technische universität dortmund

FAKULTÄT MASCHINENBAU

itpl IT in Produktion und Logistik

Masterarbeit

Entwicklung eines graphbasierten Simulators zur Simulation von Logistiknetzwerken in einem Data-Farming-Framework

verfasst von

Alexander Wuttke, B. Sc.
Matrikel-Nr.: 169108
Studiengang: Maschinenbau
alexander2.wuttke@tu-dortmund.de

Ausgegeben am: 02.11.2021
Eingereicht am: 19.04.2022

Prüferin: Dr.-Ing. Dipl.-Inform Anne Antonia Scheidler
Betreuer: M. Sc. Joachim Hunker

Inhaltsverzeichnis

1 Einleitung

Der Anspruch an die Verfügbarkeit von Waren durch Geschäfts- und Endkunden hat in den letzten Jahren immer weiter zugenommen, sodass mittlerweile eine vollständige Verfügbarkeit als beinahe selbstverständlich gilt. Logistiknetzwerke müssen in der Lage sein, diese vollständige Verfügbarkeit zu gewährleisten, damit die betreibenden Unternehmen konkurrenzfähig bleiben (Riha 2009). Damit dieses Ziel erreicht werden kann, sind eine Vielzahl von komplexen Prozessen zur Planung und zum Betrieb nötig. Darüber hinaus muss ein Logistiknetzwerk so aufgebaut sein, dass Kosten minimiert werden und das gesamte System für das betreibende Unternehmen rentabel ist. Die verschiedenartigen Ziele, wie beispielsweise niedrige Bestände und hohe Verfügbarkeit, führen zu einem Zielkonflikt, der Entscheidungsträger vor weitere Herausforderungen stellt (Seeck 2010). Moderne Logistiknetzwerke stellen somit hochkomplexe Systeme dar.

Das Management solcher Logistiknetzwerke, die auch als Supply Chains bezeichnet werden, wird unter dem Begriff Supply Chain Management zusammengefasst. Zur Unterstützung des Supply Chain Managements können Entscheidungsunterstützungssysteme eingesetzt werden (Rabe et al. 2017b). Mit ihnen kann kontextbezogenes Wissen entdeckt, aufbereitet und für Entscheidungsträger visualisiert werden. Als Grundlagen dienen Daten über das Logistiknetzwerk, welche für gewöhnlich in einer Datenbank persistiert sind. Fayyad et al. (1996) beschreiben ein Prozessmodell, mit dem Wissensentdeckung in Datenbanken realisiert werden kann. Zentrale Phase dieses Modells ist das sogenannte Data Mining, welches nach Mustern in Datenbeständen sucht und sie extrahiert.

Die nötigen Daten werden zumeist während des Betriebs des Logistiknetzwerks gesammelt. Durch die Trends der Digitalisierung und Logistik 4.0 hat die Menge der zur Verfügung stehenden Daten in den letzten Jahren deutlich zugenommen (Bousonville 2017). Jedoch weisen Daten, die auf diesem Weg erhoben werden, oftmals niedrige Qualität auf, was sich beispielsweise in fehlenden oder fehlerhaften Daten manifestiert (García et al. 2015). Als Alternative zur Datensammlung im Betrieb bietet sich die Datengenerierung durch eine Methode wie beispielsweise dem Data Farming an, bei der die Daten anhand von Simulationsexperimenten mit einem Modell des Logistiknetzwerks generiert werden und eine höhere Datenqualität erreicht werden kann (Sanchez 2018). Die erhaltenen Daten können anschließend, beispielsweise mit dem Wissensentdeckungsprozess nach Fayyad et al. (1996), analysiert werden, um verborgenes Wissen zu entdecken.

Zur Datenhaltung wird für gewöhnlich ein relationales Datenbanksystem verwendet. Zunehmend werden andere Arten von Datenbanksystemen, oftmals als NoSQL-Datenbanken bezeichnet, verwendet, die Vorteile, wie eine bessere Skalierbarkeit bei großen Datenbeständen und eingehende Geschwindigkeitsvorteile, gegenüber relationalen Datenbanksystemen bieten können (Hecht und Jablonski 2011). Ein spezifischer Typ dieser NoSQL-Datenbanken ist die Graphdatenbank, welche sich insbesondere für Daten anbietet, die untereinander eine starke Vernetzung aufweisen (Meier und Kaufmann 2016). Da Logistiknetzwerke eine netzwerkartige Struktur aufweisen und sie mathematisch als Graph beschrieben werden können, bietet sich für sie eine Speicherung in einer Graphdatenbank an.

Wird eine Graphdatenbank zur Datenhaltung verwendet, liegen die Daten in Form von Knoten und Kanten vor. Jedoch arbeiten Softwarewerkzeuge zur Weiterverarbeitung und Analyse von Daten, wie beispielsweise Simulatoren, für gewöhnlich mit einem relationalen Datenbankmodell (Gutenschwager et al. 2017). Das führt beim Betrieb solcher Werkzeuge, wenn sie in Kombination mit einer Graphdatenbank betrieben werden soll, sowohl beim Laden von Daten aus der Datenbank, als auch beim Schreiben von Daten in die Datenbank zu zusätzlichen Transformationsschritten, die einen Mehraufwand darstellen.

Zur Vermeidung der zusätzlichen Transformationsschritte kann ein Simulator verwendet werden, der Daten in Form eines Graphen einlesen, verarbeiten und ausgeben kann. Neben der Vermeidung von Transformationen erlaubt eine graphbasierte Arbeitsweise des Werkzeugs eine einfachere Integration von graphbasierten Verfahren, die in Logistiknetzwerken beispielsweise zur Tourenplanung eingesetzt werden. Allerdings existiert zurzeit, nach bestem Wissen des Autors, kein Simulator, der diese Anforderungen erfüllt.

Ziel dieser Arbeit ist es, einen ebensolchen Simulator zu entwickeln. Das Hauptziel besteht somit in der Entwicklung eines graphbasierten Simulators zur Simulation von Logistiknetzwerken, der spezialisiert ist auf die Anwendung in einem Data-Farming-Framework. Zur Erreichung des Hauptziels müssen zunächst Teilziele definiert und erreicht werden.

Das erste Teilziel stellt das Schaffen des nötigen Kontextwissens dar. Dazu gehören zum einen die technischen Grundlagen, von Themen wie Simulation, Data Farming, Graphen und Daten, und zum anderen eine Einordnung in den logistischen Kontext. Ein weiteres Teilziel ist die Identifikation und Festlegung von funktionalen und nicht-funktionalen Anforderungen an den Simulator. Dies umfasst auch die Festlegung eines Funktionsumfangs hinsichtlich zu simulierender Eigenschaften eines Logistiknetzwerks. Ein anderes Teilziel ist die Entwicklung eines Konzeptschemas, auf dessen Basis Simulationsmodelle zur Simulation durch den Simulator erstellt werden können. Dazu gehört auch die Entwicklung eines Editors mit einer grafischen Benutzeroberfläche, mit der der Nutzer ein Modell entsprechend dem Konzeptmodell erstellen kann. Der Entwurf eines geeigneten Softwaredesigns des Simulators zur Erfüllung der erhaltenen Anforderungen stellt ein weiteres Teilziel dar. Weiterhin ist die Umsetzung eines Prototyps des entworfenen Simulators ein Teilziel zur Untersuchung dessen. Abschließend wird zur Erreichung des letzten Teilziels, mithilfe des Prototyps, ein exemplarischer Data-Farming-Vorgang anhand von zwei Fallbeispielen durchgeführt und anschließend diskutiert.

Zur Erreichung der genannten Ziele wird methodisch wie folgt vorgegangen. Zunächst werden in Kapitel 2 die technischen Grundlagen beschrieben. Zu Beginn werden die Methode der Simulation und die dazugehörigen Simulatoren im Kontext von Logistiknetzwerken vorgestellt. Ein wichtiger Aspekt sind dabei Experimentierpläne, da sie eine wichtige Rolle bei der Anwendung von Data Farming spielen, und die Modellierung. Als Nächstes werden Graphen und ihre mathematische Abbildung erklärt. Weiterhin wird in das Thema Daten und Systeme zum Persistieren von Daten, den Datenbanksystemen, eingeführt. Spezielle Datenbanksysteme, auf denen der Fokus liegen wird, sind Graphdatenbanksysteme. Auch wird auf Möglichkeiten zur Modellierung von Daten eingegangen. Da der Simulator in ein Data-Farming-Framework eingebettet werden soll, wird abschließend die Methode des Data Farmings erläutert.

In Kapitel 3 wird der logistische Kontext vorgestellt und in Grundlagen eingeleitet, damit eine nötige Wissensbasis für die Problemdomäne geschaffen wird und eine Abgrenzung des Begriffs *Logistiknetzwerke* erfolgen kann. Dies umfasst insbesondere die Einleitung in Logistiknetzwerke und deren Modellierung. Weiterhin werden einige logistische Aufgaben

eines Logistiknetzwerks vorgestellt, wobei ein Fokus auf das Bestandsmanagements und der Tourenplanung gelegt wird, da diese beiden Aufgaben im zu entwickelnden Simulator exemplarisch untersucht werden.

Nachdem die technischen Grundlagen erläutert wurden und in den logistischen Kontext eingeführt wurde, wird in Kapitel 4 die Entwicklung des Simulators und dessen Umsetzung mit einem Prototyp beschrieben. Für eine zielgerichtete Arbeitsweise werden zunächst funktionale und nicht-funktionale Anforderungen abgeleitet und spezifiziert. Hierbei wird auch festgelegt, welche logistischen Fragestellungen durch die Simulation beantwortbar sein sollen. Anschließend wird ein Konzeptmodell für Logistiknetzwerke entworfen, auf dessen Basis der Nutzer für den Simulator experimentierfähige Modelle erstellen kann. Bevor die Umsetzung des Prototypen erfolgt, wird ein Softwaredesign entwickelt, wozu die Softwarearchitektur, die Beschreibung der einzelnen Softwarekomponenten und Funktionen, sowie die grafische Benutzeroberfläche gehören. Nach der Ausarbeitung des Softwaredesigns wird die Implementierung des Prototypen vorgestellt.

In Kapitel 5 wird der Prototyp anhand von zwei Fallbeispielen erprobt. Dazu werden die beiden Fallbeispiele zunächst geplant, um eine zielgerichtete Untersuchung zu ermöglichen. Nach der Planung erfolgt die Bearbeitung der Fallbeispiele. Für jedes Fallbeispiel wird zunächst ein Szenario eingeführt, eine passende Modellierung im Editor durchgeführt, ein exemplarischer Data-Farming-Vorgang vollzogen und die Ergebnisse dargestellt. Basierend auf den Erkenntnissen der Fallbeispiele und den Informationen aus Kapitel 4 wird eine Diskussion über den entwickelten Simulator geführt. Beendet wird das Kapitel mit einem Fazit der Diskussion.

Die Arbeit endet mit einer zusammenfassenden Betrachtung in Kapitel 6, in der weitere Forschungsmöglichkeiten hinsichtlich der Erprobung des entwickelten Simulators und allgemein graphbasierter Simulatoren aufgeworfen werden.

2 Technische Grundlagen des Data Farmings und der Graphen

Basis der Entwicklung eines graphbasierten Simulators zur Anwendung in einem Data-Farming-Framework sind technische Grundlagen. Zunächst werden in Abschnitt 2.1 Grundlagen der Methode Simulation vorgestellt. Dies ist nötig, da Simulation ein essentieller Bestandteil des Data Farmings und von Simulatoren ist. Dazu gehören Modelle und deren Modellierung, sowie die statistische Versuchsplanung und eine Betrachtung von Simulatoren. Anschließend wird in Abschnitt 2.2 in die Graphentheorie eingeführt, damit graphbasierte Methoden nachvollzogen werden können. Das Kapitel schließt mit einer Einführung in das Thema Daten in Abschnitt 2.3 ab, in dem auch auf die Speicherung und Modellierung von Daten eingegangen wird. Der Abschnitt beschreibt außerdem die Methode Data Farming näher.

2.1 Grundlagen der Simulation

Simulation stellt eine Methode dar, die nach VDI (2014, S. 3) als "Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierbaren Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind; insbesondere werden die Prozesse über die Zeit entwickelt". Diese Auffassung von Simulation findet sich auch in weiten Teilen der Literatur wieder (vgl. Banks 1998, Robinson 2004, Law 2015 und Gutenschwager et al. 2017). In der Definition wird der Begriff *System* verwendet, der in Anlehnung an (DIN IEC 60050-351 2014) als eine Menge von Elementen, die in Beziehungen stehen, von ihrer Umwelt abgegrenzt sind und als ein Ganzes angesehen werden, definiert ist. Es wird spezifiziert, dass das System samt seiner dynamischen Prozesse abgebildet und eine Entwicklung über die Zeit erkennbar sein soll. Solche Systeme werden als dynamische Systeme bezeichnet, die im Gegensatz zu statischen Systemen, bei denen eine Ausgangsgröße zeitgleich mit den Eingangsgrößen feststeht, durch einen Zeitfortschritt charakterisiert sind (Wunsch und Schreiber 2005). Weiterhin wird in der Definition der Begriff *experimentierbares Modell* verwendet. Der VDI (2014, S. 3) definiert Modelle als „vereinfachte Nachbildung eines geplanten oder existierenden Systems mit seinen Prozessen in einem anderen begrifflichen oder gegenständlichen System". Durch den Zusatz *experimentierbar* werden explizit solche Modelle gemeint, die ablauffähig sind, und damit zur experimentellen Analyse von System verwendet werden können, um Rückschlüsse für die Wirklichkeit zu ziehen. Die Ausführung der Modelle wird mithilfe von Simulatoren durchgeführt, zu denen weitere Informationen in Abschnitt 2.1.3 folgen. Ein Modell, das ablauffähig ist und sich für Simulation eignet, wird auch als Simulationsmodell bezeichnet (Gutenschwager et al. 2017). Eine detaillierte Betrachtung von Modellen und wie diese erstellt werden können folgt in Abschnitt 2.1.1. Der letzte Teil der Definition von Simulation durch den VDI 2014 bezieht sich auf die Zeit. Dort ist die Zeit des realen Systems gemeint, die das Modell abbildet. Diese Zeit wird auch Simulationszeit bzw. der Simulationszeitraum genannt (Kuhn und Wenzel 2008). Bezüglich des Simulationszeitraums sei

angemerkt, dass einige Modelle zunächst eine Einschwingphase durchlaufen sollten, um unerwünschte Einflüsse, die sich durch Startbedingungen eines Modells ergeben können, auszuschließen (Gutenschwager et al. 2017).

Simulation ist eine vergleichsweise zeit- und kostenaufwändige Methode (Banks 1998). Daher ist zunächst die Betrachtung der Simulationswürdigkeit ratsam. Die Anwendung von Simulation bietet sich immer dann an, wenn analytische Methoden an ihre Grenzen stoßen. Das ist dann der Fall, wenn eine gewisse Komplexität des zu untersuchenden Systems erreicht wird (Law 2015). Eine klare Definition, ab wann ein System als zu komplex für analytische Methoden gilt, existiert nicht und muss daher im Einzelfall abgeschätzt werden. Komplexitätstreiber in einem System können beispielsweise die Anwendung von stochastischen Einflüssen auf Ausgangsgrößen sein (Law 2015). Andere Rechtfertigungen der Simulationswürdigkeit sind dann gegeben, wenn nicht ausreichend erforschte Gebiete untersucht werden sollen, Experimente an realen Modellen unmöglich oder zu teuer wären, oder die Grenzen menschlicher Vorstellungskraft überschritten werden (ASIM 1997). Der VDI (2014) merkt an, dass stets ein angemessenes Kosten-Nutzen-Verhältnis gewährleistet sein sollte.

Wird die Methode der Simulation in einem Projekt eingesetzt, wird von einer Simulationsstudie gesprochen. Zur Durchführung von Simulationsstudien existiert eine Vielzahl von Vorgehensmodellen, wie beispielsweise von Landry und Oral (1993), Banks et al. (2005), Bernhard und Dragan (2007) oder Law (2015). Sie unterscheiden sich hinsichtlich Komplexität, Detailgrad und Umfang, weisen aber auch Ähnlichkeiten in ihren Prozessen auf, wie beispielsweise das Vorhandensein einer Systemanalyse. Ein Vorgehensmodell, das insbesondere im Bereich der Produktion und Logistik große Verbreitung findet, stammt von Rabe et al. (2008). Diese Arbeit orientiert sich, wenn von einer Simulationsvorgehensmodell gesprochen wird, an dem von Rabe et al. (2008), welches in der folgenden Abbildung 2.1 dargestellt ist.

Das gezeigte Vorgehensmodell in Abbildung 2.1 ist in sieben Phasen unterteilt, die als Ellipsen dargestellt sind. Die Phasen werden, entsprechend der Pfeilrichtung, von oben nach unten durchlaufen. Dabei ist zu beachten, dass einige Phasen parallel ausgeführt werden. Jede der Phasen des Vorgehensmodells hat ein entsprechendes Phasenergebnis, das über ein Rechteck mit abgerundeten Ecken dargestellt ist. Der Beginn des Vorgehensmodells ist durch den Eingang einer Zielbeschreibung gekennzeichnet. Die sieben Phasen werden im Folgenden kurz beschrieben.

Aufgabendefinition Präzisierung und Vervollständigung der Zielbeschreibung. Ergebnis ist die Aufgabenspezifikation, der alle Beteiligten zustimmen können und die innerhalb des Zeit- und Kostenrahmens der Simulationsstudie voraussichtlich realisierbar ist.

Systemanalyse Dokumentation der Zielsetzungen, Eingaben, Ausgaben, Elementen, Beziehungen zwischen Elementen, Annahmen und Vereinfachungen des zu entwickelnden Simulationsmodells. Das Ergebnis der Phase stellt das Konzeptmodell dar, das beschreibt, welche Aufgaben in welcher Weise gelöst werden sollen und welcher Umfang sowie Detailgrad dem Modell zugrunde liegen.

Modellformalisierung Überführung der Elemente und Beziehungen des Konzeptmodells in ein formales Modell, das eine Implementierung ohne weitere Analysen und Rückfragen ermöglicht.

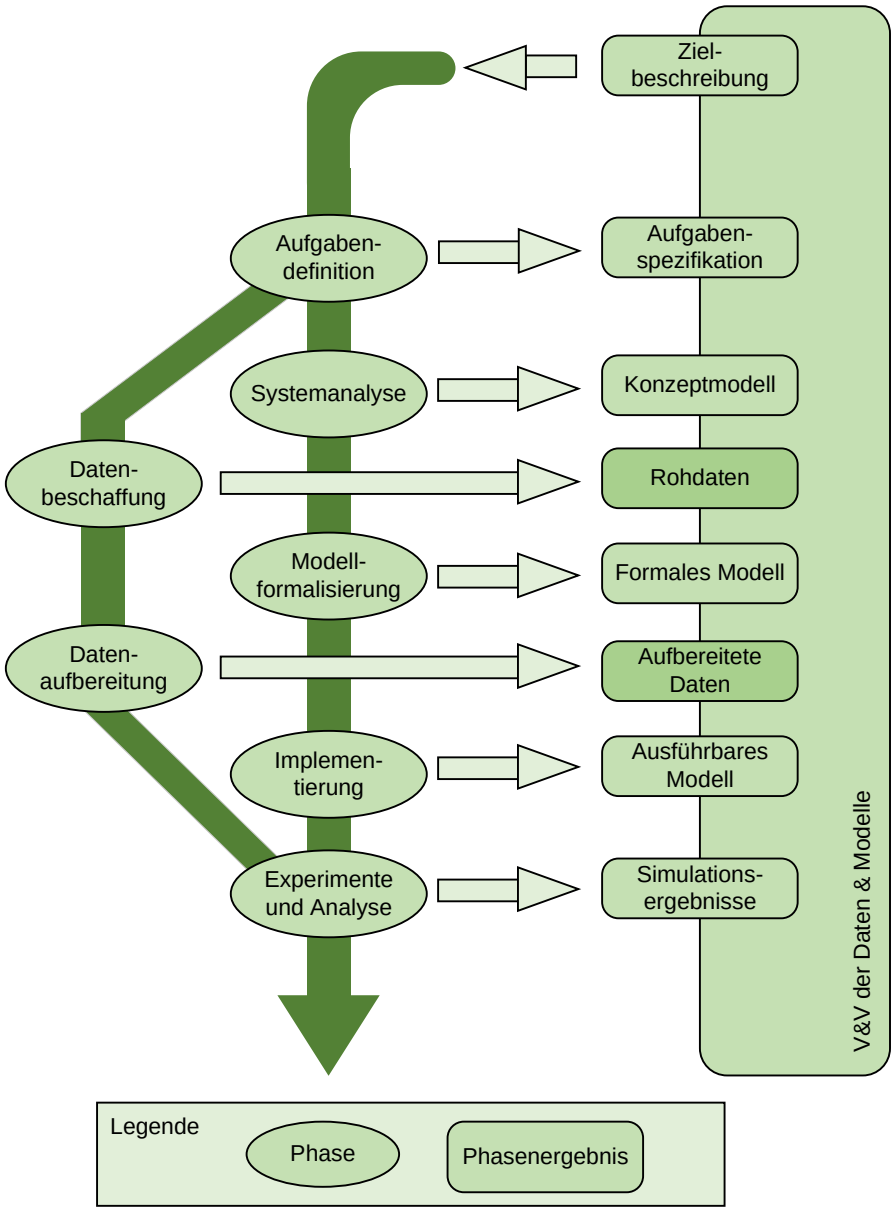


Abbildung 2.1: Simulationsvorgehensmodell nach Rabe et al. (2008)

Implementierung Überführung des formalen Modells in ein ausführbares Modell. Das erhaltene Simulationsmodell muss unter Berücksichtigung der Vorgaben des verwendeten Simulators erstellt werden.

Experimente und Analyse Zusammenführung des ausführbaren Modells und der aufbereiteten Daten um anschließend die Experimente durchzuführen. Die Ergebnisse der Experimente stellen gleichzeitig die Ergebnisse der Simulationsstudie dar.

Datenbeschaffung Akquirierung und Beschaffung von Rohdaten, die durch die Aufgabenspezifikation und dem Konzeptmodell spezifiziert wurden.

Datenaufbereitung Aufbereitung und Vorverarbeitung der Rohdaten, um diese in ausreichender Qualität und Quantität dem ausführbaren Modell zur Verfügung zu stellen.

Abbildung 2.1 ist weiterhin zu entnehmen, dass eine Verifikation und Validierung (V&V) jeglicher Phasenergebnisse und der Zielbeschreibung sichergestellt werden sollen. Die V&V haben die Aufgabe, eine konstante Überwachung der Phasenergebnisse hinsichtlich Korrektheit und Genauigkeit bei der Einhaltung der Anforderungen durchzuführen. Durch die parallele Durchführung wird sichergestellt, dass Verletzungen der geforderten Korrektheit und Genauigkeit möglichst schnell entdeckt und behoben werden können. Der V&V stehen eine Vielzahl von Techniken zur Verfügung. Je nach zu überprüfendem Gegenstand bietet sich eine Teilmenge der Techniken zur Anwendung an. Beispielhafte Techniken sind die Animation, die eine grafische Kontrolle durch den Anwender ermöglichen, der Dimensionstest, bei dem die physikalischen Einheiten der Rechnungen und Ergebnisse auf Korrektheit geprüft werden, oder das strukturierte Durchgehen, bei dem die Projektbeteiligten zusammen den zu überprüfenden Gegenstand untersuchen. Auch die Überprüfung der Rohdaten und aufbereiteten Daten fällt in den Aufgabenbereich der V&V. Die konsequente Anwendung angemessener V&V trägt maßgeblich zur Glaubwürdigkeit einer Simulationsstudie und dessen Ergebnissen bei (Rabe et al. 2008). Eine ausführliche Diskussion zum Thema V&V und angemessenen Techniken kann in Rabe et al. (2008) gefunden werden.

Die Anwendungsfelder von Simulation sind zahlreich. Verbreitete Anwendung von Simulation finden sich beispielsweise in den Bereichen Gesundheit, Militär, Personenverkehr oder Produktion (Banks et al. 2014). Auch in der Logistik ist Simulation eine häufig anzutreffende Methode (Gutenschwager et al. 2017).

2.1.1 Modelle und Modellierung

In Abschnitt 2.1 wurde der Begriff *Modell* bereits als vereinfachende Abstraktion eines realen Systems definiert. Eine andere verbreitete Definition von Modellen stammt von Stachowiak (1973), die drei Hauptmerkmale beinhaltet. Nach ihm sind die Hauptmerkmale das Abbildungsmerkmal, das Verkürzungsmerkmal und das pragmatische Merkmal.

Abbildungsmerkmal Das Abbildungsmerkmal beschreibt, dass Modelle Abbildungen realer Systeme beziehungsweise natürlicher oder künstlicher Originale sind. Sie können nicht identisch mit dem Original sein.

Verkürzungsmerkmal Modelle erfassen lediglich für den Ersteller und Benutzer relevante Attribute. Somit werden irrelevante Eigenschaften gekürzt.

Pragmatisches Merkmal Modelle ersetzen zeitweise das Original und sind immer zweckgebunden. Sie sind weiterhin in ihrem Funktionsumfang beschränkt und sind für Personen erstellt, die entweder selbst das Modell entworfen haben oder Erfahrung bzw. eine Ausbildung mit dem Umgang von ähnlichen Modellen haben.

Die Klassifikation von Modellen ist in der Literatur oftmals nicht eindeutig und kann anhand verschiedener Kriterien durchgeführt werden (Rabe et al. 2008). Kriterien zur Unterscheidung von Modellen können unter Anderem der Modellzweck, die anzuwendende Analysemethode, der zu modellierende Gegenstand oder die Beschreibungsform des erstellten Modells sein (Rabe et al. 2008). Beispielsweise klassifiziert Peters (1998) nach den fünf Modellzwecken *Beschreibung und Klassifikation*, *Erklärung und Theoriebildung*, *Prognose*, *Bewertung* sowie *Vermittlung und Demonstration*. Auf technischer Ebene wird oftmals eine Unterscheidung nach der anzuwendenden Analysemethode, beispielsweise Optimierungsmodelle und Simulationsmodelle, verwendet.

Modelle weisen grundlegende Eigenschaften auf. Rabe et al. (2008) zählen fünf wichtige Modelleigenschaften auf, die jeweils in zwei Ausprägungen einem Modell zugeordnet werden können.

Zeitverhalten Statisch oder dynamisch. Bei dynamischen Modellen ist das Verhalten von der Zeit abhängig, wohingegen die Ausgangsgrößen statischer Modelle bereits zum gleichen Zeitpunkt wie die Eingangsgrößen feststehen (Wunsch und Schreiber 2005).

