, Mengenangaben zu (klimawirkungs-)relevanten Inputs und Outputs zu erfassen (zum Beispiel Energiebedarfe, Materialien oder Maschinen). Über das Feld `impactFactor` wird angegeben, welche Klimawirkung (gemessen in kg CO2e) einer Mengeneinheit des Inventarindikators zugeordnet werden kann. Mit dem Feld `allocationFactor` wird der Anteil der Gesamtemissionen der Inventarstelle festgelegt, der einer Aktivitätsausführung angerechnet werden soll. Weiterhin wird im Feld `scope` festgehalten, ob es sich um eine Angabe für Scope 1-, Scope 2-, oder

```
1  {
2    "type": "array",
3    "items": {
4      "type": "object",
5      "properties": {
6        "amount": { "type": "number" },
7        "impactFactor": { "type": "number" },
8        "allocationFactor": { "type": "number" },
9        "scope": { "type": "number", "enum": [1,2,3] }
10     }
11   }
12 }
```

Auflistung 10.6: JSON Schema für eine Inventarstelle.

Scope 3-Emissionen handelt. Das beschriebene Schema kann in der Phase Datenerhebung und -integration verwendet werden, um einzelnen Aktivitäten Inventarstellen zuzuordnen, in denen die zugehörigen Emissionsdaten erfasst werden.

Die Berechnung der Gesamtemissionen einer Aktivitätsinstanz wird in den Transitionsbeschriftungen umgesetzt. Zur Definition entsprechender Funktionen kann der Preface-Bereich für Transitionsbeschriftungen im JSON-Netz-Editor genutzt werden. Auflistung 10.7 zeigt einen Ausschnitt des Preface-Bereichs einer entsprechenden Transitionsbeschriftung. In den Zeilen 2-8 ist eine Funktion `calculateContribution` definiert, die für eine gegebene Input-Marke den zugehörigen Wirkungsindikator ermittelt. Für eine Prozessinstanzmarke wird der im Feld `impact` gespeicherte Wert ausgegeben. Für Marken aus Inventarstellen wird der zugehörige Beitrag aus der Multiplikation des angegebenen Energieverbrauchs (`amount`) mit dem zugehörigen Wirkungsfaktor (`impactFactor`) und Allokationsfaktor (`allocationFactor`) ermittelt. Die berechneten Einzelbeiträge werden der Variable `contributions` in den Zeilen 9 und 10 als Array zugewiesen. Die Werte des Arrays werden in den Zeile 11 und 12 aufaddiert und der Variablen `impact` zugewiesen. Für Transitionen mit mehreren Output-Marken, wie $t4$ im gegebenen Prozessmodell, muss eine Allokation der Gesamtemissionen auf die erzeugten Marken stattfinden. In der Variable `outputAllocation` in Zeile 1 kann die Anzahl Output-Marken festgehalten werden, auf welche die berechneten Gesamtemissionen aufgeteilt werden sollen. In der Transitionsbeschriftung für $t4$ kann der Wert der Variable `outputAllocation` entsprechend auf 2 gesetzt werden. Auf diese Weise kann sichergestellt werden, dass beim Zusammenführen der beiden nebenläufigen Prozesspfade in $t8$, die zuvor berechneten Beiträge nicht doppelt gezählt werden.

```
local outputAllocation = 1;
local calculateContribution(token) =
  if isInstanceToken(token) then
    token.impact
  else
    token.amount * token.impactFactor * token.allocationFactor;
local contributions = std.map(calculateContribution, input_values);
local impact = sum(contributions) / outputAllocation;
```

Auflistung 10.7: Aussschnitt des Preface-Bereichs einer Transitionsbeschriftung.

Damit der berechnete Wirkungsindikator-Wert den erzeugten Marken zugewiesen wird, sind noch die Output Expressions der Transitionsbeschriftungen anzupassen. Auflistung 10.8 zeigt die angepasste Output Expression für Transition $t0$, die besagt, dass die in $p1$ zu erzeugende Prozessinstanzmarke der Wert der aus $p0$ entnommenen Marke zugewiesen wird. Das bedeutet, dass die neue Prozessinstanzmarke in $p1$ den gleichen Wert für `"instanceID"` trägt, wie die in $p0$ entnommene Prozessinstanzmarke. Dem Feld `"impact"` der neuen Prozessinstanzmarke wird der in der Variable `impact` gespeicherte Wert zugewiesen. Die vollständige Transitionsbeschriftung sowie die beschriebenen Stellenschemas sind unter https://kit-bis.github.io/#/json-nets/scenario/ einsehbar.

```
local output_p1_value = input_p0_value + { "impact": impact };
```

Auflistung 10.8: Output Expression für Wirkungsindikator-Werte.

10.1.3 Datenerhebung und -integration

In der Phase Datenerhebung und -integration geht es darum, die (für die zu untersuchenden Wirkungskategorien) relevanten Inventarindikatoren, Wirkungsfaktoren und Allokationsfaktoren zu ermitteln. Das Unternehmen entscheidet sich dazu, für die einzelnen Aktivitäten Durchschnittswerte für Energie- und Treibstoffverbräuche

zu ermitteln. Tabelle 10.1 gibt einen Überblick zu den im Unternehmen ermittelten Daten zu Scope 2-Emissionen.[1] Die Aktivitäten *t0: Verfügbarkeit prüfen*, *t2: Artikel reservieren*, *t4: Versand vorbereiten*, *t5: Rechnung erstellen*, *t6: Rechnung verschicken* und *t8: Verkaufsprozess abschließen* werden auf unternehmensinterner IT-Infrastruktur ausgeführt. Der Ausführung einer Aktivität ist für die durchschnittliche Dauer der Beanspruchung der IT-Infrastruktur ein Energieverbrauch von 0,000556 kWh zuzuordnen. Bei der Aktivität *t1: Produktion durchführen* wird festgestellt, dass der Energieverbrauch des genutzten 3D-Druckers je hergestelltem Artikel im Durchschnitt 0,24 kWh beträgt. Zur Wirkungsabschätzung der ermittelten Energiebedarfe wird für den genutzten Strommix ein Wirkungsfaktor von 0,38 kg CO2e/kWh festgestellt.

Mit den bisher ermittelten Daten ist zu erwarten, dass Prozessinstanzen, bei denen ein Artikel zu produzieren ist, weniger Energie benötigen und damit „nachhaltiger" erscheinen, als Prozessinstanzen, bei denen lediglich ein vorhandener Artikel aus dem Lager geholt wird. Im Kontext des vom Unternehmen gesetzten Ziels, Kundinnen und Kunden über die Nachhaltigkeit einer Lieferung informieren zu können, ist dieses Ergebnis zu hinterfragen. In beiden Fällen wird ein Artikel produziert, nur geschieht das im einen Fall unmittelbar bei der Ausführung des Prozesses, und im anderen Fall zu einem früheren Zeitpunkt. Den Artikeln im Lager können *eingebettete* Emissionen zugeordnet werden, um diesen Sachverhalt abzubilden. Diese eingebetteten Emissionen werden in einer zusätzlichen mit $t3$ verbundenen Inventarstelle erfasst. Die eingebetteten Emissionen entsprechen den Emissionen, die bei der Produktion eines Artikels anfallen. Im gegebenen Fall wird daher von 0,0912 kg CO2e pro Artikel ausgegangen (0,24 kWh × 0,38 kg CO2e/kWh).

Tabelle 10.1: Datenerhebung zu Scope 2-Emissionen.

Aktivität	Inventarindikator	Wirkungsfaktor
t0, t2, t4, t5, t6, t8 (Nutz. der IT-Inf.)	0,000556 kWh	0,38 kg CO2e/kWh
t1 (Produktion durchführen)	0,24 kWh	0,38 kg CO2e/kWh
t3 (Eingebettete Treibhausgas-Emissionen)	1 Stück	0,0912 kg CO2e/Stück

Bezüglich der Scope 3-Emissionen wird ermittelt, dass bei der Produktion der verwendeten IT-Infrastruktur 800,3 kg CO2e angefallen sind. Als Allokationsfaktor für die Anrechnung der Gesamtemissionen auf eine Aktivitätsausführung wird 0,00000012

[1] Bei den angegebenen Werten handelt es sich um für das Beispielszenario getroffene Annahmen basierend auf [Umw24, Boa24]. Vergleichbare Werte können aus kommerziellen Lebenszyklusanalysedatenbanken gewonnen werden [Klö09]. Neben den hier zur Illustration beschriebenen Inventarindikatoren sind für eine vollständige Analyse gegebenenfalls noch weitere Inventarindikatoren zu berücksichtigen.

angenommen (ein entsprechender Wert wird ebenfalls als eingebettete Scope 3-Emission für *t*3 angenommen). Für den Transport der Ware (*t7: Transportieren*) wird festgestellt, dass die Ware im Durchschnitt eine Strecke von 300 km beim Transport zurücklegt und 100g wiegt. Daraus ergibt sich ein Inventarindikator von 0,03 tkm. Als Wirkungsfaktor wird ermittelt, dass der Transport von einer Tonne Fracht auf einen Kilometer 0,0532 kg CO2e verursacht.[1]

Abbildung 10.2 zeigt das resultierende Modell mit ergänzten Inventarstellen für die erhobenen Daten. An den ergänzten Inventarstellen werden die zusammengetragenen Daten jeweils entsprechend dem in Auflistung 10.6 gezeigten JSON Schema eingetragen. Das resultierende Prozessmodell kann unter https://kit-bis.github.io/json-nets/#/scenario/Nachhaltigkeitsanalyse-THG eingesehen werden.

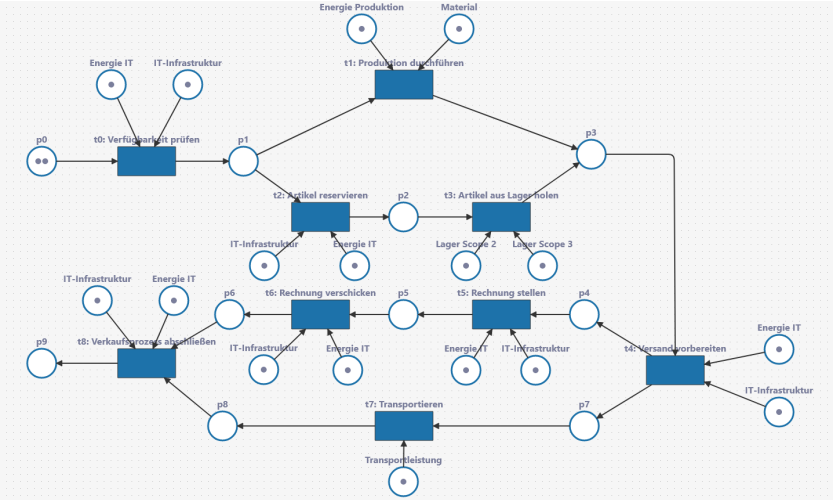


Abbildung 10.2: Prozessmodell mit Inventarstellen für die Datenerhebung.

[1] Es wird angenommen, dass der Transport von einem externen Dienstleister erbracht wird, und damit unter Scope 3 des Treibhausgasprotokolls fällt. Würde der Transport vom Unternehmen selbst mit eigenen Fahrzeugen durchgeführt werden, wären die resultierenden Treibhausgasemissionen Scope 1 zuzuordnen [WBC04].

10.1.4 Analyse und Verbesserung

Mit den ermittelten Daten können Analysen durchgeführt und Verbesserungsmöglichkeiten identifiziert werden. Zur Analyse wird beispielhaft für $i0$ eine Prozessinstanz, bei der ein Artikel neu produziert wird (Schaltfolge der Transitionen enthält $t1$), simuliert und für $i1$ der Versand eines im Lager vorhandenen Artikels (Schaltfolge der Transitionen enthält $t2$ und $t3$).[1] Auflistung 10.9 zeigt die resultierende Markierung in $p9$.

```
1  [
2    { "impact": 0.15953, "instanceID": "i0" },
3    { "impact": 0.15984, "instanceID": "i1" }
4  ]
```

Auflistung 10.9: Resultierende Markierung von $p9$ nach Durchführung der Simulation.

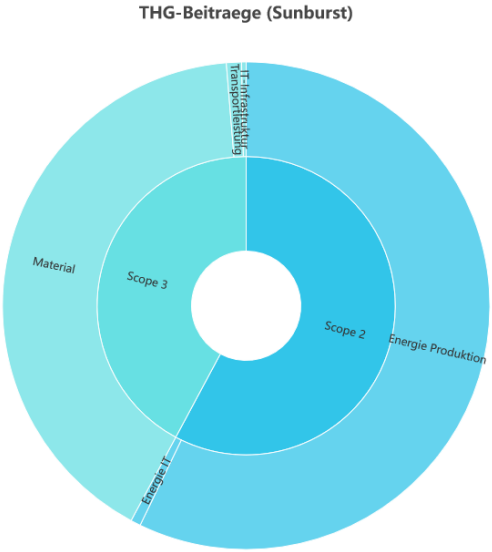


Abbildung 10.3: Beiträge zu den Gesamtemissionen der Prozessinstanz $i0$.

[1] Es wird angenommen, dass vor jedem Simulationsdurchlauf die Markierung der Inventarstellen zurückgesetzt wird.

Die resultierende Markierung in $p9$ (siehe Auflistung 10.9) zeigt, dass die Prozessinstanz, in der ein Artikel aus dem Lager geholt wird ($i1$) eine leicht höhere Klimawirkung aufweist, da hier die Energiebedarfe für $t2$ zusätzlich zu den eingebetteten Emissionen aus der Produktion anfallen. Die vom JSON-Netz-Editor bereitgestellte Visualisierung (siehe Abbildung 10.3) zeigt, dass die größten Beiträge zur Klimawirkung durch das verwendete Material (in Scope 3) und die für die Produktion benötigte Energie (in Scope 2) verursacht werden.

Verbesserungspotentiale ergeben sich damit beispielsweise durch die Umstellung auf Strom aus erneuerbaren Energien für die Produktion. Diese Verbesserungsmaßnahme lässt sich im Prozessmodell dadurch abbilden, dass der Wirkungsfaktor für die Inventarstelle *Energie Produktion* entsprechend reduziert wird.

10.2 Analyse von sozialen Nachhaltigkeitsauswirkungen

Nach der ersten Iteration des Vorgehensmodells entscheidet sich das 3D-Druck-Unternehmen dazu, den Umfang der vorgenommenen Nachhaltigkeitsanalysen zu erweitern und nun auch soziale Nachhaltigkeitsauswirkungen mit in den Blick zu nehmen. So sollen Verbesserungsmaßnahmen identifiziert werden, mit denen negative soziale Auswirkungen der Geschäftsprozesse des Unternehmens reduziert werden können. In diesem Abschnitt wird die entsprechende Durchführung der Phasen *Spezifikation des Indikatormodells*, *Datenerhebung und -integration* sowie *Analyse und Verbesserung* beschrieben.

10.2.1 Spezifikation des Indikatormodells

Für die Auswahl zu untersuchender sozialer Nachhaltigkeitsauswirkungen werden die in Richtlinien für soziale Lebenszyklusanalysen [UNE20, UNE21] definierten Wirkungskategorien herangezogen. Diese sind nach verschiedenen Anspruchsgruppen wie zum Beispiel Arbeitende, lokale Gemeinschaften oder Kundinnen und Kunden strukturiert. Beispiele für soziale Wirkungskategorien, die Arbeitende betreffen, sind Kinderarbeit, Arbeitszeiten oder Gesundheit und Arbeitsschutz. Im Folgenden wird beispielhaft die Wirkungskategorie Gesundheit und Arbeitsschutz betrachtet. [UNE21] nennt *Anzahl Unfälle* als einen Indikator zur Messung dieser Wirkungskategorie. Zur Definition der Systemgrenzen kann auf [OEC22] zurückgegriffen werden. Hier werden vergleichbar mit dem Treibhausgasprotokoll „Social Scopes“ für soziale

Nachhaltigkeitsauswirkungen definiert. Diese Social Scopes orientieren sich zwar an der Namensgebung im Treibhausgasprotokoll [WBC04], sind aber nicht deckungsgleich definiert. Diese Unterschiede sind darin begründet, dass für unterschiedliche Wirkungskategorien verschiedene Systemgrenzen relevant sind. Scope 1 betrifft nach der Definition in [OEC22] Auswirkungen auf das Wohlergehen innerhalb der Unternehmensgrenzen, insbesondere Angestellte. Scope 2 betrifft Auswirkungen, die vom Unternehmen bereitgestellte Produkte und Dienstleistungen auf Konsumentinnen und Konsumenten haben. Scope 3 betrifft Angestellte und sonstige Anspruchsgruppen in der Lieferkette des Unternehmens.

Die im Projekt beteiligten Personen mit Branchen- und Nachhaltigkeitsexpertise empfehlen, die Systemgrenze der Nachhaltigkeitsanalyse zunächst auf Scope 1 und Scope 3 zu beschränken, da die relevantesten sozialen Nachhaltigkeitsauswirkungen in der Produktion im Unternehmen (Scope 1) und entlang der Lieferkette (Scope 3) vermutet werden. Im Kontext der hier vorgestellten Methode kann die Arbeitsdauer als Inventarindikator interpretiert werden [UNE20]. Mit den zu ermittelnden Unfallraten pro Stunde für unterschiedliche Aktivitäten als Wirkungsfaktoren kann so ein Wirkungsindikator pro Aktivitätsausführung berechnet werden. Zur Eingabe und Verarbeitung der zu ermittelnden Inventarindikatoren, Wirkungsfaktoren, Wirkungsindikatoren und Allokationsfaktoren können die in Unterabschnitt 10.1.2 vorgestellten Stellenschemas und Transitionsbeschriftungen verwendet werden.

10.2.2 Datenerhebung und -integration

Das Unternehmen ermittelt für die verschiedenen Aktivitäten durchschnittliche Arbeitsdauern (als Inventarindikatoren) und durchschnittliche Unfallraten (als Wirkungsfaktoren).[1] In Tabelle 10.2 sind die erfassten Daten für Scope 1 der sozialen Nachhaltigkeitsanalyse des Geschäftsprozesses dargelegt. Es wird durchschnittlich von einer Arbeitsdauer von einer Minute je Ausführung der Aktivitäten $t0$, $t2$, $t3$, $t4$, $t5$, $t6$ und $t8$ ausgegangen. Für die Produktions-Aktivität ($t1$) wird von fünf Minuten Arbeitsdauer ausgegangen. Als Wirkungsfaktor wird für Büroarbeit eine geringere Rate an Unfällen pro Stunde ermittelt als für die Arbeit am 3D-Drucker ($t1$).

[1] Bei den angegebenen Werten handelt es sich um für das Beispielszenario getroffene Annahmen basierend auf [Ver23, Fai24, ILO24]. Vergleichbare Werte können aus kommerziellen Lebenszyklusanalysedatenbanken gewonnen werden [Gre24b]. Neben den hier zur Illustration beschriebenen Inventarindikatoren sind für eine vollständige Analyse gegebenenfalls noch weitere Inventarindikatoren zu berücksichtigen.

Tabelle 10.2: Datenerhebung für Scope 1 der sozialen Analyse.

Aktivität	Inventarindikator	Wirkungsfaktor
t0, t2, t3, t4, t5, t6, t8 (Büro)	1 Minute	0,00000874 Unfälle pro Stunde
t1 (Produktion)	5 Minuten	0,0000106 Unfälle pro Stunde

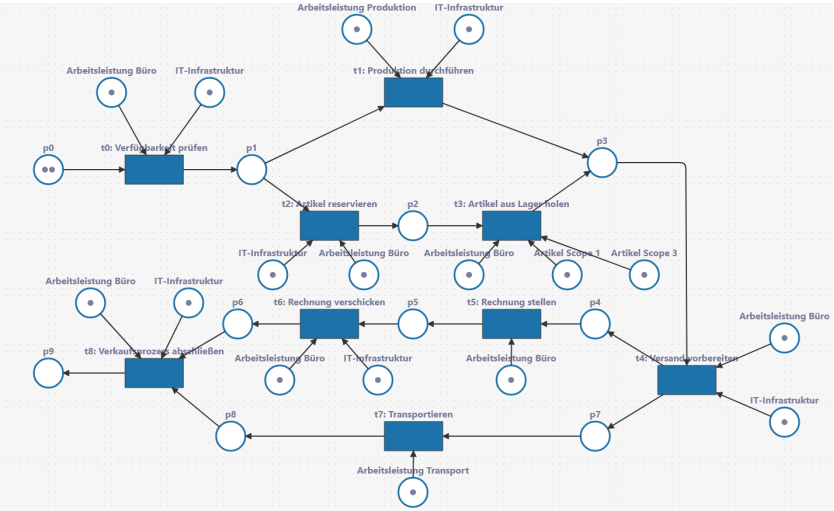


Abbildung 10.4: Prozessmodell mit Inventarstellen für die Datenerhebung.

Für soziale Nachhaltigkeitsauswirkungen in Scope 3 wird zunächst (für $t7$) eine Arbeitsdauer von 3,5 h (als durchschnittliche Dauer für den Transport und damit benötigte Arbeitszeit) ermittelt. Als durchschnittliche Unfallrate für diese Art von Arbeit werden 0,000018 Unfälle pro Stunde ermittelt. Da beim Transport nicht nur der betreffende Artikel, sondern auch weitere Ladung transportiert wird, wird ein Allokationsfaktor von 0,0001 angenommen. In Scope 3 liegen neben der sozialen Auswirkungen für die eingekaufte Transport-Dienstleistung auch solche, die in eingekauften Ressourcen wie der genutzten IT-Infrastruktur, eingebettet sind. Als Anzahl Unfälle, die durch die Produktion der genutzten IT-Infrastruktur bedingt sind, kann der Wert 0,000450133 ermittelt werden. Dieser Wert wird schließlich anteilig auf einzelne Aktivitätsausführungen angerechnet. Als Allokationsfaktor für die Aufteilung der eingebetteten sozialen Auswirkungen auf einzelne Aktivitätsausführungen wird der Wert 0,0000585206 ermittelt. Wie zuvor bei der ökologischen Nachhaltigkeitsanalyse werden für den Fall, dass ein Artikel aus dem Lager geholt wird, die in den Artikel

eingebetteten Scope 1 und Scope 3-Nachhaltigkeitsauswirkungen erfasst. Das resultierende Prozessmodell (siehe Abbildung 10.4) kann unter https://kit-bis.github.io/json-nets/#/scenario/Nachhaltigkeitsanalyse-Sozial eingesehen werden.

10.2.3 Analyse und Verbesserung

Nachdem die Daten erhoben und integriert wurden, wird das Prozessmodell wieder für zwei Prozessinstanzen, einmal für den Versand eines neu produzierten Artikels (Prozessinstanz $i0$) und einmal für den Versand eines im Lager vorhandenen Artikels (Prozessinstanz $i1$) simuliert. Für $i0$ ergibt die Analyse einen Gesamtwert von 0,000003 Unfällen und für $i1$ 0,000004 Unfällen. Abbildung 10.5 zeigt die Beiträge der einzelnen erfassten Inventarstellen zum Gesamt-Wirkungsindikatorwert von $i0$. Die Visualisierung zeigt, dass ungefähr die Hälfte der Gesamtauswirkungen jeweils Scope 1 und Scope 3 zuzuordnen sind. Den ermittelten sozialen Auswirkungen in Scope 1 kann mit Arbeitsschutzmaßnahmen im Unternehmen begegnet werden. Auf die sozialen Auswirkungen in Scope 3, die in der IT-Infrastruktur eingebettet sind, kann nicht direkt eingewirkt werden. Hier muss mit liefernden Unternehmen interagiert werden und gegebenenfalls beim Einkauf von IT-Infrastruktur gegengesteuert werden.

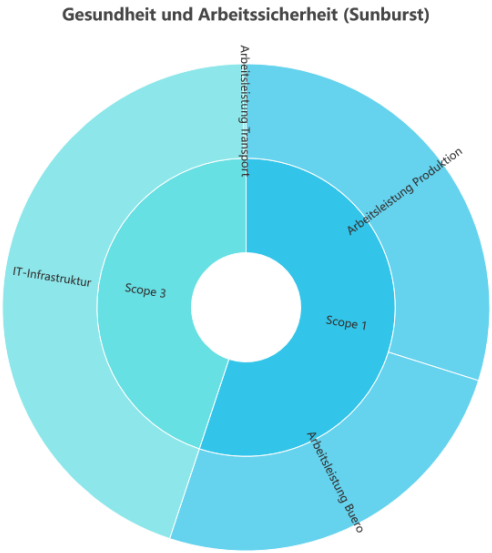


Abbildung 10.5: Beiträge zu den sozialen Auswirkungen der Prozessinstanz $i0$.

10.3 Verbesserungsmöglichkeiten

Das Vorgehensmodell sieht vor, die Validität der Analyse sowie die tatsächlichen Nachhaltigkeitsauswirkungen iterativ zu verbessern. Dazu wurden in Kapitel 7 die Verbesserungsmerkmale Prozessabdeckung, Granularität, Nachhaltigkeitsaspekte, Systemgrenzen, Datenqualität, Nachhaltigkeitsauswirkungen und Integrationstiefe beschrieben. Im Folgenden wird für jedes Verbesserungsmerkmal beschrieben, wie sich das Unternehmen im Beispielszenario ausgehend von der im letzten Abschnitt beschriebenen ersten Iteration verbessern könnte.

Prozessabdeckung und Granularität: Bisher wurde nur Prozessinstanzen eines Geschäftsprozesses untersucht. Zur Verbesserung der Prozessabdeckung könnte zusätzlich zum Beispiel der Prozess der Rücknahme von Lieferungen betrachtet werden. Zur Verbesserung der Granularität könnte das Modell der Lieferung und Produktion verfeinert werden, gegebenenfalls unter Ausnutzung der Fähigkeit von JSON-Netzen, auch Operationen auf Substrukturen von Marken darzustellen, und so zum Beispiel auch individuelle Produktkonfigurationen bestehend aus unterschiedlichen Komponenten (wie in den Beispielen in Kapitel 8) zu modellieren. So könnte eine individuelle und detaillierte Berechnung der eingebetteten Emissionen erfolgen.

Nachhaltigkeitsaspekte und Systemgrenzen: Die Nachhaltigkeitsanalyse kann erweitert werden, indem weitere ökologische (zum Beispiel Auswirkungen auf den Ozonabbau gemessen in CFCe) oder soziale (zum Beispiel exzessive Arbeitszeiten) Wirkungskategorien untersucht werden. Bei einer Erweiterung der bisherigen Systemgrenzen können auch mögliche Auswirkungen auf Kundinnen und Kunden mitbetrachtet werden.

Datenqualität: Durchschnittswerte, wie sie im Beispielszenario beispielsweise zur Messung der in der IT-Infrastruktur eingebetteten sozialen und ökologischen Auswirkungen herangezogen wurden, können aus Lebenszyklusdatenbanken gewonnen werden. Datensätze in diesen Lebenszyklusdatenbanken sind nicht immer aktuell und vollständig, sodass manchmal mit Ersatzwerten oder veralteten Werten gearbeitet werden muss. Ein weiteres Problem betrifft die Transparenz der Daten. Wenn die Daten, wie im Beispiel, lediglich in einer aggregierten Form vorliegen (Gesamtzahl der Emissionen beziehungsweise Unfälle in Scope 3), ist nicht erkennbar, wo im Lebenszyklus beziehungsweise in der Lieferkette der betreffenden Ressource die größten Nachhaltigkeitsauswirkungen auftreten. Die Weiterentwicklung und Integration von Ansätzen wie Scope3transparent (siehe Abschnitt 11.3) und Fairtronics (siehe Exkurs 10.3.1)

kann hier in Zukunft detaillierte Untersuchungen und damit auch zielgerichtetere Verbesserungsmaßnahmen ermöglichen. Mit fortgeschrittener Modellierung beziehungsweise auch Integration in die Informationssysteme des Unternehmens ließen sich auch spezifische Daten in die Analyse einbinden. Beispielsweise könnten individuelle Energiebedarfe einzelner Aktivitätsinstanzen oder tagesaktuelle Wirkungsfaktoren für den genutzten Strommix ermittelt werden.

Exkurs 10.3.1: Fairtronics

Die Herstellung, Verwendung und Entsorgung von Elektronikprodukten ist mit negativen sozialen Nachhaltigkeitsauswirkungen verbunden. Berichte über Kinderarbeit im Goldbergbau [ILO19] und so genannte „Konfliktmineralien", die bewaffnete Konflikte in Zentralafrika finanzieren [Glo15], sind Beispiele für negative soziale Auswirkungen, die bei der Gewinnung von Rohstoffen auftreten und in Elektronikprodukte eingebettet werden. Um diese Probleme angehen zu können, müssen entsprechende Daten erhoben und in geeigneter Form aufbereitet werden. Die Komplexität der Elektroniklieferketten und die Vielzahl möglicher sozialer Auswirkungen stellen jedoch eine große Herausforderung dar. Der Autor der vorliegenden Arbeit hat daher in Kooperation mit der Nichtregierungsorganisation FairLötet (https://fairloetet.de/) an einem Werkzeug „Fairtronics" zur Analyse der in Elektronikprodukte eingebetteten negativen sozialen Auswirkungen gearbeitet. Eine erste Version von Fairtronics wurde im Rahmen eines vom Prototype Fund / BMBF (prototypefund.de) geförderten Projekts über einen Zeitraum von 6 Monaten entwickelt und in der Veröffentlichung [Fri20b] beschrieben.

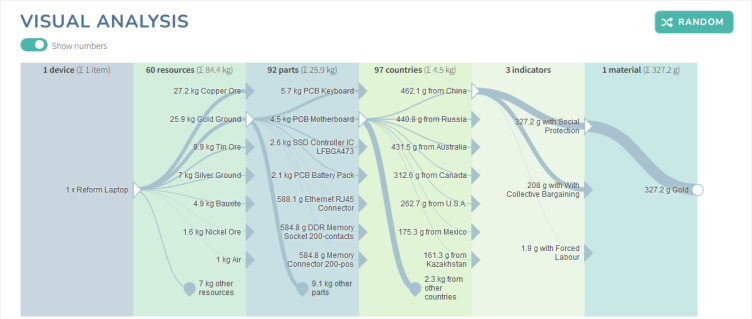


Abbildung 10.6: Indikatorberechnung in Fairtronics. Bildschirmfoto von [Fai24].

Es wurden Daten zur Komponentenzusammensetzung von Elektronikprodukten, zur Rohstoffzusammensetzung von Komponenten, zu Anteilen an der Weltproduktion verschiedener Länder und zu Arbeitsbedingungen verschiedener Länder gesammelt. Anhand dieser Daten können für unterschiedliche Elektronikprodukte Aussagen getroffen werden, in welchen Komponenten (potentiell) hohe negative soziale Auswirkungen eingebettet sind. Die Weiterentwicklung des Werkzeugs wird von FairLötet getragen und vom Autor weiter begleitet. Das Ziel der Weiterentwicklung ist, die Datenqualität und Analysemöglichkeiten weiter zu verbessern. Im aktuellen Stand stellt Fairtronics ein Analysewerkzeug bereit, das es erlaubt, die unterschiedlichen Beiträge von Komponenten, Materialien und Herkunftsländern zu identifizierten sozialen Auswirkungen zu untersuchen (siehe Abbildung 10.6).

Nachhaltigkeitsauswirkungen: Die in den Unterabschnitten 10.1.4 und 10.2.3 vorgeschlagenen Verbesserungsmaßnahmen können umgesetzt werden, um die tatsächlichen Nachhaltigkeitsauswirkungen des untersuchten Geschäftsprozesses zu verbessern. Neben unternehmensinternen Maßnahmen (wie der Umstellung auf erneuerbare Energien), sollte dazu auch die Lieferkette miteinbezogen werden. Zur Reduktion der im verwendeten Material eingebetteten Scope 3-Emissionen kann auf alternative Materialien umgestellt oder Verbesserungsmaßnahmen bei liefernden Unternehmen angestrebt werden.

Integrationstiefe: Der JSON-Netz-Editor oder andere auf den hier vorgestellten Konzepten aufbauende Softwaresysteme können mit den Informationssystemen des Unternehmens integriert werden. So können beispielsweise Daten zu gelagerten und produzierten Artikeln unmittelbar aus den Enterprise Resource Planning-Systemen des Unternehmens bezogen werden. Werden die Prozessmodelle auch zur Steuerung im Unternehmen genutzt, kann zum Beispiel die Ausführung energieintensiver Aktivitäten verzögert werden, um Produktionszeiten mit einem vorteilhaften Strommix zu nutzen [Heh24].

Diskussion der Ergebnisse

11 Evaluation und Diskussion

In diesem Kapitel werden die Ergebnisse der vorliegenden Arbeit evaluiert und diskutiert. Das Vorgehen bei der Evaluation wird in Abschnitt 11.1 vorgestellt. In Abschnitt 11.2 wird dann über die Durchführung von Tests zur Überprüfung der Ausdrucksmächtigkeit von JSON-Netzen anhand von Kontroll- und Datenflussmustern berichtet. Als Evaluation der gesamten Modellierungsmethode für Nachhaltiges Geschäftsprozessmanagement (bestehend aus Vorgehensmodell, Modellierungssprache und Modellierungswerkzeug) stellt Abschnitt 11.3 den Einsatz der Modellierungsmethode in einer Fallstudie vor. Die Erfüllung der in Kapitel 6 beschriebenen Anforderungen an die Modellierungsmethode werden in Abschnitt 11.4 zusammenfassend überprüft. Die Evaluationsergebnisse werden schließlich in Abschnitt 11.5 diskutiert.

11.1 Vorgehen bei der Evaluation

Bei der Evaluation der hier vorgestellten Modellierungsmethode wurden zwei Ziele verfolgt. Zum Einen soll die Erfüllung der aufgestellten Anforderungen (siehe Kapitel 6) überprüft werden (summative Evaluation nach [Sei11, Fra03]). Zum Anderen sollen Erkenntnisse über Weiterentwicklungsmöglichkeiten gewonnen werden (formative Evaluation nach [Sei11, Fra03, Kar16]). Je nach Art des Evaluanden (der zu evaluierenden Komponente der Modellierungsmethode) und der betreffenden Anforderung ist der Einsatz unterschiedlicher Evaluationsmethoden möglich [Pef12]. In der Literatur werden zur Evaluation von Artefakten wie der in dieser Arbeit entwickelten Modellierungsmethode unter anderem Prototypen, Fallstudien, Tests, Umfragen und Beispielszenarios als Evaluationsmethoden genannt [Ven12, Pef12]. Tabelle 11.1 zeigt, welche Evaluationsmethoden im Rahmen der vorliegenden Arbeit zur Überprüfung welcher Anforderung und Komponente der Modellierungsmethode (Evaluanden) eingesetzt wurden. Eine Anforderung kann dabei eine Komponente, mehrere Komponenten oder die gesamte Modellierungsmethode betreffen. Beispielsweise betrifft die Anforderung der *Benutzbarkeit (A 3)* sowohl die Modellierungssprache als auch das Modellierungswerkzeug. In der Tabelle sind unter der Spalte *Evaluand* die jeweiligen von

einer Evaluationsmethode beziehungsweise Anforderung betroffenen Komponenten der Modellierungsmethode aufgeführt. Dabei steht der Buchstabe „S“ für die Sprache (JSON-Netze) und „W“ für das Werkzeug (JSON-Netz-Editor). Der Buchstabe „M“ steht für die gesamte Modellierungsmethode inklusive des Vorgehensmodells für Nachhaltiges Geschäftsprozessmanagement. Zu beachten ist, dass Modellierungssprachen in vielen Fällen erst durch die Bereitstellung eines geeigneten Modellierungswerkzeugs effektiv nutzbar werden [Fra10]. Entsprechend wurde die Erfüllung der Anforderung an die Sprache (JSON-Netze) anhand von Tests, die mit dem Modellierungswerkzeug (JSON-Netz-Editor) durchgeführt wurden, überprüft. In der Tabelle ist dieser Zusammenhang mit S(W) dargestellt. Im Folgenden wird der Einsatz der Evaluationsmethoden anhand Tabelle 11.1 erläutert.

Tabelle 11.1: Übersicht zu eingesetzten Evaluationsmethoden.

Evaluationsmethode	Evaluand	Anforderung
Tests: Testfalldefinition anhand von in der Literatur beschriebenen Kontroll- und Datenflussmustern	S(W)	(A 1) Ausdrucksmächtigkeit
Beispielszenario: Demonstration der Umsetzung der Nachhaltigkeitsmuster	M S(W)	(A 2) Analysierbarkeit (A 4) Anpassbarkeit
Fallstudie: Anwendung der Methode in der Elektronikindustrie	M S(W), W S(W)	(A 2) Analysierbarkeit (A 3) Benutzbarkeit (A 4) Anpassbarkeit
Umfrage: Benutzbarkeitsumfrage mit Anwendenden	S(W), W	(A 3) Benutzbarkeit
Prototyp: Implementierung des JSON-Netz-Editors	S	(A 5) Operationalisierbarkeit

Die Anforderung *(A 1) Ausdrucksmächtigkeit* beschreibt, dass mithilfe der bereitgestellten Modellierungssprache Kontroll- und Datenflüsse von Geschäftsprozessen darstellbar sein sollen. Zur Überprüfung dieser Anforderung wurde eine *Reihe von Tests* anhand von in der Literatur beschriebenen Kontroll- und Datenflussmustern durchgeführt (siehe Abschnitt 11.2).

Das in Kapitel 10 beschriebene *Beispielszenario* demonstriert den Einsatz der Methode zur Analyse der Nachhaltigkeitsauswirkungen von Geschäftsprozessen. Es dient damit der Evaluation der Anforderung *(A 2) Analysierbarkeit*, indem gezeigt wird, dass mit der Anwendung der Modellierungsmethode Aussagen zu sozialen und ökologischen Auswirkungen eines Geschäftsprozesses gemacht werden können. Im Rahmen des Beispielszenarios wird auch demonstriert, wie die Modellierungssprache anhand

der im Modellierungswerkzeug bereitgestellten Funktionen für eine bestimmte Domäne (Nachhaltigkeitsanalysen für das beispielhafte 3D-Druck-Unternehmen) *angepasst* werden kann *(A 4)*.

Als Ergänzung zum „synthetischen" Beispielszenario wurde die Anforderung der *Analysierbarkeit (A 2)* mit einer Fallstudie (siehe Abschnitt 11.3) in einem praxisnahen Kontext evaluiert [Pef12]. Der Einsatz der Modellierungsmethode mit Anwendenden aus Unternehmen der Elektronikindustrie demonstriert auch die *Benutzbarkeit (A 3)* der Modellierungssprache und des Modellierungswerkzeugs. Weiterhin zeigt die Fallstudie auch die *Anpassbarkeit (A 4)* der Modellierungssprache für eine spezielle Domäne. Im Rahmen der Fallstudie wurde die *Benutzbarkeit (A 3)* auch mit einer quantitativen *Umfrage* untersucht.

Die Anforderung *(A 5) Operationalisierbarkeit* betrifft die Modellierungssprache. Sie soll die Entwicklung von Informationssystemen und Software-Werkzeugen unterstützen. Mit der Implementierung eines *Prototyps* (dem JSON-Netz-Editor, siehe Kapitel 9) wurde demonstriert, dass anhand der JSON-Netz-Spezifikation die Entwicklung von Software-Werkzeugen, die auch den Datenaustausch mit anderen Informationssystemen unterstützen, möglich ist.

Mit den eingesetzten Evaluationsmethoden wird jede Anforderung bezüglich der betroffenen Komponenten mindestens einmal überprüft. Nach der Beschreibung der Überprüfung der Ausdrucksmächtigkeit (Abschnitt 11.2) und der durchgeführten Fallstudie (Abschnitt 11.3) wird in Abschnitt 11.4 die Überprüfung der in Abschnitt 6.2 definierten Anforderungen zusammengefasst.

11.2 Überprüfung der Ausdrucksmächtigkeit

Im Folgenden wird die Überprüfung der in Abschnitt 6.2 definierten Anforderung der Ausdrucksmächtigkeit anhand einer Reihe von Tests beschrieben. Die Anforderung besagt, dass die bereitgestellte Modellierungssprache (JSON-Netze) die Darstellung von Kontroll- und Datenflüssen ermöglichen soll. Die Testfälle wurden anhand von in der Literatur beschriebenen Kontrollflussmustern (siehe Unterabschnitt 11.2.1) und Datenflussmustern (siehe Unterabschnitte 11.2.2 und 11.2.3) definiert.

11.2.1 Kontrollfluss

In [Aal03, Rus06] werden 20 unterschiedliche Kontrollfluss-Phänomene, sogenannte Kontrollflussmuster (engl. „Control-Flow Pattern“), beschrieben.[1] Diese Kontrollflussmuster beschreiben beispielsweise eine Folge von auszuführenden Aktivitäten (WCP 1 Sequence) oder die Möglichkeit, eine von mehreren auszuführenden Aktivitäten zu wählen (WCP 4 Exclusive Choice). Sie werden in [Aal03, Rus06] genutzt, um die Ausdrucksmächtigkeit verschiedener Prozessmodellierungssprachen wie EPKs und BPMN zu vergleichen. Beispielsweise unterstützt BPMN die Darstellung von 17 der 20 Kontrollflussmuster, während mit EPKs 9 der 20 Kontrollflussmuster darstellbar sind [Rus06]. Seit ihrer ersten Beschreibung haben die Kontrollflussmuster in der Wissenschaft und Praxis weite Verbreitung gefunden und wurden zur Evaluation von unterschiedlichen Prozesstechnologien und prozessorientierten Informationssystemen eingesetzt [Rus06, Sko16, And24]. Zur Überprüfung der Fähigkeit von JSON-Netzen, Kontrollflüsse darzustellen, wurden anhand der 20 Kontrollflussmuster Testfälle definiert. Im Folgenden wird ein Überblick über die Testergebnisse gegeben sowie über die Erstellung und Durchführung der Testfälle berichtet.

Übersicht der Testergebnisse

Tabelle 11.2 gibt eine Übersicht zu den Ergebnissen der anhand der Kontrollflussmuster durchgeführten Tests. Zur Unterstützung der Nachvollziehbarkeit wird jeweils die unter [Rus06] vergebene ID des dem Testfall zugrundeliegenden Kontrollflussmusters und der Originalname auf Englisch angegeben. Für jeden Testfall wurde ein JSON-Netz im JSON-Netz-Editor erstellt. Die Markierung des JSON-Netzes stellt dabei jeweils den Zustand einer Prozessinstanz dar. Anhand der Simulationsfunktion des JSON-Netz-Editors wurde dann überprüft, ob das JSON-Netz das im Kontrollflussmuster beschriebene Verhalten zeigt (das heißt, die erwarteten Zwischenmarkierungen und die erwartete Endmarkierung erreicht werden). Die erstellten JSON-Netze sind als Beispiele in der unter https://kit-bis.github.io/json-nets/#/controlflow/ bereitgestellten Konfiguration des JSON-Netz-Editors verfügbar. In der digitalen Version des vorliegenden Dokuments ist unter den Namen der Kontrollflussmuster in Tabelle 11.2 jeweils ein Link hinterlegt, der direkt zum entsprechenden Modell führt.

[1] Während in [Aal03] die Kontrollflussmuster natürlichsprachlich beschrieben und grafisch skizziert werden, werden sie in der späteren Arbeit [Rus06] mit einer formalen Beschreibung präzisiert. In [Rus06] werden auch weitere Kontrollflussmuster ergänzt. Die hier vorgenommene Überprüfung beschränkt sich auf die ursprünglichen 20 Kontrollflussmuster in [Aal03].

Tabelle 11.2: Ergebnisse der Testfälle zur Darstellbarkeit von Kontrollflüssen.

Testfall / Kontrollflussmuster	Ergebnis
WCP 1 Sequence	+
WCP 2 Parallel Split	+
WCP 3 Synchronization	+
WCP 4 Exclusive Choice*	+
WCP 5 Simple Merge	+
WCP 6 Multi Choice*	+
WCP 7 Structured Sychronizing Merge*	+
WCP 8 Multi-Merge	+
WCP 9 Structured Discriminator**	+
WCP 10 Arbitrary Cycles*	+
WCP 12 Multiple Instances w/o Synchronization**	+
WCP 13 Multiple Instances w/o a Priori Design-Time Knowledge**	+
WCP 14 Multiple Instances with a Priori Run-Time Knowledge**	+
WCP 15 Multiple Instances w/o a Priori Run-Time Knowledge**	+
WCP 16 Deferred Choice	+
WCP 17 Interleaved Parallel Routing	+
WCP 18 Milestone	+
WCP 19 Cancel Task	+
WCP 20 Cancel Case	+

In der Ergebnis-Spalte von Tabelle 11.2 wird mit + angezeigt, dass jeder Testfall erfolgreich durchgeführt werden konnte.[1] Im Folgenden wird die Durchführung der Tests zunächst anhand eines einfachen Beispiels (WCP 1 Sequence) beschrieben. Dabei wird auch darauf eingegangen, mit welchen Markierungen die Zustände der Prozessinstanzen dargestellt werden und welche Beschriftungen für die Umsetzung des gewünschten Verhaltens genutzt werden. Danach wird auf Sonderfälle eingegangen, in denen zur Umsetzung des im Kontrollflussmuster beschriebenen Verhaltens spezielle Markierungen und Beschriftungen eingeführt wurden (in Tabelle 11.2 mit * und ** markiert).

[1] Das Kontrollflussmuster WCP 11 Implicit Termination beschreibt, dass eine Prozessinstanz endet, *wenn keine Aktivitäten mehr auszuführen sind* (engl.: „A given process [...] instance should terminate when there are no remaining work items that are able to be done either now or at any time in the future.“) [Rus06, S. 25]. Damit wird weniger das Verhalten einer Prozessinstanz beschrieben, sondern mehr die Interpretation der Endzustände von Prozessinstanzen in einem prozessorientierten Informationssystem. Aus der Beschreibung des Kontrollflussmusters in [Rus06] ergab sich damit keine konkrete Testfalldefinition, weshalb WCP 11 in Tabelle 11.2 nicht aufgeführt wird. Wird dieses Kontrollflussmuster von einem prozessorientierten Informationssystem unterstützt, ergibt sich daraus das Problem, dass unter Umständen nicht mehr zwischen unerwünschten Endzuständen (Deadlocks) und erwünschten Endzuständen unterschieden werden kann [Rus06]. Auf Möglichkeiten, die Erreichbarkeit unterschiedlicher Zustände in JSON-Netzen zu untersuchen, wird in Kapitel 12 eingegangen.

Testbericht

Die Durchführung der Tests wird beispielhaft anhand des für das Kontrollflussmuster *WCP 1 Sequence* erstellten Testfalls erläutert. Das Kontrollflussmuster beschreibt, dass eine Aktivität B ausgeführt werden kann, nachdem eine andere Aktivität A abgeschlossen ist. Zum Beispiel kann ein Artikel versandt werden (Aktivität B), nachdem er produziert wurde (Aktivität A). Abbildung 11.1 zeigt die Startmarkierung und die Beschriftung des für den Testfall erstellten JSON-Netzes.[1] Die Transitionen A und B repräsentieren dabei die Aktivitäten A und B.

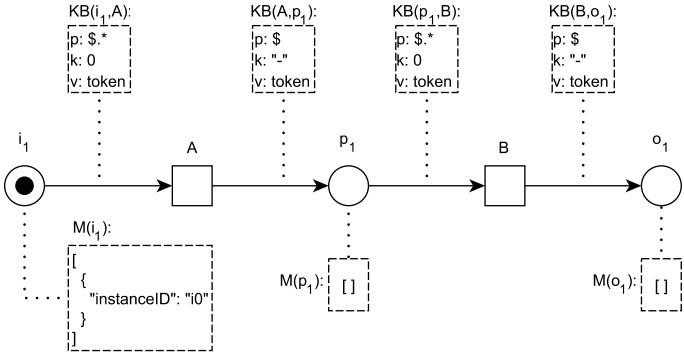


Abbildung 11.1: JSON-Netz für den Testfall WCP 1.

Zur Darstellung des Zustands einer Prozessinstanz[2] im JSON-Netz werden Prozessinstanzmarken (Objects mit einem Feld `"instanceID"`) eingeführt.[3] In Abbildung 11.1 wird gezeigt, dass in der Markierung der Stelle i_1 eine Prozessinstanzmarke liegt. Die anderen Stellen sind mit einem leeren Array markiert. Die Kanten im Vorbereich der Transitionen A und B sind mit dem Filterausdruck `"$.*"` beschriftet, womit ausgedrückt wird, dass beim Schalten der Transition ein Kind des Wurzelarrays der betreffenden Markierung ausgewählt wird. Kanten im Nachbereich einer Transition sind mit dem

[1] Die Umsetzung im JSON-Netz-Editor ist unter https://kit-bis.github.io/json-nets/#/controlflow/wcp-1-sequence einsehbar.

[2] Bei [Aal03] und [Rus06] werden andere Begriffe verwendet, die aber auf die hier verwendeten Begriffe übertragbar sind: ein „Workflow“ entspricht einem Prozesstyp, ein „Case“ ist eine Prozessinstanz und ein „Task“ ist eine Aktivität.

[3] Es wird vereinfachend davon ausgegangen, dass zu einem Zeitpunkt immer nur eine Prozessinstanz in einem Prozessmodell dargestellt wird.

Filterausdruck `"$"` beschriftet, um auszudrücken, dass beim Schalten der Transition ein Wert (hier: eine Prozessinstanzmarke) in das Wurzelarray der betreffenden Markierung eingefügt wird. Schlüsselvariablen im Vorbereich einer Transition wird der Standardwert `0` und im Nachbereich der Standardwert `"-"` zugewiesen. Wie in Unterabschnitt 8.4.4 erläutert, bedeutet das, dass beim Schalten im Vorbereich Marken an der Position 0 im Wurzelarray entfernt und im Nachbereich an das Ende des Wurzelarrays angehängt werden. Die Wertvariablen im Vor- und Nachbereich der Transitionen sind jeweils identisch. Damit wird erreicht, dass eine Transition beim Schalten im Nachbereich eine Prozessinstanzmarke mit gleicher Instanz-ID einfügt. Aufgrund der vorgenommenen Beschriftungen und Markierungen ist zunächst Transition *A* aktiviert. Sobald Transition *A* geschaltet hat, ist Transition *B* aktiviert. Nachdem Transition B einmal geschaltet hat, ist keine Transition mehr aktiviert. Das Schalten der Transitionen *A* und *B* kann als Ausführung der Aktivitäten A und B interpretiert werden. Damit zeigt das JSON-Netz das im Kontrollflussmuster beschriebene Verhalten und der Testfall wird als erfolgreich gewertet. Die meisten der in Tabelle 11.2 aufgeführten Testfälle konnten erstellt werden, indem basierend auf der Beschreibung in [Rus06] eine Netzstruktur erstellt wurde. Diese Netzstruktur wurde dann in ähnlicher Weise wie im hier gezeigten Beispiel für WCP 1 markiert und beschriftet.

Sonderfälle

Bei den in Tabelle 11.2 mit * und ** markierten Testfällen wurde für die Erstellung der zugehörigen JSON-Netze von den oben beschriebenen Standardmarkierungen und -beschriftungen abgewichen. In den Fällen, die in Tabelle 11.2 mit * markiert sind, wurde für einzelne Transitionen ein Schaltverhalten benötigt, das die Markierung einer Stelle im Nachbereich nicht verändert. Dieses Verhalten wurde damit umgesetzt, dass ein in der Markierung vorhandener leerer Array mit einem leeren Array überschrieben wird.

Abbildung 11.2 zeigt dafür beispielhaft das JSON-Netz das für den Testfall basierend auf dem Kontrollflussmusters *WCP 6 Multi Choice* erstellt wurde.[1] Das Kontrollflussmuster beschreibt, dass von mehreren möglichen Prozesspfaden einer oder mehrere gewählt werden können. Zum Beispiel kann nachdem ein Notruf beantwortet wurde (Aktivität A), entweder nur die Feuerwehr (Aktivität B) oder nur die Polizei (Aktivität C) zu einem Notfall losgeschickt werden. Je nach Art des Notrufs können aber auch beide zum Notfall geschickt werden. Wie Abbildung 11.2 zeigt, hat eine Transition *A*

[1] Siehe auch https://kit-bis.github.io/json-nets/#/controlflow/wcp-4-exclusive-choice.

zwei Stellen p_1 und p_2 im Nachbereich, welche zwei mögliche Prozesspfade darstellen, von denen einer oder auch beide ausgeführt werden können. Um diesen Sachverhalt abzubilden, werden in der Transitionsbeschriftung von A die Bedingungen `cond1` und `cond2` eingeführt.