Zeitmenge Diskret oder kontinuierlich. Bei diskreter Zeitmenge wird eine endliche Anzahl von äquidistanten Zeitpunkten verwendet. Die kontinuierliche Zeitmenge schließt jegliche Zeitpunkte ein.

Zustandsmenge Diskret oder kontinuierlich. Bei Anwendung einer diskreten Zustandsmenge sind für das Modell eine endliche Anzahl von Zuständen vorgesehen. Bei kontinuierlicher Zustandsmenge ist das Modell nicht auf vordefinierte Zustände beschränkt.

Abbildung von Zufällen Stochastisch oder deterministisch. Bei stochastischen Modellen ist das Verhalten zumindest teilweise stochastisch. Fehlen jegliche Einflüsse des Zufalls wird von deterministischen Modellen gesprochen.

Terminierung Terminieren oder nicht terminierend. Terminierende Systeme weisen definierte Startbedingungen auf und haben natürliche Ereignisse, die einen Zeitabschnitt beenden. Bei nicht terminierenden Systemen fehlen die definierten Startbedingungen und somit können vorangegangene Zeitabschnitte Folgende beeinflussen.

Anhand der Zeitmenge eines zu simulierenden Modells kann eine grundsätzliche Einteilung der Typen von Simulation getroffen werden. Der populärste Typ ist die ereignisdiskrete Simulation, in der die Berechnungen der Simulation nur zu bestimmten Zeitpunkten stattfinden (Law 2015). Der andere Typ wird kontinuierliche Simulation genannt und führt seine Berechnungen in äquidistanten Zeitschritten durch (Law 2015).

Die Erstellung oder Änderung von Modellen wird Modellbildung, Modellierung oder Modellerstellung genannt (Rabe et al. 2008). Wird das Simulationsvorgehensmodell nach Rabe et al. (2008) zugrunde gelegt (vgl. Abschnitt 2.1), sind die fünf Phasen *Aufgabendefinition*, *Systemanalyse*, *Modellformalisierung*, *Implementierung* sowie *Experimente und Analyse* relevant für die Modellbildung. Die eigentliche Modellbildung findet in den Phasen *Systemanalyse*, *Modellformalisierung* und *Implementierung* statt. In der ersten relevanten Phase wird durch die Aufgabenspezifikation das zu entwickelnde Modell an einen bestimmten Zweck gebunden, so wie es das pragmatische Merkmal von Stachowiak (1973) fordert. Weiterhin wird das System hinsichtlich seiner Systemelemente, Beziehungen der Systemelemente, Systemgrenzen und Wechselwirkungen mit der Systemumgebung klar herausgearbeitet. In den drei folgenden Phasen sind die korrespondierenden Phasenergebnisse spezielle Arten eines Modells. In der Projektphase der Systemanalyse wird als Phasenergebnis ein Konzeptmodell erhalten. Das Konzeptmodell fasst die Ergebnisse der Systemanalyse hinsichtlich zu modellierender Elemente, deren Genauigkeit und Systemgrenzen mithilfe textueller Beschreibungen, Ablaufdiagrammen oder Graphiken zusammen. Bei der Modellformalisierung stellt ein formales Modell das Phasenergebnis dar. Das formale Modell konkretisiert das Konzeptmodell mit beispielsweise textueller Beschreibung von Steuerungen in Programmablaufplänen. Das formale Modell sollte, wie auch das Konzeptmodell, unabhängig vom einzusetzenden Simulationswerkzeug sein. Ein ausführbares Modell ist das Phasenergebnis der Implementierung. Das ausführbare Modell wird auch Simulationsmodell genannt und wird so implementiert, wie es das Simulationswerkzeug vorgibt. An

dem ausführbaren Modell können die Experimente ausgeführt werden, die zu den Simulationsergebnissen führen. Attribute, die dem Simulationsmodell bei der Implementierung zugeführt werden, werden Parameter genannt. Der Prozess der Zuweisung von Parametern kann entweder manuell über beispielsweise Eingabemasken in einem Editor oder automatisch über geeignete Schnittstellen erfolgen und wird Parametrisierung genannt (Rabe et al. 2008). Parameter werden auch als Eingangsdaten des Simulationsmodells bezeichnet, die Simulationsergebnisse auch als Ergebnisdaten. Für alle genannten Phasenergebnisse ist es wichtig, eine geeignete Dokumentation zu führen, um eine Nachverfolgbarkeit zu gewährleisten und dritten Personen den Zugang zu den Ergebnissen zu erleichtern. Zudem sollte ständig eine V&V der Teilschritte erfolgen (vgl. Abschnitt 2.1).

Während des Modellierens ist der Modellierer dazu angehalten, das Modell getreu des Leitsatzes „so abstrakt wie möglich und so detailliert wie nötig “(ASIM 1997, S. 7) zu gestalten. Dies impliziert, dass so viele Details wie möglich ausgelassen werden sollen, ohne dem Modellzweck unzuträglich zu sein. Rabe et al. (2008) nennen zwei Methoden, mit denen ein Modellierer ein Modell dementsprechend abstrahieren kann. Zum einen können über die Reduktion Einzelheiten nicht modelliert werden. Zum anderen kann durch Idealisierung die Vereinfachung von Einzelheiten bewerkstelligt werden, wenn sie für den Modellzweck nötig sind. Nach VDI (2014) gibt es zwei Herangehensweisen zum strukturierten Modellieren. Die erste Herangehensweise wird Top-down-Entwurf genannt und besteht darin, das System sukzessive in kleinere Teilsysteme zu zerlegen und somit den Detailgrad der Modellierung über die Zeit zu erhöhen. Der gegensätzliche Ansatz, also das Beginnen mit detaillierten Teilsystemen und anschließender sukzessiver Vereinigung von Teilsystemen, wird nach VDI (2014) Bottom-up-Entwurf genannt. Neben den beiden genannten Herangehensweisen nennen Wenzel und Bernhard (2008) weiterhin den Middle-out-Entwurf als mögliche Alternative. Bei dieser Herangehensweise wird zunächst versucht, die am wichtigsten erscheinenden Teile des Modells zu modellieren und erst später den Rest anzuschließen, unabhängig vom Detaillevel. Rabe et al. (2008) stellen jedoch fest, dass in der Praxis selten eine Herangehensweise strikt befolgt wird, sondern Mischformen entstehen.

Ein weiteres Modellierungskonzept stellt die *agentenbasierte Modellierung* dar. Hierbei besteht das Modell aus einer Umgebung und Agenten, die darin agieren (Andrae und Pobuda 2021). Agenten sind autonome Entitäten, die mit der Umgebung und anderen Agenten in Beziehung treten und basierend darauf Entscheidungen treffen können (Law 2015). Durch die Entscheidungen der Agenten können Veränderungen in der Umgebung oder den Agenten herbeigeführt werden (Andrae und Pobuda 2021). Für weiterführende Informationen zur agentenbasierten Modellierung wird auf Andrae und Pobuda (2021) verwiesen.

Rabe et al. (2008) geben zu bedenken, dass das Modellieren immer ein subjektiver Prozess eines Modellierers ist und somit für dieselbe Modellierungsaufgabe eine Vielzahl von geeigneten Modelllösungen existiert, die auch von dem Modellierer selbst abhängt.

In diesem Abschnitt wurde bereits beschrieben, dass Modelle, bei denen der Zufall eine Rolle spielt, als stochastische Modelle bezeichnet werden. In Simulationsmodellen erfolgt die Modellierung des Zufalls, indem stochastische Größen eines Modells als Zufallsvariable modelliert werden (Gutenschwager et al. 2017). Für Zufallsvariablen wird, bei jeder Benutzung, ein zufälliger Wert verwendet. Der zufällige Wert ergibt sich aus einem Zufallsexperiment, dessen Ablauf durch eine Verteilung spezifiziert wird. Zum Beispiel könnte ein Würfelwurf über eine Zufallsvariable modelliert werden, deren Zufallsexperiment besagt, dass die Ereignisse *1*, *2*, *3*, *4*, *5* und *6* jeweils mit gleicher Wahrscheinlichkeit eintreffen. Bei der Ausführung des Simulationsmodells würde dann bei jeder Verwendung der

Zufallsvariable eines der vorgegebenen Ergebnisse, zu gleicher Wahrscheinlichkeit, zurückgegeben. Eine Zufallsvariable, die eine endliche Menge von Ereignissen (wie im Beispiel des Würfelwurfs) oder abzählbar endlich viele Zustände abbildet, wird diskrete Zufallsvariable genannt. Andere Zufallsvariablen, deren Menge von Ergebnissen nicht abzählbar ist, sind stetige Zufallsvariablen (Banks et al. 2014). Die Verteilung, wie wahrscheinlich ein Ereignis eintritt, wird über eine Verteilungsfunktion realisiert. Im Beispiel des Würfelwurfs, in dem alle Ereignisse mit gleicher Wahrscheinlichkeit auftreten, wird eine so genannte Gleichverteilung verwendet. Für diskrete und stetige Zufallsvariablen werden unterschiedliche Verteilungsfunktionen verwendet. Für detaillierte Informationen bezüglich Zufallszahlen und Verteilungsfunktionen wird auf Gutenschwager et al. (2017) oder Banks et al. (2014) verwiesen.

Zufallszahlen, die durch einen Computer generiert werden, können lediglich mithilfe deterministische Rechenvorschriften berechnet werden, die möglichst zufällig und entsprechend der definierten Verteilung wirken (Gutenschwager et al. 2017). Daher wird oftmals von Pseudo-Zufallszahlen gesprochen. Die Rechenvorschriften sind in der Lage, Folgen von Zufallszahlen zu berechnen, die als Zufallszahlenströme bezeichnet werden (Law 2015). Damit nicht immer dieselbe Folge von Zufallszahlen in einer Zufallsvariable auftreten, werden so genannte Startwerte, auch Seeds genannt, verwendet (Gutenschwager et al. 2017). Mit Ihnen kann eine Startposition in der Folge der Zufallszahlen gewählt werden. Eine nützliche Anwendung von Startwerten ergibt sich daraus, dass die Ergebnisse einer stochastischen Simulation, bei gleichem Startwert, reproduzierbar sind (Law 2015).

Für eine Simulationsstudie werden für gewöhnlich mehrere Simulationen mit unterschiedlichen Startwerten durchgeführt, um das Modell unter Einfluss verschiedener Zufallszahlen zu untersuchen. Neben den Startwerten werden jedoch auch andere Parameter variiert. Die strukturierte Planung der Untersuchung eines Simulationsmodells ist Teil der statistischen Versuchsplanung, die im anschließenden Abschnitt behandelt wird.

2.1.2 Statistische Versuchsplanung

Eine Simulationsstudie umfasst in der Regel nicht nur eine Simulation, sondern mehrere Simulationen mit variierenden Parametern und Startwerten. Die einmalige Simulation eines experimentierbaren Modells wird Simulationslauf genannt (Gutenschwager et al. 2017). Die benötigte Zeit zur Durchführung des Simulationslaufs heißt Rechenzeit (Gutenschwager et al. 2017). Ein Simulationslauf ist mit einer Parameterkonfiguration und einem Startwert assoziiert. Mit dem Startwert werden den stochastischen Algorithmen der Simulation Zufallszahlenströme zugewiesen (vgl. Abschnitt 2.1.1). Werden verschiedene Startwerte verwendet, wird sichergestellt, dass die von der Simulation verwendeten Zufallszahlen nicht deckungsgleich sind. Bei gleichbleibender Parameterkonfiguration und gleichem Startwert sind auch die Ergebnisse der Simulationsläufe gleichbleibend. Von einem Simulationsexperiment wird erst dann gesprochen, wenn eine gezielte und strukturierte empirische Untersuchung des Modellverhaltens anhand mehrerer Simulationsläufe durchgeführt wird (VDI 2014). Die Planung dieser gezielten und strukturierten empirischen Untersuchung findet in der statistischen Versuchsplanung statt. Für gewöhnlich werden für eine gewählte Parameterkonfiguration mehrere Simulationsläufe mit unterschiedlichen Startwerten durchgeführt. Solche Simulationsläufe, die gleichbleibende Parameterkonfigurationen aber unterschiedliche Startwerte aufweisen, werden Replikationen genannt (Rabe et al. 2008).

Die einzelnen Parameter der Parameterkonfiguration werden auch als Faktoren bezeichnet und können entweder qualitativ oder quantitativ sein (Gutenschwager et al. 2017). Zur Untersuchung verschiedener Parameterkonfigurationen in einem Simulationsexperiment, sollte strukturiert nach einem Experimentplan, auch Versuchsplan genannt, vorgegangen werden (Box et al. 2005). Der Experimentplan beinhaltet für jeden Simulationslauf eine zu untersuchende Parameterkonfiguration. Durch die strukturiere Vorgehensweise wird die Analyse von Ursache-Wirkungs-Beziehungen zwischen den Faktoren und den Ergebnissen ermöglicht. Ziel ist es, eine möglichst identische Abbildung der Antwortoberfläche, also der Ursache-Wirkungs-Beziehungen, des Simulationsmodells zu erhalten (Box et al. 2005). Damit ein Simulationsexperiment als erfüllt gilt, müssen alle vorgesehenen Simulationsläufe mit den entsprechenden Parameterkonfigurationen simuliert werden. Es ist zu beachten, dass mit steigender Anzahl von Faktoren der Umfang des Experimentplans typischerweise exponentiell anwächst und somit auch die Dauer des Simulationsexperiments stark erhöht wird (Siebertz et al. 2017).

Dem Anwender stehen eine Vielzahl von Arten von Experimentplänen zur Verfügung. Ein einfach zu konstruierender Experimentplan ist der vollfaktorielle Experimentplan, in dem alle möglichen Kombinationen von Stufen der Faktoren untersucht werden (Box et al. 2005). Als Stufen werden Werteniveaus bezeichnet, die einen konkret zu untersuchenden Wert beschreiben. Nach Siebertz et al. (2017) lässt sich die Anzahl der nötigen Simulationsläufe n_r aus der Zahl der Faktoren n_f und der Zahl der Stufen n_l über die folgende Gleichung 2.1 beschreiben:

$$n_r = n_l^{n_f} \tag{2.1}$$

Als Beispiel wird anhand dieser Gleichung die Anzahl der nötigen Simulationsläufe für ein Experiment mit 15 Faktoren und jeweils zwei Stufen berechnet. Das Ergebnis der Berechnung ist, dass 32.768 Simulationsdurchläufe nötig sind, um den Experimentplan vollkommen zu bearbeiten. Wird eine Rechenzeit von einer Minute pro Simulationslauf veranschlagt, wird bei sukzessiver Simulation aller Simulationsläufe eine Gesamtrechenzeit für das Simulationsexperiment von knapp 23 Tagen benötigt. Die Rechenzeit für einen Simulationslauf in der Logistik kann jedoch noch länger ausfallen, wie beispielsweise in Heger und Hildebrandt (2015) beschrieben. Auch ist die Anzahl der Faktoren und Stufen oftmals umfangreicher, sodass ein vollfaktorieller Experimentplan in vielen Fällen keine annehmbare Option ist (Gutenschwager et al. 2017). Alternativ werden teilfaktorielle Experimentpläne genutzt, die nur eine Untermenge der vollfaktoriellen Pläne untersucht, jedoch gelten hier dieselben Einschränkungen hinsichtlich des beabsichtigten Anwendungsbereichs der Logistik (Law 2015).

Andere Ansätze zur Konstruktion von Experimentplänen basieren auf mathematischen und statistischen Berechnungen, die versuchen, den funktionalen Zusammenhang von Faktoren im Vorfeld zu berechnen. Ziel ist es, eine möglichst gute Untersuchung des Modellverhaltens bei möglichst niedriger Anzahl von benötigten Simulationsläufen zu erreichen. Dazu gehören beispielsweise MiniMax bzw. MaxiMin (Johnson et al. 1990) und orthogonale Felder (Siebertz et al. 2017). Zu den effektivsten Experimentplänen gehört der Nearly Orthogonal Latin Hypercube, der beispielsweise für 29 Faktoren lediglich 257 Simulationsläufe ausführt und trotzdem zuverlässige Ergebnisse liefert (Cioppa und Lucas 2007; Sanchez 2018).

Zur Erstellung von Experimentplänen kann auf zahlreiche verfügbare Werkzeuge zurückgegriffen werden (Siebertz et al. 2017). Somit muss der Anwender sich nicht mit den oft-

mals komplizierten Berechnungsvorschriften auseinandersetzen. Die Werkzeuge sind zumeist nicht direkt in einen Simulator eingebunden.

Zwar ist die Konstruktion eines Experimentplans meistens kein Teil eines Simulators, die automatisierte Bearbeitung eines Experimentplans jedoch schon. Weitere Funktionalitäten und grundlegende Informationen zu Simulatoren werden im nächsten Abschnitt dargelegt.

2.1.3 Simulatoren

Die Durchführung einer Simulationsstudie kann mittels eines Simulators unterstützt werden. Durch die Verwendung eines für den Modellzweck geeigneten Simulators kann die benötigte Zeit zur Durchführung einer Simulationsstudie erheblich gesenkt werden, sodass sich das Kosten-Nutzen-Verhältnis der Simulation Abschnitt 2.1 positiv in Richtung des Nutzens verschiebt.

Alternative Bezeichnungen für Simulatoren sind Simulationswerkzeuge, Simulationsinstrumente, Simulationsprogramme, Simulationstools oder Simulationssysteme (Gutenschwager et al. 2017). Der VDI (2014, S. 4) definiert Simulatoren als Werkzeuge, „mit dem ein Modell zur Nachbildung des dynamischen Verhaltens eines Systems und seiner Prozesse erstellt und ausführbar gemacht werden kann“. Gutenschwager et al. (2017) zählen neben der Unterstützung bei der Modellerstellung und der Durchführung der Experimente auch die Analyse der Experimentergebnisse zu den gängigen Aufgaben eines Simulators.

Eine Einteilung von Simulatoren kann laut Noche und Wenzel (1991) hinsichtlich der Allgemeingültigkeit der Anwendung erfolgen. Sie definieren dazu fünf Ebenen:

Ebene 0 Programmiersprachen ohne Erweiterungen oder Anpassungen für simulationsspezifische Bedarfe.

Ebene 1 Programmiersprachen mit Basiskonzepten zur leichteren Modellierung von dynamischen Vorgängen.

Ebene 2 Simulationswerkzeuge mit bereits implementierten allgemeinen Komponenten für beliebige Anwendungen.

Ebene 3 Simulationswerkzeuge mit spezifischen Komponenten für einen Anwendungsbereich (z.B. Produktion und Logistik).

Ebene 4 Spezialsimulationswerkzeuge, die spezifische Komponenten für ein abgegrenztes Teilgebiet eines Anwendungsbereiches enthalten.

Die Ebenen sind in absteigender Reihenfolge nach der Allgemeingültigkeit sortiert, wobei *Ebene 0* die höchste Allgemeingültigkeit aufweist. *Ebene 4* ist somit die spezifischste Ebene. Für gewöhnlich ist die Anwendung eines möglichst spezifischen Simulators für den zu untersuchenden Modellzweck zu empfehlen, da in der Regel eine größere Geschwindigkeit bei der Modellerstellung, als auch eine höhere Güte der Ergebnisse zu erwarten sind (Gutenschwager et al. 2017).

In der Einteilung von Noche und Wenzel (1991) wird von *Komponenten* eines Simulators gesprochen. Die Hauptkomponenten eines Simulators sind laut VDI (2014) die folgenden vier Komponenten:

- Simulatorkern
- Datenverwaltung

- Benutzeroberfläche
- Schnittstelle zu externen Programmen

Für die Bereitstellung und Verarbeitung des Simulationsmodells ist der Simulatorkern zuständig. Er führt die Simulationsläufe durch und steuert die stochastischen Einflüsse (Wenzel 2018). Der Simulationskern gilt als die zentrale Komponente eines Simulators und ist für die Verknüpfung und Koordinierung der anderen Komponenten zuständig (VDI 2014).

Die Datenverwaltung ist für die Verwaltung aller Eingabe-, Experiment- und Simulationsergebnisdaten, sowie die internen Modelldaten zuständig (VDI 2014). Eingabedaten beschreiben die Daten, die der Nutzer des Simulators für das Simulationsmodell spezifiziert. Zu den Experimentdaten gehören Daten, die die Simulationsläufe in Dauer oder Startzeitpunkten beschreiben. Die internen Modelldaten sind Daten, die während der Simulation durch den Simulatorkern berechnet werden. Die Simulationergebnisdaten sind Daten, die während eines Simulationslaufs gesammelt werden.

Durch die Benutzeroberfläche erhält der Nutzer die Möglichkeit mit dem Simulator zu interagieren. Damit wird die Modellerstellung, die Parametrisierung des Modells, die Durchführung der Experimente und das Anzeigen sowie Analysieren der Experimente ermöglicht (VDI 2014). Die Möglichkeit zur interaktiven Modellerstellung mithilfe des Simulators hilft dem Nutzer, das durch den Simulator vorgesehene Konzeptmodell mit einem validen, formalen Modell umzusetzen (Gutenschwager et al. 2017).

Schnittstellen zu externen Programmen erlauben die Automatisierung von Prozessen, in denen der Simulator eingebunden ist (VDI 2014). Dazu gehören der Austausch von Datenbeständen, beispielsweise der Simulationsergebnisdaten, oder die Einbindung von Eingabedaten (VDI 2014). Da die Fähigkeit zur Generierung von Experimentplänen oftmals nicht Teil eines Simulators ist, kann über Schnittstellen ein Import von extern generierten Experimentplänen erfolgen.

Neben den vier definierten Hauptkomponenten von Simulatoren des VDI (2014) zählt Wenzel (2018) auch eine vordefinierte Modellwelt zu den Hauptkomponenten von Simulatoren. Die vordefinierte Modellwelt ergibt sich aus dem Modellierungskonzept eines Simulators. Wenzel (2018) beschreibt Modellierungskonzepte als Regelwerke zur Strukturierung und Modelleriung der zu untersuchenden Systeme, in denen die Modellbestandteile sowie deren Abhängigkeiten festgelegt werden. Eine Betrachtung von Modellierungskonzepten in der Logistik folgt im Verlauf dieser Arbeit.

Neben den aufgezeigten Hauptkomponenten sind Simulatoren oftmals in der Lage, ausgewählte V&V-Techniken umzusetzen. Beispielsweise ist die Animation eine häufige V&V-Technik, die ein Simulator anwendet, um dem Nutzer die Kontrolle des formalen Modells zu ermöglichen. Oftmals werden werkzeugspezifische V&V-Techniken angeboten, die aufgrund von Modellbesonderheiten erforderlich sind. Ein Beispiel für eine solche werkzeugspezifische V&V-Technik ist das Überprüfen, ob ein Wegenetz vollständig und somit jeder Ort des Netzes erreichbar ist.

Eine weitere Funktion, die Simulatoren bieten können, ist die parallele Simulation. Hierbei können mehrere Simulationen auf einem Computer parallel ablaufen, was auf Computern mit Mehrkern-Prozessoren zu einer besseren Auslastung der Computer Rechenleistung führen kann (Fujimoto 2015). Wird die Simulation nicht nur auf einem Computer parallel ausgeführt, sondern auf mindestens einem weiteren, wird von verteilter Simulation

gesprochen (Gutenschwager et al. 2017). Hierdurch kann die Bearbeitung des Simulationsexperiments weiter beschleunigt werden.

Es gibt eine Vielzahl von Simulatoren, die kommerziell und nicht-kommerziell angeboten werden. Sie unterscheiden sich oftmals hinsichtlich der Anwendungsbereiche, der Modellierungskonzepte oder der Granularität der Untersuchung. Dazu gehören beispielsweise DOSIMIS-3 (SDZ GmbH 2022), PlantSimulation (Bangsow 2020), AnyLogistix (The AnyLogic Company 2022b) und Arena (Rockwell Automation 2022). Zur Auswahl eines geeigneten Simulators aus dem großen Angebot der Simulatoren im Bereich der Produktion und Logistik sollten Auswahlkriterien herangezogen werden. Gutenschwager et al. (2017) zeigen fünf Kategorien von Kriterien auf, zu denen jeweils einige Beispiele genannt werden.

Allgemeine und kaufmännische Kriterien Entwicklungsgeschichte, Marktverbreitung, Lizenzpolitik und -kosten, Wartung und Support.

Ergonomische Kriterien Intuitive Bedienbarkeit, Transparenz, leichte Erlernbarkeit oder Vorhandensein von kontextbezogenen Online-Hilfefunktionen.

Funktionale Kriterien Bereitstellung von Bibliotheken, 3D-Animation, Integration von V&V-Techniken.

Kriterien zur Bewertung der Ergebnisausgabe Qualität und Verständlichkeit der Ergebnisstatistiken, Individualisierbarkeit der Berichte, Schnittstellen zur Weiterverarbeitung.

Kriterien zur Bewertung der Softwareleistung Flexibilität, Robustheit, Ausführgeschwindigkeit, Kompatibilität mit anderen Anwendungen.

In den Kriterien nach Gutenschwager et al. (2017) spielt die Softwareleistung eine Rolle, die unter Anderem durch die Kompatibilität mit anderen Anwendungen beschrieben wird. Der in dieser Arbeit zu entwickelnde Simulator soll eine Kompatibilität mit Graphdatenbanken aufweisen. Bevor diese im weiteren Verlauf der Arbeit vorgestellt werden wird zunächst im nächsten Abschnitt ein Grundverständnis der mathematischen Grundlagen vermittelt.

2.2 Graphen

Für eine strukturierte und algorithmische Untersuchung von Netzwerken bietet sich eine mathematische Beschreibung an. Ein mathematisches Beschreibungsmittel von Netzwerken und netzwerkartigen Strukturen sind Graphen, die durch die Graphentheorie definiert werden (Aigner 2015). Zum Beispiel werden Straßennetze, soziale Beziehungen oder chemische Verbindungen oftmals mit Graphen modelliert (Tittmann 2021). Netzwerkartige Strukturen sind durch das Vorhandensein einer Menge von Objekten und einer Menge von Beziehungen zwischen den Objekten charakterisiert. Analog dazu wird in der Graphentheorie von Knoten und Kanten gesprochen. Knoten entsprechen den Objekten der Netzwerke und können Orte oder Gegenstände darstellen. Kanten entsprechen den Beziehungen in Netzwerken. Am Beispiel des Straßennetzes können Knoten beispielsweise Kreuzungen, Abzweigungen oder Orte repräsentieren und die dazwischenliegenden Straßen würden durch Kanten dargestellt werden. Mathematisch ausgedrückt wird ein Graph G, bestehend aus einer Knotenmenge V und einer Kantenmenge E, mit $G{=}(V, E)$ definiert. Sind die Mengen V und E endlich, so ist auch der zugehörige Graph endlich (Tittmann 2021). Besteht ein Graph aus einer Teilmenge von Knoten und Kanten eines übergeordneten Graphen wird von einem Teilgraphen gesprochen (Krischke und Röpcke 2015).

Liegt ein Knoten v an einer Kante e, wird davon gesprochen, dass v inzident zu e ist. Zwei Knoten, die durch eine Kante verbunden sind, oder aber auch zwei Kanten, die zu demselben Knoten inzident sind, werden als adjazent bezeichnet (Krischke und Röpcke 2015). Existieren in einem Graphen für mindestens ein Knotenpaar mehr als eine verbindende Kante, wird der Graph als Multigraph bezeichnet. Graphen ohne Mehrfachkanten werden einfache Graphen genannt (Krischke und Röpcke 2015).

Um die Richtung einer Beziehung in einem Graphen auszudrücken, können die Kanten eines Graphen gerichtet sein. In diesem Fall haben sie einen Start- und Endknoten. Graphen, die gerichtete Kanten aufweisen, werden als gerichtete Graphen bezeichnet. Im Beispiel des Straßennetzes kann eine gerichtete Kante zur Modellierung einer Einbahnstraße verwendet werden. Ungerichtete Kanten sind bidirektional und haben keine definierten Start- oder Endknoten. Es ist für eine Kante zudem möglich, denselben Knoten als Start- und Endknoten aufzuweisen. In diesem Fall wird von einer *Schlinge* gesprochen. Existieren zwischen allen Knoten Kanten, ist der Graph vollständig (Tittmann 2021).

Der Grad eines Knoten spiegelt die Anzahl der ausgehenden Kanten des Knotens wider. So hat ein Knoten, der fünf ausgehende Kanten besitzt, einen Grad von Fünf. Knoten, die einen im Verhältnis zu anderen Knoten hohen Grad aufweisen, werden in ungerichteten Graphen Hubs genannt. In gerichteten Graphen werden solche Knoten Hubs genannt, die eine hohe Zahl ausgehender Kanten haben. Dahingegen werden Knoten, die der Endknoten vieler Kanten sind, als *Authorities* bezeichnet (Krischke und Röpcke 2015).

Kanten können eine Kantengewichtung, auch Kantenbewertung genannt, haben. Eine Kantengewichtung besteht aus einer reellen Zahl und kann den Informationsgehalt des Graphen steigern. Im Beispiel des Straßennetzwerks kann die Kantengewichtung als Länge der Straßensegmente verwendet werden oder als Geschwindigkeitsbegrenzung. Auch Knoten können eine Bewertung mit einer reellen Zahl erhalten (Wenger 2009). Die folgende Abbildung 2.2 stellt einen beispielhaften Graphen visuell dar.

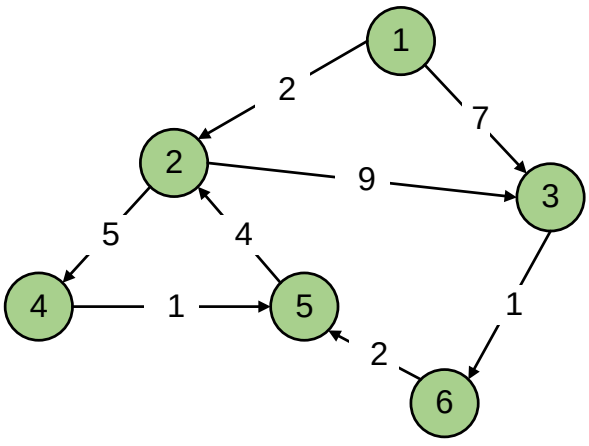


Abbildung 2.2: Beispiel eines Graphen nach Krischke und Röpcke (2015, S. 31)

Der in Abbildung 2.2 abgebildete Graph kann als einfacher und endlicher Graph identifiziert werden. Die verwendeten Pfeile und Zahlen an den Kanten lassen zudem darauf schließen, dass der Graph gerichtet und zudem gewichtet ist.

Graphen können dazu verwendet werden, strukturierte Daten und ihre netzartige Struktur zu beschreiben. Im nächsten Abschnitt wird zunächst auf den Begriff *Daten* eingegangen,

bevor im Anschluss Möglichkeiten zur Modellierung von Daten, unter anderem mithilfe von Graphen, erläutert werden.

2.3 Daten

Eine in der Fachliteratur weit verbreitete Definition und Abgrenzung des Begriffs *Daten* (Scheidler 2017) erfolgt anhand der Wissenstreppe von North (2016). In Abbildung 2.3 ist ein Auszug der Wissenstreppe dargestellt.

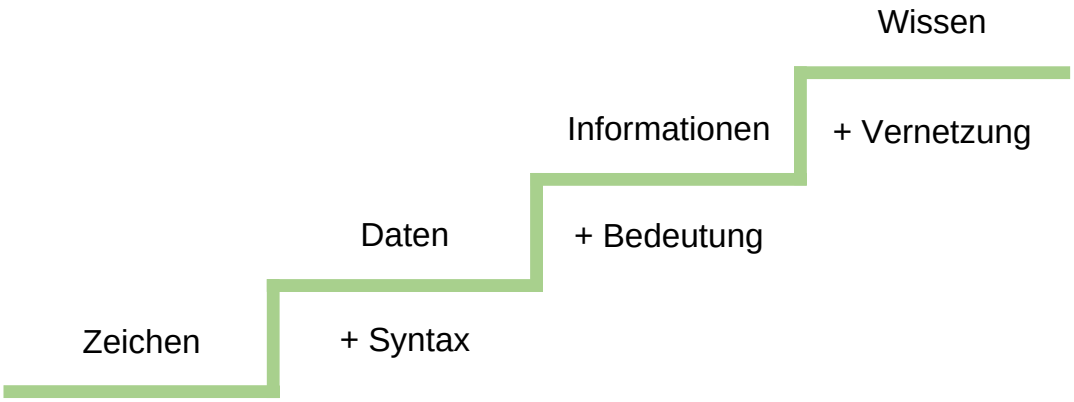


Abbildung 2.3: Auszug der Wissenstreppe nach North (2016, S. 37)

Der Abbildung 2.3 ist zu entnehmen, dass Daten zum einen gegenüber Zeichen abgegrenzt werden. Zeichen sind lediglich einzelne Buchstaben, Ziffern oder Sonderzeichen und stellen die elementarste Stufe der Treppe dar. Erst durch die Zunahme einer Ordnungsregel, auch Syntax genannt, werden aus einzelnen Zeichen Daten. Zum anderen wird der Unterschied von Daten zu Informationen aufgezeigt, der darin besteht, dass Informationen zusätzlich mit einer Bedeutung versehen sind. Die Vernetzung von Informationen lässt laut North (2016) Wissen entstehen. Übereinstimmende Definitionen lassen sich beispielsweise bei Krcmar (2015) und Probst et al. (2012) finden. Bodendorf (2006) merkt jedoch an, dass möglicherweise keine scharfe Grenze zwischen Daten und Informationen existiert, da Daten nicht vollkommen Kontextunabhängig sind.

Daten können nach Cleve und Lämmel (2020) drei Datentypen zugeordnet werden. Die Datentypen benennen und erklären sie wie folgt:

Nominale Daten Ausprägungen können nicht in eine Reihenfolge gebracht werden und nicht für Rechnungen verwendet werden. Es ist lediglich ein Vergleich möglich, ob die Ausprägungen identisch sind. Beispiele: schwarz, weiß, rot.

Ordinale Daten Ausprägungen können in eine Reihenfolge gebracht werden, eignen sich jedoch nicht zum Rechnen. Beispiele: klein, mittel, groß.

Metrische Daten Ausprägungen sind Zahlen und können in eine Reihenfolge gebracht werden. Auch ist es möglich mit ihnen zu rechnen. Beispiele: *1*, *2*, *3*.

Zudem können laut Krcmar (2015) Datensätze entweder strukturiert, semistrukturiert oder unstrukturiert sein. Zu strukturierten Daten gehören beispielsweise Transaktionsdaten, die in Geschäftsprozessen anfallen, wie Rechnungen, Lieferscheine oder Angebote (Krcmar

2015). Semistrukturierte Daten sind dort anzutreffen, wo kein festes Schema von Daten existiert, wie beispielsweise in der *Extensible Markup Language* (Meier und Kaufmann 2016). Zu unstrukturierten Daten können beispielsweise E-Mails oder Dokumente gehören (Krcmar 2015).

Die Menge der anfallenden Daten, insbesondere auch im Bereich der Logistik, hat in den letzten Jahren stark zugenommen (Huber und Kaiser 2017). Treiber dieser Entwicklung ist unter anderem die Industrie 4.0 (Andelfinger und Hänisch 2017). In diesem Zusammenhang werden große Datenbestände oftmals als *Big Data* bezeichnet. Die Interpretation, ab wann ein Datenbestand als Big Data bezeichnet werden kann, variiert in der Literatur und ist nicht klar definiert. Oftmals wird eine Definition über die drei V angeführt (Laney 2001). Zu den drei V gehören Volume (Menge), Velocity (Geschwindigkeit) und Variety (Vielfalt). Volume bezieht sich lediglich auf die anfallende Masse von Daten. Eine große Masse von Daten bedeutet also nicht, dass die dazugehörige Datenbasis als Big Data bezeichnet werden kann. Meier und Kaufmann (2016) geben als Schwellenwert Daten im Terabytebereich an. Velocity bedeutet, dass die Datenströme in Echtzeit ausgewertet und analysiert werden können (Meier und Kaufmann 2016). Variety bezeichnet das Vorhandensein von heterogenen Daten- und Dateiformaten, die strukturiert, semistrukturiert oder unstrukturiert sein können (Meier und Kaufmann 2016). Andere Autoren definieren den Begriff Big Data über fünf V. Bei den fünf V werden zusätzlich zu den drei V die Begriffe Value (Wert) und Veracity (Richtigkeit) hinzugenommen, um auf die schwankende Qualität und den schwankenden Wissensgehalt hinzuweisen (Meier und Kaufmann 2016). In der Literatur befinden zudem weitere Varianten der Definition von Big Data (Sanchez 2018), auf die aber nicht weiter eingegangen wird.

In den letzten Jahren wurde der Stellenwert von Daten in Unternehmen überdacht. Als Ergebnis werden Daten nun als wichtige Ressourcen gesehen, die, wie alle anderen Ressourcen im Unternehmen auch, geplant, gesteuert, überwacht und kontrolliert werden müssen (Meier und Kaufmann 2016). Daraus ergibt sich die Forderung nach einem unternehmensweiten Datenmanagement, um die Akquirierung, Nutzung und Kontrolle der Daten sicherzustellen und daraus eine Wertschöpfung zu generieren (Otto und Österle 2016).

Ein wichtiger Bestandteil des Umgangs mit Daten ist deren Speicherung. Dies geschieht für gewöhnlich in Datenbankssystemen, welche im anschließenden Abschnitt behandelt werden.

2.3.1 Datenbanksysteme

Zur Speicherung von Daten werden Datenbanksysteme verwendet. Meier und Kaufmann (2016, S. 2) bezeichnen sie als „eine Software zur applikationsunabhängigen Beschreibung, Speicherung und Abfrage von Daten“, die aus einer Speicherungs- und einer Verwaltungskomponente bestehen. Die Speicherungskomponente ist für das Persistieren der Daten und deren Beschreibung zuständig. Die gespeicherten Daten werden auch als Datenbasis bezeichnet (Kemper und Eickler 2015). Die Verwaltungskomponente verwaltet Zugriffs- und Bearbeitungsrechte und dient als Schnittstelle zum Anwender, der über eine Abfrage- und Manipulationssprache auf die gespeicherten Daten zugreifen kann. Aufgaben eines Datenbanksystems sind es, eine konsistente Datenbasis zu ermöglichen und diese vor unrechtmäßigen Zugriffen zu schützen (Bodendorf 2006).

Die Struktur der zu persistierenden Daten wird in dem so genannten Datenbankschema festgelegt. Der Zustand einer Datenbasis, wenn also konkrete Daten darin gespeichert

sind, wird als Datenbankausprägung bezeichnet. Das Datenbankschema einer Anwendung wird für gewöhnlich nicht verändert, die Datenbankausprägung hingegen ist konstanten Änderungen ausgesetzt (Kemper und Eickler 2015).

Ein wichtiges Konzept vieler Datenbanksysteme ist das Transaktionskonzept, das oft mit der Abkürzung Atomarität, Konsistenz, Isolation, Dauerhaftigkeit (ACID) zusammengefasst wird (Kemper und Eickler 2015). Das Konzept besagt, dass die Kommunikation mit einem Datenbanksystem in Transaktionen stattfindet. Nach Kemper und Eickler (2015) beschreibt die Abkürzung ACID die vier Eigenschaften, die eine Transaktion haben sollte:

Atomicity (Atomarität) Änderungen an der Datenbasis können allein durch Transaktionen durchgeführt werden.

Consistency (Konsistenz) Der Zustand vor und nach einer Transaktion muss zu jeder Zeit eine konsistente Datenbasis aufweisen. Führt eine Transaktion zu einem abweichenden Zustand, wird die Transaktion rückgängig gemacht, sodass der Initialzustand wiederhergestellt wird.

Isolation (Isolation) Transaktionen, die parallel ausgeführt werden, dürfen sich nicht gegenseitig beeinflussen.

Durability (Dauerhaftigkeit) Es muss sichergestellt werden, dass die Wirkung einer Transaktion dauerhaft in dem Datenbanksystem erhalten bleibt.

Eine Einteilung von Datenbanksystemen erfolgt in der Regel anhand der zugrunde liegenden Datenmodelle, die festlegen, welche Struktur die Daten haben und welche Operationen ausgeführt werden können (Kemper und Eickler 2015). In der Praxis werden die zwei Gruppen der SQL- und NoSQL-Datenbanksysteme unterschieden (Meier und Kaufmann 2016). Heutzutage werden zumeist SQL-Datenbanksysteme verwendet, jedoch finden NoSQL-Datenbanken zunehmend Anwendung (Meier und Kaufmann 2016). Die Gründe dafür sind in unterschiedlichen Stärken und Schwächen der Datenmodelle begründet. Zudem kann es sich als vorteilhaft erweisen, wenn das Datenmodell des Datenbanksystems entsprechend der Anwendung ausgewählt wird, mit dem es zusammen verwendet werden soll. Beispielsweise haben Hunker et al. (2020) in einem Experiment gezeigt, dass die Verwendung eines Graphdatenbanksystems in Verbindung mit einem Data-Mining-Tool vorteilhaft ist, da das Tool intern mit Graphstrukturen arbeitet und somit keine unnötigen Transformationen durchgeführt werden müssen, was zu einer höheren Leistungsfähigkeit des gesamten Systems führte. Im weiteren Verlauf des Abschnitts werden beispielhafte Datenmodelle vorgestellt.

SQL-Datenbanken basieren auf dem Relationenmodell. Dieses Modell verwendet Tabellen, in denen die zu speichernden Daten den Zeilen der Tabelle entsprechen. Über Referenzen können Verknüpfungen von Daten in verschiedenen Tabellen realisiert werden. Anhand der spezifizierten Referenz kann zwischen den beteiligten Tabellen navigiert werden (Kemper und Eickler 2015). Die Abfrage- und Manipulationssprache der SQL-Datenbanken ist die namensgebende und genormte Sprache SQL (Meier und Kaufmann 2016). Der Anwender kann Abfrage- und Manipulationssprachen verwenden, um Daten in die Datenbasis zu schreiben, sie auszulesen oder zu verändern (Kemper und Eickler 2015). Beispiele für SQL-Datenbanksysteme sind MySQL (Oracle Corporation 2022a), MariaDB (MariaDB Foundation 2022) oder Oracle Database (Oracle Corporation 2022b).

NoSQL-Datenbanksysteme sind eine Gruppe von Datenbanksystemen, die auf unterschiedlichen Datenmodellen basieren, die nicht-relational sind. Zu den NoSQL-Datenbanksystemen zählen Schlüssel-Wert-Datenbanksysteme, Spaltenfamilien-Datenbanksysteme, Dokument-

Datenbanksysteme, XML-Datenbanksysteme sowie Graphdatenbanksysteme (Meier und Kaufmann 2016). Weiterhin werden sie durch die Verwendung einer Abfrage- und Manipulationssprache, die nicht SQL ist, charakterisiert.

Als vorteilhaft erweisen sich NoSQL-Datenbanken bei großen Datenbeständen (vgl. Abschnitt 2.3), insbesondere wenn diese starke Vernetzungen untereinander aufweisen (Hecht und Jablonski 2011). SQL-Datenbanken hingegen weisen Schwächen im Umgang mit großen Datenbeständen auf, Kemper und Eickler (2015, S. 401) sprechen sogar von „Unzulänglichkeiten des relationalen Datenmodells für komplexere Anwendungsbereiche“. Ein beispielhaftes Experiment wurde von Partner und Vukotic und Watt (2015) durchgeführt. Die Aufgabenstellung war das Finden von Freundes-Freunden in einem sozialen Netzwerk, dass aus einer Millionen Personen mit durchschnittlich 50 Freunden bestand. Während die untersuchte relationale Datenbank bei einer Suchtiefe von vier für eine Anfrage, die etwa 600.000 Ergebnisse lieferte, knapp 28 Minuten benötigte, konnte ein Graphdatenbanksystem bereits nach 1.4 Sekunden die gesuchten Ergebnisse liefern. Eine Erklärung für die unterschiedlichen Ergebnisse liegt darin begründet, wie die Verknüpfung der Daten erfolgt und wie schnell zwischen den Daten navigiert werden kann. Insbesondere Graphdatenbanksysteme können das Navigieren zwischen Daten erleichtern, da die Daten direkte Verweise auf andere verknüpfte Daten speichern (Cook und Holder 2007). Da SQL-Datenbanken keine optimale Wahl für die Datenhaltung von komplexen Anwendungen mit Datenbeständen, die eine starke Vernetzung aufweisen, darstellen, wird im weiteren Verlauf dieses Abschnitts auf Graphdatenbanksysteme als Unterart der NoSQL-Datenbanken eingegangen, die sich insbesondere für die Speicherung von netzartigen Strukturen eignen (Meier und Kaufmann 2016).

Graphdatenbanksysteme verwenden das Graphenmodell als Datenmodell. Das bedeutet, dass ihre Daten in einer Graphstruktur gespeichert werden und somit selbst einem Netz ähneln (Gosnell und Broecheler 2020). Nach (Robinson et al. 2015) kann das Graphenmodell mit drei spezifischen Modellen umgesetzt werden. Dazu gehören das Hypergraphen-Modell, Tripel-Graphen-Modell und Eigenschaftsgraphen-Modell. Hypergraphen zeichnen sich durch die Verwendung von Beziehungen aus, die eine beliebige Anzahl von Start- und Endknoten aufweisen (Robinson et al. 2015). Das Tripel-Graphen-Modell, oft auch Resource-Description-Framework-Modell genannt, werden hauptsächlich in der Domäne der semantischen Netze verwendet und sind durch die W3C genormt (Robinson et al. 2015). Ein Tripel stellt eine Aussage dar und besteht aus Subjekt, Prädikat und Objekt. Die Bestandteile des Tripels können einzelne Literale sein oder Ressourcen bzw. Verweise wo die Ressourcen zu finden sind wie beispielsweise Internetseiten (Robinson et al. 2015). Das berühmteste der Graphenmodelle ist das Eigenschaftsgraph-Modell (Robinson et al. 2015). Es bedient sich solcher Graphen, deren Knoten und Kanten einen Namen Tragen und darüber hinaus Eigenschaften besitzen können. Die Eigenschaften bestehen aus Attribut-Wert-Paaren, die sich aus einem Attributnamen und einem zugehörigen Wert bilden lassen (Meier und Kaufmann 2016). Ein beispielhafter Eigenschaftsgraph ist in Abbildung 2.4 dargestellt.

Abbildung 2.4 zeigt die Modellierung eines Fahrzeugs, das mit einer Palette beladen ist. Das Fahrzeug und die Palette werden jeweils als Knoten modelliert. Die Relation zwischen beiden wird mit einer Kante dargestellt, die in der Abbildung als schwarzer Pfeil dargestellt ist. Die Kante trägt den Titel *beladen mit*, um eine Beschreibung zu liefern. Zusätzlich ist die Kante mit einer Eigenschaft versehen, die das Datum modellieren soll, an dem die Palette eingeladen wurde. Das Fahrzeug weist eine Kapazität von zehn Kubikmetern auf und die Palette nimmt ein Volumen von zwei Kubikmetern ein.

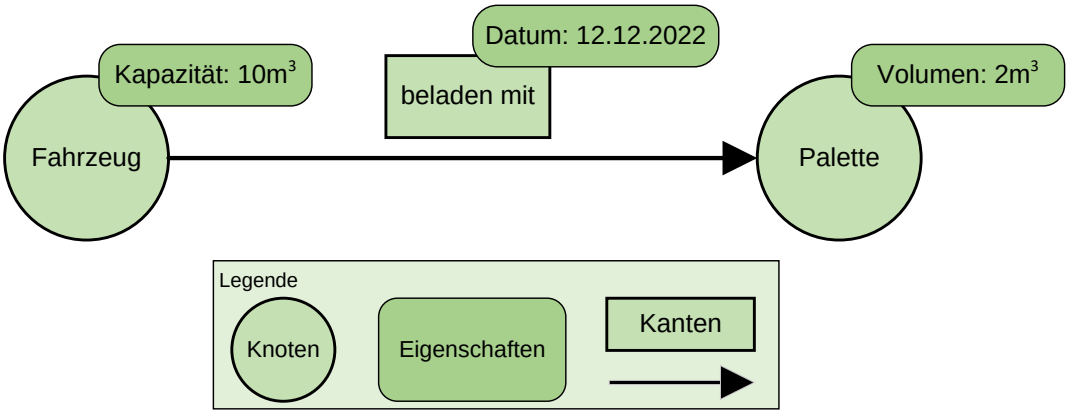


Abbildung 2.4: Beispiel eines Eigenschaftsgraphen nach Liu et al. (2021, S. 241)

Die am weitesten verbreitete Graphdatenbank ist Neo4J (solid IT 2022, Fernandes und Bernardino 2018). In einem ausführlichen Vergleich von Graphdatenbanken haben Fernandes und Bernardino (2018) die Graphdatenbank Neo4j als beste Option bezeichnet. Neo4j ist Open-Source und in Java (Bloch 2018, vgl.) implementiert. Zudem ist das Graphdatenbanksystem transaktionsbasiert. Das verwendete Datenmodell basiert auf dem Eigenschaftsgraph-Modell und erlaubt das Einführen eigener Datenbankschemata. Neo4j kann entweder mit einem Client-Server-Modell oder eingebettet betrieben werden. Zu den weiteren Merkmalen gehören gute Skalierbarkeit, hohe Zuverlässigkeit, hohe Leistung bei Speicherung und Verarbeitung von Daten, gut umgesetzte Schnittstellen für Java und die größte aktive Graphen-Community weltweit (Fernandes und Bernardino 2018). Andere populäre Graphdatenbanken sind AllegroGraph (Franz Inc. 2022), ArangoDB (ArangoDB Inc 2022) und TigerGraph (TigerGraph 2022)(solid IT 2022, Fernandes und Bernardino 2018).

Die von Neo4j verwendete Abfrage- und Manipulationssprache ist Cypher (Francis et al. 2018). Sie ist mittlerweile so weit verbreitet, dass sie als Quasi-Standard der Graphabfragesprachen gilt (Robinson et al. 2015). Basismechanismus ist der Musterabgleich, der die durch die Syntax der Sprache beschriebenen Muster in den gespeicherten Graphdaten findet (Meier und Kaufmann 2016). Cypher gilt als einfach zu erlernende und zu lesende Sprache, insbesondere für Anwender mit Vorkenntnissen anderer Datenbanksprachen wie SQL (Meier und Kaufmann 2016). Cypher wird neben Neo4J auch in anderen Datenbanken wie SAP HANA Graph verwendet und ist auch in der Forschung weit verbreitet (Francis et al. 2018). Detaillierte Einblicke und Verwendungsbeispiele können in Robinson et al. (2015) gefunden werden. Neben Cypher sind die Datenbanksprachen Gremlin und SPARQL verbreitet (Robinson et al. 2015).

Eine weitere Funktion, die Neo4j bietet, ist das Objekt-Graph-Mapping. Diese Funktionalität ist eine einfache Möglichkeit zur Kommunikation mit der Datenbank für Anwendungen, die objekt-orientiert und in Java implementiert wurden. Die Daten der Graphdatenbank werden automatisiert in Objekte der Programmiersprache Java umgewandelt oder aus diesen erzeugt. Das Object-Graph-Mapping (OGM) nutzt dazu Cypher und bietet eine Schnittstelle für die Anwendungen (Dietze et al. 2016, Neo4j 2022).

Bevor ein Datenbanksystem einer Anwendung verwendet werden kann, muss ein Datenbankschema entwickelt werden. Dieser Vorgang erfolgt in der Datenmodellierung, die das Thema des kommenden Abschnitts ist.

2.3.2 Datenmodellierung

Zum Entwurf einer Datenbank wird die Methode der Datenmodellierung angewendet. Meier und Kaufmann (2016) beschreiben den Datenbankentwurf als Vorgang mit drei Phasen. Zunächst wird eine Datenanalyse durchgeführt, daran anschließend eine Konzeption der Daten in einem Entitäten-Beziehungsmodell entworfen und schließlich darauf basierend ein Datenbankschema entwickelt. In diesem Abschnitt werden die drei genannten Phasen genauer erläutert.

Zu Beginn des Datenbankentwurfs muss der Teil der realen Welt festgelegt werden, der modelliert werden soll. Dementsprechend werden Daten und Beziehungen eingegrenzt und benannt. Es sollte eine Liste von relevanten Informationssachverhalten erstellt werden, die den ausgewählten Teil der realen Welt verbal beschreiben (Meier und Kaufmann 2016).

Im zweiten Schritt des Datenbankentwurfs werden die Ergebnisse des ersten Schritts konzeptuell modelliert. Dies geschieht für gewöhnlich anhand des Entitäten-Beziehungsmodell (Meier und Kaufmann 2016). Andere Ansätze, wie das semantische Datenmodell oder das objektorientiere Entwurfsmodell (Kemper und Eickler 2015) werden in dieser Arbeit nicht näher betrachtet. Im Entitäten-Beziehungsmodell werden die abzubildenden Sachverhalte mithilfe von Entitäten und Beziehungen modelliert. Entitäten entsprechen den Gegenständen des betrachteten Teils der realen Welt und werden auch Datenklassen genannt. Die Ergebnisse der Modellierung unter Zuhilfenahme des Entitäten-Beziehungsmodells werden für gewöhnlich visuell in einem Entitäten-Beziehungsmodell-Diagramm dargestellt.