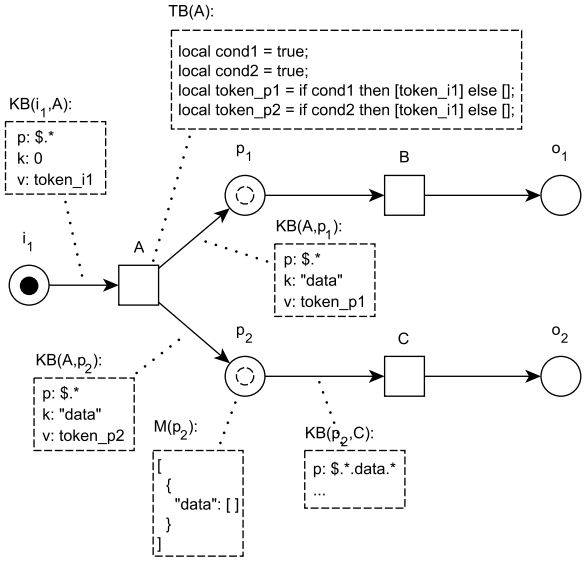


Abbildung 11.2: JSON-Netz für den Testfall WCP 6.

In den Markierungen von p_1 und p_2 wird jeweils ein Object mit dem Feld `data` und einem leeren Array als Wert vorbereitet. Diese spezielle Markierung ist in Abbildung 11.2 mit Kreisen mit gestrichelten Linien angedeutet. Durch die angepasste Transitionsbeschriftung für Transition A und die inzidenten Filterbeschriftungen wird erreicht, dass beim Schalten von A der Wert des Feldes `"data"` entweder mit einem leeren Array oder einem Array, das eine Prozessinstanzmarke enthält, überschrieben wird. Die Filter an den eingehenden Kanten von B und C werden entsprechend angepasst, sodass die jeweiligen Transitionen dann aktiviert sind, wenn im Array unter `data` ein Wert (eine Prozessinstanzmarke) vorhanden ist. Die Beschriftung der Kante (A,p_1) verwendet statt der Wertvariablen `token_p2` die Wertvariable `token_p1`, ansonsten entsprechen die Beschriftungen und Markierungen des Ausführungspfades über B denen des Ausführungspfades über C.

In den Fällen, die in Tabelle 11.2 mit ** markiert sind, wurde für einzelne Transitionen ein Schaltverhalten benötigt, das bei einem Schaltvorgang mehrere Prozessinstanzmarken auf einmal aus einer Markierung entfernt. Dieses Verhalten wurde so umgesetzt, dass beim Schalten ein Array, das eine Anzahl Prozessinstanzmarken enthält, mit einem leeren Array ersetzt wird.

Abbildung 11.3 zeigt dafür beispielhaft das JSON-Netz, das für den Testfall basierend auf dem Kontrollflussmusters *WCP 9 Structured Discriminator* erstellt wurde.[1] Das Kontrollflussmuster beschreibt eine Möglichkeit, nebenläufige Prozesspfade zusammenzuführen. Dabei wird nicht auf die Beendigung aller ausgewählten Prozesspfade gewartet, sondern bei Beendigung des ersten Prozesspfades sofort die Folgeaktivität ausgeführt. Diese Folgeaktivität wird nur einmal ausgeführt, das heißt weitere abgeschlossene Prozesspfade führen nicht zum erneuten Ausführen der Folgeaktivität. Zum Beispiel werden bei der Behandlung eines Herzstillstandes die Aktivitäten Puls überprüfen (A1) und Herzschlag überprüfen (A2) nebenläufig ausgeführt. Sobald die erste der beiden Aktivitäten abgeschlossen ist, wird die Triage-Aktivität (B) ausgelöst. Wie Abbildung 11.3 zeigt, können die unterschiedlichen zusammenzuführenden Prozesspfade mit zwei Transitionen $A1$ und $A2$ mit einer gemeinsamen Stelle p_1 im Nachbereich dargestellt werden. Die Stelle p_1 liegt im Vorbereich der Transition B.

Damit B nicht zweimal (nach dem Schalten von $A1$ und $A2$) schalten kann, werden einige Anpassungen im Vergleich zu den für den Testfall zu WCP 1 Sequence beschriebenen Standardmarkierungen und -beschriftungen vorgenommen. In p_1 ist ein leerer Array im Wurzelarray der Markierung vorbereitet. Diese spezielle Markierung ist in Abbildung 11.3 mit einem Kreis mit gestrichelten Linien angedeutet. Die Filter an den Kanten $(A1,p_1)$, $(A2,p_1)$ und (p_1,B) sind so angepasst, dass beim Schalten der inzidenten Transitionen die Prozessinstanzmarken in diesem Array (und nicht im Wurzelarray) eingefügt oder entfernt werden. Eine ergänzende Konstruktion aus den Stellen p_2 und p_3 sowie der Transition *reset* sorgt dafür, dass B nur einmal schalten kann. Ein Schalten von *reset* sorgt dann dafür, dass verbleibende Prozessinstanzmarken in p_1 entfernt werden. In der Transitionsbeschriftung von *reset* wird die Anzahl an Elementen im Array in p_1 gezählt. Die Transition kann schalten, wenn das Array $m-1$ Elemente enthält, wobei angenommen wird, dass m der Anzahl an Stellen im Vorbereich von p_1 entspricht. Das Entfernen der verbleibenden Prozessinstanzmarken in p_1 wird dadurch erreicht, dass der ganze Array aus $p1$ entnommen wird und mit einem leeren Array (siehe $KB(reset,p_1)$) ersetzt wird.

[1] Die Umsetzung im JSON-Netz-Editor ist unter https://kit-bis.github.io/json-nets/#/controlflow/wcp-9-structured-discriminator abrufbar.

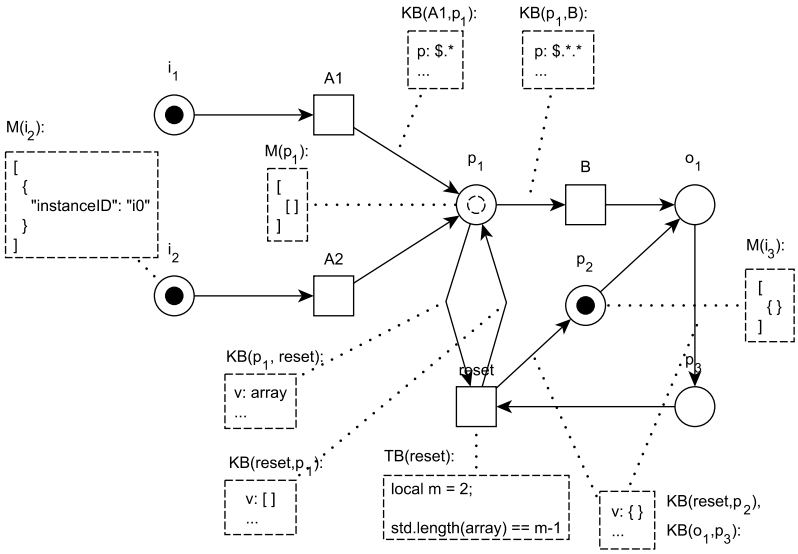


Abbildung 11.3: JSON-Netz für den Testfall WCP 9.

11.2.2 Datenfluss

Anknüpfend an das Konzept der Kontrollflussmuster, werden in [Rus04a] 40 *Datenflussmuster* (engl. „Workflow Data Pattern“) identifiziert und natürlichsprachlich beschrieben. Ergänzend dazu werden unter der Webseite [Wor24] illustrierende Animationen für jedes der Datenflussmuster bereitgestellt. Sie können in ähnlicher Weise wie die Kontrollflussmuster dazu genutzt werden, die Fähigkeiten von Prozessmodellierungssprachen und -werkzeugen, Datenflüsse darzustellen, strukturiert zu untersuchen. Die Datenflussmuster betreffen unterschiedliche Aspekte der Darstellung von Datenflüssen und sind daher den Kategorien *Datensichtbarkeit*, *Dateninteraktion*, *Datentransfer* und *Datenbasierter Kontrollfluss* zugeordnet. Zur Überprüfung der Fähigkeit von JSON-Netzen, Datenflüsse darzustellen, wurden anhand der 40 Datenflussmuster Testfälle definiert. Im Folgenden wird die Erstellung der Testfälle beschrieben und ein Überblick über die Testergebnisse gegeben.

Übersicht der Testergebnisse

Tabelle 11.3 gibt eine Übersicht zu den Ergebnissen der anhand der Datenflussmuster durchgeführten Tests. Zur Unterstützung der Nachvollziehbarkeit wird jeweils die unter [Rus04a] vergebene ID und der Originalname auf Englisch angegeben. Für jeden Testfall wurde ein JSON-Netz im JSON-Netz-Editor erstellt. Die Markierung des JSON-Netzes stellt dabei jeweils den Zustand einer Prozessinstanz dar. Anhand der Simulationsfunktion des JSON-Netz-Editors wurde dann überprüft, ob das JSON-Netz das im Kontrollflussmuster beschriebene Verhalten zeigt (das heißt, die erwarteten Zwischenmarkierungen und die erwartete Endmarkierung erreicht werden). Die erstellten JSON-Netze sind als Beispiele in der unter https://kit-bis.github.io/json-nets/#/dataflow/ bereitgestellten Konfiguration des JSON-Netz-Editors verfügbar. In der digitalen Version des vorliegenden Dokuments ist unter den Namen der Kontrollflussmuster in Tabelle 11.3 jeweils ein Link hinterlegt, der direkt zum entsprechenden Modell führt.

Tabelle 11.3: Ergebnisse der Testfälle zur Darstellbarkeit von Datenflüssen.

Kategorie	Testfall / Datenflussmuster	Ergebnis
Datensichtbarkeit	WDP 1 Task Data	+
	WDP 3 Scope Data	+
	WDP 4 Multiple Instance Data	+
	WDP 5 Case Data	+
	WDP 6 Folder Data	+
	WDP 7 Workflow Data	+
	WDP 8 Environment Data*	+
Dateninteraktion	WDP 9 Task to Task	+
	WDP 12 To Multiple Instance Task	+
	WDP 13 From Multiple Instance Task	+
	WDP 14 Case to Case	+
	WDP 15/19/23 Data Interaction – Comp. to Env. – Push*	+
	WDP 16/20/24 Data Interaction – Env. to Comp. – Pull*	+
Datentransfer	WDP 27 Data Transfer by Value – Incoming	+
	WDP 28 Data Transfer by Value – Outgoing	+
	WDP 29 Data Transfer Copy In / Copy Out	+
	WDP 31 Data Transfer by Reference – With Lock	+
	WDP 32 Data Tranformation – Input	+
	WDP 33 Data Tranformation – Output	+
Datenbasierter Kontrollfluss	WDP 34 Task Precondition – Data Existence	+
	WDP 35 Task Precondition – Data Value	+
	WDP 36 Task Postcondition – Data Existence	+
	WDP 37 Task Postcondition – Data Value	+
	WDP 39 Data-based Task Trigger	+
	WDP 40 Data-based Routing	+

In der Ergebnis-Spalte von Tabelle 11.3 wird mit + angezeigt, dass jeder Testfall erfolgreich durchgeführt werden konnte.[1] Die Testfälle sind den entsprechenden Kategorien der zugehörigen Datenflussmuster, *Datensichtbarkeit*, *Dateninteraktion*, *Datentransfer* und *Datenbasierter Kontrollfluss* zugeordnet. Im Folgenden wird die Durchführung der Tests für jede der Kategorien beschrieben. Danach wird auf Sonderfälle eingegangen, die in Tabelle 11.2 mit * markiert sind, sowie auf Datenflussmuster, für die kein Testfall erstellt werden konnte.

Testbericht

Im Folgenden wird beispielhaft die Durchführung der anhand der Datenflussmuster in den Kategorien *Datensichtbarkeit*, *Dateninteraktion*, *Datentransfer* und *Datenbasierter Kontrollfluss* erstellten Tests erläutert. Es wird jeweils der mit den Datenflussmustern einer Kategorie betrachtete Aspekt erläutert und dann die zugehörigen Tests beschrieben.

Die Datenflussmuster *WDP 1* bis *WDP 8* beschreiben den Aspekt der *Datensichtbarkeit* in Geschäftsprozessen [Rus04a]. Ein Datenobjekt kann beispielsweise nur für eine Aktivitätsinstanz (*WDP 1 Task Data*) oder für eine ganze Prozessinstanz (*WDP 5 Case Data*) verfügbar sein. Die Gültigkeit eines Datenobjekts nur innerhalb einer Aktivitätsinstanz (*WDP 1 Task Data*) wird im für den zugehörigen Testfall erstellten JSON-Netz damit demonstriert, dass die Belegung einer Variablen für einen Schaltvorgang (einer Ausführung einer Aktivität) unabhängig von weiteren Schaltvorgängen derselben Transition ist. Ein Datenobjekt, das für alle Aktivitätsinstanzen einer Prozessinstanz verfügbar sein soll (*WDP 5 Case Data*) kann als zusätzliches Feld in einer Prozessinstanzmarke[2] von Transition zu Transition „weitergereicht“ werden. Es ist auch möglich, im Vorbereich einer Transition mehrere Prozessinstanzmarken mit der gleichen Instanz-ID aber unterschiedlichen Werten für das zusätzliche Feld zu erzeugen, und so *WDP 4 Multiple Instance Data* umzusetzen. Die Verfügbarkeit eines Datenobjekts für Aktivitätstypen oder Prozesstypen kann dadurch erreicht werden, dass eine

[1] Neben den unterschiedlichen Datenflussphänomenen, die mit den Datenflussmustern beschrieben werden, unterscheiden sich datenorientierte Prozessmodellierungssprachen auch darin, welche Datentypen darstellbar sind [Rus04a]. [Rus04a] unterscheidet die Darstellbarkeit unterschiedlicher Konstrukte, wie einfache Datentypen (zum Beispiel „string“, „integer“, „date“, „time“) und komplexe Datentypen (zum Beispiel „composition“ und „array“). JSON-Netze beherrschen über das JSON-Datenmodell sowohl einfache als auch komplexe Datentypen (siehe Abschnitt 8.3). Auch spezielle Datentypen wie Datums- und Zeitangaben können mit JSON Schema-Typisierungen umgesetzt werden [Wri22b].

[2] Siehe Erläuterungen zu Prozessinstanzmarken in JSON-Netzen bei der Beschreibung der Kontrollfluss-Testfälle in Unterabschnitt 11.2.1.

zusätzliche Datenspeicher-Stelle eingeführt wird, die das Datenobjekt enthält. Diese Datenspeicher-Stelle wird für den Testfall zu *WDP 7 Workflow Data* mit allen Transitionen des Prozessmodells im Vor- und Nachbereich verbunden. So können alle Aktivitäten lesend und schreibend auf das entsprechende Datenobjekt zugreifen. Unterschiedliche Einschränkungen der Verfügbarkeit (*WDP 3 Scope Data* und *WDP 6 Folder Data*) können mit ergänzenden Beschriftungen und Markierungen umgesetzt werden.

Der Aspekt der *Dateninteraktion* wird mit den Datenflussmustern *WDP 9* bis *WDP 26* beschrieben [Rus04a]. Ein Datenobjekt kann beispielsweise zwischen Aktivitätsinstanzen (*WDP 9 Task to Task*) oder auch zwischen Prozessinstanzen (*WDP 14 Case to Case*) ausgetauscht werden. Wie bereits für die Erstellung des Testfalls zu *WDP 5 Case Data* beschrieben, kann ein Datenobjekt, das zwischen verschiedenen Aktivitätsinstanzen ausgetauscht werden soll (*WDP 9 Task to Task*), als zusätzliches Feld einer Prozessinstanzmarke „weitergereicht“ werden. Über das Einführen einer Datenspeicher-Stelle kann ein Datenobjekt auch zwischen Prozessinstanzen (*WDP 14 Case to Case*) ausgetauscht werden. Auch Aktivitäten, die innerhalb einer Prozessinstanz mehrmals ausgeführt werden (*WDP 12 To Multiple Instance Task* und *WDP 13 From Multiple Instance Task*), können mit einem JSON-Netz umgesetzt werden.

Die Datenflussmuster der Kategorie *Datentransfer* (*WDP 27* bis *WDP 33*) beschreiben unterschiedliche Arten, wie ein Datenobjekt zwischen Prozesskomponenten ausgetauscht wird [Rus04a]. So kann beispielsweise das Datenobjekt selbst (*WDP 27/28 Data Transfer by Value*) oder eine Referenz auf ein Datenobjekt (*WDP 30/31 Data Transfer by Reference*) zwischen Prozesskomponenten ausgetauscht werden. In JSON-Netzen kann ein Datenobjekt als Wert in die Markierung einer Stelle von einer Transition eingefügt und von einer anderen Transition gelesen und gelöscht werden. Damit kann die Weitergabe von Datenobjekten „by Value“, also *WDP 27 Data Transfer by Value – Incoming* und *WDP 28 Data Transfer by Value – Outgoing* in einem Testfall demonstriert werden. Über ergänzende Konstruktionen lassen sich aber auch Mechanismen wie das Kopieren von Werten zur Verarbeitung (*WDP 29 Data Transfer Copy In / Copy Out*) oder das Weiterreichen einer Referenz auf ein in einem gemeinsamen Datenspeicher vorgehaltenen Wert (*WDP 31 Data Transfer by Reference – With Lock*) umsetzen. Mit den Datenflussmustern *WDP 32 Data Transformation – Input* und *WDP 33 Data Transformation – Output* wird unterschieden, ob bei der Ausführung einer Aktivität Eingabewerte (Input) oder Ausgabewerte (Output) manipuliert werden. In JSON-Netzen können in Transitionsbeschriftungen Werte in Variablen, die in den Filtertupeln der inzidenten Kanten definiert wurden, manipuliert werden. Diese Variablen können im Vorbereich (Input) oder im Nachbereich (Output) der Transition (Aktivität) definiert sein.

Unterschiedliche Arten, wie der Kontrollfluss auf Basis von vorhandenen Daten beeinflusst werden kann, werden mit den Datenflussmustern der Kategorie *Datenbasierter Kontrollfluss* (*WDP 34* bis *WDP 40*) beschrieben [Rus04a]. Die Datenflussmuster *WDP 34* bis *WDP 37* beschreiben die Möglichkeit, das Vorhandensein eines bestimmten Datenobjekts (gegebenenfalls unter Einschränkungen für einen Wert) als Vor- oder Nachbedingung für die Ausführung einer Aktivität zu definieren. Stellen im Vor- und Nachbereich einer Transition lassen sich als Vor- und Nachbedingungen einer Aktivität interpretieren. Mit entsprechenden Stellenschemas oder Kanten- und Transitionsbeschriftungen lässt sich damit jeweils ein entsprechendes JSON-Netz erstellen. In ähnlicher Weise ist auch die Umsetzung eines Schaltverhaltens umsetzbar, das nur unter gewissen Bedingungen einen bestimmten Wert in die Markierung einer Stelle schreibt. Dieser Wert ermöglicht dann wieder als „Auslöser" das Schalten einer bestimmten Transition (*WDP 39 Data-based Task Trigger*). Die Auswahl von Prozesspfaden (*WDP 40 Data-based Routing*) ist in JSON-Netzen ebenfalls in Abhängigkeit von verschiedenen Datenobjekten möglich (siehe dazu auch die in Unterabschnitt 11.2.1 beschriebene Umsetzung von *WCP 6*).

Sonderfälle

Die in Tabelle 11.2 mit * markierten Datenflussmuster beschreiben die Verfügbarkeit von Daten aus externen Systemen (*WDP 8 Environment Data*) sowie unterschiedliche Arten der Interaktion mit externen Systemen (*WDP 15/19/23 Data Interaction – Comp. to Env. – Push* und *WDP 16/20/24 Data Interaction – Env. to Comp. – Pull*). Mit *WDP 15/19/23 Data Interaction – Comp. to Env. – Push* sind Datenflussmuster zusammengefasst, die beschreiben, dass eine Prozesskomponente die Übergabe eines Datenobjekts an ein externes System auslösen kann. Das Gegenstück dazu bilden die Datenflussmuster *WDP 16/20/24 Data Interaction – Env. to Comp. – Pull*, die beschreiben, dass eine Prozesskomponente ein Datenobjekt von einem externen System anfordern kann. Für diese Datenflussmuster wurden keine separaten Testfälle erstellt. Mit den Testfällen zu den Datenflussmustern *WDP 1-7* wird jedoch demonstriert, dass mit JSON-Netzen unterschiedliche Arten von Prozesskomponenten wie Aktivitätsinstanzen, Prozessinstanzen, Prozesstypen darstellbar sind. Weiterhin wird im Bericht zur durchgeführten Fallstudie (siehe Abschnitt 11.3) gezeigt, dass mit den im JSON-Netz-Editor umgesetzten Funktionen die Markierung einer Stelle an ein externes System übergeben oder von einem externen System angefordert werden kann. Diese Demonstrationen ist in Tabelle 11.3 als erfolgreicher Test aufgeführt.

Für manche Datenflussmuster konnten keine Testfälle erstellt werden. Wie im Folgenden beschrieben wird, ist das jeweils mit Einschränkungen der bisher im JSON-Netz-Editor bereitgestellten Funktionen und nicht mit grundlegenden Einschränkungen in der Spezifikation von JSON-Netzen zu begründen. Davon betroffen ist eine Gruppe von Datenflussmustern, die voraussetzen, dass ein externes System die Übertragung eines Datenobjekts anstoßen kann. Das betrifft die Datenflussmuster *WDP 18/22/26 Data Interaction – Component to Environment – Pull-oriented*, *WDP 17/21/25 Data Interaction – Environment to Component – Push-oriented* sowie *WDP 38 Event-based Task Trigger*. Bisher ist im JSON-Netz-Editor lediglich die Möglichkeit vorgesehen, dass Anwendende des JSON-Netz-Editors eine Datenübertragung zu oder von einem externen System anstoßen (siehe die Erläuterungen zu *WDP 15/19/23* und WDP *16/20/24*). Mit der Implementierung entsprechender Schnittstellen im JSON-Netz-Editor können künftig auch Testfälle für diese Datenflussmuster erstellt werden.

Eine weitere Gruppe von Datenflussmustern setzt voraus, dass ein Prozess in Unterprozesse verfeinert werden kann. Das betrifft die Datenflussmuster *WDP 2 Block Data*, *WDP 10 Block Task to Sub-Workflow Decomposition* und *WDP 11 Sub-Workflow Decomposition to Block Task*. Unterprozesse können mit Petri-Netzen beispielsweise als Verfeinerungen von Transitionen dargestellt werden, bei denen eine Transition t durch eine Netzstruktur mit den Stellen ${}^\bullet t \cup t^\bullet$ als Schnittstelle ersetzt wird [Rei10]. Bisher ist diese Möglichkeit jedoch nicht im JSON-Netz-Editor implementiert, weshalb kein zugehöriger Testfall erstellt werden konnte.

Die Datenflussmuster *WDP 30 Data Transfer by Reference – Unlocked* und *WDP 31 Data Transfer by Reference – With Lock* unterscheiden, ob mehrere Aktivitäten nebenläufig auf ein Datenobjekt zugreifen können (*Unlocked*) oder nicht (*With Lock*). Mit den bestehenden Funktionen zur (teil-)automatisierten Simulation im JSON-Netz-Editor kann zu einem Zeitpunkt immer nur eine Transition schalten, womit lediglich ein Testfall für *WDP 31* erstellt werden konnte. Weitere Aspekte des nebenläufigen Zugriffs mehrerer Transitionen auf dasselbe Datenobjekt werden in Unterabschnitt 11.2.3 untersucht.

11.2.3 Manipulation von Substrukturen

Transitionen in JSON-Netzen können, in ähnlicher Weise wie in den verwandten Nested-Relation/Transitionsnetzen [Obe96b] und XML-Netzen [Len01] auch Substrukturen aus komplexen Datenobjekten entfernen oder einfügen. Dadurch können beispielsweise im Vergleich zu gefärbten Petri-Netzen Datenflussphänomene im

Zusammenhang mit komplexen Datenobjekten (zum Beispiel der Zugriff auf Rechnungspositionen in Rechnungen) kompakter dargestellt werden. Darüber hinaus können dadurch mehrere Transitionen nebenläufig auf dasselbe Datenobjekt zugreifen [Obe96c]. In den in [Rus04a] beschriebenen Datenflussmustern wurden diese Aspekte des Zugriffs auf komplexe Datenobjekte nicht untersucht. Für die Erstellung von Testfällen wird daher auf in [Obe96c] beschriebene Beispiele zurückgegriffen.

Tabelle 11.4: Ergebnisse der Testfälle zur Manipulierbarkeit von Substrukturen.

Testfall	Ergebnis
SMP 1 Delete Substructure	+
SMP 2 Insert Substructure	+
SMP 3 Concurrent Manipulation of Substructures	(+)

Wie Tabelle 11.4 zeigt, wurden basierend auf den in [Obe96c] beschriebenen Beispielen ergänzende Datenflussmuster definiert. Diese sind (1) das Löschen einer Substruktur in einem komplexen Datenobjekt (*SMP 1 Delete Substructure*), (2) das Einfügen eine Substruktur in ein komplexes Datenobjekt (*SMP 2 Insert Substructure*) und (3) der nebenläufige Zugriff mehrerer Aktivitäten auf unterschiedliche Substrukturen desselben Datenobjekts (*SMP 3 Concurrent Manipulation of Substructures*). Als Testfälle wurden entsprechende in [Obe96c] für Nested-Relation/Transitionsnetze beschriebene Beispiele als JSON-Netz rekonstruiert. Anhand der im JSON-Netz-Editor bereitgestellten Funktionen zur Simulation eines JSON-Netzes wurde dann untersucht, ob das JSON-Netz das gewünschte Verhalten zeigt. Die umgesetzten JSON-Netze sind unter https://kit-bis.github.io/json-nets/#/substructures/ als Beispiele verfügbar und als Links in der Tabelle 11.4 hinterlegt. Das Einfügen und Löschen von Substrukturen in ein komplexes Datenobjekt lässt sich mit JSON-Netzen unmittelbar umsetzen (Ergebnis + in Tabelle 11.4). Das in [Obe96c] beschriebene Beispiel für den nebenläufigen Zugriff auf mehrere Substrukturen desselben Datenobjekts lässt sich ebenfalls als JSON-Netz nachbilden. Anhand dieses Testfalls lässt sich zeigen, dass mehrere Transitionen für eine konfliktfreie Belegung (siehe Unterabschnitt 8.4.1) von Substrukturen desselben Datenobjekts aktiviert sind und schalten können. Der bisher implementierte Simulationsalgorithmus eines JSON-Netzes erlaubt allerdings nicht das gleichzeitige oder nebenläufige Schalten von Transitionen. Da der Sachverhalt mit einem JSON-Netz darstellbar ist, entsprechende Funktionen im JSON-Netz-Editor aber bisher noch nicht implementiert wurden, wird der Testfall zu *SMP 3 Concurrent Manipulation of Substructures* mit (+) bewertet.