Das erarbeitete Entitäten-Beziehungsmodell wird im dritten Schritt der Datenmodellierung in das Datenbankschema überführt. Hierzu muss festgelegt werden, welches Datenmodell in der Datenbank verwendet werden soll, wie beispielsweise das im vorherigen Abschnitt vorgestellte Relationenmodell oder Graphenmodell. Auch die Ergebnisse dieses Schritts sollten visuell in einer für das gewählte Datenmodell angemessenen Darstellung festgehalten werden.

Bei einer Überführung in das Graphenmodell entsprechen die Entitäten den Knoten und die Beziehungen den Kanten. In Meier und Kaufmann (2016)) ist eine detaillierte Anleitung anhand von sieben Regeln zur Überführung eines Entitäten-Beziehungsmodells in ein Graphenmodell verfügbar.

Zwar wurde bisher beschrieben, wie ein Datenbankentwurf erfolgen kann und wie Daten gespeichert werden können, jedoch wurde bisher nicht auf die Quellen von Daten eingegangen. Eine Betrachtung von Quellen von Daten erfolgt im nächsten Abschnitt.

2.3.3 Data Farming

Die meisten existierenden Daten in großen Datenbeständen der Logistik sind Beobachtungsdaten. Das bedeutet, sie wurden in der Realität beobachtet, beziehungsweise gemessen und erfasst (Gadatsch und Landrock 2017). Beobachtungsdaten weisen jedoch oftmals Mängel hinsichtlich der Qualität auf. García et al. (2015) definieren Datenqualität über die drei Elemente Richtigkeit, Vollständigkeit sowie Konsistenz und stellen fest, dass Beobachtungsdaten in den meisten Fällen keine dieser Elemente in hohem Maße erfüllt. Eine Alternative zur Sammlung von Daten an einem realen System stellt Data Farming dar. Die Methode des Data Farmings generiert Daten mithilfe der Simulation, genauer mithilfe der Durchführung von Experimenten an Simulationsmodellen. Sanchez (2018) sieht die durch Data Farming erhaltenen Daten in zwei Punkten als überlegen an. Zum einen würde die

Menge der Daten, die zur strukturierten Analyse eines Modells nötig sind, im Vergleich zu Beobachtungsdaten niedriger ausfallen, da nur vielversprechende Kombinationen von Faktoren untersucht würden und damit Redundanzen verringert werden könnten. Zum anderen könnten die Ursache-Wirkungs-Beziehungen in dem Modell besser untersucht und nachvollzogen werden, da eine Beziehung von Ergebnisdaten zu Faktorkombinationen einfach hergestellt werden kann.

Zunächst wurde Data Farming seit der ersten Vorstellung durch Brandstein und Horne (1998) hauptsächlich in der militärischen Forschung angewendet. Mittlerweile sind andere Anwendungsbereiche, wie in der Medizin (Mayo et al. 2016) oder Produktion (Feldkamp et al. 2018), hinzugekommen. Im Bereich der Logistik beschreiben Hunker et al. (2021) ein Farming-for-Mining-Framework zur Wissensentdeckung in Supply Chains. Dort wird Data Farming genutzt, um eine Simulationsdatenbasis für Data Mining zu erschaffen, damit das Verhalten einer Supply Chain analysiert werden kann. Eine zentrale Aussage von Hunker et al. (2021) ist, dass ein Simulationsmodell in einem solchen Farming-for-Mining-Framework speziell auf die Anforderungen des Data Minings ausgerichtet werden sollte, um eine zielgerichtete Generierung von Daten und somit eine Zeitersparnis zu erreichen. Eine spezielle Art des Data Mining stellt das Graph Mining dar, welches auf Daten in Form eines Graphen angewandt werden kann, um Muster zu extrahieren (Cook und Holder 2007). Für weitere detaillierte Informationen zu Graph Mining wird auf Cook und Holder (2007) verwiesen.

In Abschnitt 2.3 wurden bereits die drei V als Charakterisierung von Big Data vorgestellt. Im Kontext von Data Farming spricht Sanchez (2018) von den drei F des Data Farmings zur Charakterisierung der generierten Datenbestände: *Factors* (Faktoren), *Features* (Funktionalitäten) und *Flexibility* (Flexibilität). Factors bezieht sich auf eine breite und heterogene Auswahl von wichtigen Faktoren für die Simulation. Damit ist die Auswahl von Faktoren gemeint, die qualitativ, quantitativ oder kontinuierlich sein können und kleine bis große Wertebereiche aufweisen. Es sind auch Faktoren für Simulationsparameter wie den Simulationszeitraum oder die Länge der Einschwingphase möglich. Mit Features ist gemeint, dass eine große Anzahl von Simulationsantworten in den generierten Daten berücksichtigt werden sollen, also verschiedene Stellen der Antwortoberfläche der Simulation. Flexibility bezieht sich auf die Möglichkeit, eine Vielzahl von Fragen durch die Daten beantworten zu können, und zwar auch solche, die nicht im vornherein bekannt waren.

Die Bedeutung einer effizienten statistischen Versuchsplanung wird von Sanchez (2018) hervorgehoben, damit das Data Farming in akzeptabler Zeit die gewünschten Ergebnisse liefert. Insbesondere die Anwendung von raumfüllenden Versuchsplänen eignet sich laut Sanchez (2018) für Data Farming, da so ein hoher Grad an Flexibilität sichergestellt werden kann.

Der Ablauf des Data Farmings kann anhand der *Loop of Loops* von Horne und Seichter (2014) beschrieben werden. Die Umsetzung einer Anwendung, die sich an diesem Ablaufmodell orientiert, kann als Data-Farming-Framework bezeichnet werden. In Abbildung 2.5 ist die Loop of Loops dargestellt.

Der Abbildung 2.5 lassen sich die fünf relevanten Bereiche des Data Farmings entnehmen, nämlich den schnellen Szenario-Prototypenbau, die Modellentwicklung, die statistische Versuchsplanung, die leistungsstarken Berechnungen sowie die Analyse und Visualisierung. Der schnelle Szenario-Prototypenbau besteht aus der Definition der in der Simulation abzubildenden Teile des realen Systems, sowie der Ein- und Ausgangsdaten der Simulation. Das Ziel ist es, in kurzer Zeit ein Szenario zu erhalten, das die Fragestellung beantwor-

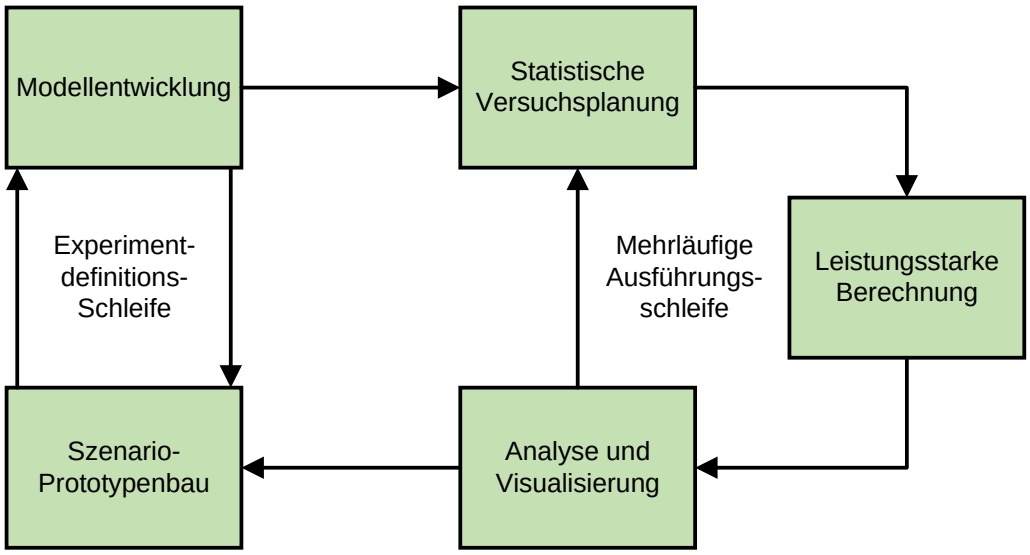


Abbildung 2.5: Die Loop of Loops nach Horne und Seichter (2014, S. 2356)

ten und einfach in ein Modell überführt werden kann. Bei der Modellentwicklung wird der Prototyp in ein ausführbares Modell überführt. Der Prototypenbau und die Modellentwicklung sind iterativ durchzuführen. Das iterative Vorgehen der beiden Phasen stellt eine Schleife dar, die Horne und Seichter (2014) als die Experimentdefinitions-Schleife beschreiben. Nach dem Abschluss der Bearbeitung dieser Schleife werden Experimente für das ausführbare Modell mit einer statistischen Versuchsplanung geplant. In der Phase der leistungsstarken Berechnung werden die geplanten Simulationsläufe des Experiments mit leistungsstarker Software und Hardware ausgeführt. Die leistungsstarke Berechnung kann mit Supercomputern ausgeführt werden, üblicherweise werden jedoch Netzwerke von mehreren Computern dazu verwendet (Sanchez 2018). Sanchez (2018) betont, dass Entwickler von Simulationswerkzeugen eine bessere Anbindung an solche Netzwerke von Computern anbieten sollten und eine bessere Ausnutzung der parallelen Kapazitäten von Mehrkern-Computern nutzen sollten. Den Bereich der Analyse und Visualisierung definieren Horne und Seichter (2014) als Prozess zum Hervorheben von nützlichen Informationen, dem Ziehen von Schlussfolgerungen und der Entscheidungsunterstützung auf Basis von statistischen, zusammenfassenden und präsentierenden Techniken. Das iterative Durchlaufen der Bereiche statistische Versuchsplanung, leistungsstarke Berechnung sowie Analyse und Visualisieren stellt die zweite Schleife dar, die als mehrläufige Ausführungsschleife bezeichnet wird. Eine weitere Schleife in der Loop of Loops ergibt sich durch das iterative Durchlaufen der beiden anderen Schleifen. Dies bedeutet, dass der gesamte Data Farming Prozess als iterativ angesehen werden und so lange stetig verbessert werden kann, bis gewünschte Ergebnisse erzielt werden.

In diesem Kapitel wurden alle benötigten technischen Grundlagen dargelegt, um die Entwicklung eines graphbasierten Simulators für ein Data-Farming-Framework nachvollziehen zu können. Im nächsten Kapitel erfolgt die Vorstellung des logistischen Kontexts, um den Simulator speziell für die Anwendung mit Logistiknetzwerken zu entwickeln.

3 Logistischer Kontext

Zum Verständnis der Arbeit wird in diesem Kapitel grundlegendes Wissen der Domäne Logistik, insbesondere hinsichtlich von Logistiknetzwerken, erläutert. Dazu wird zunächst der Begriff Logistik betrachtet mit einem Fokus auf den in der Logistik ablaufenden Prozesse. Daran anschließend werden exemplarisch vier grundlegende Aufgaben der Logistik herausgegriffen und vorgestellt. Am Ende des Kapitels wird der Begriff Logistiknetzwerk für diese Arbeit definiert und die Modellierung sowie Simulation solcher Netzwerke beschrieben.

3.1 Logistik und Logistiksysteme

Der Einfluss der Logistik auf die Unternehmensziele ist in den Bereichen Qualität, Kosten, Lieferung und Flexibilität signifikant (Schuh et al. 2013). Dennoch ist der Begriff der Logistik in der Literatur nicht scharf definiert und hat hinsichtlich seines Verständnisses eine Wandlung durchlaufen.

Eine klassische, weit verbreitete Definition von Logistik geschieht über die so genannten 7R der Logistik (Gleißner und Femerling 2012). Sie besagen, dass das richtige Produkt, in der richtigen Menge, in der richtigen Qualität, am richtigen Ort, zur richtigen Zeit, zu den richtigen Kosten, für den richtigen Kunden zur Verfügung stehen soll (Plowman 1964). Sie beschreiben die ergebnisorientierten Grundgedanken, die der Logistik zugrunde liegen. Eine andere Definition, die die Grenzen von Logistik aufzeigen soll, stammt von Pfohl (2018) und bezeichnet die Aufgabe der Logistik als die raum-zeitliche Güterverteilung mithilfe von Bewegungs- und Lagerprozessen. Außerhalb des Aufgabenbereichs liegen nach ihm die Güterbereitstellung, die durch Produktionsprozesse wie Gewinnungs-, Verarbeitungs- und Bearbeitungsprozesse erfolgt, und die Güterverwendung, die durch Konsumtionsprozesse, wie Gebrauchs- und Verbrauchsprozesse, stattfindet. Konsumtionsprozesse sind der Auslöser für jegliche Güterverteilung und den daraus resultierenden Güterfluss (Heiserich et al. 2011). Die physischen Kernleistungen der Logistik sind nach diesen Definitionen das Transportieren, Umschlagen und Lagern, welche zusammenfassend als Transport-, Umschlags-, und Lagerprozesse (TUL-Prozesse) bezeichnet werden (Heiserich et al. 2011). Auch Fleischmann (2008) sieht in den TUL-Prozessen die primären Leistungsprozesse der Logistik.

Im Laufe der Zeit haben einige Autoren ein weiter gefasstes Verständnis des Begriffs Logistik entwickelt, das über einfache TUL-Prozesse hinausgeht. Beispielsweise formuliert Bretzke (2020, S. 3-4): „Logistik umfasst die Gestaltung, Planung, Abstimmung, Steuerung, Durchführung und Kontrolle aller Ressourcen und Aktivitäten, die den Fluss von Transaktionsobjekten zwischen definierten Herkunftsorten (Quellen) und definierten Zielorten (Senken) beeinflussen und zeitgerecht auf einen bestimmten Bedarf ausrichten“. Eine gängige Sichtweise auf die Logistik hinsichtlich ihrer Querschnittsfunktionen beinhaltet die Unterteilung in die Beschaffungs-, Produktions- und Distributionslogistik (Schulte 2016). Kurz zusammengefasst beschäftigt sich die Beschaffungslogistik mit der Verfügbarkeit von

Gütern, die Produktionslogistik mit wertsteigernden Transformationsprozessen und die Distributionslogistik mit der Überführung von Gütern zu Kunden (Schuh et al. 2013). Auch die zunehmende Digitalisierung der Logistik führt zu einer Wandlung des Begriffs. (Hausladen 2016) fordert eine angepasste Sichtweise auf die klassischen 7R der Logistik, die nach ihrer Meinung um ein weiteres R für die richtigen Informationen ergänzt werden sollten und somit zu den 8R der Logistik würden.

Ein verwandter Begriff zu Logistik ist Supply Chain Management. Einige Autoren sind sogar der Meinung, dass Logistik und Supply Chain Management dasselbe beschreiben, wenn das weiter gefasste Verständnis des Logistikbegriffs zugrunde gelegt wird (z. B. Heiserich et al. 2011 und Waters 2011). Bei ihrer Argumentation wird dem Supply Chain Management lediglich das Management von logistischen Aktivitäten zugerechnet und damit eine Gleichsetzung der Begriffe begründet. Andere Autoren sehen die Logistik dem Supply Chain Management untergeordnet, da Supply Chain Management alle logistischen Aktivitäten einschließt und zusätzlich Marketing-, Produktions-, Innovations- und Entwicklungsprozesse einschließt (z.B. Dobhan 2012 oder Sucky 2008). Auch das Council of Supply Chain Management Professionals grenzt Logistik von Supply Chain Management ab, indem sie in Logistik lediglich den Teil des Supply Chain Managements sehen, der für die Planung, Implementierung und Kontrolle des Flusses und der Lagerung von Gütern, Diensten und verbundenen Informationen zwischen dem Ursprung der Güter und deren Ort des Konsums verantwortlich ist (Pfohl 2018).

Die Ausführung logistischer Prozesse, also das Ineinandergreifen von Bewegungs- und Lagerungsprozessen, geschieht in Logistiksystemen (Pfohl 2018). Systeme sind, wie bereits in Abschnitt 2.1 beschrieben, eine Menge von Elementen, die in Beziehungen stehen und von ihrer Umwelt abgegrenzt sind. Die Menge der Elemente kann als ein Ganzes gesehen werden. Im Fall von Logistik sind die Systemelemente entweder Logistikunternehmen, deren Hauptaufgabe die Raum- und Zeitüberbrückung ist, oder Unternehmen, deren Teilaufgabe die Durchführung logistischer Prozesse ist, wie Industrie-, Handels-, oder Dienstleistungsunternehmen (Pfohl 2018). Systeme werden grundsätzlich in soziale, technische und sozio-technische Systeme unterschieden (Jockisch und Rosendahl 2009). In sozialen Systemen ist der Mensch zentraler Bestandteil. Bei der Modellierung solcher Systeme müssen alle wichtigen psychischen Komponenten und zwischenmenschliche Interaktionen adäquat abgebildet werden. Technische Systeme beinhalten ausschließlich technische bzw. physikalische Elemente. Bei den sozio-technischen Systemen, die technische und soziale Elemente vereinen, kommen beide Arten von Komponenten vor und müssen abgebildet werden. Ein logistisches System wird den sozio-technischen Systemen zugeordnet, da es Unternehmen inklusive derer Mitarbeiter und technischen Anlagen beinhaltet.

Sucky (2008) definiert Logistiksysteme als Systeme, die aus einer endlichen Menge von Akteuren bestehen, wie Verlader, Dienstleister oder Empfänger, die über Kunden-Lieferanten-Beziehungen verbunden sind. Die Akteure haben unterschiedliche Standorte, wie Produktionsstätten, Hubs oder Läger. Die Standorte sind über verschiedenartige Beziehungen verbunden, die sich nach eingesetzten Transportmitteln und Fließobjekten kategorisieren lassen. Zu den Fließobjekten zählt Sucky (2008) Güter-, Informations- und Finanzflüsse, die zur logistischen Leistungserstellung benötigt werden. Hier geht seine Auffassung über die reine Güterverteilung hinaus, jedoch ist ein Güterfluss davon abhängig, dass im Vorfeld Informationen zwischen Orten ausgetauscht werden, um einen Bedarf zu kommunizieren und entsprechend finanziell zu vergüten (Pfohl 2018).

3.2 Exemplarische Aufgaben der Logistik

Im vorangehenden Abschnitt werden die TUL-Prozesse als die Kernleistung der Logistik definiert. Die Aufgaben, die durch die TUL-Prozesse erledigt werden sollen, erfüllen die Grundaufgaben der Logistik und werden daher aus als Kernaufgaben der Logistik bezeichnet (Schuh et al. 2013). Schuh et al. (2013) zählen zu den Kernaufgaben der Logistik das Transportieren, Umschlagen, Kommissionieren, Verpacken, Lagern und Disponieren. Im Folgenden werden zwei exemplarische Kernaufgaben herausgegriffen und näher erläutert. Zunächst wird in Abschnitt 3.2.1 das Lagern betrachtet, welches die zeitliche Güterverteilung abbildet. Zudem behandelt der Abschnitt mit der Bestandsplanung eine weitere Aufgabe der Logistik, die für die Aufgabe des Lagerns unterstützende Tätigkeiten erbringt. Im darauf folgenden Abschnitt 3.2.2 wird für die räumliche Güterverteilung das Transportieren beleuchtet. Auch hier wird mit der Tourenplanung eine weitere logistische Aufgabe beleuchtet, die für den Kernprozess des Transportieren wichtig ist.

3.2.1 Lagern und Bestandsplanung

Das Lagern wird als das geplante Liegen von Gütern im Materialfluss definiert (VDI 1970). Es erfüllt eine Ausgleichsfunktion hinsichtlich Differenzen in der zeitlichen und mengenmäßigen Nachfrage von Gütern (Stich et al. 2013). Die Aufgabe des Lagerns wird von Lägern übernommen, in denen Bestände von Gütern gehalten werden. Da Bestände durch zu- und abfließende Güter ständigen Änderungen ausgesetzt sind, sollte eine Bestandsplanung erfolgen. Die Aufgabe der Bestandsplanung ist Teil der Beschaffungslogistik und kann mithilfe von Lagerhaltungspolitiken durchgeführt werden (Schuh et al. 2013).

Lagerhaltungspolitiken beinhalten Antworten auf die Häufigkeit der Ermittlung des Lagerbestands, den Zeitpunkt der Bestellauslösung für neue Güter und dem Umfang der Bestellung (Inderfurth 1996). Sie werden zumeist im Fall stationärer Nachfrage, das bedeutet die Nachfrage bleibt über einen längeren Zeitraum konstant, eingesetzt und führen Bestellentscheidungen anhand vordefinierter Regeln durch (Manitz 2015).

Lagerhaltungspolitiken, die die Bestellungen unabhängig von anderen Akteuren im Logistiknetzwerk auslösen, werden einstufige Lagerhaltungspolitiken genannt (Tempelmeier 2015). Für Informationen zu mehrstufigen Lagerhaltungspolitiken, die den systemweiten Lagerbestand bei ihren Entscheidungen berücksichtigen, wird auf Tempelmeier (2015) verwiesen. Im folgenden sollen lediglich einstufige Lagerhaltungspolitiken thematisiert werden. Wichtige Größen sind die Entscheidungsvariablen des Bestellpunkts s, des Bestellzyklus r, der Bestellmenge q und des Bestellniveaus S. Entsprechend der gerade genannten Entscheidungsvariablen gibt es die (s, q)-Politik, (r, s)-Politik und (s, S)-Politik. Die (s, q)-Politik verwendet als Bestellpunkt einen definierten Meldebestand und hat eine konstante Bestellmenge. Sinkt der Lagerbestand auf den Meldebestand, wird eine Bestellung mit der konstanten Bestellmenge ausgelöst. Ist der Lagerbestand so niedrig, dass die übliche Bestellmenge nicht ausreichen würde, den Lagerbestand über den Meldebestand zu bringen, können entsprechend mehrere Bestellungen mit der konstanten Bestellmenge ausgelöst werden. Bei der (r, S)-Politik wird in einem konstanten Bestellzyklus, also in konstanten zeitlichen Abständen, eine Bestellung ausgelöst, die den Lagerbestand auf ein konstantes Bestellniveau hebt. Die (s, S)-Politik prüft, ob der Lagerbestand den Bestellpunkt erreicht hat und löst dann eine Bestellung aus, die den Lagerbestand auf ein konstantes Bestellniveau hebt. Bei der (s, q)- und der (s, S)-Politik kann zwischen kontinuierlicher und periodischer Bestandsüberwachung unterschieden werden. Kontinuierliche Bestandsüber-

wachung bedeutet, dass eine Bestellung sofort ausgeführt wird, sobald der Meldebestand erreicht ist. Bei periodischer Bestandsüberwachung wird die Feststellung des Erreichens des Meldebestands und die Bestellauslösung zur nächsten Periode ausgeführt. In der Praxis ist eine Periodenlänge für gewöhnlich mindestens einen Tag lang (Tempelmeier 2015).

3.2.2 Transportieren und Tourenplanung

Neben dem Lagern ist das Transportieren eine wichtige Kernaufgabe der Logistik, die entlang des gesamten Logistiksystems auftritt. Eine Unterscheidung wird zwischen innerbetrieblichen und außerbetrieblichen Transportprozessen gemacht (Schuh et al. 2013). Die außerbetrieblichen Transportprozesse sind für Transporte zwischen verschiedenen Standorten in einem Logistiksystem zuständig und verwenden meist Transportmittel mit hohen Kapazitäten wie Lastkraftwagen (LKW), Flugzeuge, Züge oder Schiffe (Schuh et al. 2013). Innerbetriebliche Transportprozesse hingegen sind für die Verteilung von Gütern innerhalb eines Standorts zuständig und verwenden zumeist Flurförderfahrzeuge (Schuh et al. 2013).

Für den Fall, dass ein Standort viele Kunden aufweist, die Güter von dem Standort erhalten sollen und es beim Transport in Frage kommt, mit einem Fahrzeug mehrere Kunden zu bedienen, wird eine Tourenplanung durchgeführt. Die Tourenplanung ist eine Logistikaufgabe der Distributionslogistik (Wenger 2009). Die Tourenplanung ist immer dann nötig, wenn die Bestellmengen der einzelnen Knoten bezogen auf die Kapazität der Auslieferungsfahrzeuge relativ niedrig ist und das Erfüllen mehrerer Aufträge auf einer Tour somit möglich ist. Der Fall niedriger Bestellmengen bei hoher Anzahl von Bestellungen ist insbesondere in der Konsumgüterindustrie oftmals gegeben (Wenger 2009). Die Aufgabe der Tourenplanung ist es, unter Berücksichtigung der Anzahl der zur Verfügung stehenden Fahrzeuge und deren Kapazitätsbeschränkungen, kostenoptimale Touren zu planen (Schuh et al. 2013). Eine Tour ist eine Menge von Aufträgen, die von einem Transportmittel bedient wird und die Menge geplanter Touren wird auch Tourenplan genannt (Wenger 2009). Problemstellungen der Tourenplanung werden oftmals mit Methoden des Operations Research gelöst (Domschke 2018).

Bei dem Standardproblem der Tourenplanung, das im weiteren Verlauf exemplarisch behandelt wird, handelt es sich um das Capacitated Vehicle Routing Problem (CVRP) (Wenger 2009). Das CVRP unterliegt nach Wenger (2009) folgenden allgemeinen Annahmen:

- Es existiert eine Menge von Aufträgen, die hinsichtlich der Bestellmenge und dem Lieferort bekannt sind.
- Alle Aufträge müssen an demselben Depot beginnen und enden.
- Die akkumulierte Bestellmenge aller Aufträge darf nicht durch nur eine Lieferung befriedigbar sein.
- Die Distanzen zwischen den Bestellorten untereinander und dem Depot müssen bekannt und symmetrisch sein (richtungsunabhängig).
- Die Fahrzeiten müssen konstant sein.
- Die Flotte der Transportmittel muss beliebig groß und homogen sein.
- Jedes Element der Flotte kann maximal eine Tour durchführen.

Aus dem allgemeinen CVRP können spezialisierte Probleme abgeleitet werden. So ist es beispielsweise möglich, die Anzahl der Elemente in der Flotte der Transportmittel zu begrenzen.

Zur Lösung des CVRP wird ein System in Graphenform verwendet. Die Knoten des Graphen werden durch die Kunden gebildet. Auch der Standort, an von dem die Güter losgeschickt werden und zu dem die eingesetzten Fahrzeuge zurückkehren, wird mit einem Knoten dargestellt. Alle Knoten erhalten eine Bewertung. Die Bewertung entspricht der nachgefragten Menge an Gütern. Der Versandstandort erhält eine Bewertung von Null. Der Graph ist vollständig und die Kanten erhalten Bewertungen, die dem Aufwand zum Befahren der Strecke darstellen.