11.3 Einsatz der Modellierungsmethode in einer Fallstudie

In diesem Abschnitt wird über den Einsatz der vorgestellten Modellierungsmethode für nachhaltiges Geschäftsprozessmanagement in einem Projekt für die Treibhausgasbilanzierung in der Elektronikindustrie berichtet. Der Einsatz der Modellierungsmethode im Projekt dient der Evaluation ihres Nutzens für Unternehmen [Öst11]. Die Form des folgenden Berichts orientiert sich dabei an Empfehlungen für das Verfassen chronologischer Fallstudien in [Blo03]. Als Quellen dienen der Projektantrag, die Projektwebseite, Notizen des Autors, Besprechungsprotokolle und Dokumente wie virtuelle Whiteboards und Berichte, die im Projekt erarbeitet wurden.

11.3.1 Das Projekt Scope3transparent

Das Projekt Scope3transparent wurde in der deutschen Nationalen Klimaschutzinitiative[1] vom Bundesministerium für Wirtschaft und Klimaschutz gefördert. Über die dreijährige Projektlaufzeit (Oktober 2021 bis September 2024) hinweg sollten im Projekt Maßnahmen zur Reduktion von Treibhausgas-Emissionen in der Lieferkette von Elektronikprodukten entwickelt werden. Projektpartner waren das Fraunhofer IZM[2] (eine Forschungseinrichtung mit Fokus auf robuste und zuverlässige Elektronik), AfB gGmbH[3] (ein gemeinnütziges Unternehmen, das gebrauchte IT-Geräte aufarbeitet und verkauft), Umwelttechnik BW[4] (eine baden-württembergische Landesagentur, die Unternehmen mit Kompetenzen im Bereich Umwelttechnik, Ressourceneffizienz und Klimaschutz unterstützt) und die Forschungsgruppe Betriebliche Informationssysteme[5] des Karlsruher Instituts für Technologie, die insbesondere zum Thema Nachhaltiges Geschäftsprozessmanagement forscht. Als assoziierter Partner war weiterhin der Verein FairLötet e.V.[6] dabei, der sich für soziale Gerechtigkeit in den Lieferketten der Elektronikbranche einsetzt.

Der Hintergrund des Projekts ist, dass aktuell viele Unternehmen schon Nachhaltigkeitsziele in Bezug auf Treibhausgas-Emissionen formulieren, dabei jedoch nur auf Treibhausgas-Emissionen abzielen, die direkt im Unternehmen oder bei der

[1] https://www.klimaschutz.de
[2] https://www.izm.fraunhofer.de/
[3] https://www.afb-group.de/
[4] https://www.umwelttechnik-bw.de
[5] https://aifb.kit.edu/web/Betriebliche_Informationssysteme
[6] https://fairloetet.de/

Produktion eingekaufter Energie entstehen (Scope 1 und Scope 2 im Treibhausgasprotokoll [WBC04]). In vielen Industrien und insbesondere in der Herstellung von IT- und Elektronikprodukten kann jedoch ein Großteil der einem Produkt zuordenbaren Treibhausgas-Emissionen der Lieferkette (Scope 3) entstammen [Sco23]. Im Projekt sollten daher Maßnahmen zur Bewusstseinsschaffung und zur Reduzierung von Treibhausgas-Emissionen mit Fokus auf die Lieferkette beziehungsweise Scope 3 in der IT- und Elektronikindustrie entwickelt werden. Zu den umgesetzten Maßnahmen zählen

- die Ermittlung von Scope 3-Emissionen mit Pilot-Unternehmen,
- der Aufbau einer Datenbank mit Treibhausgas-Fußabdrücken[1] von relevanten Materialien, Komponenten, Baugruppen und Geräten,
- die Entwicklung eines Praxisleitfadens für Unternehmen,
- die Durchführung einer Schulungskampagne für Unternehmen,
- die Entwicklung eines unterstützenden Software-Werkzeugs,
- die Entwicklung von Schulungsmaterialien für Repair-Cafés sowie die Beteiligung an Repair-Cafés,
- die Ableitung von Empfehlungen für die Politik.

Im Folgenden liegt der Fokus des Berichts auf der Entwicklung und Anwendung des unterstützenden Software-Werkzeugs, welche der Autor als Leiter des Teilprojekts beim Projektpartner Forschungsgruppe Betriebliche Informationssysteme verantwortete. Das Software-Werkzeug wurde im Rahmen der Vorbereitung und Durchführung der Schulungskampagne für Unternehmen eingesetzt, wobei die in dieser Arbeit vorgestellte Methode zum Einsatz kam.

11.3.2 Treibhausgasbilanzierung in der Elektronikindustrie

Zu Beginn des Projekts wurde von den Projektpartnern eine Bedarfsanalyse in Form einer Online-Umfrage mit Unternehmen durchgeführt [Sco22]. An der Online-Umfage haben 114 Unternehmensvertreterinnen und Vertreter aus unterschiedlichen, der Elektronikindustrie zuzuordnenden Branchen wie Automation, Fahrzeugelektrik oder Batterie-Herstellung teilgenommen. Die Teilnehmenden waren unter anderem

[1] Ein Treibhausgas-Fußabdruck entspricht einer Produkt-Lebenszyklusanalyse für eine Wirkungskategorie (Klimawirkung), siehe auch Kapitel 3.

in der Geschäftsführung, im Nachhaltigkeitsmanagement und als Produktionsleitung beschäftigt. Sie gaben zu 64 Prozent an, dass sie in einem Unternehmen beschäftigt sind, das mehr als 250 Beschäftigte hat. Die übrigen Teilnehmenden gaben an, dass sie in kleineren Unternehmen beschäftigt sind oder machten keine Angaben. Den Angaben zufolge erhalten über die Hälfte der Unternehmen Anfragen von Kundinnen und Kunden zu den Treibhausgas-Fußabdrücken ihrer Produkte. Insgesamt gehen 90 Prozent der Unternehmen davon aus, dass das in Zukunft der Fall sein wird. Scope 1 und Scope 2 werden bereits von zwei Dritteln der Unternehmen in Produkt- oder Unternehmensbezogenen Treibhausgasbilanzen erfasst. 28 Prozent befassen sich auch mit vorgelagerten Treibhausgas-Emissionen entlang der Lieferkette (Scope 3). Von den Unternehmen, die ihre Scope 3-Emissionen untersuchen, setzt ein Großteil eine hybride Methode ein. Das bedeutet, dass zur Bilanzierung der Treibhausgas-Emissionen eine Kombination aus zuliefererspezifischen Daten, ausgabenbasierten Schätzungen und Durchschnittsdaten (zum Beispiel aus Lebenszyklusanalyse-Datenbanken) genutzt wird. Die Bilanzierung der Scope 3-Emissionen erfolgt im Allgemeinen für ausgewählte Produkte (ausgewählt nach Kriterien wie hohe Einkaufsausgaben, hohe Nachfrage oder auf spezifische Anfragen) und nur in 6 Prozent der Fälle (auf die Gesamtzahl der Teilnehmenden gerechnet) für das gesamte Produktportfolio.

Technisch wird die Datenerfassung und Bilanzierung bei den meisten Unternehmen mit Excel umgesetzt. Daneben werden auch Software-Werkzeuge für Lebenszyklusanalysen wie Umberto [iPo24], GaBi [Sph24] und SimaPro [PRé24] eingesetzt. Nur 5 Prozent der Unternehmen gaben an, dass sie durchgängige IT-Systeme zur Datenerfassung in der Lieferkette einsetzen. Als Herausforderung bei der Treibhausgasbilanzierung wird die Komplexität der Lieferkette und des Produktportfolios genannt. Weiterhin beklagen Unternehmen Schwierigkeiten bei der Kommunikation mit anderen Unternehmen in ihrer Lieferkette. Technische Herausforderungen werden in den Bereichen Datenkonsistenz, Datenlücken und Verfügbarkeit generischer Daten gesehen. Unternehmen sehen Lösungsansätze in der Bereitstellung branchenspezifischer Standards, der Bereitstellung von Berechnungswerkzeugen und Datenbanken sowie der spezifischen Unterstützung des Datenaustauschs in der Lieferkette.

11.3.3 Durchführung der Unternehmensschulungen

In den im Projekt durchgeführten eintägigen Schulungen sollten jeweils ca. 12 Teilnehmende aus Unternehmen in einem Theorieteil die Grundlagen der Treibhausgasbilanzierung kennenlernen. In einem Praxisteil sollte das Gelernte dann anhand einer

praxisnahen Aufgabenstellung angewendet werden. Bei der Vorbereitung und Durchführung des Praxisteils kam, wie im Folgenden beschrieben, die in dieser Arbeit beschriebene Modellierungmethode zum Einsatz. Die Phasen des Vorgehensmodells aus Kapitel 7 wurden dabei insgesamt vier Mal durchlaufen. Abbildung 11.4 stellt den zeitlichen Ablauf der Iterationen zusammen mit den Schulungsterminen dar. Das Schulungskonzept wurde zunächst gemeinsam mit den Projektpartnern erarbeitet und einmal innerhalb des Projektteams, im Dezember 2023, pilotiert. Im Anschluss fand jeweils eine Schulung im Januar, März und April 2024 statt.

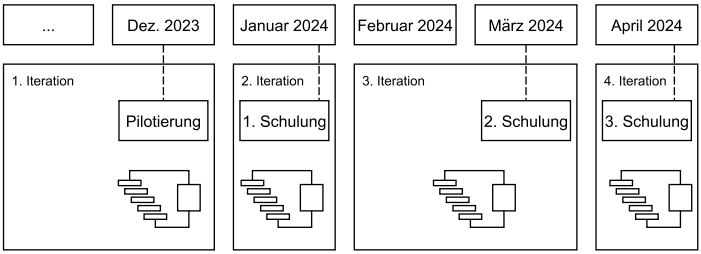


Abbildung 11.4: Zeitlicher Ablauf der Fallstudie.

Die Phase *Spezifikation des Indikatormodells* fand dabei jeweils während den Vorbereitungen eines Schulungstermins statt. Im Rahmen des Projekts waren der Fokus auf der Wirkungskategorie Klimaänderung und die Systemgrenzendefinitionen nach dem Treibhausgasprotokoll (Scope 1, Scope 2 und Scope 3) vorgegeben. Im Austausch mit den *Nachhaltigkeits- und Branchenexpertinnen und -experten* im Projekt entwarf der Autor in seiner Rolle als *Werkzeugexperte* eine Konfiguration des JSON-Netz-Editors mit Stellenschemas zur Erfassung relevanter Inventarindikatoren und Wirkungsfaktoren, sowie vorbereiteten Transitionsbeschriftungen zur Berechnung der resultierenden Wirkungsindikatoren. Relevante Inventarindikatoren waren Mengen der Treibhausgase NF3, SF6 und CF4 (für Scope 1), Energieverbräuche (für Scope 2) sowie unterschiedliche Materialien und Komponenten (für Scope 3). Wirkungsfaktoren für verschiedene Treibhausgase wurden direkt (in vorbereiteten Transitionsbeschriftungen) im JSON-Netz-Editor hinterlegt. Für Energieverbräuche war vorgesehen, dass Anwendende Wirkungsfaktoren in Abhängigkeit von unterschiedlichen Strommixen selbst angeben. In Bezug auf Materialien und Komponenten sollten Wirkungsfaktoren relativ zu ihrem Gewicht, zu ihrer Fläche oder einer Stückzahl angegeben werden können.

In der Phase *Geschäftsprozessidentifikation* erstellen *Anwendende* ein Prozessmodell des zu untersuchenden Geschäftsprozess erstellen. Im Rahmen der Schulungen wurde ein fiktives aber praxisnahes Szenario der Lieferkette einer Kameraproduktion vorbereitet. Abbildung 11.5 zeigt eine Darstellung der Lieferkette, wie sie auch den Schulungsteilnehmenden präsentiert wurde. Die Teilnehmenden wurden in Gruppen eingeteilt und jeweils einem der Unternehmen der Lieferkette zugeordnet. Für jedes dieser Unternehmen wurde eine Beschreibung des entsprechenden Herstellungsprozesses (zum Beispiel Herstellung eines Wafers oder Herstellung einer Leiterplatte) bereitgestellt. Aufgrund zeitlicher Beschränkungen, und da bei den Unternehmensschulungen der Fokus auf der Erfassung von Treibhausgasemissionen lag, wurde jeweils für die Teilnehmenden ein zugehöriges Prozessmodell vorbereitet. Dieses konnte zu Beginn einer Schulung von den Teilnehmenden inspiziert und mit der Prozessbeschreibung abgeglichen werden. Den Teilnehmenden wurde im Rahmen der Schulung eine Konfiguration des JSON-Netz-Editors im Assistenzmodus (siehe Unterabschnitt 9.5.5) bereitgestellt. Das heißt, sie konnten die Netzstrukturen der bereitgestellten Prozessmodelle bearbeiten und jeweils vorbereitete Stellen- oder Transitionstypen auswählen. Die Markierung der Stellen konnte dann anhand von Formularen, die aus den vorbereiteten Stellenschemas generiert wurden, bearbeitet werden.

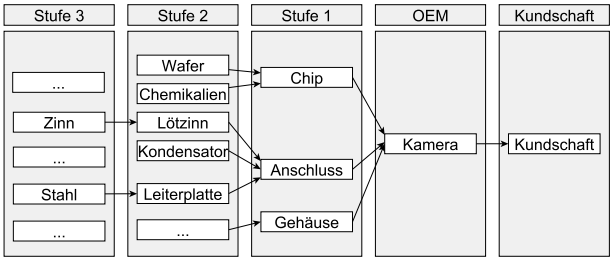


Abbildung 11.5: Vereinfachte Lieferkette einer Kameraherstellung.

Die Phase *Datenerhebung und -integration* bildete den zeitlichen Schwerpunkt der Schulung. Jede Gruppe erhielt Unterlagen, welche unternehmensinterne Daten wie Stücklisten und Energieverbräuche darstellten, sowie Auszüge aus beispielhaften Datenbanken mit Wirkungsfaktoren. Die Teilnehmenden haben zunächst diese unternehmensinternen Daten eingebunden und die bereitgestellten Prozessmodelle entsprechend erweitert. Abbildung 11.6 zeigt beispielhaft ein Formular, wie es von Teilnehmenden zur Bearbeitung einer Stellenmarkierung (für die Erfassung von Scope 1-Daten) ausgefüllt werden konnte. Anstelle von generischen Daten (aus den

bereitgestellten Datenbanken) konnten spezifische Scope 3-Daten von liefernden Unternehmen (den anderen Gruppen in der Schulung) eingebunden werden, um die Datenqualität zu verbessern. Der Austausch der Daten erfolgte über die in Unterabschnitt 9.5.4 beschriebene Funktion des JSON-Netz-Editors, Stellenmarkierungen über eine REST-Schnittstelle zu veröffentlichen oder einzubinden. Als Maß für die sich verbessernde Datenqualität wurde in der bereitgestellten Konfiguration des JSON-Netz-Editors der Primärdatenanteil von Wirkungsindikatoren berechnet. Der Primärdatenanteil gab an, wie hoch der Beitrag von spezifischen (also im Unternehmen erhobenen und nicht aus generischen Datenbanken stammenden) Datenquellen war [WBC21].

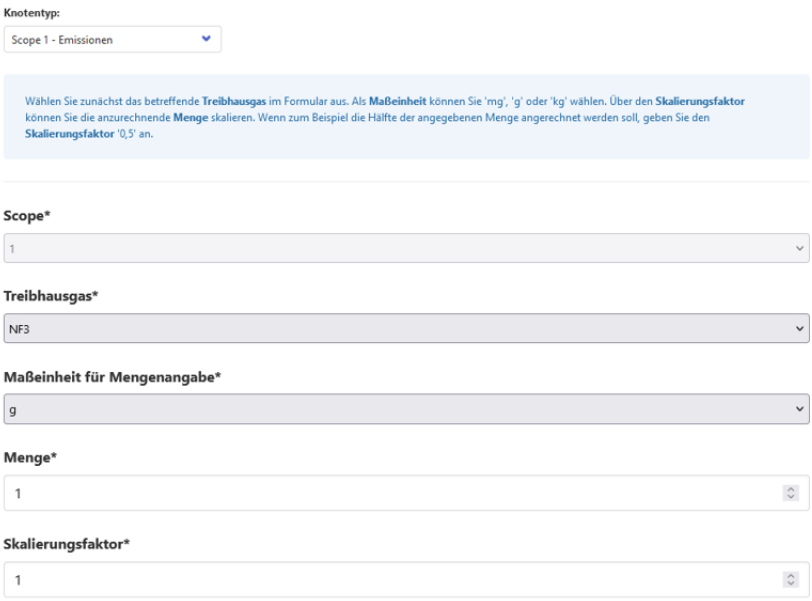


Abbildung 11.6: Formular zur Eingabe von Scope 1-Daten.

Die Teilnehmenden konnten zu jedem Zeitpunkt die Funktion des JSON-Netz-Editors zur teilautomatischen Simulation eines JSON-Netzes nutzen (Phase *Analyse und Verbesserung*). Mithilfe dieser Funktion konnte der Produktionsprozess einer

Komponente oder eines Produkts simuliert werden. Anhand der vorbereiteten Transitionsbeschriftungen wurde dann die Gesamt-Treibhausgasbilanz (in kg CO2e) einer Prozessinstanz, der Primärdatenanteil sowie Beiträge einzelner Scope 1-, Scope 2- und Scope 3-Emissionen berechnet und visualisiert. Von den Teilnehmenden wurden in abschließenden Diskussionsrunden mögliche Verbesserungsmaßnahmen vorgeschlagen. Beispielsweise konnte mit der Verlagerung des Produktionsstandorts ein geringerer Treibhausgas-Emissionsfaktor für Scope 2-Emissionen angenommen und die Gesamtbewertung damit verbessert werden. Dabei wurde auch die Plausibilität der Ergebnisse mit den im Projekt beteiligten Personen mit Branchen- und Nachhaltigkeitsexpertise diskutiert.

In jedem Schulungstermin wurde eine Benutzbarkeitsevaluation mit der System Usability Scale (SUS) Methode [Bro95] durchgeführt. Diese diente der Evaluation des JSON-Netz-Editors sowie der Aufnahme der Rückmeldungen und Verbesserungsvorschläge der Teilnehmenden. Die SUS-Methode stellt einen Fragebogen mit zehn Fragen bereit, mit dem die Benutzbarkeit eines Software-Werkzeugs bewertet werden kann. Anhand einer vorgegebenen Formel lässt sich aus den Antworten ein Wert zwischen 0 und 100 berechnen. Die Auswertung der Fragebögen ergab für die Probeschulung den Durchschnittswert 76,4, für die erste Iteration der Schulung den Wert 74,1, für die zweite 74,3 und für die dritte 74,1. SUS-Werte über 70 können nach [Ban08] als gute Benutzbarkeit interpretiert werden, was darauf hindeutet, dass der JSON-Netz-Editor von den Teilnehmenden angenommen wurde. In einem Freitextfeld hatten die Teilnehmenden darüber hinaus die Möglichkeit, Rückmeldungen oder Verbesserungsvorschläge mitzuteilen. Diese Verbesserungsvorschläge wurden in den einzelnen Iterationen genutzt, um den JSON-Netz-Editor und das Schulungskonzept weiter zu verbessern. Einige Teilnehmenden äußerten auch Interesse an weiterführenden Einsatzmöglichkeiten des JSON-Netz-Editors, was in Folgegesprächen weiter diskutiert wurde. Denkbar wäre beispielsweise eine Anbindung an Enterprise Resource Planning-Systeme der Unternehmen.

11.4 Anforderungsüberprüfung

In Kapitel 6 wurden basierend auf den vorgestellten Grundlagen Anforderungen an eine Modellierungsmethode für Nachhaltiges Geschäftsprozessmanagement aufgestellt. Im Folgenden wird jeweils die Anforderungsdefinition aus Kapitel 6 wiedergegeben und basierend auf der durchgeführten Evaluation zusammengefasst, wie die jeweilige Anforderung erfüllt wurde.

(A 1) Ausdrucksmächtigkeit: Die Modellierungssprache soll das zu untersuchende Phänomen (Geschäftsprozesse) darstellen können [Fra03]. Dazu zählt die Darstellbarkeit des Kontroll- und Datenflusses von Geschäftsprozessen.

Evaluation: Mit JSON-Netzen wird als Modellierungssprache eine Variante Höherer Petri-Netze zur Beschreibung von Geschäftsprozessen bereitgestellt. Höhere Petri-Netze sind für die Darstellung des Kontroll- und Datenflusses von Geschäftsprozessen geeignet [Aal15]. Für JSON-Netze wurde diese Eignung mit einer Reihe von erfolgreich durchgeführten Tests demonstriert.

(A 2) Analysierbarkeit: Mithilfe der Modellierungsmethode sollen, basierend auf den in Abschnitt 4.3 definierten Nachhaltigkeitsmustern, Aussagen zu den Nachhaltigkeitsauswirkungen eines Geschäftsprozesses ermöglicht werden [Obe96a, Fra03]. Dies bedeutet, dass Aussagen zu verschiedenen Inputs und Outputs eines Geschäftsprozesses (NMG 1), zu seinen sozialen und ökologischen Auswirkungen bezüglich unterschiedlicher Systemgrenzen (NMG 2 und NMG 3) sowie zur Allokation der Nachhaltigkeitsauswirkungen (NMG 4) gemacht werden können.

Evaluation: Die bereitgestellte Modellierungssprache ermöglicht die Simulation von Geschäftsprozessen. Über die Einbindung von Nachhaltigkeitsdaten können Aussagen über die Nachhaltigkeitsauswirkungen eines Geschäftsprozesses getroffen werden. Weiterhin wird ein Vorgehensmodell bereitgestellt, das auf Konzepten des Geschäftsprozessmanagements und der Lebenszyklusanalyse beruht. In einem Beispielszenario (siehe Kapitel 10) und einer Fallstudie (siehe Abschnitt 11.3) wurde die Darstellbarkeit der in Abschnitt 4.3 definierten Nachhaltigkeitsmuster demonstriert. Es wurden unterschiedliche Inputs und Outputs erfasst (zum Beispiel Energie, Arbeitszeit, Produkte), soziale und ökologische Auswirkungen (Klimawirkung, Arbeitsunfälle) für unterschiedliche Systemgrenzen (Scopes des Treibhausgasprotokolls [WBC04] und soziale Scopes nach [OEC22]) sowie Allokationen vorgenommen. Anhand der bereitgestellten Konzepte können auch weitere ökologische und soziale Auswirkungen betrachtet werden.

(A 3) Benutzbarkeit: Die Modellierungssprache und das Modellierungswerkzeug sollen für potentielle Anwendende leicht benutzbar sein [Fra03].

Evaluation: Es wurde zwischen technisch versierten Anwendenden und Anwendenden ohne Kenntnis der JSON-Syntax unterschieden. JSON wird in der Literatur als „leichtgewichtiges" Datenmodell beschrieben [Bou20] und hat bei Entwicklerinnen und Entwicklern weite Anwendung gefunden [Sta24, Goo24b]. Bei der Entwicklung des Modellierungswerkzeugs (siehe Kapitel 9) stand der Entwurf einer Benutzungsoberfläche, die Anwendende bei der Modellierung unterstützt und anleitet, im Fokus. Dazu wurden in der Literatur beschriebene Benutzbarkeitsheuristiken umgesetzt sowie geeignete Benutzungskonzepte in vergleichbaren Software-Werkzeugen identifiziert und übertragen. Weiterhin wurden ergänzende Funktionen implementiert, die auch Anwendenden ohne Kenntnis der JSON-Syntax die Benutzung des JSON-Netz-Editors ermöglichen. In einer Fallstudie wurde gezeigt, wie auch diese Zielgruppe Nachhaltigkeitsanalysen mithilfe des JSON-Netz-Editors durchführen kann. Die Benutzbarkeit des JSON-Netz-Editors wurde von den Anwendenden als gut bewertet.

(A 4) Anpassbarkeit Die Modellierungssprache soll für individuelle Anwendungsdomänen (spezielle Unternehmen, Branchen und Nachhaltigkeitsaspekte) anpassbar sein [Fra03].

Evaluation: Das Vorgehensmodell wurde generisch formuliert, sodass es für unterschiedliche Unternehmen, Branchen und Nachhaltigkeitsaspekte anwendbar ist. Im JSON-Netz-Editor wurden Funktionen umgesetzt, mit denen domänenspezifische Konfigurationen bereitgestellt werden können.

(A 5) Operationalisierbarkeit: Die Modellierungssprache soll die Entwicklung von Informationssystemen unterstützen [Fra03]. Daraus folgt, dass anhand der Spezifikation der Modellierungssprache Software-Werkzeuge umgesetzt werden können. Diese Software-Werkzeuge sollen auch den Datenaustausch mit anderen Informationssystemen unterstützen.

Evaluation: Die formale Spezifikation der JSON-Netze unterstützt die Entwicklung von Modellierungswerkzeugen und prozessorientierten Informationssystemen [Fra03]. Weiterhin wird die Operationalisierbarkeit durch die Integration des JSON-Datenmodells unterstützt. So lassen sich Datentypen des JSON-Datenmodells unmittelbar in Datentypen gängiger Programmiersprachen übersetzen [Dro22]. Es ist als Datenaustauschformat für Webdienste und als Speicherformat für NoSQL-Datenbanksysteme weit verbreitet [Bou20]. Diese Eigenschaften des JSON-Datenmodells begünstigen die Entwicklung von prozessorientierten Informationssystemen auf der Grundlage von JSON-Netzen. Mit der prototypischen

Entwicklung des JSON-Netz-Editors wurde die Operationalisierbarkeit von JSON-Netzen demonstriert. Auch die Unterstützung für den Datenaustausch mit (externen) Informationssystemen wurde im Rahmen der Fallstudie mit der Einbindung einer externen REST-Schnittstelle für den Austausch von Nachhaltigkeitsdaten gezeigt.

11.5 Diskussion

Im Folgenden werden die Beiträge und die Grenzen der hier vorgestellten Arbeit diskutiert. Die Diskussion erfolgt anhand von drei Aspekten. (1) Die Spezifikation der JSON-Netze und die im JSON-Netz-Editor gegebene Modellierungs- und Analyseunterstützung werden anderen existierenden Ansätzen zur datenorientierten Prozessmodellierung gegenübergestellt. (2) Der Beitrag der vorgestellten Methode im Vergleich zu den in der Literatur beschriebenen Ansätzen für Nachhaltiges Geschäftsprozessmanagement (siehe Kapitel 5) wird untersucht. (3) Die vorliegende Arbeit wird im Kontext aktueller Herausforderungen der Nachhaltigen Entwicklung sowie bestehender Ansätze für Nachhaltigkeitsanalysen besprochen. Die Diskussion bildet damit die Grundlage für die Vorstellung von Weiterentwicklungsmöglichkeiten und anknüpfenden Arbeiten in Kapitel 12.