Das CVRP gehört zu den NP-vollständigen Problemen und somit steigt die Rechenzeit exponentiell mit der Problemgröße (Wenger 2009). Das bedeutet, dass das Problem so komplex ist, dass eine exakte Lösung für mittelgroße Probleminstanzen lediglich unter Einsatz großer Rechenzeit möglich ist. Für große Probleminstanzen ist eine Berechnung oftmals mit aktuellen Rechenkapazitäten von Computern nicht realisierbar. Eine Alternative zur exakten Berechnung stellt die Verwendung einer Heuristik dar, welche in heuristischen Verfahren angewendet werden. Durch die Anwendungen von heuristischen Verfahren wird im Allgemeinen nicht die optimale Lösung eines Problems gefunden, dafür aber zulässige gute Lösungen in deutlich kürzerer Zeit (Domschke et al. 2015). Somit ist es möglich, unter Verwendung von Heuristiken, NP-vollständige Probleme, wie das CVRP, in kurzer Zeit ausreichend gut zu lösen.

Zu den bekanntesten heuristischen Verfahren zur Lösung des CVRP zählt das Savings-Verfahren von Clarke und Wright (1964) (Schuh et al. 2013). Darauf basierend haben Pichpibul und Kawtummachai (2012) den Improved Clarke and Wright Savings Algorithm (ICWA) entworfen. In einem von ihnen durchgeführten Vergleich konnte der ICWA deutlich bessere Ergebnisse erzielen als in der Originalform. Auch im Vergleich zu anderen heuristischen Verfahren liefert der ICWA hervorragende Ergebnisse, obgleich es hinsichtlich seiner Komplexität vergleichsweise einfach aufgebaut ist. In einer kritischen Analyse stellen Sörensen et al. (2019) fest, dass die Ergebnisse mit der beschriebenen Implementierung nicht reproduzierbar sind und der ICWA nicht zu den leistungsstärksten Verfahren gehört, sondern eher als überdurchschnittlich starkes Verfahren angesehen werden sollte. Nichtsdestotrotz wird in dieser Arbeit der ICWA exemplarisch behandelt, da es zu den intuitivsten und einfachsten Verfahren zählt und mindestens überdurchschnittliche Ergebnisse erwartet werden können.

Das ursprüngliche Verfahren nach Clarke und Wright (1964) beginnt damit, für jeden Kunden, der beliefert werden soll, eine Tour zum Depot zu erstellen. Im Folgenden wird eine Liste von Einsparungen (*Savings*) hinsichtlich der gefahrenen Strecke berechnet, die durch die Kombination von Touren entstehen würden. Anschließend wird die Liste sortiert und die Touren vereinigt, die die höchsten Einsparungen realisieren. Für den weiteren Verlauf gibt es eine parallele und eine sequentielle Variante des Verfahrens. Bei der parallelen Variante wird nach einer Vereinigung von Touren die Liste der Einsparungen aktualisiert und wiederum die Tour vereinigt, die die höchste Einsparung bietet, unabhängig von den vorherig vereinten Touren. Dies wird so lange wiederholt, bis keine Kombinationen aufgrund der Kapazitätsbeschränkungen möglich sind oder alle Touren bereits vereinigt wurden. Bei der sequentiellen Variante werden, sobald eine neue Tour durch Vereinigung erhalten wurde, die höchsten Einsparungen gesucht, die mit der begonnenen Tour kombinierbar sind. Dies wird so lange gemacht, bis die Kapazität des Transportmittels für die Tour ausgeschöpft ist und keine weiteren Touren mehr gebildet werden können. Die erwähnten Einsparungen können mithilfe der nachfolgenden Gleichung 3.1 berechnet werden:

$$s_{ij}{=}d_{0i} + d_{0j} - d_{ij} \quad \text{für} \quad i, j > 0 \quad \text{und} \quad i \neq j \tag{3.1}$$

Das Verfahren von Pichpibul und Kawtummachai (2012) basiert auf der parallelen Variante des Verfahrens von Clarke und Wright (1964). Sie schlagen vor, die Tourenplanung nicht nur anhand einer sortierten Liste der Einsparungen zu berechnen, sondern iterativ mehrere Listen einzubeziehen, deren Reihenfolge von Einsparungen teilweise Zufallsfaktoren unterliegt. Durch das iterative Verfahren werden mehrere Tourenpläne erstellt, von dem der beste ausgewählt werden kann. Das zufallsbehaftete Auswahlverfahren bei der Erstellung der Listen von Ersparnissen wird als eine Kombination des *Turnierauswahlverfahrens* (Goldberg et al. 1989) und der *Roulette Wheel Selection* (Holland 1992) beschrieben. Das Flussdiagramm des ICWA kann wie folgt in Abbildung 3.1 angegeben werden:

Das iterative Verfahren aus Abbildung 3.1 kann nach einer beliebigen Anzahl von Durchläufen beendet werden. Eine Erhöhung der Iteration führt folgerichtig zu einer erhöhten Berechnungszeit des Verfahrens. In den Experimenten in der Arbeit von Pichpibul und Kawtummachai (2012) stellen die Autoren fest, dass in fast allen Fällen die beste Lösung in weniger als 2500 Iterationen gefunden wird.

3.3 Logistiknetzwerke

In Abschnitt 3.1 wurden Logistiksysteme bereits als Systeme zur raum-zeitlichen Güterverteilung definiert. Schuh et al. weisen darauf hin, dass Logistiksysteme, vorausgesetzt sie bestehen aus mehreren Standorten, eine netzwerkartige Struktur von Standorten und deren Beziehungen untereinander aufweisen. Sucky (2008) leitet den Begriff des Logistiknetzwerks aus solchen Logistiksystemen ab. Zu Analyse- und Darstellungszwecken von Logistiksystemen können sie als Netzwerk, genauer als Logistiknetzwerk, modelliert werden. Wörtlich definiert Sucky (2008, S. 934) Logistiknetzwerke als „eine durch Abstraktion gewonnene, vereinfachte Abbildung eines realen Logistiksystems, wobei Knoten und Pfeile [Anmerkung: Kanten] die relevanten Elemente des Realsystems und deren Beziehungen darstellen“. Er ergänzt weiterhin, dass sowohl die Knoten als auch die Kanten Merkmale und Merkmalsausprägungen aufweisen können, um die Akteure und Beziehungen in dem Logistiknetzwerk zu beschreiben. Zu den Akteuren zählt Simchi-Levi et al. (2000) Zulieferer, Produktionsstätten, Warenhäuser, Distributionszentren und Kunden. Einige Autoren, wie beispielsweise Gudehus (2012), merken an, dass ein weltumspannendes Logistiknetzwerk existiert, über das alle Güter der globalisierten Welt transportiert werden. Dementsprechend stellen Logistiknetzwerke von Unternehmen bzw. Gruppen von Unternehmen lediglich „Sub-Logistiknetzwerke“ dar, deren Grenzen genau definiert werden müssen.

Ein verwandter Begriff ist der der Supply Chain, welcher in der Literatur weit verbreitet ist. Je nach Verständnis einer Supply Chain kann der Begriff synonym zu Logistiknetzwerk verwendet werden (Schuh et al.). Der Begriff Supply Chain wird in der deutschen Literatur oftmals als Lieferkette oder Wertschöpfungskette übersetzt. Jedoch geben einige Autoren wie beispielsweise Pfohl (2018) oder Mattfeld und Vahrenkamp (2014) zu bedenken, dass moderne Supply Chains weniger einer Kette als einem Netzwerk ähneln, da zwischen den Teilnehmern einer Supply Chain oftmals nicht nur sequenzielle Verbindungen, sondern auch vielfältige Vernetzungen untereinander existieren. Somit kann unter Supply Chains auch ein Liefernetzwerk bzw. ein Wertschöpfungsnetzwerk verstanden werden. Wird eine Supply Chain als Liefernetzwerk verstanden, ist sie ein Logistiknetzwerk. Nach Ansicht

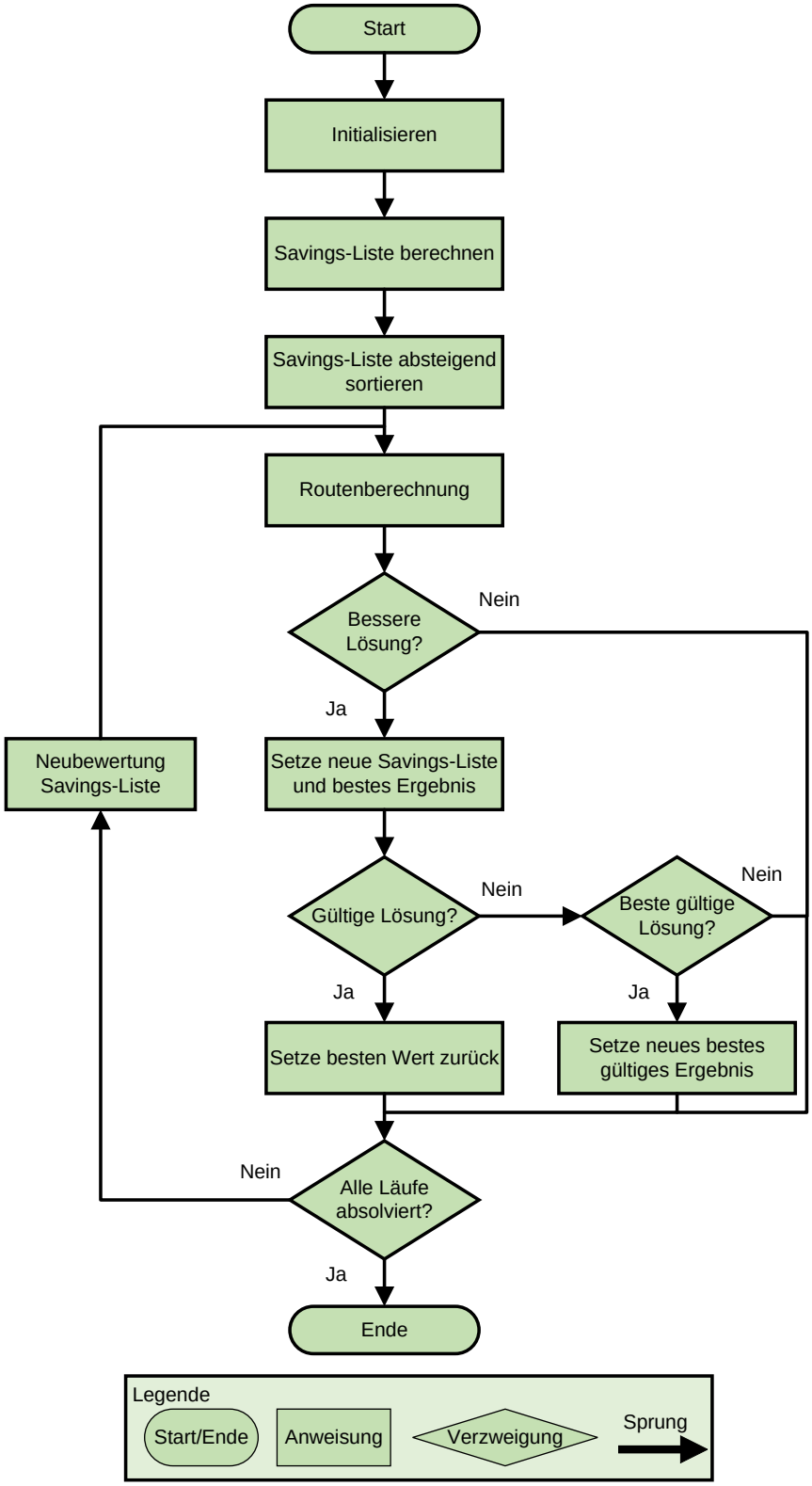


Abbildung 3.1: Flussdiagramm des ICWA nach Pichpibul und Kawtummachai (2012, S. 309)

einiger Autoren umfasst eine Supply Chain mehr als lediglich den logistischen Teil. Einige Definitionen unterstreichen den wertschöpfenden Charakter einer Supply Chain aus der prozessorientierten Sicht, so sieht Klaus et al. (2012, S. 554) die Supply Chain als eine „[...] Abfolge von Aktivitäten, die notwendig ist, um Kunden bzw. Märkte erfolgreich zu versorgen [...]“. Sucky (2021, S. 76) lässt in seine Definition einer Supply Chain die Ausrichtung auf ein bzw. mehrere Endprodukte einfließen und folgert: „Eine Supply Chain ist ein produkt- oder produktgruppenbezogenes, unternehmensübergreifendes Wertschöpfungsnetzwerk“. Nach dieser Definition sind auch wertschöpfende Prozesse wie die Verarbeitung von Gütern Teil einer Supply Chain. Der umfassenderen Auffassung einer Supply Chain als Wertschöpfungsnetzwerk folgend wäre eine synonyme Verwendung der Begriffe Supply Chain und Logistiknetzwerk nicht zutreffend. In diesem Fall würde ein Logistiknetzwerk den Teil einer Supply Chain darstellen, der lediglich auf Wertschöpfungsprozesse beschränkt wird, die der Logistik zugeordnet werden.

Ein wichtiger Punkt zur Charakterisierung von Logistiknetzwerken ist der Kundenentkopplungspunkt. Er gibt an, an welcher Stelle im Logistiknetzwerk die planbasierten Aktivitäten, damit ist das Anlegen von Beständen basierend auf Prognosen gemeint, in die auftragsgetriebenen Aktivitäten, die durch Kundenaufträge hervorgerufen werden, übergehen (Stich et al. 2013). Planbasierte Aktivitäten gehören dem so genannten Push-Prinzip an, während die auftragsorientierten Aktivitäten dem Pull-Prinzip zugeordnet werden. Je nach Lage des Kundenentkopplungspunktes, können Logistiknetzwerke als Make-to-Stock, Assemble-to-Order, Make-to-Order oder Engineer-to-Order betrieben werden (Stich et al. 2013). Bei Make-to-Stock-Netzwerken liegt der Kundenentkopplungspunkt zwischen Endmontage und dem Kundenauftrag und bedingt somit, dass Güter in Auslieferungslägern vorgehalten werden. Bei Assemble-to-Order geschieht die Montage zu den Endprodukten erst mit dem Eingang von Kundenaufträgen. Das Make-to-Order-Prinzip beginnt mit der Fertigung der Zwischenprodukte erst, sobald ein Auftrag eingegangen ist und bei dem Engineering-to-Order-Prinzp wird zunächst das Endprodukt entworfen.

Im Folgenden werden Möglichkeiten zur Modellierung von Logistiknetzwerken in Abschnitt 3.3.1 erläutert und anschließend wird in Abschnitt 3.3.2 auf die Simulation von Logistiknetzwerken eingegangen.

3.3.1 Modellierung von Logistiknetzwerken

Um die Sichtweise des Modellierers für die Modellbildung festzulegen, werden so genannte Modellierungskonzepte verwendet, die Hilfsmittel zur Modellbildung zur Verfügung stellen. Beispielsweise ist das Dortmunder Prozesskettenmodell zu nennen, zu dem weitere Informationen in Jungmann und Uygun (2009) gefunden werden können. Gutenschwager et al. (2017) beschreiben vier weitere Modellierungskonzepte, die oftmals im Kontext von Simulatoren in der Logistik verwendet werden. Dazu gehören das bausteinorientierte, das objektorientierte und das theoretische Modellierungskonzept, sowie das Sprachkonzept.

Das objektorientierte Modellierungskonzept ist aus dem objektorientierten Paradigma (Lutz 1997) abgeleitet und basiert auf Klassenbildung, Kommunikation und Vererbung. Dem Modellierer werden Basisklassen zur Verfügung gestellt, von denen Objekte erzeugt oder abgeleitet werden können. Die Basisklassen werden, wie auch bei dem bausteinorientierten Konzept, in einer Bibliothek angeboten. Aufgrund der Ähnlichkeit zum bausteinorientierten Konzept wird die objektorientierte Modellierung oftmals innerhalb von bausteinorientierten Konzepten als Ergänzung verwendet, um dem Modellierer die Möglichkeit zur Bildung von spezialisierten Unterbausteinen zu geben. Zu den theoretischen Modellierungs-

konzepten zählt Lunze (2012) beispielsweise Automaten oder Petri-Netze (vgl. Deininger 2019). Sie sind speziellen Strukturkonzepten unterworfen und müssen dementsprechend modelliert werden. Bei Sprachkonzepten werden dem Anwender Sprachkonstrukte und Modellbildungsmechanismen zur Verfügung gestellt, mit dem er das Modell in Form einer Sprache umsetzen kann. Programmiersprachen und Simulationssprachen sind typische Vertreter des Sprachkonzepts (Wenzel 2018). Simulatoren, die ein Sprachkonzept umsetzen, sind somit von hoher Allgemeingültigkeit gekennzeichnet (Noche und Wenzel 1991).

Bei dem bausteinorientierten Modellierungskonzept wird eine Bausteinbibliothek mit vordefinierten Bausteinen bereitgestellt, die untereinander verknüpft werden können und somit ein Modell erstellt werden kann. Verbraeck und Valentin (2008) beschreiben die Bausteine als eigenständige, interoperable, wiederverwendbare und austauschbare Einheiten, die ihre interne Struktur von der Umwelt abkapseln und ihre Funktionen über definierte Schnittstellen anbieten. Die Bausteine besitzen interne Ablauflogiken und können durch den Modellierer parametrisiert werden, um ein gewünschtes Verhalten abzubilden. Weiterhin kann hinterlegt werden, welche Bausteine miteinander verknüpft werden können und wie deren Informationsaustausch untereinander aussehen muss.

Unabhängig vom gewählten Modellierungskonzept sollte das erstellte Modell der netzwerkartigen Struktur von Logistiknetzwerken gerecht werden (Bause et al. 2000). Zudem sollten die in Abschnitt 3.1 identifizierten Systemelemente abgebildet werden, um von einem Modell eines Logistiknetzwerks sprechen zu können. Gutenschwager et al. (2017) weisen darauf hin, dass Modelle in der Logistik eine Vielzahl von stochastischen Vorgängen aufweisen. Bei der Modellierung des Zufalls werden für diskrete Zufallsvariablen oftmals Bernoulli-, Binomial-, Poisson- und diskrete Gleichverteilungen verwendet. Im stetigen Fall finden Exponential-, Erlang-k-, Normal-, Gamma-, Dreiecks- und stetige Gleichverteilungen Verwendung (Rabe et al. 2008). Stich et al. (2013) regen zudem an, eine Repräsentation des Logistiknetzwerks als Graphen in das Modell zu inkludieren, um eine Reihe leistungsfähiger und quantitativer Analysemethoden für logistische Fragestellungen zu ermöglichen.

Um ein fehlerfreies Modell entsprechend dem vorgesehenen Konzept zu erhalten, sollte eine ausführliche V&V durchgeführt werden (vgl. Abschnitt 2.1). Es ist zu empfehlen, eine ständige V&V begleitend zum Modellierungsprozess durchzuführen und nicht erst am Ende, um frühzeitig auf etwaige Missstände reagieren zu können (Rabe et al. 2008).

In Abschnitt 2.1.1 wurde bereits beschrieben, dass die Modellierung von Werkzeugen unterstützt wird. Im Fall von Simulationsmodellen existiert eine große Menge von Spezial-Software-Werkzeugen, die oftmals direkt in einem Simulator integriert sind (Gutenschwager et al. 2017). In der Domäne der Logistik verwenden diese Werkzeuge meistens das bausteinorientierte Modellierungskonzept. Hier wird zwischen offenen und geschlossenen Werkzeugen unterschieden (Gutenschwager et al. 2017). Offene Werkzeuge erlauben es, eigene Bausteine einzubringen, während geschlossene Werkzeuge nur eine vordefinierte Bausteinbibliothek anbieten (Gutenschwager et al. 2017). Es existieren jedoch auch Mischformen, in denen Bausteine teilweise erweitert werden können (Gutenschwager et al. 2017).

Simulationsmodelle von Logistiknetzwerken weisen oftmals verschiedene Detailebenen auf. Typisch ist beispielsweise, dass interne Prozesse von Standorten abgebildet werden, jedoch auch die Transporte zwischen den Standorten, was eine übergeordnete Detailebene darstellt. Dies geht oftmals damit einher, dass auch die Visualisierung der Modelle auf verschiedenen, hierarchischen Ebenen stattfindet. Zur Visualisierung der Transporte zwischen Standorten wird oftmals Kartenmaterial in geeignetem Maßstab verwendet (Manivannan 1998).

Simulationsmodelle werden mit der Intention erstellt, mit ihnen Experimente durchzuführen. Im nächsten Abschnitt wird auf die Simulation speziell von Logistiknetzwerken eingegangen.

3.3.2 Simulation von Logistiknetzwerken

Logistiknetzwerke stellen für gewöhnlich komplexe Systeme mit dynamischen und stochastischem Verhalten dar, bei denen analytische Methoden schnell an ihre Grenzen stoßen (Schmuck 2014). Entsprechend diesem Umstand ist der Einsatz von Simulation gerechtfertigt. Auch andere Autoren wie Law (2015) oder Gutenschwager et al. (2017) weisen auf die außerordentlich gute Eignung von Simulation in Logistiknetzwerken hin.

Die Methode der Simulation findet im Bereich der Logistik weitere Verbreitung (Law 2015, Gutenschwager et al. 2017). Die meiste Anwendung findet die ereignisdiskrete Simulation (Rose und März 2011). Der VDI (2014) sieht Anwendungen für Simulation sowohl im innerbetrieblichen als auch in überbetrieblichen logistischen Aufgaben. Dabei ist die Anwendung nicht auf eine bestimmte Phase des technischen Lebenszyklus beschränkt, sondern wird in allen Phasen, also Planung, Realisierung und Betrieb, angewendet. Ein Fokus auf den Bereich der Planungsphase ist laut mehrerer Untersuchungen von Jahangirian et al. (2010), Smith (2003) und Gutenschwager et al. (2017) erkennbar. Nach ihnen sind etwa 80% aller Simulationsstudien auf diesen Bereich ausgerichtet. Spieckermann (2005, S. 2) stellte fest, dass im Bereich der Logistikprojekte „die Planungsabsicherung mit Simulation heute eine Selbstverständlichkeit ist“ und bereits im Jahr 2005 fast keine Autobauer mehr ohne Simulation ihrer bereichs- und standortübergreifenden Liefernetzwerken auskäme. Typische Anwendungen in der Planungsphase sind laut VDI (2014) beispielsweise die Ermittlung von Kapazitätsgrenzen, die Identifizierung von Schwachstellen und die Neuplanung von Anlagen. Ein Grund für die Vielfache Anwendung der Simulation in der Planungs- und Realisierungsphase sind die Realisierung monetärer und zeitlicher Ersparnisse. Zur Veranschaulichung wird auf die folgende Abbildung 3.2 verwiesen.

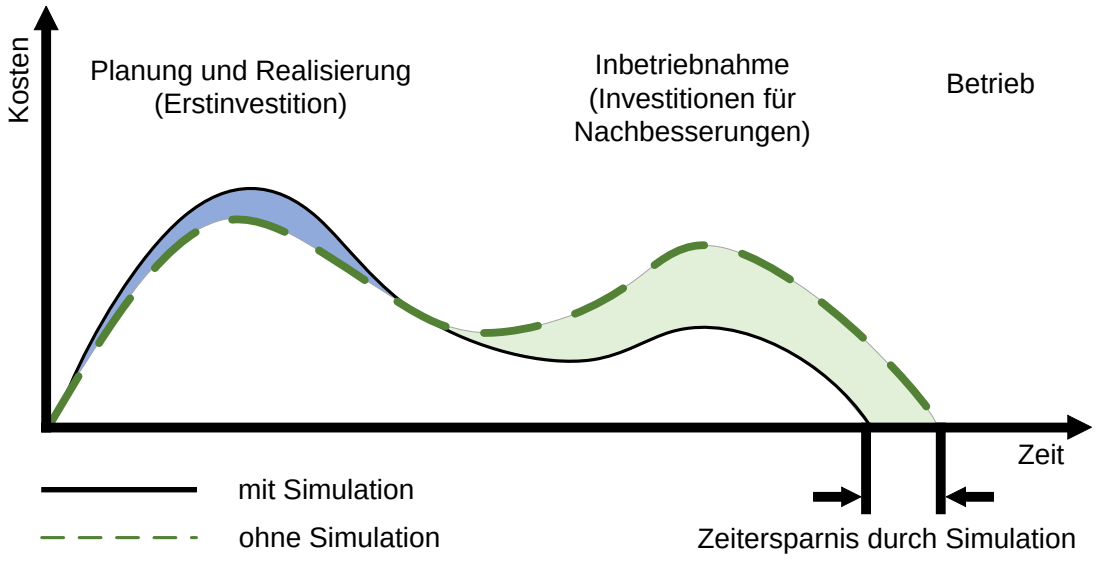


Abbildung 3.2: Schematischer Verlauf der Kosten eines Projekts in der Logistik nach Gutenschwager et al. (2017, S. 48)

Der abgebildete zeitliche Verlauf in Abbildung 3.2 zeigt die erwarteten Kosten eines Projekts unter Einsatz und ohne Einsatz der Simulation. Wird eine Simulationsstudie zur Planungs- bzw. Realisierungsphase durchgeführt, ist zunächst mit höheren Kosten durch den vermehrten Arbeits- und Zeitaufwand zu rechnen. Im weiteren Verlauf wird jedoch ersichtlich, dass in der Phase der Inbetriebnahme gesenkte Kosten zu erwarten sind, sowie eine verkürzte Länge der Phase erzielt werden kann. Eine Quantifizierung der entstandenen Kosten kann über Integration festgestellt werden. Die in der Abbildung eingefärbten Flächen zeigen die Differenz der bestimmten Integrale in den entsprechenden Integrationsbereichen. Die blau gefärbte Fläche stellt die zunächst entstandenen Mehrkosten durch die Simulationsstudie dar und die grüne Fläche die gesparten Kosten. In dem gezeigten Fall übersteigt die grüne Fläche die blaue und zeigt somit eine positive Bilanz hinsichtlich der Kosten auf. Die Abbildung zeigt lediglich einen idealtypischen Verlauf, der durch die Anwendung der Simulation in einem Projekt entstehen kann. Im Einzelfall können die tatsächlichen Ergebnisse abweichen. Darüber hinaus sei angemerkt, dass die Abbildung nicht die Qualität der Planungsergebnisse mit einbezieht. Die erhöhte Qualität der Planung kann erheblich zu Kostenersparnissen im laufenden Betrieb beitragen und eine weitere Rechtfertigung für Simulation in Produktion und Logistik darstellen.