11.5.1 Beitrag zur datenorientierten Prozessmodellierung

JSON-Netze unterstützen die datenorientierte Geschäftsprozessmodellierung, indem sie ein flexibles, einfach nutzbares und weit verbreitetes Datenmodell mit Petri-Netzen kombinieren. Sie können damit als eine Alternative zu anderen Varianten Höherer Petri-Netze wie gefärbten Petri-Netzen [Jen81], Prädikat/Transitionsnetzen [Gen81], Nested-Relation/Transitionsnetzen [Obe96c] oder XML-Netzen [Len03] eingeordnet werden.

Unabhängig von spezifischen Unterschieden bezüglich der gegebenen Ausdrucksmöglichkeiten, Benutzbarkeit oder Werkzeugunterstützung eignen sich alle genannten Varianten Höherer Petri-Netze dazu, sowohl den Kontroll- als auch den Datenfluss in Geschäftsprozessen darzustellen [Aal15]. Eine spezielle Ausdrucksmöglichkeit von JSON-Netzen ist, dass Sachverhalte dargestellt werden können, in denen eine Transition beim Schalten Substrukturen eines Datenobjekts verarbeitet (siehe Unterabschnitt 11.2.3). Diese Fähigkeit unterscheidet JSON-Netze (gemeinsam mit Nested-Relation/Transitions- und XML-Netzen) von anderen höheren Petri-Netzen.

Damit lassen sich entsprechende Sachverhalte (zum Beispiel die Bearbeitung eines Auftrags mit Auftragspositionen, die Herstellung eines Elektronikprodukts mit mehreren Komponenten, ...) kompakter darstellen [Obe96c]. Weiterhin kann auf diese Weise bei der Simulation und Ausführung von Geschäftsprozessen auch ein nebenläufiger Zugriff auf unterschiedliche Substrukturen desselben Datenobjekts umgesetzt werden [Obe96c]. Die unterschiedlichen Varianten Höherer Petri-Netze unterscheiden sich weiterhin im jeweils verwendeten Datenmodell für Stellenmarkierungen. Die Vor- und Nachteile des verwendeten Datenmodells übertragen sich damit auch auf die datenorientierte Prozessmodellierung mit einem Höheren Petri-Netz. Gefärbte Petri-Netze nutzen für die Typisierung von Stellen das Datenmodell einer Programmiersprache [Aal97]. Prädikat/Transitions-, Nested-Relation/Transitions-, und XML-Netze haben gemein, dass sie Petri-Netze mit Datenmodellen verknüpfen, die auch in Datenbanksystemen zum Einsatz kommen [Obe96c, Len03]. Im Kontext der Entwicklung von Informationssystemen hat der Einsatz von Datenbank-Datenmodellen bei der datenorientierten Prozessmodellierung den Vorteil, dass Daten in der Form verarbeitet werden können, in der sie in Datenbanken gespeichert vorliegen [Len03]. XML-Netze haben hier den weitergehenden Vorteil, dass es sich bei XML um ein flexibles und textbasiertes Datenmodell handelt [Nur09], das sich auch für den Datenaustausch zwischen Datenbanken beziehungsweise zur Kommunikation mit externen Informationssystemen eignet [Len03]. Gefärbte Petri-Netze können aufgrund des gewählten Programmiersprachen-Datenmodells als Software-näher bezeichnet werden und Prädikat/Transitions-, Nested-Relation/Transitions- und XML-Netze als Datenbank-näher. Sollen, basierend auf einer der gegebenen Varianten Höherer Petri-Netze, Software-Werkzeuge und (prozessorientierte) Informationssysteme entwickelt werden, müssen jedoch Datenbanken und Software miteinander interagieren. Eine Herausforderung in der Softwareentwicklung ist, dass Typsysteme von Programmiersprachen und Datenbanken nicht direkt, beziehungsweise nur mit zusätzlichem Aufwand, ineinander übertragbar sind (der sogenannte „Impedance Mismatch“ [Ire09, Col20, Hew20]). JSON hat als Datenbank-Datenmodell spezifische Vorteile und nimmt auch bei der Herausforderung des Impedance Mismatch eine Sonderrolle ein. Es ist ein flexibles und textbasiertes Datenmodell und eignet sich damit genauso wie XML für den Datenaustausch [Nur09]. In vielen Anwendungsfällen hat es XML als Datenmodell abgelöst. So werden seit ungefähr der Mitte der 2010er Jahre bei Google und auf Stackoverflow, einer Austauschplattform für Entwicklerinnen und Entwickler, mehr Anfragen mit Bezug zu JSON als zu XML gestellt [Sta24, Goo24b]. In der Literatur wird seine Beliebtheit bei Entwicklerinnen und Entwicklern unter anderem damit erklärt, dass es als leichtgewichtig (in Bezug auf Syntaxregeln) und lesbar wahrgenommen wird [Nur09, Hew20]. Darüber hinaus hilft die Nutzung des

JSON-Datenmodells, das Problem des Impedance Mismatch zu verringern [Hew20]. So kommt das JSON-Datenmodell einerseits in (NoSQL-)Datenbanksystemen zum Einsatz und ist andererseits auch unmittelbar (im Fall von JavaScript) oder mit wenig Aufwand in das Typsystem von gängigen Programmiersprachen übertragbar [Nur09]. Diese Stärken des JSON-Datenmodells zeigten sich auch bei der Entwicklung des JSON-Netz-Editors und insbesondere bei der Bereitstellung von Funktionen zur Unterstützung des Datenaustauschs und der Benutzbarkeit (siehe Kapitel 9). Die Wahl von JSON als Datenmodell für die datenorientierte Prozessmodellierung mit Petri-Netzen unterstützt somit die Entwicklung von benutzungsfreundlichen, prozessorientierten Informationssystemen, die auch mit externen Systemen und Datenbanken interagieren.

Die Evaluation mit Kontroll- und Datenflussmustern in Abschnitt 11.2 zeigt auch Grenzen der Spezifikation und Implementierung von JSON-Netzen auf. Im Vergleich zu beispielsweise gefärbten Petri-Netzen fehlt bisher die Möglichkeit hierarchische Verfeinerungen von Prozessmodellen darzustellen. Eine Ergänzung entsprechender Ausdrucksmöglichkeiten würde die Bewertung der Umsetzbarkeit der untersuchten Kontroll- und Datenflussmuster (wie zum Beispiel WDP 2) weiter verbessern. Daneben sind auch Grenzen von JSON als gewähltem Datenmodell zu nennen. Beispielsweise gibt es mehr formale Untersuchungen der Eigenschaften des XML-Datenmodells und von XML-Anfrage- und Schemasprachen [Bou20]. Auch die Standardisierung von Anfrage- und Schemasprachen ist für das XML-Datenmodell weiter vorangeschritten. Während die Standards für Schema- und Anfragesprachen für das XML-Datenmodell stabil und etabliert sind [W3C17b, W3C17a], befinden sich viele Standards im Umfeld des JSON-Datenmodells, wie beispielsweise für JSON Schema [Wri22b] oder JSONPath [IET24], noch im Entwurfsstadium. Insbesondere gefärbte Petri-Netze wurden in zahlreichen wissenschaftlichen Arbeiten untersucht und angewandt, sodass für sie in größerem Umfang Unterstützung für Simulationen und formale Analysen bereitsteht [Jen93, Jen90, Jen94]. Zukünftig können formale Untersuchungen und Adaptionen von Petri-Netz-Eigenschaften wie Beschränktheit, Terminierung oder Lebendigkeit [Rei10] sowie die Ausarbeitung eines Simulationskonzepts, die Analysefähigkeiten von JSON-Netzen und JSON-Netz-Werkzeugen erweitern.

11.5.2 Beitrag zum Nachhaltigen Geschäftsprozessmanagement

In der vorliegenden Arbeit wurde ein Ansatz zur Modellierung, Analyse und Verbesserung der Nachhaltigkeit von Geschäftsprozessen vorgestellt. Basierend auf einer Gegenüberstellung von Konzepten des (konventionellen) Geschäftsprozessmanagements und bestehenden Methoden der Nachhaltigkeitsanalyse (Lebenszyklusanalyse) wurde eine Methode für Nachhaltiges Geschäftsprozessmanagement erarbeitet. Im Folgenden werden die Beiträge und Grenzen des vorliegenden Ansatzes im Vergleich zu den in Kapitel 5 vorgestellten bestehenden Ansätzen für Nachhaltiges Geschäftsprozessmanagement diskutiert.

Der Ansatz der vorliegenden Arbeit wurde in einem Software-Werkzeug umgesetzt und in einem realitätsnahen Kontext in Unternehmensschulungen evaluiert. Im Vergleich dazu haben andere bestehende Ansätze (siehe Kapitel 5) einen geringeren Reifegrad, da sie sich auf die Beschreibung von Konzepten und Beispielen beschränken, die noch nicht in einem Software-Werkzeug umgesetzt wurden. Darüber hinaus bietet der vorliegende Ansatz in mehreren Aspekten eine höhere Aussagekraft bei der Modellierung und Analyse der Nachhaltigkeitsauswirkungen von Geschäftsprozessen als bestehende Ansätze. In der vorliegenden Arbeit wurden für die Modellierung und Analyse der Nachhaltigkeitsauswirkungen von Geschäftsprozessen Konzepte aus bestehenden Methoden der Nachhaltigkeitsanalyse (Lebenszyklusanalyse) abgeleitet. Dazu zählt die Einführung einer Unterscheidung von Inventarindikatoren und Wirkungsindikatoren. Diese Unterscheidung beruht auf dem in der Lebenszyklusanalyse angewandten Prinzip, aus identifizierten Stoffströmen eines Systems Nachhaltigkeitsauswirkungen abzuleiten [ISO06a]. So kann beispielsweise zwischen der Ermittlung des Ölverbrauchs eines Geschäftsprozesses und der Ermittlung der Klimawirkung eines Geschäftsprozesses unterschieden werden. Der Ölverbrauch eines Geschäftsprozess kann als Inventarindikator erfasst und über die Umrechnung in einen Wirkungsindikator (zum Beispiel Klimawirkung gemessen in kg CO2e) kann eine Aussage über die Nachhaltigkeitsauswirkungen des Geschäftsprozesses getroffen werden. Daneben wurde, ebenfalls basierend auf Konzepten der Lebenszyklusanalyse [ISO06a], in der vorliegenden Arbeit das Konzept von Systemgrenzen für Nachhaltigkeitsanalysen im Geschäftsprozessmanagement eingeführt. Bei der Nachhaltigkeitsanalyse eines Geschäftsprozesses sind demnach Kriterien festzulegen, in welchem Umfang Inventarindikatoren zu ermitteln, und wie anzugebende Wirkungsfaktoren zu interpretieren sind. Die beschriebenen Konzepte wurden in einem Vorgehensmodell integriert und es wurde gezeigt, wie anhand des Vorgehensmodells unterschiedliche ökologische und soziale Nachhaltigkeitsauswirkungen von Geschäftsprozessen untersucht werden

können. In den Ansätzen von Houy et al. [Hou11, Hou12], Zhu et al. [Zhu15] und Betz [Bet14] wird die Unterscheidung von Inventar- und Wirkungsindikatoren sowie die Definition von Systemgrenzen nicht unterstützt. Die Ansätze von Hoesch-Klohe et al. [Gho10, Hoe10c, Hoe10a, Hoe10b, Hoe12a, Hoe12b], Recker et al. [Rec11, Rec12] und Wesumperuma et al. [Wes11, Wes13, Wes15] orientieren sich am Treibhausgasprotokoll [WBC04] und nicht an den allgemeineren Prinzipien der Lebenszyklusanalyse. Daher wurden mit ihnen bisher lediglich Analysen der Klimawirkung von Geschäftsprozessen gezeigt und nicht für weitere Nachhaltigkeitsapekte.

Die in dieser Arbeit vorgenommene Einführung eines Allokationsfaktors ermöglicht die anteilige Anrechnung von Inventar- oder Wirkungsindikatoren auf Aktivitätsinstanzen. Das Thema Allokation wurde auch in den Ansätzen von Hoesch-Klohe et al. [Gho10, Hoe10c, Hoe10a, Hoe10b, Hoe12a, Hoe12b], Recker et al. [Rec11, Rec12] und Wesumperuma et al. [Wes11, Wes13, Wes15] behandelt. Diese Ansätze beschreiben (zum Teil modellgestützte) Vorgehensweisen, um mit genutzten Ressourcen zusammenhängende Nachhaltigkeitsauswirkungen auf Aktivitäten oder Aktivitätsinstanzen anzurechnen. Beispielsweise werden in [Hoe10c, Hoe10b] unterschiedliche Zusammenhänge zwischen Ressourcen (zum Beispiel ein Drucker benötigt Papier und Energie) beschrieben und zur Analyse der Nachhaltigkeitsauswirkungen eines Geschäftsprozesses herangezogen. Diese Modellierungsansätze könnten in Zukunft adaptiert und als Konfiguration im JSON-Netz-Editor bereitgestellt werden.

Die hier vorgestellte Arbeit adressiert eine Reihe der in Abschnitt 5.3 auf Basis einer systematischen Literaturrecherche identifizierten Forschungsbedarfe für Nachhaltiges Geschäftsprozessmanagement. In einem interdisziplinären Projekt wurden Konzepte der Lebenszyklusanalyse erfolgreich mit Konzepten des Geschäftsprozessmanagements kombiniert (siehe der Bericht zur Fallstudie in Abschnitt 11.3). Im Unterschied zu vergleichbaren Ansätzen für Nachhaltiges Geschäftsprozessmanagement, die in vielen Fällen in einem konzeptionellen Stadium sind, wurden die hier vorgestellten Konzepte in einem Software-Werkzeug und in einer praxisnahen Anwendung evaluiert. Weiterhin stellt die vorgenommene Integration von Konzepten der Lebenszyklusanalyse eine fundierte Basis für die Betrachtung von Nachhaltigkeitsaspekten im Geschäftsprozessmanagement dar. So konnte gezeigt werden, dass die hier vorgestellten Konzepte eine umfangreichere Betrachtung von Nachhaltigkeitsaspekten (insbesondere soziale Nachhaltigkeitsaspekte, siehe Abschnitt 10.2) unterstützen.

In der Anwendung der Methode in einem Beispiel-Szenario (siehe Kapitel 10) und in einer Fallstudie (siehe Abschnitt 11.3) wurde gezeigt, wie der JSON-Netz-Editor konfiguriert werden kann, um Nachhaltigkeitsanalysen für verschiedene Nachhaltigkeitsaspekte und Anwendungsfälle zu unterstützen. Die Anwendung des Vorgehensmodells und die Nutzung beziehungsweise Anpassung des JSON-Netz-Editors setzt dabei Fachwissen zu Nachhaltigkeitsanalysen und zu JSON-Technologien voraus. Hier können das Vorgehensmodell und Konfigurationen des JSON-Netz-Editors in Zukunft noch weiter ausgearbeitet werden, um Hilfestellungen für unterschiedliche Nachhaltigkeitsaspekte und Anwendungsfälle zu bieten.

11.5.3 Beitrag zur Nachhaltigen Entwicklung

In Kapitel 4 wurden Konzepte und Methoden des Geschäftsprozessmanagements und der Nachhaltigkeitsanalyse einander gegenübergestellt. Aus der Gegenüberstellung wurde das Argument abgeleitet, dass (zu entwickelnde) Methoden für Nachhaltiges Geschäftsprozessmanagement einen Beitrag bei der Operationalisierung von Nachhaltigkeitsanalysen leisten können. Mit der in dieser Arbeit vorgestellten Modellierungsmethode wurde demonstriert, wie dieser Beitrag realisiert werden kann, indem beispielsweise Simulationen von Geschäftsprozessen durchgeführt werden können, oder der Datenaustausch zur Analyse von Treibhausgasemissionen in Elektronikliefernetten unterstützt wird. Die Bereitstellung von spezialisierten, leicht benutzbaren Software-Werkzeugen sowie Lösungen für verbesserte Datenintegration wird sowohl in der Literatur [Cir12, Bac19] als auch von der Industrie selbst [Sco22] als aktuelle Herausforderungen der Nachhaltigkeitsanalyse gesehen. Es wurde gezeigt, dass mit der hier vorgestellten Modellierungsmethode spezialisierte Software-Werkzeuge erstellt werden können, um diesem Bedarf zu begegnen. Die Modellierungsmethode soll dabei bestehende Nachhaltigkeitsanalysemethoden nicht ersetzen sondern ergänzen. Beispielsweise wird im illustrativen Szenario davon ausgegangen, dass Inventarindikatoren für die Anrechnung auf Aktivitätsinstanzen bekannt, beziehungsweise ermittelbar sind. Dagegen dient der in Abschnitt 3.4 beschriebene Ansatz der Stoffstromnetze, der im Lebenszyklusanalyse-Werkzeug Umberto [iPo24] umgesetzt ist, auch dazu, unbekannte Stoffströme (zum Beispiel anhand von anderen, bekannten Stoffströmen und funktionalen Zusammenhängen) in einem Prozess zu ermitteln [Häu97, Möl95]. Aufbauend auf der hier vorliegenden Arbeit können in Zukunft Ansätze der Lebenszyklusanalyse und des Geschäftsprozessmanagements tiefer integriert werden. So können beispielsweise Schnittstellen geschaffen werden, um in

Lebenszyklusanalyse-Werkzeugen ermittelte Werte als Inventar- oder Wirkungsindikatoren in Geschäftsprozessmodelle einzubinden. Auf der anderen Seite können beispielsweise auch in Geschäftsprozessmanagement-Werkzeugen durchgeführte Simulationen Daten für Lebenszyklusanalysen liefern.[1] Auf anknüpfende Ideen und Ansätze dazu wird im folgenden Kapitel näher eingegangen.

[1] Ähnliche Ideen beschreiben zum Beispiel [Wid14, Rei13] für ökologische und soziale Lebenszyklusanalysen. Hier wird jedoch nicht explizit auf Konzepte des Geschäftsprozessmanagements eingegangen, sondern Ansätze zur Kombination von Materialflussanalysen und diskreten Simulationen vorgestellt.

12 Zusammenfassung und Ausblick

Angesichts sich verschärfender globaler Herausforderungen wie der Klimakrise wird von Unternehmen zunehmend gefordert, dass sie die sozialen und ökologischen Auswirkungen ihres Handelns identifizieren und verbessern. Konzepte, Methoden und Werkzeuge des Nachhaltigen Geschäftsprozessmanagements können Unternehmen dabei unterstützen. Die vorliegende Arbeit stellt dazu eine innovative Modellierungsmethode bereit, die es Unternehmen ermöglicht, die Nachhaltigkeitsaspekte ihrer Geschäftsprozesse systematisch und umfassend zu untersuchen sowie kontinuierlich zu verbessern. Die systematische Kombination von Konzepten des Geschäftsprozessmanagements und der Lebenszyklusanalyse stellt dabei einen wesentlichen Fortschritt im Bereich des Nachhaltigen Geschäftsprozessmanagements dar. Bei der Entwicklung der Modellierungsmethode wurden konzeptuell anspruchsvolle Herausforderungen gelöst (formale Spezifikation der JSON-Netze, Synthese von Konzepten der Lebenszyklusanalyse und des Geschäftsprozessmanagements), um im Ergebnis eine praxisrelevante Lösung für aktuelle gesellschaftliche und unternehmerische Herausforderungen zu bieten. Im Folgenden werden die Ergebnisse der vorliegenden Arbeit zusammengefasst und ein Ausblick auf mögliche Weiterentwicklungen der vorgestellten Modellieurngsmethode sowie anknüpfende Arbeiten gegeben.

12.1 Nachhaltigkeitsmuster für Geschäftsprozesse

Zur Vorbereitung des Methodenentwurfs wurden zunächst bestehende Konzepte und Methoden des Geschäftsprozessmanagements untersucht (siehe Kapitel 2). Konzepte und Methoden des Geschäftsprozessmanagements zielen darauf ab, die Leistung von Unternehmen zu verbessern und stellen IT-Unterstützung für die Modellierung, Analyse, Verbesserung und Umsetzung von Geschäftsprozessen bereit [Wes19]. Für die Bereitstellung von Konzepten und Methoden für Nachhaltiges Geschäftsprozessmanagement ist demnach zunächst die Frage zu klären, wie die Nachhaltigkeitsauswirkungen von Geschäftsprozessen modelliert werden können. Dazu wurden im nächsten Schritt bestehende Nachhaltigkeitsanalysemethoden untersucht (siehe Kapitel 3). Der

Fokus lag dabei auf der Lebenszyklusanalyse, als zentraler und umfassender Ansatz der Nachhaltigkeitsanalyse [ISO06a]. Auf der Grundlage einer Gegenüberstellung von Konzepten des Geschäftsprozessmanagements mit Konzepten der Lebenszyklusanalyse wurden Nachhaltigkeitsmuster für Geschäftsprozesse identifiziert (siehe Kapitel 4). Diese Nachhaltigkeitsmuster beschreiben unterschiedliche Aspekte der Nachhaltigkeitsauswirkungen von Geschäftsprozessen. Ihre Umsetzung in Ansätzen für Nachhaltiges Geschäftsprozessmanagement ermöglicht es, in der Lebenszyklusanalyse bewährte Konzepte der Analyse von Nachhaltigkeitsauswirkungen auf das Geschäftsprozessmanagement zu übertragen. Sie beschreiben die Unterscheidung zwischen nachhaltigkeitsrelevanten Inputs und Outputs eines Geschäftsprozesses (NMG 1), seinen Nachhaltigkeitsauswirkungen für unterschiedliche Wirkungskategorien (NMG 2) sowie seinen Nachhaltigkeitsauswirkungen für unterschiedliche Systemgrenzen (NMG 3). Weiterhin wird mit dem Nachhaltigkeitsmuster NMG 4 die Allokation von Nachhaltigkeitsauswirkungen zwischen Prozesskomponenten beschrieben. Die Nachhaltigkeitsmuster dienten im weiteren Verlauf der Arbeit der strukturierten Untersuchung bestehender Ansätze für Nachhaltiges Geschäftsprozessmanagement sowie der Definition und Überprüfung von Anforderungen für die in dieser Arbeit entwickelte Modellierungsmethode. Es wurde festgestellt, dass bestehende Ansätze für Nachhaltiges Geschäftsprozessmanagement die identifizierten Nachhaltigkeitsmuster nur unzureichend unterstützen (siehe Kapitel 5).

12.2 Entwicklung der Modellierungsmethode

Als Beitrag zur Unterstützung der Operationalisierung von Nachhaltigkeitsanalysen in Unternehmen wurde eine Modellierungsmethode für Nachhaltiges Geschäftsprozessmanagement entwickelt und vorgestellt. Aus den zuvor erarbeiteten Grundlagen des Nachhaltigen Geschäftsprozessmanagements wurden Anforderungen an die Modellierungsmethode abgeleitet (siehe Kapitel 6). Sie soll einerseits die Modellierung des Kontroll- und Datenflusses von Geschäftsprozessen unterstützen (Ausdrucksmächtigkeit). Andererseits soll sie Aussagen über die Nachhaltigkeitsauswirkungen der modellierten Geschäftsprozesse ermöglichen (Analysierbarkeit). Weiterhin soll sie leicht von Anwendenden benutzbar sein (Benutzbarkeit) und die Umsetzung von Software-Werkzeugen sowie die Integration mit betrieblichen Informationssystemen unterstützen (Operationalisierbarkeit). Die aufgestellten Anforderungen wurden mit den drei Komponenten der Modellierungsmethode, dem Vorgehensmodell, der Modellierungssprache und dem Modellierungswerkzeug umgesetzt.

Das Vorgehensmodell beschreibt, wie ein Unternehmen die Nachhaltigkeit seiner Geschäftsprozesse modellieren, analysieren und verbessern kann (siehe Kapitel 7). Dazu werden erstmalig Konzepte der Lebenszyklusanalyse mit Ansätzen des Geschäftsprozessmanagements kombiniert, um so eine tiefgehende und fundierte Betrachtung von Nachhaltigkeitsaspekten bei der Analyse von Geschäftsprozessen zu ermöglichen. Es bezieht dabei die definierten Nachhaltigkeitsmuster für Geschäftsprozesse ein und ist unabhängig von konkreten Prozessmodellierungssprachen oder -werkzeugen.

JSON-Netze wurden als datenorientierte Prozessmodellierungssprache entwickelt (siehe Kapitel 8), um das Vorgehensmodell technisch zu unterstützen. Sie kombinieren dazu JSON, ein flexibles, einfach nutzbares und weit verbreitetes Datenmodell [Goo24b, IET17, Sta24], mit Petri-Netzen. Auf diese Weise ermöglichen sie die Modellierung und Analyse von Geschäftsprozessen unter Einbeziehung von Nachhaltigkeitsaspekten. Sie eignen sich darüber hinaus in besonderer Weise für die Operationalisierung von Nachhaltigkeitsanalysen, da das verwendete Datenmodell sowohl die Software-Entwicklung als auch die Integration von Datenbanken unterstützt.

Der JSON-Netz-Editor unterstützt als Modellierungswerkzeug die Anwendung der Modellierungssprache (siehe Kapitel 9). Er zeichnet sich durch seine hohe Benutzbarkeit und Anpassbarkeit aus und ermöglicht es Unternehmen, die entwickelte Modellierungsmethode effektiv zu nutzen. Im JSON-Netz-Editor wurden Grundfunktionen zur Modellierung mit JSON-Netzen umgesetzt. Dazu zählen die grafische Modellierung von Netzstrukturen, das Editieren von Stellen, Transitionen und Kanten sowie die Simulation von JSON-Netzen. Für jede der genannten Funktionen wurden unterstützende Mechanismen, wie zum Beispiel Rückmeldungen über gültige Markierungen, implementiert. Darüber hinaus wurden ergänzende Funktionen umgesetzt, welche die Nutzung des JSON-Netz-Editors auch für Anwendende ohne Kenntnis der JSON-Syntax ermöglichen. Zu diesen ergänzenden Funktionen zählen unter anderem die Generierung von Formularen aus Stellenschemas sowie die Visualisierung von Daten in Markierungen.

12.3 Evaluation der Modellierungsmethode

Die Erfüllung der einzelnen Anforderungen bezüglich der unterschiedlichen Komponenten der Modellierungsmethode wurden mit verschiedenen Evaluationsmethoden untersucht (siehe Kapitel 11). Der JSON-Netz-Editor demonstriert als Prototyp die software-technische Umsetzbarkeit der JSON-Netz-Spezifikation. Zusätzlich wurde in

einem Beispielszenario gezeigt, wie mit dem JSON-Netz-Editor anhand des Vorgehensmodells die Nachhaltigkeitsauswirkungen von Geschäftsprozessen analysiert werden können (siehe Kapitel 10). Basierend auf in der Literatur beschriebenen Kontroll- und Datenflussmustern wurde eine Reihe von Tests für den JSON-Netz-Editor durchgeführt. Die erfolgreich durchgeführten Tests demonstrieren, dass mittels JSON-Netzen Kontroll- und Datenflüsse von Geschäftsprozessen darstellbar sind.

Neben dem Beispielszenario wurde als weitere Demonstration der Einsatzmöglichkeiten der Modellierungsmethode eine Fallstudie in der Elektronikindustrie durchgeführt. Zusammen mit Projektpartnern wurde in einem BMBF-geförderten Projekt in der Nationalen Klimainitiative ein Schulungskonzept für Treibhausgasbilanzierungen in Elektronikunternehmen ausgearbeitet. An insgesamt drei Terminen wurde den jeweils 8 bis 15 Teilnehmenden aus Unternehmen der Elektronikindustrie zunächst Grundlagen der Treibhausgasbilanzierung vermittelt. In einem Praxisteil sollten die Teilnehmenden die vermittelten Inhalte für ein realistisches Szenario, welches die Lieferkette einer Kameraherstellung umfasst, anwenden. Bei der Vorbereitung und Durchführung der Schulungen kam das in dieser Arbeit entwickelte Vorgehensmodell und der JSON-Netz-Editor zum Einsatz. Zur Vorbereitung der Schulungen wurde gemeinsam mit den Nachhaltigkeits- und Branchenexpertinnen und -experten ein Indikatormodell entworfen und als Konfiguration des JSON-Netz-Editors umgesetzt. Während der Schulungen konnten die Teilnehmenden bereitgestellte Unternehmensdaten in Prozessmodelle integrieren. Darüber hinaus wurden über eine REST-Schnittstelle Daten zwischen den Teilnehmenden (die verschiedene Unternehmen der Lieferkette repräsentierten) ausgetauscht. Anhand der im JSON-Netz-Editor bereitgestellten Analyse- und Visualisierungsmöglichkeiten wurden von den Teilnehmenden mögliche Maßnahmen zur Reduzierung der Treibhausgasemissionen im Schulungsszenario diskutiert. Zur Untersuchung der Benutzbarkeit des JSON-Netz-Editors wurde eine Benutzbarkeitsumfrage durchgeführt. Die Auswertung der Benutzbarkeitsumfrage ergab, dass der JSON-Netz-Editor von den Teilnehmenden gut angenommen wurde.

12.4 Weiterentwicklungen und anknüpfende Arbeiten

Die in dieser Arbeit vorgestellte Modellierungsmethode bietet eine Grundlage für Weiterentwicklungen und anknüpfende Arbeiten. Im Folgenden wird dargelegt, wie die mit dem JSON-Netz-Editor gegebene Modellierungs- und Analyseunterstützung weiterentwickelt werden kann (Unterabschnitt 12.4.1). In Unterabschnitt 12.4.2 werden anknüpfende Arbeiten zur Untersuchung theoretischer Aspekte im Zusammenhang

mit JSON-Netzen beschrieben. Unterabschnitt 12.4.3 gibt einen Überblick zu anknüpfenden Arbeiten zur Erweiterung der Modellierungsmethode. Schließlich werden in Unterabschnitt 12.4.4 Anknüpfungspunkte zu aktuellen Arbeiten im Bereich Process Mining aufgezeigt.

12.4.1 Erweiterung des JSON-Netz-Editors

Die Bereitstellung von benutzbaren Modellierungswerkzeugen wird in der Literatur als Herausforderung der datenorientierten Prozessmodellierung beschrieben [Rei17]. Ein Schwerpunkt bei der Entwicklung des JSON-Netz-Editors lag daher auf der Bereitstellung einer leicht zu bedienenden Benutzungsoberfläche. Weiterführende Arbeiten können über die systematische Untersuchung und Adaption von Benutzbarkeitsheuristiken einen Beitrag dazu leisten, datenorientierte Prozessmodellierung für breitere Anwendungsgruppen zu erschließen. Ein Ansatzpunkt hierfür liegt in der Adaption von Low-Code-Programmierkonzepten [Hir23]. Zur Unterstützung der Erstellung von Transitionsbeschriftungen kann beispielsweise eine Bibliothek von Funktionen bereitgestellt werden, bei denen nur noch einzelne Parameter konfiguriert werden müssen. Dies würde es Anwendenden ermöglichen, komplexe Aufgaben mit geringerem Aufwand zu erledigen und so die Produktivität bei der Modellierung steigern. Auch neue Ansätze zur KI-gestützten Generierung von Prozessmodellen [For24] und Wirkungsindikatoren [Mai24] aus unstrukturierten Daten können genutzt werden, um Anwendende bei der Erstellung von JSON-Netzen sowie bei der Erhebung und Integration von Nachhaltigkeitsdaten zu unterstützen.

12.4.2 Untersuchung und Weiterentwicklung von JSON-Netzen

Im Rahmen der Evaluation wurde gezeigt, dass mit JSON-Netzen unterschiedliche Kontroll- und Datenflussphänomene darstellbar sind. Die Weiterentwicklung der Spezifikation kann die gegebenen Ausdrucksmöglichkeiten weiter steigern. Dazu gehört die Möglichkeit, beim Schalten einer Transition das Entfernen oder Einfügen mehrerer Werte zuzulassen. Die in Kapitel 8 vorgestellte Spezifikation von JSON-Netzen ist dazu so zu erweitern, dass mehrere der von einem Filterausdruck ausgewählten Werte entfernt oder eingefügt werden können. Dadurch können gewisse Sachverhalte, wie beispielsweise die in Unterabschnitt 11.2.1 beschriebenen Sonderfälle, mit JSON-Netzen kompakter dargestellt werden.

Das Schaltverhalten von JSON-Netzen kann künftig mit der Adaption von Petri-Netz-Eigenschaften wie Deadlock-Freiheit oder der Erreichbarkeit bestimmter Markierungen [Rei10] näher untersucht werden. Darauf aufbauend können entsprechende Algorithmen bereitgestellt werden, mit deren Hilfe beispielsweise die zuverlässige Ausführung eines Prozessmodells gewährleistet werden kann. Ein Anknüpfungspunkt hierfür ist die Arbeit [Bad15], in der formale Eigenschaften von Petri-Netzen mit semistrukturierten Daten untersucht werden. Für die Markierung von Stellen werden dabei nicht Daten in einem konkreten Datenmodell wie JSON oder XML verwendet, sondern allgemein Datenstrukturen, die als Bäume darstellbar sind. In [Bad15] wird gezeigt, dass (für die gegebene Definition eines Höheren Petri-Netzes) Eigenschaften wie Erreichbarkeit ohne zusätzliche Einschränkungen für mögliche Beschriftungen und Markierungen nicht entscheidbar sind. Mit der Einführung von gewissen Einschränkungen (zum Beispiel für die zulässige Baumtiefe) kann jedoch Entscheidbarkeit gezeigt werden [Bad15].

[Bad15] betrachtet dabei nicht das Einfügen und Entfernen von Substrukturen in Marken (beziehungsweise Teilbäumen in Markierungen), wie es mit Nested-Relation/Transitions-, XML- und JSON-Netzen möglich ist. Für die Übertragbarkeit der Ergebnisse von [Bad15] muss daher zunächst die Einschränkung vorgenommen werden, dass beim Schalten einer Transition ausschließlich ganze Marken verarbeitet werden. Künftig sind daher im nächsten Schritt auch formale Eigenschaften für JSON-Netze zu untersuchen, in denen der Zugriff auf Substrukturen von Marken zugelassen ist. Ein Ansatzpunkt hierfür ist die Eigenschaft des JSON-Datenmodells, dass jeder Teilbaum eines JSON-Baums selbst wieder ein gültiges JSON-Dokument darstellt (siehe Abschnitt 8.3). Jeder Teilbaum einer gegebenen Markierung lässt sich demnach als individuelle Marke interpretieren. Das Entfernen oder Einfügen eines Werts (eines Teilbaums und dessen Teilbäumen) in einer Markierung entspricht in dieser Interpretation des Schaltverhaltens von JSON-Netzen dem Entfernen oder Einfügen einer variablen Anzahl von Marken. Demnach kann das Gewicht der Kanten als dynamisch und abhängig von der gegebenen Markierung und Beschriftung angesehen werden. In der in Unterabschnitt 2.3.2 gegebenen Definition von elementaren Petri-Netzen werden beim Schalten eine Marke aus einer Stelle entfernt beziehungsweise in eine Stelle eingefügt. Die Eigenschaften von Petri-Netzen, in denen eine dynamische Anzahl von Marken eingefügt und entfernt werden kann, wird beispielsweise in [Cia94] untersucht. Die hier skizzierte Interpretation des Schaltverhaltens von JSON-Netzen bietet damit einen Ansatzpunkt, um auf bestehenden Ergebnissen der Untersuchung formaler Eigenschaften von Petri-Netzen aufzubauen.

Die gegebenen Analysefähigkeiten können darüber hinaus mit der Entwicklung eines umfangreichen Simulationskonzepts verbessert werden. Dazu können bestehende Petri-Netz-Simulationskonzepte (siehe zum Beispiel [Obe96c, Geh19]) für JSON-Netze adaptiert werden. Die Implementierung von Funktionen zur (zufälligen) Generierung unterschiedlicher Markierungen würde es ermöglichen, variierende Daten (zum Beispiel variierende Strombedarfe oder andere Inventarindikatoren) bei der Simulation mit JSON-Netzen zu berücksichtigen. In Kombination mit einem erweiterten Simulationsalgorithmus, der potentielle Schaltfolgen ermittelt, würden so umfangreichere und aussagekräftigere Analysen ermöglicht. Bei der Entwicklung eines Simulationsalgorithmus für JSON-Netze ist ebenfalls die Möglichkeit des nebenläufigen Zugriffs mehrerer Transitionen auf Substrukturen desselben Datenobjekts näher zu untersuchen (siehe Unterabschnitt 11.2.3). Die effiziente Ermittlung von konfliktfreien Belegungen (siehe Unterabschnitt 8.4.1), unter denen Transitionen aktiviert sind, stellt dabei eine zu lösende Herausforderung dar.

12.4.3 Erweiterung der Modellierungsmethode

Die Definition und Interpretation von Nachhaltigkeitsindikatoren wird im JSON-Netz-Editor über die Einführung von Inventarstellen und ergänzenden Visualisierungsmöglichkeiten unterstützt (siehe Kapitel 10). In [Büd20] wird ein geeignetes Dashboard-Konzept beschrieben, mit dem die gegebenen Visualisierungsmöglichkeiten erweitert werden können. Das Dashboard-Konzept sieht vor, neben dem Prozessmodell ein Indikatormodell darzustellen. Das Indikatormodell visualisiert die Berechnung einer Nachhaltigkeitsbewertung von Inventarindikatoren über Wirkungsindikatoren bis hin zu einer aggregierten Gesamtbewertung. Indikatormodell und Prozessmodell können dynamisch verfeinert werden und ermöglichen so detaillierte Analysen zu spezifischen Nachhaltigkeitsaspekten über verschiedene Prozessstufen hinweg. Mit der Bereitstellung eines solchen Dashboards kann für Anwendende der Zugang zu wichtigen Nachhaltigkeitsdaten erleichtert und die Entscheidungsfindung unterstützt werden.

Die bereitgestellte Modellierungsunterstützung kann um weitere Sprachkonzepte erweitert werden. Dazu zählt die Unterstützung der Modellierung von Unterprozessen, Ressourcen und Organisationsstrukturen. Unterprozesse können mit Petri-Netzen beispielsweise als Verfeinerungen von Transitionen dargestellt werden, bei denen eine Transition t durch eine Netzstruktur mit den Stellen ${}^\bullet t \cup t^\bullet$ als Schnittstelle ersetzt wird [Rei10]. Entsprechende Funktionen lassen sich auch für JSON-Netze spezifizieren und im JSON-Netz-Editor umsetzen. Auf diese Weise kann die Ausdrucksmächtigkeit für

diesen Aspekt der Prozessmodellierung mit JSON-Netzen erweitert werden. Mit der Bereitstellung von Sprachkonzepten zur Modellierung von Ressourcen und Organisationsstrukturen kann in Zukunft auch die Analyse von entsprechenden Fragestellungen (zum Beispiel: Welche Ressourcen oder Organisationseinheiten beeinflussen die Nachhaltigkeitsbewertung eines Geschäftsprozesses?) unterstützt werden.

Die Integration von Nachhaltigkeitsaspekten kann zu umfangreichen und komplexen Prozessmodellen führen kann. Über die Einführung eines Sichtenkonzepts [Fra03] können künftig (für bestimmte Fragestellungen) irrelevante Aspekte ausgeblendet oder bei Bedarf zusätzliche Aspekte eingeblendet werden. Die Umsetzung eines solchen Konzepts würde es Anwendenden ermöglichen, sich auf die wesentlichen Komponenten zu konzentrieren. Gleichzeitig bietet es notwendige Flexibilität, um bei Bedarf auch auf detailliertere Informationen zugreifen zu können.

Domänenspezifische Modellierungskonzepte erhöhen die Produktivität der Modellierung für Anwendende [Fra13]. In der in dieser Arbeit entwickelten Modellierungsmethode wird über ein konfigurierbares Modellierungswerkzeug die Bereitstellung domänenspezifischer Modellierungskonzepte (zum Beispiel für Treibhausgasanalysen in der Elektronikindustrie wie in Abschnitt 11.3 beschrieben) unterstützt. Zur Handhabung einer Vielzahl von unterschiedlichen domänenspezifischer Modellierungsumgebungen empfiehlt [Fra14] die Einführung eines Mehrebenmodellierungskonzepts. Mehrebenenmodellierungskonzepte ermöglichen die Instanziierung von domänenspezifischen Modellierungssprachen auf unterschiedlichen Abstraktionsebenen [Fra14]. Nachhaltigkeitsanalysen stellen ein vielversprechendes Anwendungsgebiet für die Weiterentwicklung von Mehrebenenmodellierungskonzepten dar [Nol19, Kac18]. Auch in technischer Hinsicht bietet die hier vorliegende Arbeit vielversprechende Anknüpfungspunkte für die Entwicklung eines Mehrebenenmodellierungskonzepts. So empfiehlt [Fra14], dass für die Umsetzung der Mehrebenenmodellierung die traditionelle Unterscheidung zwischen Typ und Instanz aufzulösen ist. Das prototypenbasierte Vererbungskonzept der Programmiersprache JavaScript (und damit das JSON-Datenmodell) kann die Umsetzung eines Mehrebenenmodellierungskonzepts technisch unterstützen, da hier Vererbungsbeziehungen zwischen Instanzen (Objekten) beschrieben werden können [Red13, ECM22, Neu09]. So können auf der Grundlage der hier vorgestellten Arbeit Konzepte entwickelt werden, um komplexe Modellierungsszenarien einfacher und intuitiver zu gestalten und die Wiederverwendbarkeit von Modellen zu fördern.

12.4.4 Analyseunterstützung mit Process Mining

Neben Ansätzen, mit denen die Fähigkeiten der hier vorgestellten Modellierungsmethode erweitert werden, können auf Grundlage der vorliegenden Arbeit auch weitere Prozesstechnologien für die Operationalisierung von Nachhaltigkeitsanalysen entwickelt werden. Beispielsweise können Process Mining-Ansätze dazu genutzt werden, um auf Basis von Event-Log-Daten Prozessmodelle zu generieren und diese um Nachhaltigkeitsdaten anzureichern [Gra23, Ort21]. Die in Kapitel 4 beschriebenen Nachhaltigkeitsmuster bieten eine Grundlage dafür, fortschrittliche Process Mining-Ansätze zu entwickeln, in denen Nachhaltigkeitsaspekte strukturiert und umfassend betrachtet werden. Vielversprechend erscheint dabei auch die Kombination mit Compliance Checking-Ansätzen, da sie eine umfassende Überwachung und Kontrolle der Nachhaltigkeitsauswirkungen von Geschäftsprozessen ermöglichen [Sch20]. Bei einer gegebenen Zielsetzung (zum Beispiel Grenzwerte für Treibhausgasemissionen) kann so die Prozessausführung in Echtzeit überwacht werden, um die Einhaltung von Nachhaltigkeitszielen und gesetzlichen Bestimmungen sicherzustellen. So können Unternehmen künftig dabei unterstützt werden, die Nachhaltigkeitsauswirkungen ihrer Geschäftsprozesse auch während der Ausführung zu verbessern und dabei auf sich ändernde regulatorische Anforderungen zu reagieren.

Die Einbindung von Nachhaltigkeitsdaten erfordert darüber hinaus die Erweiterung bestehender Process Mining-Algorithmen um eine Datenflussperspektive [Gra23]. Auch hierfür bieten JSON-Netze als datenorientierte Prozessmodellierungssprache Anknüpfungspunkte. Die Kombination von JSON-Netzen mit Process Mining-Ansätzen erlaubt es, sowohl Kontroll- als auch Datenflussinformationen in Event Logs detailliert zu erfassen. Im Vergleich zu bestehenden Ansätzen, die Datenflüsse im Process Mining mit „object-centric“ Petri-Netzen (einer Variante gefärbter Petri-Netze) darstellen [Aal20], bieten JSON-Netze flexiblere Möglichkeiten zur Modellierung komplexer Datenstrukturen. Die Entwicklung von JSON-Netz-basierten Process Mining Algorithmen kann damit in künftig eine präzisere Abbildung realer Datenstrukturen ermöglichen und so die Qualität der abgeleiteten Prozessmodelle verbessern.

12.5 Fazit

Unter Schlagworten wie Digital Twins [Par21], Internet of Things [Jan20] oder Smart Manufacturing [Lu20] wird die Verknüpfung der digitalen Welt mit realen Prozessen vorangetrieben. Diese fortschreitende Digitalisierung kann es in Zukunft ermöglichen, die Nachhaltigkeitsauswirkungen von Geschäftsprozessen unmittelbar bei ihrer Ausführung sichtbar zu machen. So könnte zum Beispiel eine Augmented Reality Brille in einer Smarten Fabrik nicht nur unterschiedliche mögliche Aktivitäten darstellen [Bel21], sondern auch die damit zusammenhängenden Nachhaltigkeitsauswirkungen unter Einbeziehung von aktuellen Kontextdaten visualisieren. Mit solchen Maßnahmen kann ein besseres Bewusstsein für Nachhaltigkeitsauswirkungen und Zusammenhänge geschaffen und es können (im Sinne der Nachhaltigkeit) bessere Entscheidungen ermöglicht werden. Eine Voraussetzung dafür ist, dass die Erhebung von Nachhaltigkeitsindikatoren auf einem umfassenden, nachvollziehbaren und transparenten Nachhaltigkeitskonzept beruht [Klö09, ISO06a].Darüber hinaus sollte beachtet werden, dass es nicht um die Verbesserung der Nachhaltigkeitsindikatoren selbst, sondern um die Verbesserung der tatsächlichen Nachhaltigkeitsauswirkungen geht [Kol22, Kle25]. Dazu ist es notwendig, dass sowohl die Erhebung von Nachhaltigkeitsindikatoren, als auch ihre Interpretation im Austausch mit beteiligten Anspruchsgruppen geschieht [Gro12]. Auch hierzu können Prozessmodelle einen Beitrag leisten, indem sie eine gemeinsame Kommunikationsgrundlage bieten [Bor24]. Die vorliegende Arbeit bietet damit vielfältige Anknüpfungspunkte für weiterführende Entwicklungen, die Unternehmen dabei unterstützen, die Nachhaltigkeitsauswirkungen ihrer Geschäftsprozesse zu analysieren und zu verbessern.

Literatur

[Aal03] Aalst, Wil M. P. van der; Hofstede, Arthur H. M. ter; Kiepuszewski, Bartek und Barros, Ana P.: „Workflow Patterns“. In: *Distributed and Parallel Databases* 14.1 (2003), S. 5–51.

[Aal15] Aalst, Wil M. P. van der: „Business process management as the “Killer App” for Petri nets“. In: *Software & Systems Modeling* 14.2 (2015), S. 685–691.

[Aal20] Aalst, Wil M. P. van der und Berti, Alessandro: „Discovering Object-Centric Petri Nets“. In: *Fundamenta Informaticae* 175.1 (2020), S. 1–40.

[Aal97] Aalst, Wil M. P. van der: „Verification of workflow nets“. In: *Application and Theory of Petri Nets*. Toulouse, Frankreich, 1997.

[Abi97] Abiteboul, Serge: „Querying Semi-Structured Data“. In: *International Conference on Database Theory*. Delphi, Griechenland, 1997.

[Alp15] Alpers, Sascha; Becker, Christoph; Oberweis, Andreas und Schuster, Thomas: „Microservice Based Tool Support for Business Process Modelling“. In: *International Enterprise Distributed Object Computing Workshop*. Adelaide, Australien, 2015.

[Amr14] Amran, Azlan und Keat Ooi, Say: „Sustainability reporting: meeting stakeholder demands“. In: *Strategic Direction* 30.7 (2014), S. 38–41.

[And24] Andree, Kerstin; Hoang, Mai; Dannenberg, Felix; Weber, Ingo und Pufahl, Luise: „Discovery of Workflow Patterns - A Comparison of Process Discovery Algorithms“. In: *Cooperative Information Systems*. Groningen, Niederlande, 2024.

[Apa24] Apache: Apache ECharts. 2024. url: https://echarts.apache.org (besucht am 07. 07. 2024).

[Ard08] Ardagna, Danilo; Cappiello, Cinzia; Lovera, Marco; Pernici, Barbara und Tanelli, Mara: „Active energy-aware management of business-process based applications“. In: *European Conference on a Service-Based Internet*. Madrid, Spanien, 2008.

[Bac19] Bach, Rebecca; Mohtashami, Negar und Hildebrand, Linda: „Comparative Overview on LCA Software Programs for Application in the Façade Design Process“. In: *Journal of Facade Design and Engineering* (2019), S. 13–26.

[Bad15] Badouel, Eric; Hélouët, Loïc und Morvan, Christophe: „Petri nets with semi-structured data“. In: *International Conference on Application and Theory of Petri Nets and Concurrency*. Brüssel, Belgien, 2015.

[Bal11] Balzert, Silke; Kleinert, Thomas; Fettke, Peter und Loos, Peter: Vorgehensmodelle im Geschäftsprozessmanagement. Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI): Veröffentlichungen des Instituts für Wirtschaftsinformatik, 2011.

[Ban08] Bangor, Aaron; Kortum, Philip T. und Miller, James T.: „An Empirical Evaluation of the System Usability Scale“. In: *International Journal of Human-Computer Interaction* 24.6 (2008), S. 574–594.

[Ban15] Bandara, Wasana; Furtmueller, Elfi; Gorbacheva, Elena; Miskon, Suraya und Beekhuyzen, Jenine: „Achieving rigor in literature reviews: Insights from qualitative data analysis and tool-support“. In: *Communications of the Association for Information Systems* 37.1 (2015), S. 154–204.

[Bel21] Bellalouna, Fahmi: „The Augmented Reality Technology as Enabler for the Digitization of Industrial Business Processes: Case Studies“. In: *Procedia CIRP* 98 (2021), S. 400–405.

[Bet14] Betz, Stefanie: „Sustainability aware process management using XML-Nets“. In: *International Conference on Informatics for Environmental Protection*. Oldenburg, Deutschland, 2014.

[Bjø18] Bjørn, Anders; Owsianiak, Mikołaj; Molin, Christine und Hauschild, Michael Zwicky: „LCA History“. In: *Life Cycle Assessment*. Hrsg. von Hauschild, Michael Zwicky; Rosenbaum, Ralph K. und Olsen, Stig Irving. Springer, 2018, S. 17–30.

[Blo03] Blonk, Heico van der: „Writing Case Studies in Information Systems Research“. In: *Journal of Information Technology* 18.1 (2003), S. 45–52.

[Boa24] Boavizta: Datavizta. 2024. url: https://dataviz.boavizta.org/ (besucht am 07. 07. 2024).

[Boe14] Boell, Sebastian K und Cecez-Kecmanovic, Dubravka: „A hermeneutic approach for conducting literature reviews and literature searches“. In: *Communications of the Association for Information Systems* 34.1 (2014), S. 257–286.

[Bor24] Bork, Dominik; David, Istvan; España, Sergio; Guizzardi, Giancarlo; Proper, Henderik A und Reinhartz-Berger, Iris: „The Role of Modeling in the Analysis and Design of Sustainable Systems: A Panel Report“. In: *Communications of the Association for Information Systems* 54 (2024), S. 911–936.

[Bou17] Bourhis, Pierre; Reutter, Juan L.; Suárez, Fernando und Vrgoč, Domagoj: „JSON: Data model, query languages and schema specification“. In: *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. Chicago, IL, USA, 2017.

[Bou20] Bourhis, Pierre; Reutter, Juan L. und Vrgoč, Domagoj: „JSON: Data model and query languages“. In: *Information Systems* 89 (2020).

[Bro09] Brocke, Jan vom; Simons, Alexander; Niehaves, Björn; Riemer, Kai; Plattfaut, Ralf und Cleven, Anne: „Reconstructing the giant: On the importance of rigour in documenting the literature search process“. In: *European Conference on Information Systems*. Verona, Italien, 2009.

[Bro95] Brooke, John: „SUS: A quick and dirty usability scale“. In: *Usability Evaluation in Industry* 189 (1995).

[Büd20] Büdel, Vera; Fritsch, Andreas und Oberweis, Andreas: „Integrating sustainability into day-to-day business: a tactical management dashboard for O-LCA“. In: *International Conference on ICT for Sustainability*. Bristol, UK, 2020.

[Bun97] Buneman, Peter: „Semistructured data“. In: *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. Tucson, AZ, USA, 1997.

[Bur24] Burgmer, Christoph: JSONPath Comparison. 2024. url: https://cburgmer.github.io/json-path-comparison/ (besucht am 07. 07. 2024).

[Cam24] Camunda: Camunda Modeler. 2024. url: https://camunda.com/try-modeler/ (besucht am 07. 07. 2024).

[Cap11a] Cappiello, Cinzia; Fugini, Maria Grazia; Ferreira, Alexandre Mello; Plebani, Pierluigi und Vitali, Monica: „Business process co-design for energy-aware adaptation“. In: *International Conference on Intelligent Computer Communication and Processing*. Cluj-Napoca, Rumänien, 2011.

[Cap11b] Cappiello, Cinzia; Fugini, Mariagrazia; Pernici, Barbara und Plebani, Pierluigi: „Green information systems for sustainable IT“. In: *Information Technology and Innovation Trends in Organizations*. Hrsg. von D'Atri, Alessandro; Ferrara, Maria; George, Joey F. und Spagnoletti, Paolo. Springer, 2011, S. 153–160.