Schmuck (2014) zählt zu den Gründen, die für den Einsatz von Simulation in Logistiknetzwerken sprechen, vier weitere Punkte:

- Erhöhtes Systemverständnis
- Berücksichtigung unerwarteter und zufälliger Ereignisse
- Untersuchung des Einflusses einzelner Ereignisse auf das gesamte Netzwerk
- Risikominimierung bei Planungs- und Konfigurationsänderungen

In Abschnitt 3.2.2 wurde die Tourenplanung als häufig verwendetes Optimierungsproblem in Modellen von Logistiknetzwerken identifiziert. Zur Lösung des Problems bietet sich nicht nur die Optimierung an, sondern auch eine kombinierte Anwendung von Optimierung und Simulation. Dies kann zu einer hierarchischen Architektur führen, in der die Simulation die führende bzw. dominante Methode ist, die die untergeordnete Optimierung steuert und deren Ergebnisse verwendet (März und Krug 2011). Es sind auch alternative Architekturen denkbar, wie beispielsweise die Optimierung als führendes System. Eine Kombination von Simulation mit einer Meta-Heuristik (eine Heuristik als führendes System) wird nach Juan und Rabe (2013) als Simheuristik bezeichnet.

Ein Anwendungsbereich in der Logistik Domäne ist die City Logistik. Sie beschäftigt sich mit der effektiven Gestaltung von Transporten in urbanen Gebieten (Savelsbergh und van Woensel 2016). Eine der ersten Simulationsstudien auf diesem Gebiet wurde von McDermott im Jahr 1975 durchgeführt und behandelt Tourenplanung in New York City. Neuere Untersuchungen wurden beispielsweise von Rabe et al. (2018) durchgeführt, die die Einführung urbaner Konsolidierungszentren in der Metropolregion Athen (Griechenland) mithilfe diskreter Ereignissimulation und dazugehöriger Tourenplanung bewerteten. Voruntersuchungen zu der Simulationsstudie in Athen sind in Gruler et al. (2017) nachzulesen, die detaillierter auf die Tourenplanung in einer zweistufigen Supply Chain mithilfe des Savings-Algorithmus (siehe Abschnitt 3.2.2) eingehen. Ein Beispiel eines Simulators auf Basis des Bausteinprinzips ist in Staritz et al. (2021) zu finden. Dort wird der Einfluss autonomer Fahrzeuge auf Verkehr, Umwelt und ökonomischen Aspekten mit dem Simulator AnyLogic (The AnyLogic Company 2022a) untersucht.

Weitere Beispiele für standortübergreifende Simulation von Logistiknetzwerken können im Bereich des Werkstoffhandels gefunden werden. Rabe et al. (2017b) beschreiben die Kombination ereignisdiskreter Simulation im Verbund mit SimHeustrik im Rahmen eines Entscheidungsunterstützungssystems. In einer anderen Arbeit von Rabe et al. (2017a) wird eine konzeptionelle Verknüpfung der ereignisdiskreten Simulation mit einem agentenbasierten Ansatzes mit bestärkendem Lernen zur Entscheidungsunterstützung in Logistiknetzwerken des Werkstoffhandels beschrieben.

4 Entwicklung eines Simulators für Logistiknetzwerke

Auf Basis der in Kapitel 2 erörterten technischen Grundlagen und dem in Kapitel 3 vorgestellten logistischen Kontext erfolgt in diesem Kapitel die Entwicklung eines Simulators für Logistiknetzwerke.

Der Name des Simulators lautet *LogFarm* und leitet sich aus der Problemdomäne der Logistik („Log“) sowie dem beabsichtigten Verwendungszweck innerhalb eines Data-Farming-Frameworks („Farm“) ab. Das Vorgehen bei der Entwicklung orientiert sich an dem Wasserfallmodell von Fairley (1985), das eine vereinfachte Form des ursprünglichen Wasserfallmodells von Royce (1970) ist. Es stellt ein Vorgehensmodell zur Softwareentwicklung dar und besteht aus fünf Phasen. Das Wasserfallmodell ist in Abbildung 4.1 abgebildet.

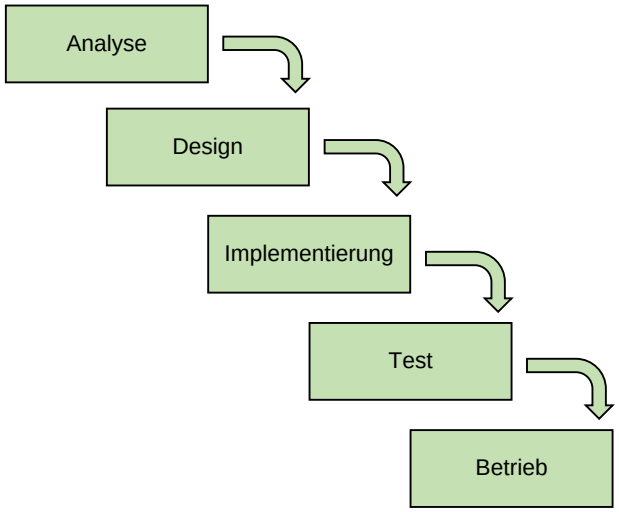


Abbildung 4.1: Wasserfallmodell nach Fairley (1985)

Wie in Abbildung 4.1 zu erkennen ist, startet das Vorgehen in der ersten Phase mit dem Namen Analyse. In dieser Phase werden die Anforderungen, die sich an die Anwendung ergeben, abgeleitet, spezifiziert und dokumentiert. Die beschriebenen Schritte werden für den Simulator in Abschnitt 4.1 vorgenommen. Die zweite Phase ist das Design des Systems. Zusammenfassend werden im Systemdesign die Struktur, Abläufe und Schnittstellen der Software erarbeitet. In Abschnitt 4.2 wird zunächst die Struktur der Simulationsmodelle durch ein Konzeptmodell festgelegt und anschließend das Softwaredesign des Simulators in Abschnitt 4.3 beschrieben. Die dritte Phase des Wasserfallmodells ist die Implementierung. In dieser Phase wird die geplante Anwendung samt aller Komponenten realisiert. Die Komponenten werden getestet, um einen fehlerfreien Betrieb sicherzustellen. Die Implementierung eines Prototypen des Simulators ist in Abschnitt 4.4 beschrieben. Zu der Implementierung des Prototyps gehört auch eine grafische Benutzeroberfläche, die in Ab-

schnitt 4.5 vorgestellt wird. Phase vier des Wasserfallmodells ist der Test der Software. Diese Phase umfasst umfangreiche Softwaretests. Die fünfte und letzte Phase ist der Betrieb, in der die Software in den Realbetrieb übergeht und laufend gewartet und verbessert werden soll. Die letzten beiden Phasen des Wasserfallmodells werden nicht explizit in dieser Arbeit behandelt, stattdessen werden in Kapitel 5 zwei beispielhafte Anwendungen des Simulators anhand von Fallbeispielen ausführlich beschrieben und diskutiert.

4.1 Ableitung und Spezifikation von Anforderungen

Damit eine zielgerichtete Entwicklung des Simulators möglich ist, müssen Anforderungen an die Anwendung identifiziert und spezifiziert werden. Eine Anforderung beschreibt eine Bedingung, die erfüllt sein muss, oder eine Fähigkeit, die eine Anwendung besitzen muss, um der Zielbeschreibung zu entsprechen (Pohl und Rupp 2015). Anforderungen können als funktional oder nicht-funktional klassifiziert werden. Funktionale Anforderungen haben direkten Einfluss auf das Ergebnis der Anwendung und werden als spezifisch für eine Anwendung angesehen. Nicht-funktionale Anforderungen hingegen beschreiben Qualitätsanforderungen, wie beispielsweise die Performanz, Verfügbarkeit oder Skalierbarkeit der Anwendung (Pohl und Rupp 2015). In diesem Abschnitt werden drei funktionale und fünf nicht-funktionale Anforderungen spezifiziert und diskutiert.

Anforderung I: Die Anwendung muss Daten eines Logistiknetzwerks in Form eines Graphen mittels Data Farming generieren

Diese Anforderung stellt den Zweck der Anwendung dar und wird direkt aus dem Hauptziel der vorliegenden Arbeit abgeleitet. Hierdurch wird festgelegt, dass der Simulator für die Methode Data Farming (vgl. Abschnitt 2.3.3) verwendet werden soll und somit innerhalb eines Data-Farming-Frameworks Verwendung finden kann. Als Simulationsgegenstand sind Logistiknetzwerke spezifiziert, welche in Abschnitt 3.3 beschrieben werden. Die durch das Data Farming erhaltenen Daten sollen nicht in tabellarischer Form vorliegen, sondern in Form von Graphen.

Da für den Simulationsgegenstand Logistiknetzwerke keine Einschränkungen vorliegen, werden generische Logistiknetzwerke betrachtet, die nicht branchenspezifisch sind oder eine bestimmte Architektur aufweisen. Logistiknetzwerke sind komplexe Systeme, in denen eine Vielzahl von Prozessen ablaufen und verschiedene Logistikaufgaben bearbeitet werden (vgl. Abschnitt 3.3). Aufgrund diesen Umstands ist eine allumfassende Simulation in dieser Arbeit nicht zu realisieren und sollte eingegrenzt werden. LogFarm soll den Fokus auf die logistischen Kernaufgaben legen, die in Abschnitt 3.1 als die TUL-Prozesse identifiziert wurden. Zudem sollen die Prozesse, die innerhalb der Standorte der Teilnehmer der Logistiknetzwerke ablaufen, nicht im Detail betrachtet werden. Das bedeutet, dass die Betrachtungsebene hauptsächlich Standorte und deren Beziehungen untereinander einschließt.

Anforderung II: Die Anwendung muss graphbasiert arbeiten

Diese Anforderung lässt sich ebenfalls aus dem Hauptziel der Arbeit ableiten. Neben der Generierung von Daten in Form eines Graphen soll die Anwendung weitestgehend graphbasiert arbeiten. Dafür muss geklärt werden, was unter dem Begriff graphbasierter Simulator verstanden wird. In dieser Arbeit wird ein Simulator als graphbasiert verstanden, wenn die intern verwendete Datenhaltung ähnlich dem Graphdatenmodell erfolgt und einige der verwendeten Algorithmen auf Basis von Graphen durchgeführt werden. Für den

Simulator bedeutet das insbesondere, dass die zu simulierenden Simulationsmodelle in einer Graphenrepräsentation vorliegen und Vorteile durch graphbasierte Lösungsansätze aus diesem Umstand gezogen werden. Entsprechend der Anforderung einer internen Datenhaltung ähnlich des Graphdatenmodells sollen die Objekte und Daten der Programmierung lediglich über Relationen verbunden werden und nicht in tabellarischer Form vorliegen.

Da die Objekte in einem Logistiknetzwerk, wie beispielsweise Standorte, Artikel oder Fahrzeuge, vielfältige Eigenschaften aufweisen und zudem deren Beziehungen oftmals mit Eigenschaften beschrieben werden können, bietet sich die Verwendung des Eigenschaftsgraphen-Modells als spezielle Art des Graphenmodells an (vgl. Abschnitt 2.3.1). Es sei angemerkt, dass zur Beschreibung von Gütern im Zusammenhang von LogFarm zumeist der Begriff *Artikel* verwendet wird.

Anforderung III: Die Anwendung muss eine grafische Benutzeroberfläche aufweisen und eine interaktive Modellbildung erlauben

Zur Erfüllung dieser Anforderung muss mithilfe einer geeigneten Bibliothek eine grafische Oberfläche erstellt werden. Das Ziel der grafischen Benutzeroberfläche ist es, dem Benutzer den Funktionsumfang der Anwendung auf einfache Weise zugänglich zu machen. Dazu gehören beispielsweise die Funktionen zur Parametrisierung, Durchführung und Überwachung von Experimenten, die zur Anwendung des Data Farmings benötigt werden. Zudem können V&V Methoden, wie beispielsweise die Animation, einfach in eine Benutzeroberfläche integriert werden.

Um Simulationen mit dem Simulator durchführen zu können ist es notwendig, dass ein ausführbares Modell erstellt wird (vgl. Abschnitt 2.1). Das Modell muss entsprechend dem vorgesehenen Datenbankschema erstellt werden. Um diesen Vorgang zu unterstützen, soll eine grafische Benutzeroberfläche entwickelt werden, in der der Anwender interaktiv ein valides Modell erstellen kann. Diese grafische Benutzeroberfläche, die in die gesamte Benutzeroberfläche der Anwendung eingebettet werden soll, wird fortan als Editor bezeichnet. Durch die Verwendung eines Editors kann sichergestellt werden, dass der Nutzer ein Modell entsprechend der vorgegebenen Syntax und Semantik des Konzeptmodells der Simulationsmodelle erstellen kann. Für das Modellierungskonzept (vgl. Abschnitt 3.3.1) des Editors soll das bausteinorientierte Konzept angewendet werden, das sich insbesondere für die interaktive Modellbildung logistischer Systeme eignet (Gutenschwager et al. 2017) und durch die spezialisierten Bausteine genau auf den gebotenen Funktionsumfang von LogFarm angepasst werden kann. Der Simulator soll als geschlossenes Werkzeug (vgl. Abschnitt 3.3.1) ausgeführt werden. Das bedeutet, der Benutzer kann keine eigenen Bausteine der Bausteinbibliothek hinzufügen. Das Ziel ist, bereits durch die vorgegebenen Bausteine eine geeignete Modellierung einer Vielzahl von Logistiknetzwerken zu ermöglichen.

Anforderung I bis Anforderung III stellen die funktionalen Anforderungen zur Ausarbeitung der Zweckbestimmung von LogFarm Anwendung dar. Es folgen nun die fünf nichtfunktionalen Anforderungen.

Anforderung IV: Der Simulator muss ein agentenbasiertes Modell durch kontinuierliche Simulation verarbeiten und stochastische Vorgänge abbilden können

Aus der Methode der Simulation lässt sich diese Anforderung ableiten. Durch die netzwerkartige Struktur von Logistiknetzwerken, in denen verschiedene Akteure agieren, bietet sich für die Modellierung ein agentenbasierter Ansatz an (vgl. Abschnitt 2.1.1). Die Akteure der Logistiknetzwerke werden durch Agenten abgebildet, die selbstständig Entscheidungen treffen. Aufgrund der in Anforderung I beschriebenen Betrachtungsebene, die eine

vergleichsweise makroskopische Sicht darstellt, kann auch der Abstand zwischen zwei Zeitpunkten, zu denen das Modell simuliert wird, vergleichsweise groß gewählt werden. Durch die Wahl verhältnismäßig großer Zeitabstände kann davon ausgegangen werden, dass stetige Veränderungsprozesse zwischen den Abständen stattfinden und somit die Simulation zeitgesteuert und kontinuierlich erfolgen kann (vgl. Abschnitt 2.1.1). Das bedeutet, die Simulation soll in äquidistanten Zeitschritten durchgeführt werden. Die Festlegung des verwendeten Zeitschritts erfolgt in Abschnitt 4.2. In Abschnitt 3.3.1 wurde beschrieben, dass es für Modelle in der Logistik charakteristisch ist, stochastische Prozesse aufzuweisen. Daher muss auch LogFarm in der Lage sein, stochastische Prozesse angemessen abzubilden.

Anforderung V: Die Anwendung muss große Datenbestände verarbeiten können

Zwar stellen das ausführbare Modell und der externe Experimentierplan keine großen Datenmengen dar, jedoch besteht bei der Anwendung von Data Farming für gewöhnlich die Absicht, große Datenbestände (vgl. Abschnitt 2.3) zu generieren. Dementsprechend muss die Anwendung die anfallenden Datenmengen verarbeiten und in geeigneter Weise halten können.

Für die Datenhaltung soll ein Graphdatenbanksystem verwendet werden, das in Abschnitt 2.3.1 bereits als bevorzugtes Mittel zur Datenhaltung großer Datenbestände mit starker Vernetzung vorgestellt wurde. Die Entscheidung beruht auf drei Argumenten. Da der zu entwickelnde Simulator darauf ausgerichtet ist, große Datenbestände zu generieren, bietet sich die Verwendung einer Graphdatenbank in diesem Aspekt an. Ein weiteres Argument ist, dass wenn die Anwendung, wie durch Anforderung II gefordert, graphbasiert arbeitet, die interne Struktur bereits einem Graph ähnelt und keine Transformation beispielsweise in das relationale Modell nötig wird. Als letztes Argument wird die erwartete Vernetzung der zu speichernden generierten Daten angeführt. Zwar hängt der Grad der Vernetzung maßgeblich von dem verwendeten Simulationsmodell ab, jedoch sind Logistiknetzwerke in der Regel komplexe Systeme die in Netzwerken modelliert werden und somit auch die Ergebnisdaten starke Vernetzungen aufweisen.

Anforderung VI: Die Anwendung muss leistungsstarke Berechnungen unterstützen

Ein Grundbestandteil des Data Farmings ist die leistungsstarke Berechnung (vgl. Abschnitt 2.3.3). Eine konkrete Rechenleistung, die mindestens erreicht werden sollte, wird nicht spezifiziert. Vielmehr soll hervorgehoben werden, dass große Simulationsexperimente und die Verarbeitung großer Datenmengen rechenintensive Prozesse sind, die in annehmbarer Zeit erledigt werden müssen. Was als annehmbare Zeitspanne gilt, ist ein subjektives Empfinden des Nutzers und abhängig vom beabsichtigten Anwendungsfall. Wird die Simulation beispielsweise dazu verwendet, Entscheidungen im operativen Bereich zu unterstützen, müssen die Ergebnisse schneller vorliegen, als in taktischen oder strategischen Bereichen. Um die Wahrscheinlichkeit zu steigern, dass der Data-Farming-Prozess in annehmbarer Zeit durchgeführt werden kann, sollte eine möglichst gute Auslastung der verfügbaren Rechenkapazitäten des Computers realisiert werden. Zusätzlich zur Optimierung einzelner Computer soll es möglich sein, den Data-Farming-Prozess mit dem Simulator in einem Cluster von Computern auszuführen. Dadurch wird es dem Nutzer ermöglicht, auf einfache Weise die Rechenleistung zu erhöhen.

Im Rahmen dieser Arbeit wird auf die Umsetzung und Erprobung der Anwendung hinsichtlich der Fähigkeit zur Clusterung mehrerer Computer in einem Netzwerk verzichtet. Die zu entwickelnde Anwendung soll jedoch prinzipiell die Verteilung der Rechenlast auf mehrere Computer unterstützen. In dieser Arbeit liegt der Fokus bei der Realisierung der

leistungsstarken Berechnung auf der Optimierung auf einem Computer. Dazu wird eine parallele Simulation bzw. Ausführung der Experimente angestrebt, um von den Möglichkeiten eines Mehrkern-Computers Gebrauch zu machen. Jedem zur Verfügung stehenden Kern soll die Ausführung eines Teils des Experimentplans zugewiesen werden um die Gesamtrechenzeit zur vollständigen Abarbeitung des Experimentplans zu reduzieren.

Anforderung VII: Die Anwendung muss einen extern erstellten Experimentierplan verwenden können

Ein weiterer essentieller Bestandteil des Data Farmings ist die statistische Versuchsplanung (vgl. Abschnitt 2.1.2). Wie in Abschnitt 2.1.2 beschrieben, existiert bereits eine große Auswahl von Softwarelösungen, die eine statistische Versuchsplanung vornehmen können, um einen Experimentierplan zu erhalten. Die Anwendung soll sich diesen Umstand zu Nutze machen und die Verwendung eines extern berechneten Experimentierplans ermöglichen. Dazu soll eine geeignete Schnittstelle zur Verfügung gestellt werden, über die mit einem vordefinierten Datenformat der Experimentierplan der Anwendung zugeführt werden kann.

Als Datenformat zum Austausch des Experimentplans werden Comma Separated Values (CSV) in einem Format, kurz CSV-Format, verwendet werden (Shafranovich 2005). Es stellt eines der einfachsten und am weitesten verbreiteten Dateiformate zum Informationsaustausch dar und wird durch viele Softwarelösungen zur Generierung von Experimentierplänen verwendet. Das Format besteht aus Datensätzen, die mehrere Datenfelder haben, in denen jeweils ein Datum gespeichert ist. Die Datensätze werden zumeist über Zeilenumbrüche getrennt und die Datenfelder durch Kommata. Zudem ist das Format menschenlesbar und mit vielen Anwendungen kompatibel. Die Schnittstelle zum Einlesen des Experimentierplans ist ein Leseprozess, den der Anwender über die grafische Benutzeroberfläche mit einem Bedienelement initiieren kann. Es muss sichergestellt werden, dass die mit dem Editor von LogFarm erstellten Simulationsmodelle Faktoren aufweisen können, die durch den Experimentplan gesteuert werden können.

Anforderung VIII: Die Anwendung muss weiterentwickelt werden können

Die in dieser Arbeit vorgestellte Form des Simulators stellt lediglich einen Prototypen dar, der ein grundsätzliches Vorgehen bei der Entwicklung graphbasierter Simulatoren im Bereich der Logistiknetzwerke aufzeigt. Um für anschließende Arbeiten, insbesondere durch Dritte, verwendbar zu sein, muss eine ausreichende Dokumentation und Verfügbarkeit des Programmcodes des Simulators öffentlich zugänglich sein. Für die Dokumentation wird zunächst auf die vorliegende Arbeit und die ausführlichen Kommentare im Programmcode selbst verwiesen. Um die öffentliche Verfügbarkeit des Codes zu gewährleisten, soll eine Veröffentlichung des gesamten Programmcodes und der dazugehörigen Ressourcen im Internet angestrebt werden. Somit stellt LogFarm eine Open-Source-Software dar, die durch Dritte eingesehen, geändert und genutzt werden kann. Die Veröffentlichung soll unter der Lizenz GPLv3 erfolgen GNU (2022).

Um der internationalen Forschungsgemeinschaft die Zugänglichkeit zu LogFarm zu erleichtern, soll die Sprache des Programmcodes und der grafischen Benutzeroberfläche Englisch sein. Dementsprechend sollen beispielsweise auch die Zahlenformate an den englischsprachigen Raum angepasst werden.

Die abgeleiteten Anforderungen stellen die erste Phase des Wasserfallmodells dar. Im nächsten Abschnitt wird basierend auf den Anforderungen ein Konzeptmodell für Simulationsmodelle von Logistiknetzwerken entwickelt, was der zweiten Phase des Wasserfallmodells zugeordnet werden kann.

4.2 Konzeptmodell eines generischen Logistiknetzwerks

In diesem Abschnitt wird ein Konzeptmodell eines generischen Logistiknetzwerks entworfen, welches als Metamodell für solche Modelle dient, die der Simulator verarbeiten kann. Der Modellzweck des Metamodells ist daher die Beschreibung und Festlegung eines Rahmens, innerhalb dessen Modelle von Nutzern erstellt werden können. Zur Erarbeitung des Konzeptmodells wird anhand des Top-down-Ansatzes (vgl. Abschnitt 2.1.1) vorgegangen. Das bedeutet, dass ausgehend vom Simulationsgegenstand die nötigen Detaillierungen vorgenommen werden.

Zunächst wird eine Systemanalyse durchgeführt (vgl. Abschnitt 2.1.1). Sämtliche Logistiknetzwerke weisen dynamische Prozesse auf, das heißt, sie verändern sich mit zeitlichem Fortschritt (vgl. Abschnitt 2.1.1). Dementsprechend müssen auch Modelle zur Abbildung von Logistiknetzwerken dynamisch sein. Ein weiterer Teil der Systemanalyse ist die Festlegung des Umfangs und des Detaillierungsgrads des Modells. In Abschnitt 3.3 wurde aufgezeigt, dass Logistiknetzwerke komplexe Systeme sind und eine umfassende Abbildung folgerichtig umfangreich ist. In Anforderung I wurde bereits beschrieben, dass die Betrachtungsebene der Logistiksysteme auf Standorte und deren Beziehungen untereinander fokussiert ist und die TUL-Prozesse als Kernleistung der Logistik abgebildet werden sollen. Bei der Abbildung der TUL-Prozesse wird bei den Transport- und Lagerprozessen ein höherer Detailgrad verwendet als bei Umschlagsprozessen. Innerbetriebliche Transportprozesse werden nicht abgebildet. Daher sind im folgenden, wenn von Transportprozessen gesprochen wird, speziell Distributionsprozesse gemeint, die außerbetriebliche Transporte darstellen.

Neben den logistischen Aufgaben der TUL-Prozesse werden zwei weitere logistische Aufgaben abgebildet. Zur Unterstützung der Lagerprozesse ist es möglich, die Bestandsplanung zu integrieren. Spezifisch können den Lägern einstufige Lagerhaltungspolitiken zugewiesen werden, mit denen regelbasierte Nachbestellungen von Gütern erfolgen (vgl. Abschnitt 3.2.1). Die zweite zusätzlich abbildbare Aufgabe unterstützt Transportprozesse. Gegeben dem Fall, dass ein Akteur des Logistiknetzwerks Bestellungen von mehreren Kunden an einem Tag bedienen will, bietet sich die Verwendung der Tourenplanung an (vgl. Abschnitt 3.2.2). Bei der Tourenplanung wird die Anzahl der zur Verfügung stehenden Fahrzeuge und deren Kapazitäten berücksichtigt.

Ein Aspekt bei der Festlegung des Detaillierungsgrads ist die Auswahl der zeitlichen Auflösung. Durch die Auswahl eines niedrigen Detaillierungsgrads kann die zeitliche Auflösung niedrig gewählt werden. Das bedeutet, es finden diskrete Zeitsprünge mit vergleichsweise hohem Abstand statt, wie in Anforderung IV gefordert. Für die Modellierung der Logistiknetzwerke auf Standortebene wird eine Simulation auf Tagesbasis als ausreichend genau festgelegt. Somit betragen die äquidistanten Zeitschritte der Simulation einen Tag bzw. 24 Stunden. Jedoch werden einige Eigenschaften des Logistiknetzwerks, wie beispielsweise die Produktionsrate von Gütern, stundenbezogen angegeben, um verschiedene Längen von Arbeitstagen zu berücksichtigen. Die stundenbezogenen Werte werden für jeden Simulationsschritt mit der Arbeitszeit des betreffenden Tages multipliziert, um die Tageswerte zu erhalten. Die Länge der Arbeitstage in Stunden und Daten über Betriebsferien oder Feiertage werden in einem Kalender innerhalb des Modells gesammelt. Dabei kann jedem betroffenen Element des Modells ein individueller Kalender zugewiesen werden. Da keine Modellierung auf Stundenebene erfolgt, werden keine Zeitfenster, wie z. B. von neun Uhr bis 17 Uhr spezifiziert, sondern nur die Anzahl der Stunden hinterlegt.

Die Systemelemente in einem Logistiknetzwerk sind die darin handelnden Akteure (vgl. Abschnitt 3.3). Die abzubildenden Akteure in LogFarm werden entsprechend der Definition von Simchi-Levi et al. (2000) wie folgt festgelegt:

- Zulieferer
- Produktionsstätten
- Warenhäuser
- Kunden

In der Aufzählung werden die von Simchi-Levi et al. (2000) genannten Distributionszentren nicht berücksichtigt, da diese in dem Metamodell den Warenhäusern gleichgesetzt werden. Im weiteren Verlauf der Arbeit werden die aufgelisteten Ausprägungen von Akteuren auch als Rollen bezeichnet.