[Cap13] Cappiello, Cinzia; Plebani, Pierluigi und Vitali, Monica: „Energy-aware process design optimization“. In: *International Conference on Cloud and Green Computing*. Karlsruhe, Deutschland, 2013.

[Cap14] Cappiello, Cinzia; Melià, Paco; Pernici, Barbara; Plebani, Pierluigi und Vitali, Monica: „Sustainable choices for cloud applications: A focus on CO2 emissions“. In: *International Conference on ICT for Sustainability*. Stockholm, Schweden, 2014.

[Che24] Chester, David: dchester/jsonpath. 2024. url: https://github.com/dchester/jsonpath (besucht am 07. 07. 2024).

[Cia94] Ciardo, Gianfranco: „Petri nets with marking-dependent arc cardinality: Properties and analysis“. In: *Application and Theory of Petri Nets*. Zaragoza, Spanien, 1994.

[Cir12] Ciroth, Andreas: „Software for Life Cycle Assessment“. In: *Life Cycle Assessment Handbook: A Guide for Environmentally Sustainable Products*. Hrsg. von Curan, Mary Ann. Wiley, 2012, S. 143–158.

[cli24] client.IO: jointJS. 2024. url: https://www.jointjs.com/ (besucht am 07. 07. 2024).

[Col20] Colley, Derek; Stanier, Clare und Asaduzzaman, Md: „Investigating the Effects of Object-Relational Impedance Mismatch on the Efficiency of Object-Relational Mapping Frameworks:“ In: *Journal of Database Management* 31.4 (2020), S. 1–23.

[Con07] Connelly, Steve: „Mapping Sustainable Development as a Contested Concept“. In: *Local Environment* 12.3 (2007), S. 259–278.

[Cou19a] Couckuyt, Dries und Van Looy, Amy: „A systematic review of green business process management“. In: *Business Process Management Journal* 26.2 (2019), S. 421–446.

[Cou19b] Couckuyt, Dries und Van Looy, Amy: „Green BPM as a business-oriented discipline: A systematic mapping study and research agenda“. In: *Sustainability* 11.15 (2019).

[Dav93] Davenport, Thomas H: Process innovation – reengineering work through information technology. Harvard Business School Press, 1993.

[Del11] Delmas, Magali A und Burbano, Vanessa Cuerel: „The drivers of greenwashing“. In: *California Management Review* 54.1 (2011), S. 64–87.

[Deu19] Deutschlandfunk: Kehrseite der Energiewende: Lithium-Abbau in Südamerika. 2019. url: https://www.deutschlandfunk.de/lithium-abbau-in-suedamerika-kehrseite-der-energiewende-100.html (besucht am 05. 12. 2023).

[Dro22] Droettboom, Michael: Understanding JSON Schema. Space Telescope Science Institute, 2022.

[Dud24] Dudenredaktion: Nachhaltigkeit auf Duden online. 2024. url: https://www.duden.de/node/100643/revision/100679 (besucht am 04. 01. 2024).

[Dum18] Dumas, Marlon; La Rosa, Marcello; Mendling, Jan und Reijers, Hajo A.: Fundamentals of business process management. 2. Aufl. Springer, 2018.

[Duq21] Duque-Grisales, Eduardo und Aguilera-Caracuel, Javier: „Environmental, Social and Governance (ESG) Scores and Financial Performance of Multilatinas: Moderating Effects of Geographic International Diversification and Financial Slack“. In: *Journal of Business Ethics* 168.2 (2021), S. 315–334.

[Ecl24] EclipseSource: JSONForms. 2024. url: https://jsonforms.io/ (besucht am 07. 07. 2024).

[ECM17] ECMA: The JSON Data Interchange Syntax. 2017.

[ECM22] ECMA: ECMAScript 2022 Language Specification. 2022.

[Elk18] Elkington, John: „25 years ago I coined the phrase “Triple bottom line.” here’s why it’s time to rethink it.“ In: *Harvard Business Review* (2018).

[Elk94] Elkington, John: „Towards the sustainable corporation: Win-win-win business strategies for sustainable development“. In: *California Management Review* 36.2 (1994), S. 90–100.

[Elk97] Elkington, John: Cannibals with forks: the triple bottom line of 21st century business. Capstone Publishing Limited, 1997.

[Ern22] Ernst, Robin-Alexander; Gerken, Maike; Hack, Andreas und Hülsbeck, Marcel: „SMES' reluctance to embrace corporate sustainability: The effect of stakeholder pressure on self-determination and the role of social proximity". In: *Journal of Cleaner Production* 335 (2022).

[Fai24] FairLötet e.V.: Fairtronics. 2024. url: new.fairtronics.org (besucht am 07. 07. 2024).

[Fie00] Fielding, Roy Thomas: „Architectural Styles and the Design of Network-based Software Architectures". Diss. Irvine, CA, USA: University of California, 2000.

[Fin14a] Finkbeiner, Matthias: „The international standards as the constitution of life cycle assessment: The ISO 14040 series and its offspring". In: *Background and Future Prospects in Life Cycle Assessment*. Hrsg. von Klöpffer, Walter. Springer, 2014, S. 85–106.

[Fin14b] Finkbeiner, Matthias u. a.: „Challenges in Life Cycle Assessment: An Overview of Current Gaps and Research Needs". In: *Background and Future Prospects in Life Cycle Assessment*. Hrsg. von Klöpffer, Walter. Springer, 2014, S. 207–258.

[Fis98] Fischer, Thomas; Biskup, Hubert und Müller-Luschnat, Günther: „Begriffliche Grundlagen für Vorgehensmodelle". In: *Vorgehensmodelle für die betriebliche Anwendungsentwicklung*. Hrsg. von Kneuper, Ralf; Müller-Luschnat, Günther und Oberweis, Andreas. Vieweg+Teubner, 1998, S. 13–31.

[For24] Forell, Martin und Schüler, Selina: „Modeling meets Large Language Models". In: *Modellierung 2024 Satellite Events*. Potsdam, Deutschland, 2024.

[Fra03] Frank, Ulrich und Laak, Bodo L. van: Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen. 34. Universität Koblenz-Landau: Arbeitsbericht des Instituts für Wirtschaftsinformatik, 2003.

[Fra10] Frank, Ulrich: Outline of a method for designing domain-specific modelling languages. 42. ICB-Research Report, 2010.

[Fra13] Frank, Ulrich: „Domain-specific modeling languages: Requirements analysis and design guidelines". In: *Domain Engineering: Product Lines, Languages, and Conceptual Models*. Hrsg. von Reinhartz-Berger, Iris; Sturm, Arnon; Clark, Tony; Cohen, Sholom und Bettin, Jorn. Springer, 2013, S. 133–157.

[Fra14] Frank, Ulrich: „Multilevel Modeling: Toward a New Paradigm of Conceptual Modeling and Information Systems Design“. In: *Business & Information Systems Engineering* 6.6 (2014), S. 319–337.

[Fri07] Frischknecht, Rolf; Althaus, Hans-Jörg; Bauer, Christian; Doka, Gabor; Heck, Thomas; Jungbluth, Niels; Kellenberger, Daniel und Nemecek, Thomas: „The Environmental Relevance of Capital Goods in Life Cycle Assessments of Products and Services“. In: *International Journal of Life Cycle Assessment* (2007).

[Fri20a] Frischknecht, Rolf: Lehrbuch der Ökobilanzierung. Springer, 2020.

[Fri20b] Fritsch, Andreas; Beschke, Sebastian; Drucks, Tamara; Jekutsch, Sebastian und Lutzweiler, Samuel: „Fairtronics: a Social Hotspot Analysis Tool for Electronics Products“. In: *Electronics Goes Green*. Berlin, Deutschland, 2020.

[Fri22] Fritsch, Andreas; Hammerstein, Johanna von; Schreiber, Clemens; Betz, Stefanie und Oberweis, Andreas: „Pathways to Greener Pastures: Research Opportunities to Integrate Life Cycle Assessment and Sustainable Business Process Management Based on a Systematic Tertiary Literature Review“. In: *Sustainability* 14.18 (2022).

[Fri23a] Fritsch, Andreas: „Mit prozessorientierten Informationssystemen zu nachhaltigen Geschäftsmodellen“. In: *Red Stack Business News* 6 (2023), S. 54–61.

[Fri23b] Fritsch, Andreas; Schüler, Selina; Forell, Martin und Oberweis, Andreas: „Modelling and Execution of Data-Driven Processes with JSON-Nets“. In: *Business Process Modeling, Development, and Support*. Zaragoza, Spanien, 2023.

[Gar04] Garriga, Elisabet und Melé, Domènec: „Corporate social responsibility theories: Mapping the territory“. In: *Journal of Business Ethics* 53.1 (2004), S. 51–71.

[Geh19] Gehlot, Vijay: „From Petri NETS to Colored Petri NETS: A Tutorial Introduction to NETS Based Formalism For Modeling And Simulation“. In: *Winter Simulation Conference*. National Harbor, MD, USA, 2019.

[Gen14] Genero Bocco, Marcela; Cruz-Lemus, José A. und Piattini Velthuis, Mario G.: Métodos de investigación en ingeniería del software. RA-MA, 2014.

[Gen81] Genrich, Hartmann J. und Lautenbach, Kurt: „System modelling with high-level Petri nets“. In: *Theoretical Computer Science* 13.1 (1981), S. 109–135.

[Gho10] Ghose, Aditya; Hoesch-Klohe, Konstantin; Hinsche, Lothar und Le, Lam Son: „Green business process management: A research agenda“. In: *Australasian Journal of Information Systems* 16.2 (2010), S. 103–117.

[Git24] GitHub, Inc: GitHub. 2024. url: https://github.com/ (besucht am 07. 07. 2024).

[Glo15] Global Witness: Conflict Minerals in Eastern Congo. 2015. url: https://www.globalwitness.org/en/campaigns/conflict-minerals/conflict-minerals-eastern-congo/ (besucht am 07. 07. 2024).

[Goh15] Gohar, Shahrzad Roohy und Indulska, Marta: „Business process management: Saving the planet?“ In: *Australasian Conference on Information Systems*. Adelaide, Australien, 2015.

[Goh20] Gohar, Shahrzad Roohy und Indulska, Marta: „Environmental sustainability through green business process management“. In: *Australasian Journal of Information Systems* 24 (2020), S. 1–30.

[Goo24a] Google: Google Docs. 2024. url: https://www.google.de/intl/de/docs/about/ (besucht am 07. 07. 2024).

[Goo24b] Google: Google Trends. 2024. url: https://trends.google.com/trends/explore?date=all&geo=US&q=json,xml (besucht am 07. 07. 2024).

[Goo24c] Google: Jsonnet. 2024. url: https://jsonnet.org/ (besucht am 15. 01. 2023).

[Gör03] Göransson, Bengt; Gulliksen, Jan und Boivie, Inger: „The usability design process – integrating user-centered systems design in the software development process“. In: *Software Process: Improvement and Practice* 8.2 (2003), S. 111–131.

[Gös07] Gössner, Stefan: JSONPath - XPath for JSON. 2007. url: https://goessner.net/articles/JsonPath/ (besucht am 08. 01. 2023).

[Grä13] Gräuler, Matthias und Teuteberg, Frank: „Experimental evaluation of a process benchmarking tool in a green business process management context“. In: *International Conference on Business Informatics*. Leipzig, Deutschland, 2013.

[Gra23] Graves, Nina; Koren, István und Aalst, Wil M.P. van der: „ReThink Your Processes! A Review of Process Mining for Sustainability“. In: *International Conference on ICT for Sustainability*. Rennes, Frankreich, 2023.

[Gre24a] GreenDelta: openLCA. 2024. url: https://www.openlca.org/ (besucht am 07. 07. 2024).

[Gre24b] GreenDelta: Psilca. 2024. url: https : / / psilca . net/ (besucht am 07. 07. 2024).

[Gro12] Groen, Bianca A. C.; Wouters, Marc J. F. und Wilderom, Celeste P. M.: „Why do employees take more initiatives to improve their performance after co-developing performance measures? A field study“. In: *Management Accounting Research* 23.2 (2012), S. 120–141.

[Gui11] Guinée, Jeroen B.; Heijungs, Reinout; Huppes, Gjalt; Zamagni, Alessandra; Masoni, Paolo; Buonamici, Roberto; Ekvall, Tomas und Rydberg, Tomas: „Life Cycle Assessment: Past, Present, and Future“. In: *Environmental Science & Technology* 45.1 (2011), S. 90–96.

[Hal19] Haldar, Stuti: „Towards a conceptual understanding of sustainability-driven entrepreneurship“. In: *Corporate Social Responsibility and Environmental Management* 26.6 (2019), S. 1157–1170.

[Ham94] Hammer, Michael und Champy, James: Reengineering the corporation: a manifesto for business revolution. Brealey, 1994.

[Har15] Harmon, Paul: „The scope and evolution of business process management“. In: *Handbook on Business Process Management 1*. Hrsg. von Brocke, Jan vom und Rosemann, Michael. 2. Aufl. Springer, 2015, S. 37–80.

[Häu95] Häuslein, Andreas und Hedemann, Jan: „Die Bilanzierungssoftware Umberto und mögliche Einsatzgebiete“. In: *Stoffstromanalysen in Ökobilanzen und Öko-Audits*. Hrsg. von Schmidt, Mario und Schorb, Achim. Springer, 1995, S. 59–78.

[Häu97] Häuslein, Andreas und Hedemann, Jan: „Die Grundfunktionen von Umberto“. In: *Ökobilanzierung mit Computerunterstützung*. Hrsg. von Schmidt, Mario und Häuslein, Andreas. Springer, 1997, S. 37–50.

[Hav24] Haverbeke, Marijn und Heine, Adrian: CodeMirror. 2024. url: https: //codemirror.net/ (besucht am 07. 07. 2024).

[Heh24] Hehnle, Philipp; Behrendt, Maximilian; Weinbrecht, Luc und Correa, Carl: „Carbon-Aware Process Execution for Green Business Process Management“. In: *International Conference on Enterprise Information Systems*. Angers, Frankreich, 2024.

[Hei00] Heiskanen, Eva: „Managers' interpretations of LCA: Enlightenment and responsibility or confusion and denial?“ In: *Business Strategy and the Environment* 9.4 (2000), S. 239–254.

[Hei02a] Heijungs, R und Suh, S: The computational structure of life cycle assessment. Eco-efficiency in industry and science. Springer, 2002.

[Hei02b] Heiskanen, Eva: „The institutional logic of life cycle thinking“. In: *Journal of Cleaner Production* 10.5 (2002), S. 427–437.

[Her19] Hernández González, Anaisa; Calero, Coral; Pérez Parra, Dianelys und Mancebo, Javier: „Approaching green BPM characterisation“. In: *Journal of Software: Evolution and Process* 31.2 (2019), S. 1–26.

[Hew20] Hewasinghage, Moditha; Nadal, Sergi und Abelló, Alberto: „On the Performance Impact of Using JSON, Beyond Impedance Mismatch“. In: *New Trends in Databases and Information Systems*. Lyon, Frankreich, 2020.

[Hil15] Hilty, Lorenz Manuel und Aebischer, Bernard: „ICT for Sustainability: An Emerging Research Field“. In: *ICT Innovations for Sustainability*. Hrsg. von Hilty, Lorenz Manuel und Aebischer, Bernard. Springer, 2015, S. 3–36.

[Hir23] Hirzel, Martin: „Low-Code Programming Models“. In: *Communications of the ACM* 66.10 (2023), S. 76–85.

[Hoe10a] Hoesch-Klohe, Konstantin und Ghose, Aditya: „Business process improvement in Abnoba“. In: *International Conference on Service-Oriented Computing Workshops*. San Francisco, CA, USA, 2010.

[Hoe10b] Hoesch-Klohe, Konstantin und Ghose, Aditya: „Carbon-aware business process design in abnoba“. In: *International Conference on Service-Oriented Computing*. San Francisco, CA, USA, 2010.

[Hoe10c] Hoesch-Klohe, Konstantin; Ghose, Aditya und Lê, Lam Son: „Towards green business process management“. In: *International Conference on Services Computing*. Miami, FL, USA, 2010.

[Hoe12a] Hoesch-Klohe, Konstantin und Ghose, Aditya: „Environmentally aware business process improvement in the enterprise context“. In: *Harnessing Green IT: Principles and Practices*. Hrsg. von Murugesan, San und Gangadharan, G. R. Wiley, 2012, S. 265–282.

[Hoe12b] Hoesch-Klohe, Konstantin und Ghose, Aditya: „Making use of scenarios for environmentally aware system design“. In: *International Workshop on Requirements Engineering for Sustainable Systems*. Essen, Deutschland, 2012.

[Hol01] Holliday, C.: „Sustainable growth, the DuPont way“. In: *Harvard Business Review* 79.8 (2001), S. 129–134.

[Hor24] Horus software GmbH: Horus Business Modeler. 2024. url: https://www.horus.biz/de/produkte/business-modeler/ (besucht am 07. 07. 2024).

[Hou11] Houy, Constantin; Reiter, Markus; Fettke, Peter und Loos, Peter: „Towards green BPM - sustainability and resource efficiency through business process management“. In: *Business Process Management Workshops*. Hoboken, NJ, USA, 2011.

[Hou12] Houy, Constantin; Reiter, Markus; Fettke, Peter; Loos, Peter; Hoesch-Klohe, Konstantin und Ghose, Aditya: „Advancing business process technology for humanity: Opportunities and challenges of green BPM for sustainable business activities“. In: *Green Business Process Management: Towards the Sustainable Enterprise*. Hrsg. von Brocke, Jan vom; Seidel, Stefan und Recker, Jan. Springer, 2012, S. 75–92.

[Hua15] Huang, Kuo-Feng; Dyerson, Romano; Wu, Lei-Yu und Harindranath, G.: „From Temporary Competitive Advantage to Sustainable Competitive Advantage“. In: *British Journal of Management* 26.4 (2015), S. 617–636.

[IET13] IETF: JSONPointer. 2013.

[IET17] IETF: The JavaScript object notation (JSON) data interchange format. 2017.

[IET24] IETF: JSONPath RFC9535. 2024.

[ILO19] ILO: Child labour in mining, poor working conditions take centre stage in inter-regional meeting. 2019. url: https://www.ilo.org/resource/news/child-labour-mining-poor-working-conditions-take-centre-stage-inter (besucht am 07. 07. 2024).

[ILO24] ILO: ILOSTAT. 2024. URL: https : / / ilostat . ilo . org/ (besucht am 07. 07. 2024).

[iPo24] iPoint: Umberto. 2024. URL: https://www.ifu.com/de/umberto/ (besucht am 07. 07. 2024).

[Ire09] Ireland, Christopher; Bowers, David; Newton, Michael und Waugh, Kevin: „A Classification of Object-Relational Impedance Mismatch“. In: *International Confernce on Advances in Databases, Knowledge, and Data Applications*. Gosier, Guadeloupe, Frankreich, 2009.

[ISO06a] ISO: ISO 14040:2006 Umweltmanagement - Ökobilanz - Grundsätze und Rahmenbedingungen. 2006.

[ISO06b] ISO: ISO 14044:2006 Umweltmanagement - Ökobilanz - Anforderungen und Anleitungen. 2006.

[ISO14] ISO: ISO/TS 14072:2014 Environmental management - Life cycle assessment - requirements and guidelines for organizational life cycle assessment. 2014.

[Jan20] Janiesch, Christian u. a.: „The Internet-of-Things Meets Business Process Management: A Manifesto“. In: *IEEE Systems, Man, and Cybernetics Magazine* 6.4 (2020), S. 34–44.

[Jen81] Jensen, Kurt: „Coloured petri nets and the invariant-method“. In: *Theoretical Computer Science* 14.3 (1981), S. 317–336.

[Jen90] Jensen, Kurt: „Coloured Petri Nets: A High Level Language for System Design and Analysis“. In: *Advances in Petri Nets*. Bonn, Deutschland, 1990.

[Jen93] Jensen, Kurt: „An Introduction to the Theoretical Aspects of Coloured Petri Nets“. In: *A Decade of Concurrency: Reflections and Perspectives*. Noordwijkerhout, Niederlande, 1993.

[Jen94] Jensen, Kurt: Coloured petri nets: Basic concepts, analysis methods and practical use. Springer, 1994.

[Jer24] Jeremy, Thomas: Bulma. 2024. URL: https : / / bulma . io/ (besucht am 07. 07. 2024).

[JSO23] JSON Type Definition Contributors: JSON Type Definition. 2023. URL: https://jsontypedef.com/ (besucht am 07. 07. 2024).

[Kac18] Kaczmarek-Heß, M.; Nolte, M.; Fritsch, Andreas und Betz, S.: „Practical experiences with multi-level modeling using FMMLx : A hierarchy of domain-specific modeling languages in support of life-cycle assessment“. In: *MODELS 2018 Workshops*. Kopenhagen, Dänemark, 2018.

[Kan08] Kannengiesser, Udo: „Subsuming the BPM life cycle in an ontological framework of designing“. In: *International Workshop CIAO! and International Workshop EOMAS*. Montpellier, Frankreich, 2008.

[Kap96] Kaplan, Robert S. und Norton, David P.: The balanced scorecard: translating strategy into action. Harvard Business Press, 1996.

[Kar02] Karagiannis, Dimitris und Kühn, Harald: „Metamodelling platforms“. In: *International Conference EC-Web*. Aix-en-Provence, Frankreich, 2002.

[Kar15] Karagiannis, Dimitris: „Agile modeling method engineering“. In: *Panhellenic Conference on Informatics*. Athen, Griechenland, 2015.

[Kar16] Karagiannis, Dimitris: „Conceptual modelling methods: The AMME agile engineering approach“. In: *International Conference on Informatics in Economy*. Cluj-Napoca, Rumänien. 2016.

[Kas14] Kasah, Tarek: „LCA of a newsprint paper machine: a case study of capital equipment“. In: *International Journal of Life Cycle Assessment* 19.2 (2014), S. 417–428.

[Kit07] Kitchenham, Barbara und Charters, Stuart: Guidelines for performing systematic literature reviews in software engineering. University of Durham: EBSE Technical Report, 2007.

[Kit10] Kitchenham, Barbara A.; Budgen, David und Brereton, O. Pearl: „The value of mapping studies – A participant-observer case study“. In: *International Conference on Evaluation and Assessment in Software Engineering*. London, UK, 2010.

[Kle25] Klessascheck, Finn; Weber, Ingo und Pufahl, Luise: „SOPA: a framework for sustainability-oriented process analysis and re-design in business process management“. In: *Information Systems and e-Business Management* (2025).

[Klö09] Klöpffer, Walter und Grahl, Birgit: Ökobilanz (LCA): Ein Leitfaden für Ausbildung und Beruf. Wiley-VCH, 2009.

[Klö14] Klöpffer, Walter und Grahl, Birgit: Life cycle assessment (LCA) a guide to best practice. Wiley-VCH, 2014.

[Kol22] Kolk, Berend van der: „Numbers Speak for Themselves, or Do They? On Performance Measurement and Its Implications“. In: *Business & Society* 61.4 (2022), S. 813–817.

[Kos18] Koschmider, Agnes; Oberweis, Andreas und Stucky, Wolffried: „A petri net-based view on the business process life-cycle“. In: *Enterprise Modelling and Information Systems Architectures (EMISAJ)* 13 (2018), S. 47–55.

[Lag23] Lagemann, Hanna: Conception and Development of a Modeling Editor for JSON Nets. Masterarbeit am KIT, Betreuer: Andreas Fritsch, Prüfer: Andreas Oberweis, 2023.

[Lél91] Lélé, Sharachchandra M.: „Sustainable Development: A Critical Review“. In: *World Development* 19.6 (1991), S. 607–621.

[Len01] Lenz, Kirsten und Oberweis, Andreas: „Modeling interorganizational workflows with XML nets“. In: *Hawaii International Conference on System Sciences*. Maui, HI, USA, 2001.

[Len03] Lenz, Kirsten und Oberweis, Andreas: „Inter-organizational business process management with XML nets“. In: *Petri Net Technology for Communication-Based Systems*. Hrsg. von Ehrig, Hartmut; Reisig, Wolfgang; Rozenberg, Grzegorz und Weber, Herbert. Springer, 2003, S. 243–263.

[Lev06] Levy, Yair und J. Ellis, Timothy: „A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research“. In: *Informing Science* 9 (2006), S. 181–212.

[Löf11] Löfgren, Birger und Tillman, Anne-Marie: „Relating manufacturing system configuration to life-cycle environmental performance: discrete-event simulation supplemented with LCA“. In: *Journal of Cleaner Production* 19.17 (2011), S. 2015–2024.

[Lu20] Lu, Yuqian; Xu, Xun und Wang, Lihui: „Smart manufacturing process and system automation – A critical review of the standards and envisioned scenarios“. In: *Journal of Manufacturing Systems* 56 (2020), S. 312–325.

[Lub15] Lubbecke, Patrick; Reiter, Markus; Fettke, Peter und Loos, Peter: „Simulation-based decision support for the reduction of the energy consumption of complex business processes“. In: *Hawaii International Conference on System Sciences*. Kauai, HI, USA, 2015.

[Lub16] Lubbecke, Patrick; Fettke, Peter und Loos, Peter: „Towards ecological workflow patterns as an instrument to optimize business processes with respect to ecological goals“. In: *Hawaii International Conference on System Sciences*. Koloa, HI, USA, 2016.

[Lüb16] Lübbecke, Patrick; Fettke, Peter und Loos, Peter: „Sustainability patterns for the improvement of IT-Related business processes with regard to ecological goals“. In: *Business Process Management Workshops*. Rio de Janeiro, Brasilien, 2016.

[Mac17] Maciel, João Carlos: „The core capabilities of green business process management - a literature review“. In: *International Conference on Business Informatics*. St. Gallen, Schweiz, 2017.

[Mai24] Maibaum, Frederik; Kriebel, Johannes und Foege, Johann Nils: „Selecting textual analysis tools to classify sustainability information in corporate reporting“. In: *Decision Support Systems* (2024).

[Man95] Mankins, John C: Technology readiness levels. NASA, 1995, S. 1995.

[Mar15] Martínez-Blanco, Julia; Lehmann, Annekatrin; Chang, Ya-Ju und Finkbeiner, Matthias: „Social organizational LCA (SOLCA) - a new approach for implementing social LCA“. In: *International Journal of Life Cycle Assessment* 20.11 (2015), S. 1586–1599.

[Mar20] Martínez-Blanco, Julia; Forin, Silvia und Finkbeiner, Matthias: „Challenges of organizational LCA: lessons learned from road testing the guidance on organizational life cycle assessment“. In: *International Journal of Life Cycle Assessment* 25.2 (2020), S. 311–331.

[Mei19] Meier, Andreas und Kaufmann, Michael: „NoSQL Databases“. In: *SQL & NoSQL Databases*. Springer, 2019, S. 201–218.

[Mel10] Melville, Nigel P.: „Information Systems Innovation for Environmental Sustainability“. In: *MIS Quarterly* 34.1 (2010), S. 1–21.

[Mey11] Meyer, Andreas; Smirnov, Sergey und Weske, Mathias: Data in business processes. Universität Potsdam, 2011.

[Mic24] Microsoft: Visual Studio Code. Version 1.91. 2024.

[Mir18] Miralles-Quirós, María; Miralles-Quirós, José und Valente Gonçalves, Luis: „The Value Relevance of Environmental, Social, and Governance Performance: The Brazilian Case“. In: *Sustainability* 10.3 (2018).

[Möl95] Möller, Andreas und Rolf, Arno: „Methodische Ansätze zur Erstellung von Stoffstromanalysen unter besonderer Berücksichtigung von Petri-Netzen“. In: *Stoffstromanalysen in Ökobilanzen und Öko-Audits*. Hrsg. von Schmidt, Mario und Schorb, Achim. Springer, 1995, S. 33–58.

[Möl97] Möller, Andreas: „Berechnungsverfahren unter Umberto“. In: *Ökobilanzierung mit Computerunterstützung*. Hrsg. von Schmidt, Mario und Häuslein, Andreas. Springer, 1997, S. 115–130.

[Mon24] MongoDB Inc: MongoDB. 2024. url: https://www.mongodb.com/docs/manual/ (besucht am 07. 07. 2024).

[Moo09] Moody, D.: „The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering“. In: *IEEE Transactions on Software Engineering* 35.6 (2009), S. 756–779.

[Mor02] Morris, Julian: „What is sustainable development and how can we achieve it ?“ In: *Sustainable development: Promoting progress or perpetuating poverty?* Hrsg. von Morris, Julian. Profile Books, 2002.

[Mor24] Morote, Eduardo San Martin: Pinia. 2024. url: https://pinia.vuejs.org/ (besucht am 07. 07. 2024).

[Mue05] Muehlen, Michael zur und Ho, Danny Ting-Yi: „Risk management in the BPM lifecycle“. In: *Business Process Management Workshops*. Nancy, Frankreich, 2005.

[Mül97] Müller-Beilschmidt, Peter: „Software zur Unterstützung der Ökobilanzierung – ein Überblick“. In: *Ökobilanzierung mit Computerunterstützung*. Hrsg. von Schmidt, Mario und Häuslein, Andreas. Springer, 1997, S. 3–10.

[Mye86] Myers, Brad A.: „Visual programming, programming by example, and program visualization: a taxonomy“. In: Computer Human Interaction Conference. Boston, MA, USA, 1986.

[Mye96] Myerson, George und Rydin, Yvonne: The language of environment: A new rhetoric. UCL Press, 1996.

[n8n24] n8n GmbH: n8n. Version 1.24. 2024.

[Net06] Netjes, Mariska; Reijers, Hajo A. und Aalst, Wil M.P. van der: „Supporting the BPM life-cycle with FileNet“. In: *EMMSAD Workshop*. Luxemburg, Luxemburg, 2006.

[Neu09] Neumayr, Bernd; Grün, Katharina und Schrefl, Michael: „Multi-level domain modeling with m-objects and m-relationships“. In: *Asia-Pacific Conference on Conceptual Modelling*. Wellington, Neuseeland, 2009.

[Nol19] Nolte, Mario; Kaczmarek-Heß, Monika; Fritsch, Andreas und Betz, Stefanie: „A Hierarchy of DSMLs in Support of Product Life-Cycle Assessment“. In: *International Conference on Business Informatics*. Siegen, Deutschland, 2019.

[Now11a] Nowak, Alexander; Leymann, Frank; Schleicher, Daniel; Schumm, David und Wagner, Sebastian: „Green business process patterns“. In: *Conference on Pattern Languages of Programs*. Portland, OR, USA, 2011.

[Now11b] Nowak, Alexander; Leymann, Frank; Schumm, David und Wetzstein, Branimir: „An architecture and methodology for a four-phased approach to green business process reengineering“. In: *International Conference on Information and Communication on Technology*. Toulouse, Frankreich, 2011.

[Now12] Nowak, Alexander; Binz, Tobias; Fehling, Christoph; Kopp, Oliver; Leymann, Frank und Wagner, Sebastian: „Pattern-driven green adaptation of process-based applications and their runtime infrastructure“. In: *Computing. Archives for Scientific Computing* 94.6 (2012), S. 463–487.

[Now13a] Nowak, Alexander; Binz, Tobias; Leymann, Frank und Urbach, Nicolas: „Determining power consumption of business processes and their activities to enable green business process reengineering“. In: *International Enterprise Distributed Object Computing Conference*. Vancouver, BC, Kanada, 2013.

[Now13b] Nowak, Alexander und Leymann, Frank: An overview on implicit green business process patterns. Universität Stuttgart, 2013.

[Now13c] Nowak, Alexander und Leymann, Frank: „Green business process patterns – part II“. In: *International Conference on Service-Oriented Computing and Applications*. Koloa, HI, USA, 2013.

[Now13d] Nowak, Alexander und Leymann, Frank: „Green enterprise patterns“. In: *Conference on Pattern Languages of Programs*. Monticello, IL, USA, 2013.

[Now14a] Nowak, A.; Breitenbücher, U. und Leymann, F.: „Automating green patterns to compensate CO2 emissions of cloud-based business processes“. In: *International Conference on Advanced Engineering Computing and Applications in Sciences*. Rom, Italien, 2014.

[Now14b] Nowak, Alexander: „Green business process management: Methode und realisierung“. Diss. Universität Stuttgart, 2014.

[Nur09] Nurseitov, Nurzhan; Paulson, Michael; Reynolds, Randall und Izurieta, Clemente: „Comparison of JSON and XML Data Interchange Formats: A Case Study“. In: International Conference on Computer Applications in Industry and Engineering. San Francisco, CA, USA, 2009.

[Obe96a] Oberweis, Andreas: Modellierung und Ausführung von Workflows mit Petri-Netzen. B. G. Teubner, 1996.

[Obe96b] Oberweis, Andreas und Sander, Peter: „Information system behavior specification by high level Petri nets“. In: *ACM Transactions on Information Systems* 14.4 (1996), S. 380–420.

[Obe96c] Oberweis, Andreas: „An integrated approach for the specification of processes and related complex structured objects in business applications“. In: *Decision Support Systems* 17.1 (1996), S. 31–53.

[OEC22] OECD: Measuring the social performance of firms through the lens of the OECD Well-being Framework. OECD Policy Insights on Well-being, Inclusion and Equal Opportunity. 2022.

[OMG13] OMG: Business Process Model and Notation (BPMN), Version 2.0.2. 2013.

[Ong14] Ong, Kian Win; Papakonstantinou, Yannis und Vernoux, Romain: „The SQL++ Unifying Semi-structured Query Language, and an Expressiveness Benchmark of SQL-on-Hadoop, NoSQL and NewSQL Databases“. In: *CoRR* (2014).

[Opi14a] Opitz, Nicky; Krüp, Henning und Kolbe, Lutz Maria: „Environmentally sustainable business process management - developing a green BPM readiness model“. In: *Pacific Asia Conference on Information Systems*. Chengdu, China, 2014.

[Opi14b] Opitz, Nicky; Krüp, Henning und Kolbe, Lutz Maria: „Green business process management – a definition and research framework“. In: *Hawaii International Conference on System Sciences*. Waikoloa, Hawaii, USA, 2014.

[Ort21] Ortmeier, Christian; Henningsen, Nadja; Langer, Adrian; Reiswich, Alexander; Karl, Alexander und Herrmann, Christoph: „Framework for the integration of Process Mining into Life Cycle Assessment“. In: *Procedia CIRP* 98 (2021), S. 163–168.

[Öst11] Österle, Hubert; Becker, Jörg; Frank, Ulrich; Hess, Thomas; Karagiannis, Dimitris; Krcmar, Helmut; Loos, Peter; Mertens, Peter; Oberweis, Andreas und Sinz, Elmar J: „Memorandum on design-oriented information systems research“. In: *European Journal of Information Systems* 20.1 (2011), S. 7–10.

[Ott12] Ottmann, Thomas und Widmayer, Peter: Algorithmen und Datenstrukturen. 5. Aufl. Spektrum, 2012. 774 S.

[Par21] Park, Gyunam und Aalst, Wil M.P. van der: „Realizing A Digital Twin of An Organization Using Action-oriented Process Mining“. In: *International Conference on Process Mining*. Eindhoven, Niederlande, 2021.

[Pat15] Patton, Jeff und Economy, Peter: User story mapping: die Technik für besseres Nutzerverständnis in der agilen Produktentwicklung. Übers. von Hildebrandt, Petra. O'Reilly, 2015.

[Pef12] Peffers, Ken; Rothenberger, Marcus; Tuunanen, Tuure und Vaezi, Reza: „Design Science Research Evaluation“. In: *Design Science Research in Information Systems: Advances in Theory and Practice*. Las Vegas, NV, USA, 2012.

[Per08] Pernici, Barbara; Ardagna, Danilo und Cappiello, Cinzia: „Business process design: Towards service-based green information systems“. In: *IFIP World Computer Congress*. Milano, Italien, 2008.

[Pet08] Petersen, Kai; Feldt, Robert; Mujtaba, Shahid und Mattsson, Michael: „Systematic Mapping Studies in Software Engineering“. In: *International Conference on Evaluation and Assessment in Software Engineering*. Bari, Italien, 2008.

[Pet15] Petersen, Kai; Vakkalanka, Sairam und Kuzniarz, Ludwik: „Guidelines for conducting systematic mapping studies in software engineering: An update“. In: *Information and Software Technology* 64 (2015), S. 1–18.

[Pez16] Pezoa, Felipe; Reutter, Juan L.; Suarez, Fernando; Ugarte, Martín und Vrgoč, Domagoj: „Foundations of JSON schema“. In: *International World Wide Web Conferences*. Genua, Schweiz, 2016.

[Pob21] Poberezkin, Evgeny: Ajv JSON Schema validator. 2021. URL: https://ajv.js.org/ (besucht am 07. 07. 2024).

[Poh19] Pohl, Johanna: „Diskussion zum Thema Ressourcen und Prozesse". ICT4S Summer School. Leiden, Niederlande, 2019.

[Pol19] Polyvyanyy, Artem; Werf, Jan Martijn E. M. van der; Overbeek, Sietse und Brouwers, Rick: „Information systems modeling: language, verification, and tool support". In: *International Conference on Advanced Information Systems Engineering*. Rom, Italien, 2019.

[Por85] Porter, Michael E.: Creating and sustaining competitive advantage: Management logics, business models, and entrepreneurial rent. The Free Press, 1985.

[PRé24] PRé Sustainability: SimaPro. 2024. URL: https://simapro.com/ (besucht am 07. 07. 2024).

[Pur19] Purvis, Ben; Mao, Yong und Robinson, Darren: „Three pillars of sustainability: in search of conceptual origins". In: *Sustainability Science* 14.3 (2019), S. 681–695.

[Rah17] Rahim, Norfhadzilahwati: „Sustainable Growth Rate and Firm Performance: a Case Study in Malaysia". In: *International Journal of Management, Innovation & Entrepreneurial Research* 3.2 (2017), S. 48–60.

[Ray03] Raymond, Eric Steven: The Art of Unix Programming. Pearson, 2003.

[Reb04] Rebitzer, Gerald; Ekvall, Tomas; Frischknecht, Rolf; Hunkeler, David; Norris, Gregory; Rydberg, Tomas; Schmidt, Wulf Peter; Suh, Sangwon; Weidema, Bo und Pennington, David: „Life cycle assessment". In: *Environment International* 30.5 (2004), S. 701–720.

[Rec11] Recker, Jan; Rosemann, Michael und Gohar, Ehsan Roohi: „Measuring the carbon footprint of business processes". In: *Business Process Management Workshops*. Hoboken, NJ, USA, 2011.

[Rec12] Recker, Jan; Rosemann, Michael; Hjalmarsson, Anders und Lind, Mikael: „Modeling and analyzing the carbon footprint of business processes". In: *Green Business Process Management: Towards the Sustainable Enterprise*. Hrsg. von Brocke, Jan vom; Seidel, Stefan und Recker, Jan. Springer, 2012, S. 93–109.

[Red13] Redondo, Jose Manuel und Ortin, Francisco: „Efficient support of dynamic inheritance for class- and prototype-based languages". In: *Journal of Systems and Software* 86.2 (Feb. 2013), S. 278–301.

[Rei10] Reisig, Wolfgang: Petrinetze. Springer, 2010.

[Rei13] Reinhard, Jürgen; Zah, Rainer; Wohlgemuth, Volker und Jahr, Paul: „Applying Life Cycle Assessment within Discrete Event Simulation: Practical Application of the Milan/EcoFactory Material Flow Simulator". In: *International Conference on Informatics for Environmental Protection*. Hamburg, Deutschland, 2013.

[Rei14] Reiter, Markus; Fettke, Peter und Loos, Peter: „Towards green business process management: Concept and implementation of an artifact to reduce the energy consumption of business processes". In: *Hawaii International Conference on System Sciences*. Waikoloa, HI, USA, 2014.

[Rei17] Reijers, Hajo A.; Vanderfeesten, Irene; Plomp, Marijn G. A.; Van Gorp, Pieter; Fahland, Dirk; Crommert, Wim L. M. van der und Garcia, H. Daniel Diaz: „Evaluating data-centric process approaches: Does the human factor factor in?" In: *Software & Systems Modeling* 16.3 (2017), S. 649–662.

[Rob22] Robie, Jonathan; Fourny, Ghislain; Brantner, Matthias; Florescu, Daniela; Westmann, Till und Zaharioudakis, Markos: JSONiq. 2022. url: https://www.jsoniq.org/ (besucht am 07. 07. 2024).

[Rum90] Rummler, Geary A. und Brache, Alan P.: Improving Performance: How to manage the white space on the organizational chart. Jossey-Bass, 1990.

[Rus04a] Russell, Nick: „Workflow data patterns". In: *International Conference on Conceptual Modeling*. Shanghai, China, 2004.

[Rus04b] Russell, Nick: Workflow resource patterns. Eindhoven University of Technology, 2004.

[Rus06] Russell, Nick; Hofstede, Arthur H. M. ter; Der Aalst, Wil M. P. van und Mulyar, Nataliya: „Workflow Control-Flow Patterns: A Revised View". In: *BPM Center Report* (2006).

[Saa18] Saake, Gunther; Sattler, Kai-Uwe und Heuer, Andreas: Datenbanken: Konzepte und Sprachen. MITP-Verlag, 2018.

[Sah18] Sahatqija, Kosovare; Ajdari, Jaumin; Zenuni, Xhemal; Raufi, Bujar und Ismaili, Florije: „Comparison between relational and NOSQL databases". In: *International Convention on Information and Communication Technology, Electronics and Microelectronics*. Opatija, Kroatien, 2018.

[SAP24] SAP Signavio: Signavio Process Manager. 2024. url: https://www.signavio.com/de/products/process-manager/ (besucht am 07. 07. 2024).

[Sar15] Saryerwinnie, James: JMESPath. 2015. url: https://jmespath.org/ (besucht am 07. 07. 2024).

[Sch12] Schönthaler, Frank; Vossen, Gottfried; Oberweis, Andreas und Karle, Thomas: Business processes for business communities. Springer, 2012.

[Sch13] Scheer, August-Wilhelm: ARIS - vom Geschäftsprozess zum Anwendungssystem. Springer, 2013.

[Sch17] Schoormann, Thorsten; Behrens, Dennis und Knackstedt, Ralf: „Sustainability in business process models: A taxonomy-driven approach to synthesize knowledge and structure the field". In: *International Conference on Information Systems*. Seoul, Korea, 2017.

[Sch19] Schoormann, Thorsten; Kutzner, Kristin; Pape, Sebastian und Knackstedt, Ralf: „Elevating social sustainability in business processes: A pattern-based approach". In: *International Conference on Information Systems*. München, Deutschland, 2019.

[Sch20] Schreiber, Clemens: „Automated Sustainability Compliance Checking Using Process Mining and Formal Logic". In: *International Conference on ICT for Sustainability*. Bristol, UK, 2020.

[Sch94] Scheer, August-Wilhelm: Business process engineering: reference models for industrial enterprises. Springer Science & Business Media, 1994.

[Sch97a] Schmidt, Mario: „Stoffstromnetze zwischen produktbezogener und betrieblicher Ökobilanzierung". In: *Ökobilanzierung mit Computerunterstützung*. Hrsg. von Schmidt, Mario und Häuslein, Andreas. Springer, 1997, S. 11–24.

[Sch97b] Schmidt, Mario und Häuslein, Andreas: „Der Einstieg in Umberto – ein einfaches Beispiel". In: *Ökobilanzierung mit Computerunterstützung*. Hrsg. von Schmidt, Mario und Häuslein, Andreas. Springer, 1997, S. 27–36.

[Sco22] Scope3transparent Konsortium: Wo steht die Lieferkette elektronischer Produkte und Komponenten in Sachen Scope-3-Bilanzierung? 2022.

[Sco23] Scope3transparent Konsortium: Bilanzierung von Treibhausgasen in der Lieferkette elektronischer Komponenten und Produkte. 2023.

[Sei11] Sein; Henfridsson; Purao; Rossi und Lindgren: „Action Design Research“. In: *MIS Quarterly* 35.1 (2011), S. 37.

[Sib85] Sibertin-Blanc, C: „High Level Petri Nets with Data Structure“. In: European Workshop on Application and Theory of Petri Nets. Espoo, Finnland, 1985.

[Sko16] Skouradaki, Marigianna; Ferme, Vincenzo; Pautasso, Cesare; Leymann, Frank und Hoorn, André van: „Micro-Benchmarking BPMN 2.0 Workflow Management Systems with Workflow Patterns“. In: *International Conference on Advanced Information Systems Engineering*. Ljubljana, Slowenien, 2016.

[Sno21] Snoeck, Monique; De Smedt, Johannes und De Weerdt, Jochen: „Supporting data-aware processes with MERODE“. In: *Enterprise, Business-Process and Information Systems Modeling*. Melbourne, Australien, 2021, S. 131–146.

[Sno23] Snoeck, Monique; Verbruggen, Charlotte; De Smedt, Johannes und De Weerdt, Jochen: „Supporting data-aware processes with MERODE“. In: *Software and Systems Modeling* 22.6 (2023), S. 1779–1802.

[Sph24] Sphera: Life Cycle Assessment Software and Data | Sphera (GaBi). 2024. url: https://sphera.com/solutions/product-stewardship/life-cycle-assessment-software-and-data/ (besucht am 07. 07. 2024).

[Sta19] Stadtländer, Maren; Schoormann, Thorsten und Knackstedt, Ralf: „How Software Promotes the Integration of Sustainability in Business Process Management“. In: *International Conference on Business Informatics*. Siegen, Deutschland, 2019.

[Sta24] Stack Overflow: Stack Overflow Trends. 2024. url: https://insights.stackoverflow.com/trends?tags=json%2Cxml%2Ccsv%2Csoap (besucht am 07. 07. 2024).

[Sta73] Stachowiak, Herbert: Allgemeine Modelltheorie. Wien, New York: Springer, 1973.

[Ste19] Steinau, Sebastian; Marrella, Andrea; Andrews, Kevin; Leotta, Francesco; Mecella, Massimo und Reichert, Manfred: „DALEC: a framework for the systematic evaluation of data-centric approaches to process management software“. In: *Software & Systems Modeling* 18.4 (2019), S. 2679–2716.

[Sto12] Stolze, Carl; Semmler, Gebke und Thomas, Oliver: „Sustainability in business process management research - a literature review“. In: *Americas Conference on Information Systems*. Seattle, WA, USA, 2012.

[Sto24] Stofer, Manuel: manuelstofer/json-pointer. 2024. url: https://github.com/manuelstofer/json-pointer (besucht am 07. 07. 2024).

[Tay11] Taylor, Frederick Winslow: The principles of scientific management. Harper & Brothers, 1911.

[Ten01] Tennant, Geoff: Six sigma: SPC and TQM in manufacturing and services. Gower Publishing, 2001.

[The23] The Apache Software Foundation: CouchDB. 2023. url: https://couchdb.apache.org/ (besucht am 07. 07. 2024).

[Umw24] Umweltbundesamt: Probas. 2024. url: https://www.probas.umweltbundesamt.de/ (besucht am 07. 07. 2024).

[UNE15] UNEP und SETAC: Guidance on organizational life cycle assessment. 2015.

[UNE20] UNEP: Guidelines for Social Life Cycle Assessment of Products and Organizations. 2020.

[UNE21] UNEP: Methodological Sheets for Subcategories in Social Life Cycle Assessment (S-LCA). 2021.

[Ung04] Unger, Nicole; Beigl, Peter und Wassermann, Gudrun: General requirements for LCA software tools. Wien, Österreich, 2004.

[Uni87] United Nations General Assembly: Report of the world commission on environment and development: Our common future (A/42/427). 1987.

[Van14] Van Looy, Amy; De Backer, Manu und Poels, Geert: „A conceptual framework and classification of capability areas for business process maturity“. In: *Enterprise Information Systems* 8.2 (2014), S. 188–224.

[Ven12] Venable, John; Pries-Heje, Jan und Baskerville, Richard: „A Comprehensive Framework for Evaluation in Design Science Research“. In: *Design Science Research in Information Systems. Advances in Theory and Practice*. Las Vegas, NV, USA, 2012.

[Ver23] Verwaltung-Berufsgenossenschaft: VBG-Jahresbericht 2023. 2023.

[W3C12] W3C: W3C XML Schema Definition Language (XSD) 1.1. 2012.

[W3C17a] W3C: XPath 3.1. 2017.

[W3C17b] W3C: XQuery 3.1. 2017.

[WBC04] WBCSD und WRI: The Greenhouse Gas Protocol. 2004.

[WBC21] WBCSD: Pathfinder Framework: Guidance for the Accounting and Exchange of Product Life Cycle Emission. 2021.

[Web02] Webster, Jane und Watson, Richard T.: „Analyzing the Past to Prepare for the Future: Writing a Literature Review“. In: *MIS Quarterly* 26.2 (2002), S. xiii–xxiii.

[Wei98] Weitz, Wolfgang: „Combining structured documents with high-level petri-nets for workflow modeling in internet-based commerce“. In: *International Journal of Cooperative Information Systems* 7.4 (1998), S. 275–296.

[Wes11] Wesumperuma, Ashini; Ginige, Jeewani Anupamal; Ginige, Athula und Hol, Ana: „A framework for multi-dimensional business process optimization for GHG emission mitigation“. In: *Australasian Conference on Information Systems*. Sydney, Australien, 2011.

[Wes13] Wesumperuma, Ashini; Ginige, Athula; Ginige, Jeewani Anupama und Hol, Ana: „Green activity based management (ABM) for organisations“. In: *Australasian Conference on Information Systems*. Melbourne, Australien, 2013.

[Wes15] Wesumperuma, Ashini Emalka: „Multi-dimensional business process optimisation for greenhouse gas (GHG) emission management“. Diss. University of Western Sydney, 2015.

[Wes19] Weske, Mathias: Business process management: Concepts, languages, architectures. 3. Aufl. Springer, 2019.

[Whi14] White, Gareth R.T. und James, Peter: „Extension of process mapping to identify “green waste”“. In: *Benchmarking* 21.5 (2014), S. 835–850.

[Wid14] Widok, Andi H. und Wohlgemuth, Volker: „Social Sustainability and Manufacturing Simulation“. In: *International Conference on Advances in System Simulation*. Nizza, Frankreich, 2014.

[Woh00] Wohlgemuth, Volker und Page, Bernd: „Einbettung von Transportmodellen und diskreten Simulationsmodellen in Stoffstromnetze“. In: *Conference on Environmental Informatics*. Bonn, Deutschland, 2000.

[Woh06] Wohlgemuth, Volker; Page, Bernd und Kreutzer, Wolfgang: „Combining discrete event simulation and material flow analysis in a component-based approach to industrial environmental protection“. In: *Environmental Modelling & Software* 21.11 (2006), S. 1607–1617.

[Woh14] Wohlin, Claes: „Guidelines for snowballing in systematic literature studies and a replication in software engineering“. In: *International Conference on Evaluation and Assessment in Software Engineering*. London, UK, 2014.

[Wol19] Wolfson, Adi; Dominguez-Ramos, Antonio und Irabien, Angel: „From Goods to Services: The Life Cycle Assessment Perspective“. In: *Journal of Service Science Research* 11.1 (2019), S. 17–45.

[Wor24] Workflow Patterns Initiative: Workflow Patterns. 2024. url: www.workflowpatterns.com (besucht am 06. 07. 2024).

[Wri22a] Wright, A.; Andrews, H. und Hutton, B.: JSON Schema Validation: A Vocabulary for Structural Validation of JSON. 2022. url: https://json-schema.org/draft/2020-12/json-schema-validation.html (besucht am 07. 07. 2024).

[Wri22b] Wright, A.; Andrews, H.; Hutton, B. und Dennis, G.: JSON Schema: A Media Type for Describing JSON Documents. 2022. url: https://json-schema.org/draft/2020-12/json-schema-core.html (besucht am 07. 07. 2024).

[You24] You, Evan: Vue.js. 2024. url: vuejs.org (besucht am 07. 07. 2024).

[Zen14] Zendoia, Jose; Woy, Udisien; Ridgway, Nicola; Pajula, Tiina; Unamuno, Gorka; Olaizola, Aratz; Fysikopoulos, Apostolos und Krain, Roland: „A specific method for the life cycle inventory of machine tools and its demonstration with two manufacturing case studies“. In: *Journal of Cleaner Production* 78 (2014), S. 139–151.

[Zhu15] Zhu, Xinwei; Zhu, Guobin; Broucke, Seppe Vanden und Recker, Jan: „On merging business process management and geographic information systems: Modeling and execution of ecological concerns in processes“. In: *International Conference on Geo-Informatics in Resource Management and Sustainable Ecosystem*. Ypsilanti, MI, USA, 2015.

Abbildungsverzeichnis

Tabellenverzeichnis

Auflistungsverzeichnis

Abkürzungsverzeichnis

BPMN	Business Process Model and Notation
CFCe	CFC-Äquivalente
CO2e	CO2-Äquivalente
DTD	Document Type Definition
EPK	Ereignisgesteuerte Prozessketten
JSON	JavaScript Object Notation
REST	Representational State Transfer
SGML	Standard Generalized Markup Language
SO2e	SO2-Äquivalente
SQL	Structured Query Language
XML	Extensible Markup Lanuguage