Zwischen den genannten Akteuren soll der Austausch von Gütern und Informationen möglich sein. Unter Gütern werden Rohstoffe, Halbzeuge, unfertige Produkte und Endprodukte verstanden. Zur weiteren Begrenzung des Modellumfangs und Detaillierungsgrads wird der Austausch von Informationen auf das Aufgeben von Bestellungen begrenzt.

Der Austausch von Gütern soll über Relationen erfolgen, die Transportstrecken, wie Straßen oder Schienenwege, abbilden. Im Folgenden werden diese Relationen als Transportrelationen bezeichnet. Die Relationen, über die Bestellungen abgewickelt werden, sind Lieferantenbeziehungen. Zwischen Akteuren können entweder keine oder genau eine Transportbeziehung bestehen, sowie keine oder genau eine Lieferantenbeziehung.

Die Punkte, an denen Güter in das Logistiknetzwerk ein- oder austreten, bilden die Systemgrenze. Nach dem Fließprinzip werden Eintrittpunkte Quellen und Austrittspunkte Senken genannt. Über die Systemgrenze wird lediglich der Austausch von Gütern erlaubt, der Austausch von Informationen wird nicht betrachtet. Jedes Modell muss mindestens eine Quelle und eine Senke aufweisen, um einen Güterfluss zu ermöglichen. Ohne Quelle würden, abgesehen von einem Initialbestand, keine Güter in dem Logistiknetzwerk existieren und ohne eine Senke würden alle Lager kontinuierlich gefüllt werden. Um einen Güterfluss zu initiieren, müssen neben den primären TUL-Prozesse also auch solche Prozesse des Systems abgebildet werden, die neue Güter in das System ein- oder ausbringen. In dieser Arbeit werden zu den einbringenden Prozessen Produktions- und Transformationsprozesse gezählt. Für ausbringende Prozesse werden Konsum- und zugehörige Nachfrageprozesse betrachtet.

Ein weiterer Teil des Konzeptmodells ist die Abbildung stochastischer Prozesse, um Anforderung IV zu erfüllen. Dabei können zwei Arten von Prozessen abgebildet werden. Zum einen soll die Nachfrage eines Akteurs des Logistiknetzwerks stochastisch modelliert werden können, um schwankende Bedarfe nachzubilden. Zum anderen sollen die Produktions- und Transformationsprozesse Störungen unterliegen können, deren Intensität durch stochastische Prozesse bestimmt werden.

Das Ziel des Data Farmings ist die Generierung von Daten. In LogFarm sollen vorerst zwei Arten von Daten generiert werden. Dazu gehören einerseits Bestellungen und andererseits Berichte über Lagerbestände. Die Betrachtung von Bestellungen erlaubt Einblicke in das transaktionale Verhalten des Logistiknetzwerks und kann beispielsweise Lieferengpässe aufdecken. Durch Berichte über die Lagerbestände, die täglich erfolgen sollen, kann beispielsweise wichtiges Wissen zur Reduzierung der Lagerbestände eines Logistiknetzwerks erlangt werden.

Im anschließenden Abschnitt wird das vorgestellte Konzeptmodell formalisiert und grundlegende Designentscheidungen für LogFarm getroffen.

4.3 Softwaredesign

Im Softwaredesign sollen die wichtigsten Aspekte der Software entworfen und beschrieben werden. Dazu gehören die Softwarearchitektur, das Datenbankschema und die grafischen wie nicht-grafischen Schnittstellen (Sommerville 2016). Eine entsprechende Beschreibung dieser Aspekte für den zu entwickelnden Simulator werden in diesem Abschnitt dargelegt. Zudem wird auch der Simulatorkern und ein Experiment Manager entworfen sowie das algorithmische Design der exemplarisch bearbeiteten, logistischen Aufgaben der Bestandsplanung und Tourenplanung beschrieben.

4.3.1 Bausteinbibliothek und komponentenbasierte Bausteine

In Anforderung III wird gefordert, dass Logfarm als geschlossener Simulator im Bausteinprinzip entwickelt werden soll. In diesem Abschnitt werden die Bausteine identifiziert, die dem Nutzer in der angebotenen Bausteinbibliothek zur Verfügung gestellt werden. Darüber hinaus wird die Gliederung der Funktionalitäten der Bausteine in Komponenten vorgestellt.

In Abschnitt 3.3.1 wurde das bausteinorientierte Modellierungskonzept im Allgemeinen bereits erläutert. Mit den Bausteinen muss es möglich sein, jegliche im Konzeptmodell (vgl. Abschnitt 4.2) herausgearbeiteten Aspekte in einem ablauffähigen Modell zu realisieren. Dazu wird zunächst ein Modell mit Bausteinen erstellt und parametrisiert, bevor es durch den Editor in das ablauffähige Modell überführt wird. Um alle Aspekte des Konzeptmodells abbilden zu können, müssen alle Akteure des Logistiknetzwerks, deren primären und nebensächlichen Prozesse sowie die Beziehungen zwischen den Akteuren, abgebildet werden. Entsprechend dem Konzeptmodell soll dies auf einer aggregierten, makroskopischen Ebene auf Standortbasis geschehen. Als fundamentale Bausteine werden daher die Standorte der Akteure des Logistiknetzwerks festgelegt. Um den Funktionsumfang der Bausteine zu definieren, werden zunächst Gemeinsamkeiten und Unterschiede der Akteure in dem Konzeptmodell diskutiert.

Allen Akteuren in einem Logistiknetzwerk ist gemeinsam, dass sie einen identifizierenden Namen und einen physischen Ort aufweisen. Der physische Ort kann über eine geographische Koordinate spezifiziert werden, die aus einem Längen- und Breitengrad besteht. Weiterhin ist es allen Akteuren möglich, Beziehungen zu anderen Akteuren des Logistiknetzwerks zu haben und mit diesen Güter oder Informationen auszutauschen. Auch weisen die Akteure eine individuelle tägliche Arbeitszeit auf, die für jeden Wochentag unterschiedlich ausfallen kann. Folglich besitzen alle Akteure eine gemeinsame Basis.

Eine Unterscheidung der Akteure kann hinsichtlich der von ihnen erbrachten bzw. bei ihnen ablaufenden Prozesse erfolgen. In Abschnitt 4.2 wurde bereits festgelegt, dass Distributions-, Lager, Produktions-, Transformations-, Konsum- und Nachfrageprozesse betrachtet werden. Auch wenn es zwischen den Akteuren eine Schnittmenge hinsichtlich der Prozesse gibt, ist eine eindeutige Identifizierung der Rolle des Akteurs in dem Logistiknetzwerk über die Gesamtmenge der Prozesse möglich. Beispielsweise ist die Rolle eines Akteurs, der lediglich Nachfrage-, Distributions- und Lagerprozesse abbildet, eindeutig als Warenhaus zu identifizieren, obwohl eine Produktionsstätte diese Prozesse auch abbilden könnte,

jedoch zusätzlich Produktions- und Transformationsprozesse aufweist. Eine tabellarische Übersicht, in der ersichtlich ist, welche Prozesse für eine Rolle identifizierend sind, ist in Tabelle 4.1 dargestellt.

Tabelle 4.1: Rollen in einem Logistiknetzwerk und ihre Prozesse

Rolle	Prod.	Tran.	Kons.	Nach.	Lage.	Dist.
Zulieferer	●	○	-	-	●	●
Produktionsstätten	○	●	○	○	●	●
Warenhäuser	-	-	-	○	●	●
Kunden	-	-	●	●	-	-

Legende:
● zwingend ○ optional - unzulässig
Prod. = Produktionsprozesse Trans. = Transformationsprozesse Kons. = Konsumprozesse
Nach. = Nachfrageprozesse Lage. = Lagerprozesse Dist. = Distributionsprozesse

Die Abbildung der sechs Prozesse wird im Kontext des Simulators durch jeweils eine Komponente realisiert. Die Komponenten stellen gekapselte Funktionalitäten dar und können kombiniert werden. Bausteine sind die übergeordnete Struktur, der die Komponenten zugehörig sind. Der Funktionsumfang eines Bausteins lässt sich aus den zugehörigen Komponenten ableiten. Somit muss für die Festlegung der Funktionsumfänge der Bausteine spezifiziert werden, welche Komponenten sie aufweisen. In den Komponenten zur Beschreibung der Produktions- und Transformationsprozesse können optional Störungen eingebracht werden.

Zusätzlich zu den sechs Komponenten zur Beschreibung von Prozessen gibt es eine Standortkomponente, die jeder Akteur im Logistiknetzwerk aufweist und Informationen zum Namen, täglicher Arbeitszeit, physischem Ort, sowie Verbindungen zu anderen Akteuren beinhaltet.

Bausteine sind Vorlagen, mit denen Standorte von Akteuren mit einer spezifischen Rolle in einem Logistiknetzwerk abgebildet werden können. Jeder Baustein weist eine Standortkomponente auf und eine zulässige Kombination von Komponenten der sechs abzubildenden Prozesse zur Erfüllung seiner Rolle gemäß Tabelle 4.1. Das bedeutet, dass die Komponenten, deren Prozesse in der Tabelle als zwingend angegeben werden, in dem Baustein immer vorhanden sind. Komponenten, deren Prozesse als optional spezifiziert wurden, können dem Baustein optional durch den Benutzer hinzugefügt werden. Unzulässige Prozesse können einem Baustein nicht verfügbar gemacht werden.

Neben den vier bereits vorgestellten Bausteinen gibt es weiterhin den Kundengruppenbaustein. Damit soll es dem Benutzer möglich gemacht werden, viele Kunden ohne viel Aufwand zu erzeugen. Sollen beispielsweise in einem Logistiknetzwerk 100 Kunden betrachtet werden, wäre die Modellierung und Parametrisierung jedes Kunden mit einem Baustein sehr aufwendig. Der Kundengruppenbaustein weist dieselben Komponenten wie ein Kundenbaustein auf. Die Komponenten und deren Parametrisierung dienen als Vorlage und werden auf alle durch die Kundengruppe modellierten Kunden übertragen. Der Unterschied zu einem normalen Kundenbaustein besteht darin, dass eine Kundengruppe kein Objekt des Simulationsmodells auf Datenebene ist, sondern bei der Überführung in ein ausführbares Simulationsmodell eine vorher spezifizierte Anzahl von Kunden entsprechend der Vorlage generiert wird. Dazu kann im Kundengruppenbaustein die Zahl zu generierender

Kunden festgelegt werden. Außerdem kann ein geografischer Raum spezifiziert werden, in dem die Kunden zufällig und gleich verteilt generiert werden. Der geografische Raum wird durch vier Werte beschrieben, die den minimalen und maximalen Längen- sowie Breitengrad darstellen. Die zufällige Verteilung der Kunden findet während der Modellerstellung statt und wird danach für das Simulationsmodell festgelegt.

4.3.2 Datenanalyse und Datenbankschema

In diesem Abschnitt wird das Konzeptmodell aus Abschnitt 4.2 auf der Datenebene modelliert. In Abschnitt 2.3.2 wurde beschrieben, dass für die Datenmodellierung zunächst eine Datenanalyse erfolgt, darauf basierend ein konzeptionelles Datenmodell erstellt und schließlich ein Datenbankschema abgeleitet wird. Das konzeptionelle Entitäten-Beziehungsmodell-Diagramm eines Logistiknetzwerks in LogFarm ist in Abbildung 4.2 abgebildet.

Eine vom Autor durchgeführte Datenanalyse hat das konzeptionelle Datenmodell des Simulators, wie im Entitäten-Beziehungsmodell-Diagramm (Kemper und Eickler 2015) in Abbildung 4.2 abgebildet, ergeben. Dort sind die Zusammenhänge des Konzeptmodells auf einer hohen Abstraktionsebene dargestellt. Der Abbildung ist zu entnehmen, dass ein Logistiknetzwerk aus mindestens einem Akteur bestehen muss, um zu existieren. Die Funktionsfähigkeit eines Logistiknetzwerks wird jedoch erst durch mindestens zwei Akteure hergestellt. Die Akteure weisen die Attribute Standort-Nummer, Name, Arbeitszeit und geografische Koordinate auf. Zudem ist der Abbildung zu entnehmen, dass jeder Akteur eine beliebige Anzahl von Relationen zu anderen Akteuren besitzen kann. Akteure haben die Möglichkeit, beliebig viele unterschiedliche Artikel bereitzustellen, zu lagern oder zu verbrauchen. Artikel werden durch eine Artikelnummer charakterisiert und tragen einen Namen zur besseren Lesbarkeit der Daten. Auf weitere Attribute, wie beispielsweise Volumen, Gewicht oder Wert der Artikel wird verzichtet, um den Entwicklungsaufwand zu limitieren. Der Austausch von Artikeln zwischen den Akteuren findet über Bestellungen statt. Bestellungen werden durch eine Bestellnummer, ein Bestelldatum und ein Lieferdatum beschrieben. Zudem muss einer Bestellung mindestens ein Artikel zugewiesen sein. Bestellungen können von Akteuren aufgegeben oder erfüllt werden. Zur tatsächlichen Erfüllung von Bestellungen werden Fahrzeuge verwendet. Akteure verwenden eine beliebige Anzahl von Fahrzeugen, die die Artikel transportieren können. Die Fahrzeuge sind durch eine Fahrzeug-Nummer eindeutig identifizierbar und weisen zudem einen Modellnamen und eine Kapazität auf. Die Kapazität bezieht sich lediglich auf die Menge der transportierbaren Artikel, da keine Informationen über Volumen oder Gewicht vorliegen.

Im weiteren Verlauf dieses Abschnitts wird das Datenbankschema detailliert abgeleitet. Wie in Anforderung I und Anforderung II beschrieben, erfolgt die Modellierung im Eigenschaftsgraphen-Modell für Graphdatenbanksysteme (vgl. Abschnitt 2.3.1).

Artikel, Fahrzeuge und Fahrzeugflotten

Zunächst wird die Modellierung von Artikeln, Fahrzeugen und Fahrzeugflotten thematisiert, die in Abbildung 4.3 dargestellt ist.

Die Modellierung von Artikeln wurde aus dem Entitäten-Beziehungsmodell-Diagramm in Abbildung 4.2 übernommen, jedoch wurde die Artikelnummer nicht explizit als Eigenschaft modelliert, sondern nur der Name. Zur eindeutigen Identifizierung eines Artikels in der Graphdatenbank wird die eindeutige Identifikationsnummer des Knoten verwendet, die automatisch durch das Graphdatenbanksystem generiert wird und daher nicht vom Benutzer spezifiziert werden muss. Derselbe Umstand gilt auch für Fahrzeuge und Standorte

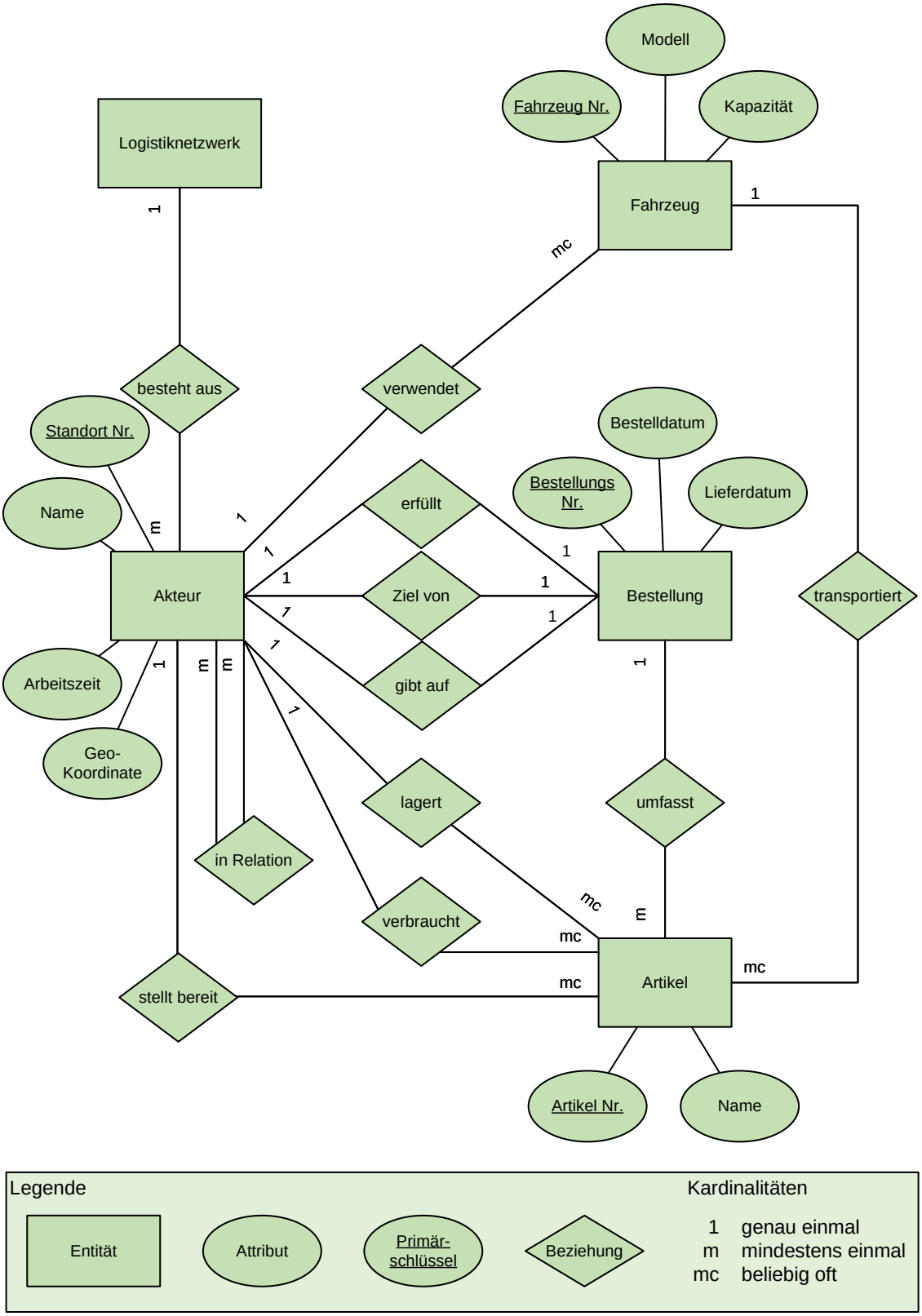


Abbildung 4.2: Entitäten-Beziehungsmodell-Diagramm des generischen Konzeptmodells

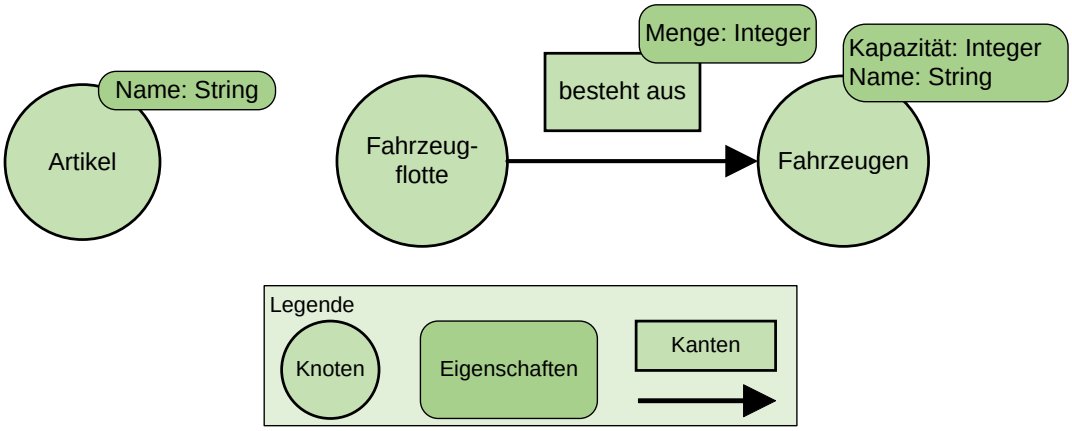


Abbildung 4.3: Eigenschaftsgraphen-Modell von Artikeln, Fahrzeugen und Fahrzeugflotten

von Akteuren. Hinter den Namen der Eigenschaften sind die verwendeten Datentypen spezifiziert, mit denen die Eigenschaften modelliert sind. Für die Simulationsmodelle wurden der Datentyp *Integer* für Ganzzahlen, *Float* für Gleitkommazahlen und *String* für Texte verwendet.

Fahrzeuge weisen Eigenschaften zur Spezifikation des Modellnamens und der Kapazität auf. Fahrzeugflotten sind ein Verbund von mehreren Fahrzeugen. Die Anzahl von Fahrzeugen eines Typs in einer Fahrzeugflotte können über die Mengeneigenschaft der Kante, über die die Fahrzeuge verbunden sind, festgelegt werden.

Verteilungen

Zur Abbildung stochastischer Prozesse werden Verteilungen verwendet, die in LogFarm, wie in Abbildung 4.4 dargestellt, modelliert werden.

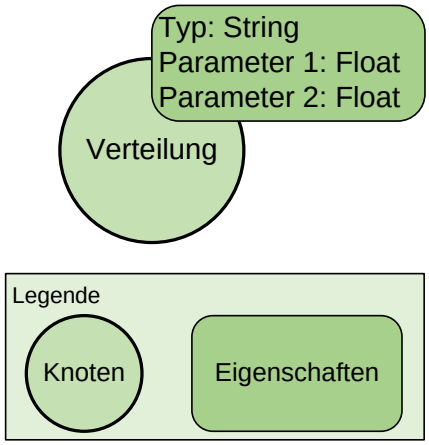


Abbildung 4.4: Eigenschaftsgraphen-Modell von Verteilungen

Über die *Typ* Eigenschaft kann der Typ der Verteilung spezifiziert werden. Zunächst werden dem Benutzer die Normal-, Gamma- und Exponentialverteilung zur Auswahl gestellt, welche zu den häufig verwendeten Verteilungen in der Logistik gehören (vgl. Abschnitt 3.3.1). Verteilungen werden durch Parameter charakterisiert. Im Fall der durch

den Simulator angebotenen Verteilungen werden jeweils zwei Parameter zur eindeutigen Festlegung der Verteilungen benötigt. Daher weist die Modellierung on Abbildung 4.4 zwei weitere Eigenschaften für eben diese Parameter auf.

Störungen

Eine Anwendung von Verteilungen findet beispielsweise in Störungen statt. Wie im Konzeptmodell beschrieben, können Störungen in der Produktions- und Transformationskomponente auftreten. Die Modellierung einer Störung erfolgt wie in Abbildung 4.5 abgebildet.

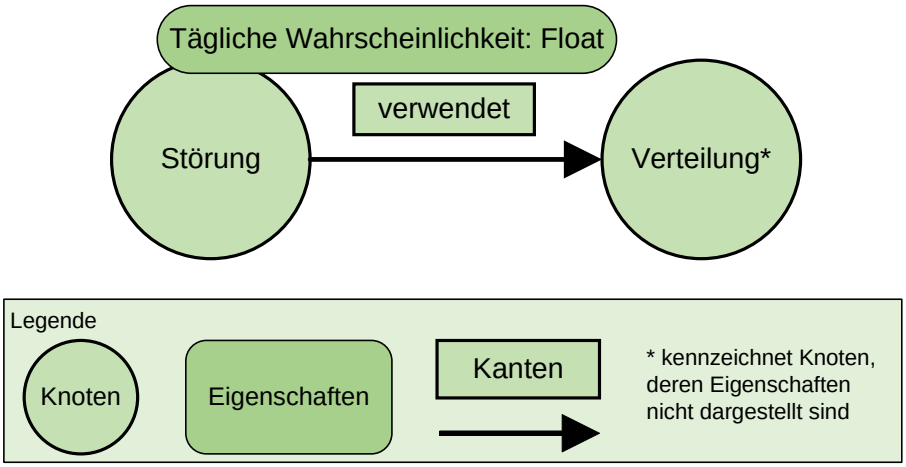


Abbildung 4.5: Eigenschaftsgraphen-Modell von Störungen

Das Auftreten der Störung ist durch eine tägliche Wahrscheinlichkeit definiert, die über die gleichnamige Eigenschaft spezifiziert werden kann. Im Konzeptmodell wurde beschrieben, dass die Störung verschiedene Intensitäten aufweisen kann und stochastisch beschrieben werden soll. Daher sind Störungen mit einer Verteilung verknüpft, die das Intensitätslevel der Störung abbildet.

Standortkomponente

Die Modellierung im Eigenschaftsgraphen-Modell ist in Abbildung 4.6 abgebildet.

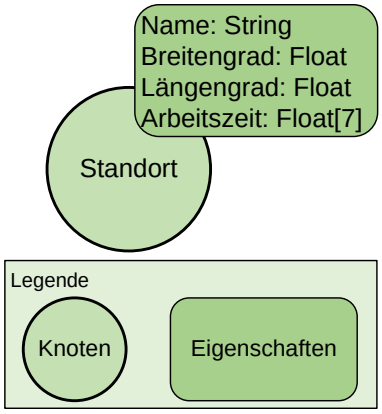


Abbildung 4.6: Eigenschaftsgraphen-Modell der Standortkomponente

Die Standortkomponente stellt eine simple Struktur zur Speicherung des Namens, der täglichen Arbeitszeit und der geographischen Koordinate des Standorts dar. Im Konzeptmodell wurde bereits beschrieben, dass die geographischen Koordinaten aus den beiden Bestandteilen Längen- und Breitengrad besteht. Die tägliche Arbeitszeit kann für jeden Tag der Woche separat spezifiziert werden, wodurch sich insgesamt sieben Werte ergeben.

Produktionskomponente

Die Aufgabe der Produktionskomponente ist es, neue Artikel zu erschaffen, beziehungsweise diese in das System zu bringen. Die Modellierung ist in Abbildung 4.7 dargestellt.

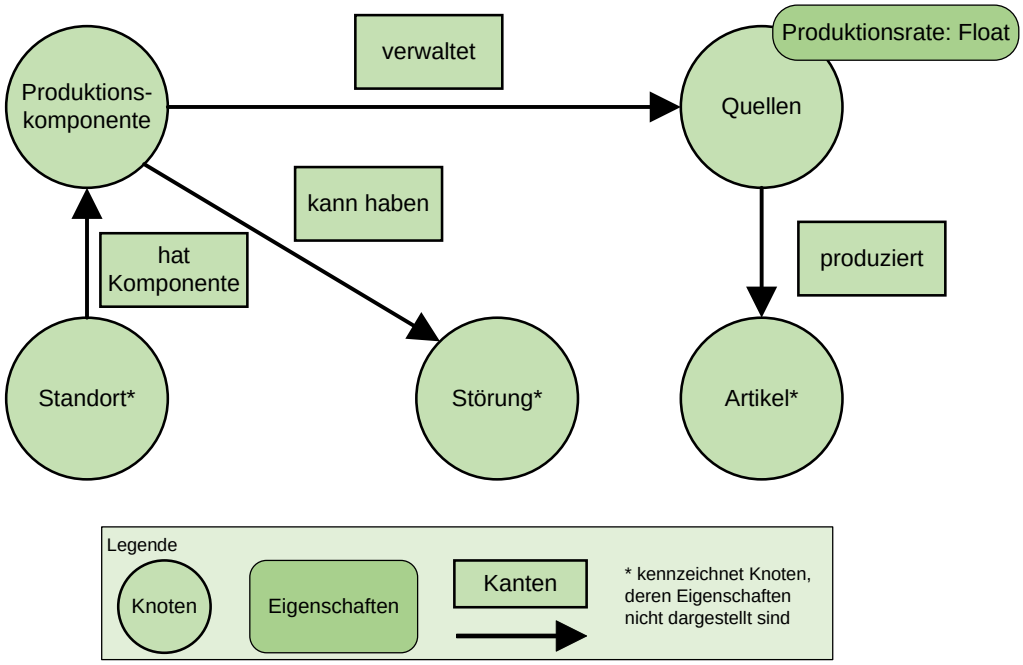


Abbildung 4.7: Eigenschaftsgraphen-Modell der Produktionskomponente

Es soll möglich sein, beliebig viele verschiedene Artikel in beliebiger Quantität einzubringen. Die Komponente verhält sich somit wie eine bzw. mehrere Quellen nach dem Fließprinzip. Die Produktionskomponente kann beliebig viele Quellen aufweisen. Wie im Konzeptmodell beschrieben, wird die Produktionsrate der Quellen stundenbezogen angegeben. Zudem ist es möglich, optional eine Störung einzubringen.

Transformationskomponente

Das Eigenschaftsgraph-Modell der Transformationskomponente ist in Abbildung 4.8 dargestellt.

Die Transformationskomponente umfasst keine Quellen, wie sie in der Produktionskomponente zu finden sind, sondern transformiert Artikel, die bereits im System vorhanden sind. Ein Transformationsprozess besteht dabei aus mindestens einem Artikel, der verbraucht wird, und mindestens einem Artikel, der erzeugt wird. Ein Beispiel für einen Transformationsprozess ist das Verschweißen von zwei Artikeln zu einem Neuen. Ein anderes Beispiel ist ein Entpackungsvorgang, bei dem der verbrauchte Artikel eine bepackte Palette darstellt und neue Güter in das System eingebracht werden, die vorher auf der Palette waren. In der Transformationskomponente können beliebig viele Transformationsprozesse hinter-

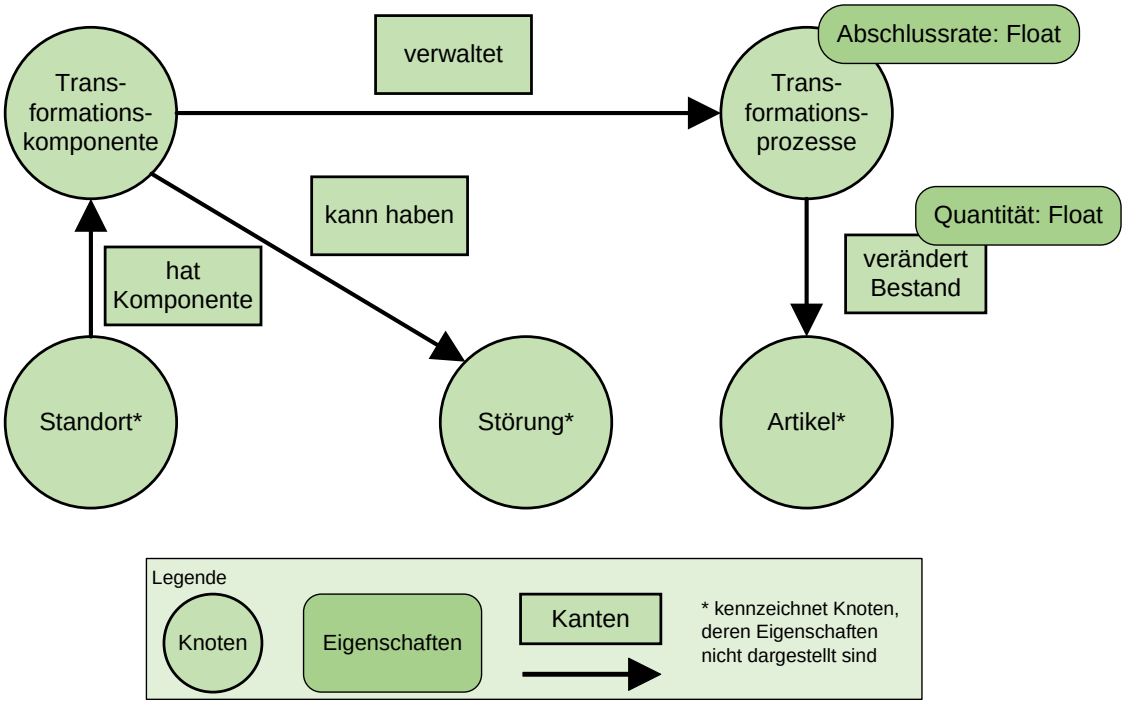


Abbildung 4.8: Eigenschaftsgraphen-Modell der Transformationskomponente

legt werden. Die Modellierung ähnelt stark der Produktionskomponente, jedoch weist die Kante zwischen den Knoten *Transformationsprozesse* und *Artikel* eine Eigenschaft auf, mit der spezifiziert werden kann, mit welcher Quantität der Bestand eines Artikels sich pro Abschluss der Transformation verändern soll.

Lagerkomponente

Das geplante Liegen von Gütern im Güterfluss wird durch die Lagerkomponente abgebildet. Die Modellierung ist in Abbildung 4.9 zu sehen.

Die Lagerkomponente kann mehrere Läger verwalten. Jedes Lager wird durch seine Kapazität charakterisiert. Es muss spezifiziert werden, welche Artikel in den Lägern gehalten werden können und wie die Bestandshaltung erfolgen soll. Dazu stehen dem Nutzer die in Abschnitt 3.2.1 beschriebenen einstufigen Lagerhaltungspolitiken (s,q)-, (r,s)- und (r,S)-Politik zur Verfügung. Den Politiken ist jeweils eine Identifikationsnummer zugeordnet, über die sie im Modell eindeutig identifiziert werden kann. Die Lagerhaltungspolitiken werden jeweils durch zwei Parameter beschrieben, die generisch als Parameter 1 und Parameter 2 bezeichnet werden. Um den Bestand zum Beginn der Simulation des Lagers abzubilden, kann über den Initialbestandsfaktor die prozentuale Auslastung spezifiziert werden. Der Initialbestand ist vorerst nur pauschal für alle Artikel spezifizierbar, somit ist der Anfangsbestand aller Artikel gleich verteilt.

Distributionskomponente

Liegen einem Standort unerfüllte Bestellungen vor, ist es die Aufgabe der Distributionskomponente diese, wenn möglich, zu bedienen. Es findet eine Priorisierung der Bestellungen hinsichtlich des Bestelldatums statt, sodass ältere Bestellungen zuerst bedient werden. Die Modellierung im Eigenschaftsgraphen-Modell ist in Abbildung 4.10 dargestellt.

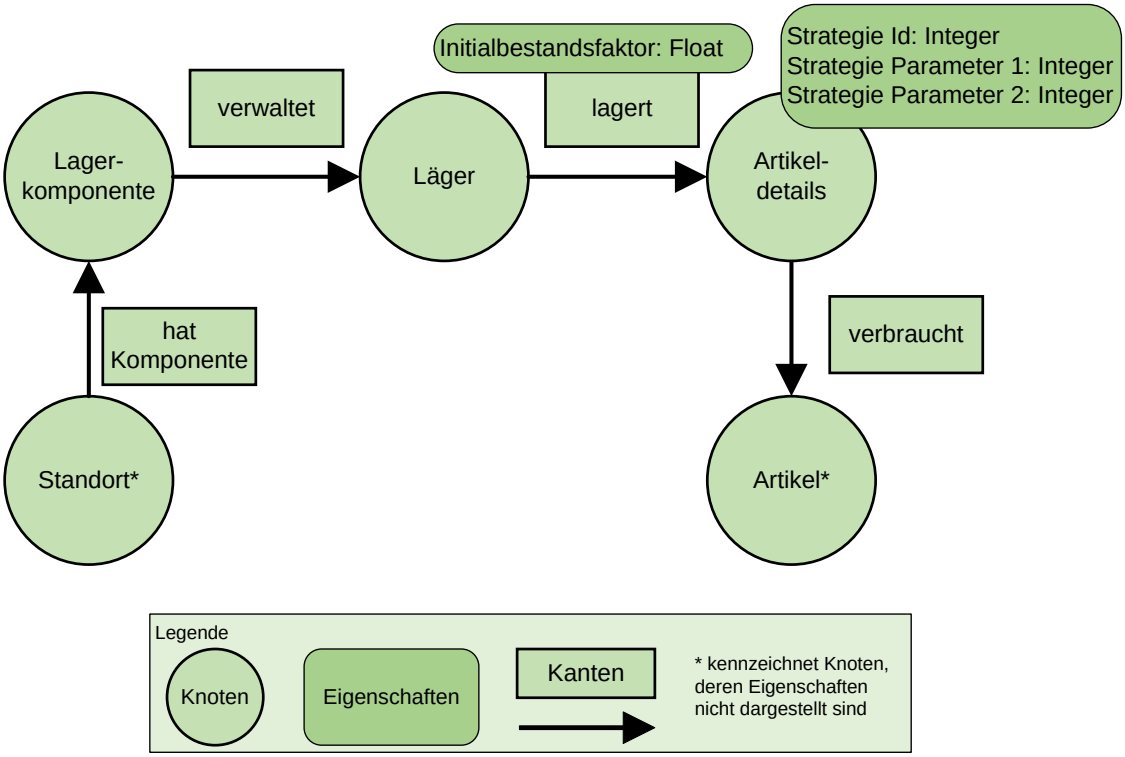


Abbildung 4.9: Eigenschaftsgraphen-Modell der Lagerkomponente

Der Nutzer muss für jeden Kundenstandort spezifizieren, ob dieser in die statische oder dynamische Tourenplanung des Standorts eingeschlossen werden soll. Die statische Tourenplanung bedient sich atomarer Transportrelationen zwischen dem Standort und seinen Kunden. Das bedeutet, dass jedes Fahrzeug lediglich Touren mit einer Station, nämlich dem Kunden, fährt. Bei der dynamischen Tourenplanung wird eine Tourenplanung auf Basis des verbesserten ICWA durchgeführt (vgl. Abschnitt 3.2.2). Soll die Belieferung eines Kunden statisch erfolgen, wird dieser über eine Kante direkt mit der Distributionskomponente verbunden. Kunden, die in der dynamischen Tourenplanung berücksichtigt werden sollen, werden mit einem zusätzlichen Knoten namens *Dynamische Routinggruppe* verbunden. Sowohl für die statische als auch die dynamische Planung werden dedizierte Fahrzeugflotten verwendet. Die Distanzen zwischen den Standorten berechnet der Simulator anhand der geographischen Koordinaten mithilfe der Haversine-Formel (Sinnott 1984).

Konsumkomponente

Die Konsumkomponente ist für die Entfernung von Artikeln aus dem System verantwortlich und übernimmt somit die Aufgabe einer Senke. In Abbildung 4.11 ist die Modellierung der Komponente dargestellt

Die Komponente kann die Entfernung beliebig vieler Artikel entsprechend einer spezifizierten Konsumrate modellieren. Die Konsumrate ist, wie die anderen Raten auch, in Artikeln pro Stunde anzugeben. Die Modellierung der Komponente ist ähnlich zu der Produktionskomponente.

Nachfragekomponente

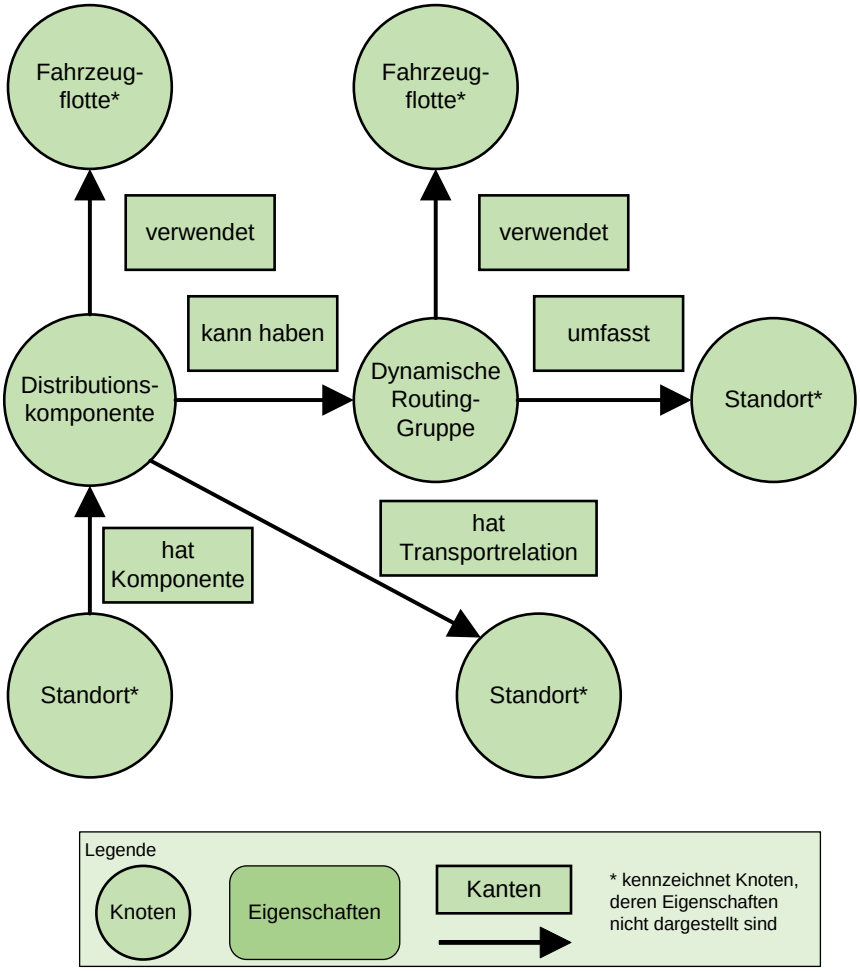


Abbildung 4.10: Eigenschaftsgraphen-Modell der Distributionskomponente

Mit der Nachfragekomponente können keine Güter aus dem System entfernt werden, sondern lediglich Bestellungen aufgegeben werden. Neben den Bestellungen, die durch die Lagerhaltungspolitiken ausgelöst werden, ist es der einzige Weg, einen Güterfluss hervorzurufen. Abbildung 4.12 zeigt die Modellierung der Komponente im Eigenschaftsgraphen-Modell.

Zur Modellierung der Nachfrage stehen dem Nutzer stochastische Beschreibungsmittel zur Verfügung. Zunächst wird über eine Verteilung die Häufigkeit des Auftretens von Nachfrage bestimmt. Des Weiteren werden einem Knoten mit dem Namen *Artikelwähler* mehrere *Artikelwählereinträge* über Kanten zugewiesen, die jeweils mit einer Gewichtung versehen sind. Die Gewichtung bleibt konstant und modelliert, mit welcher Wahrscheinlichkeit ein Eintrag ausgewählt wird. Jedem Eintrag ist ein Artikel und eine Verteilung für die Menge der resultierenden Nachfrage zugewiesen. Löst die Verteilung des zeitlichen Abstands eine Nachfrage aus, wird der Artikelwähler entsprechend der Gewichtungen der Einträge zufällig einen Eintrag auswählen und eine Nachfrage für den Artikel des Eintrags mit beliebiger Quantität generieren.

Dynamische Parameter über den Experimentplan

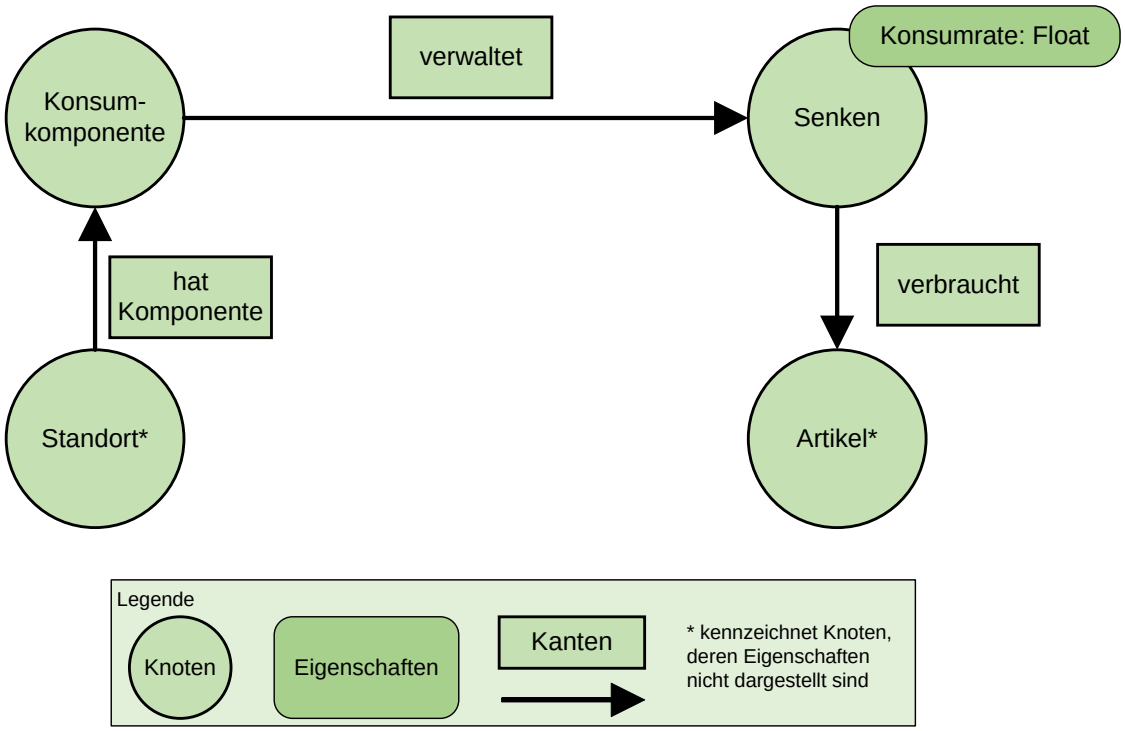


Abbildung 4.11: Eigenschaftsgraphen-Modell der Konsumkomponente

Das Konzeptmodell sieht vor, wie in Anforderung VII gefordert, einige Parameter des Simulationsmodells dynamisch über den Experimentplan zu steuern. Das bedeutet, es gibt statische und dynamische Parameter, die auch Faktoren genannt werden. Sollen Faktoren im Modell verwendet werden, müssen diese auf Datenebene gekennzeichnet werden. Dafür verwendet LogFarm ein Konstrukt, das als Experimentplan-Koppler bezeichnet wird. Die Modellierung des Konstrukts ist in Abbildung 4.13 dargestellt.

Der Experimentplan-Koppler stellt eine Relation zwischen Faktoren und Stellen im Experimentplan her. Dazu wird eine Kante vom Experimentplan-Koppler zu einem Knoten hergestellt, der einen Faktor beinhalten soll. Über die Eigenschaft *Eigenschaft* kann mit einem String die Eigenschaft des Zielknotens spezifiziert werden, die als Faktor verwendet werden soll. Mit der Eigenschaft *Faktor Id* kann die Stelle im Experimentplan spezifiziert werden, die den Faktor steuern soll. Der Ansatz kann für jegliche Eigenschaften des Modells verwendet werden. Das gilt sowohl für quantitative als auch für qualitative Parameter. Eine Stelle des Experimentplans kann zudem mehrere Faktoren steuern.

Generierte Daten

Die Aufgabe des Simulators ist es, Daten zu generieren. Auch diese Daten müssen in einer Graphdatenbank gespeichert werden (vgl. Anforderung V). Wie diese Daten im Eigenschaftsgraphen-Modell abgebildet werden, ist in Abbildung 4.14 zu sehen.

Das Konzeptmodell beschreibt als zu generierende Daten die aufkommenden Bestellungen und Lagerbestandsmeldungen der Simulationsläufe. Um die Daten immer eindeutig einem Simulationslauf zuordnen zu können, werden sie auf Datenebene mit einem Knoten verbunden, der den Namen *Simulationslauf* trägt. Dieser Knoten hat als Eigenschaften drei Werte. Mit der Eigenschaft *Designpunkt* wird beschrieben, welcher Simulationslauf des

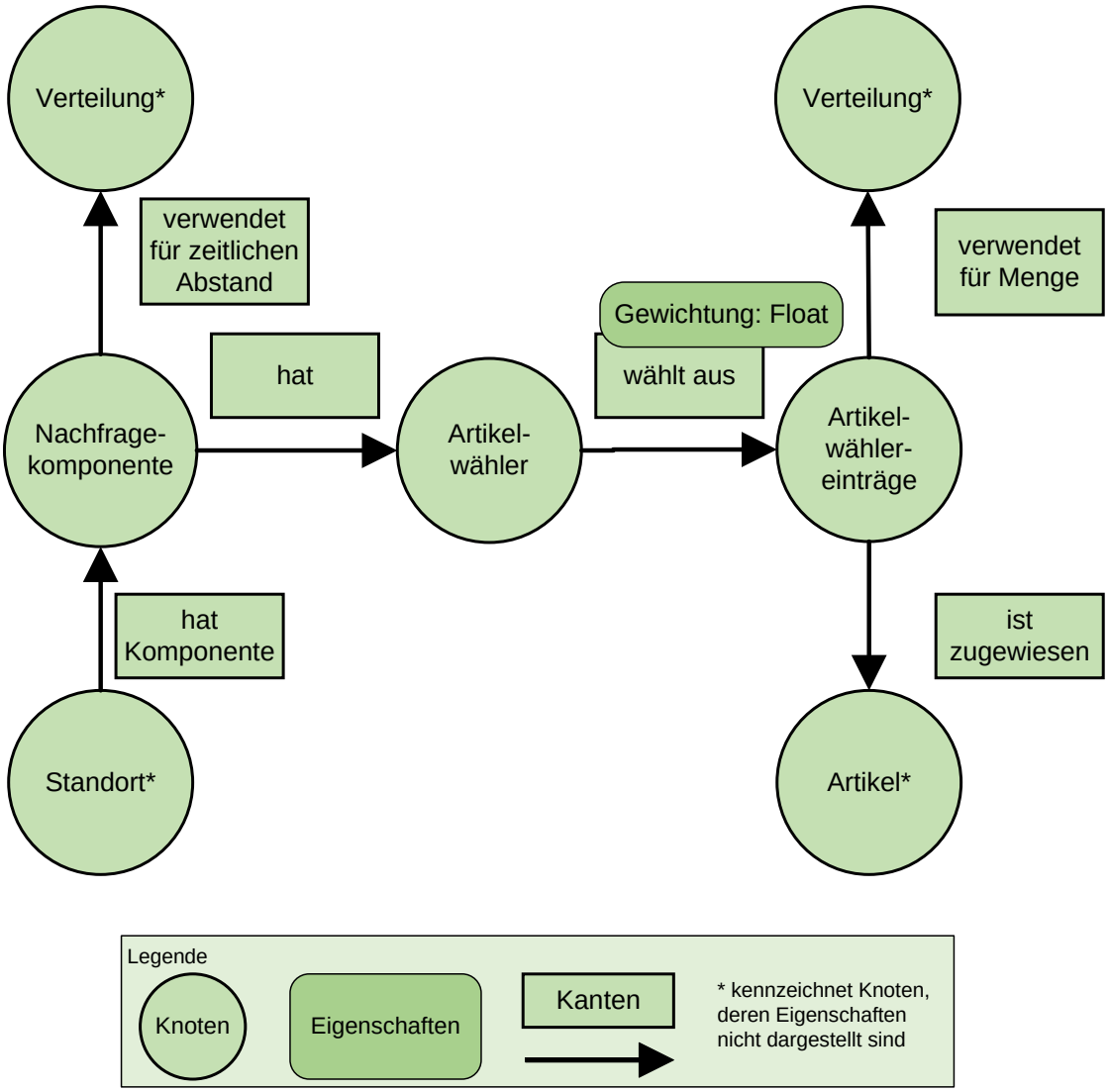


Abbildung 4.12: Eigenschaftsgraphen-Modell der Nachfragekomponente

Experimentplans untersucht wurde. Mit der Eigenschaft *Replikation* kann spezifiziert werden, in welcher Replikation des Designpunkts die Daten generiert wurden. Die Eigenschaft *Lauf Id* beschreibt eine Identifikationsnummer, die jedem Simulationslauf zugeordnet ist.

Bestellungen sind mit jeweils fünf weiteren Knoten verbunden. Zwei Knoten spezifizieren die Standorte, von denen die Bestellungen ausgehen und an die sie gerichtet sind. Weitere zwei Knoten legen die Daten fest, an denen die Bestellungen aufgegeben wurden und wann sie geliefert wurden. Der letzte Knoten spezifiziert, welche Artikel die Bestellung umfasst. Die Menge der Artikel wird über die dazugehörige Kante bestimmt.

Die Lagerbestandsmeldungen werden täglich angefertigt. Sie sind jeweils mit einem Datum verbunden, das diesen Tag beschreibt. Zudem ist der Knoten adjazent zu dem Standort, der das Lager verwaltet. Weitere Kanten existieren zu den Artikeln, die sich im Lager befinden können. Die Kanten weisen die Eigenschaft *Menge* auf, über die der Lagerbestand modelliert wird.

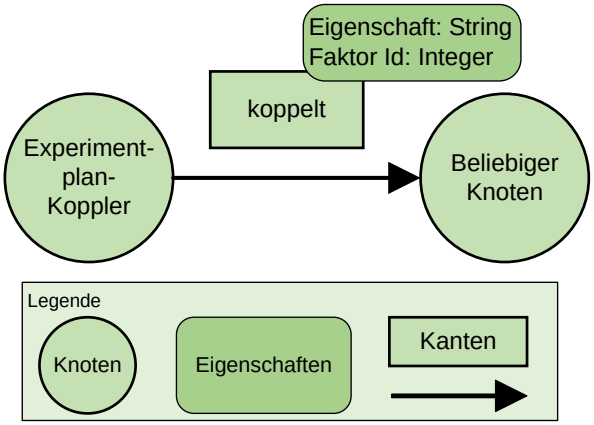


Abbildung 4.13: Eigenschaftsgraphen-Modell des Experimentplan-Kopplers

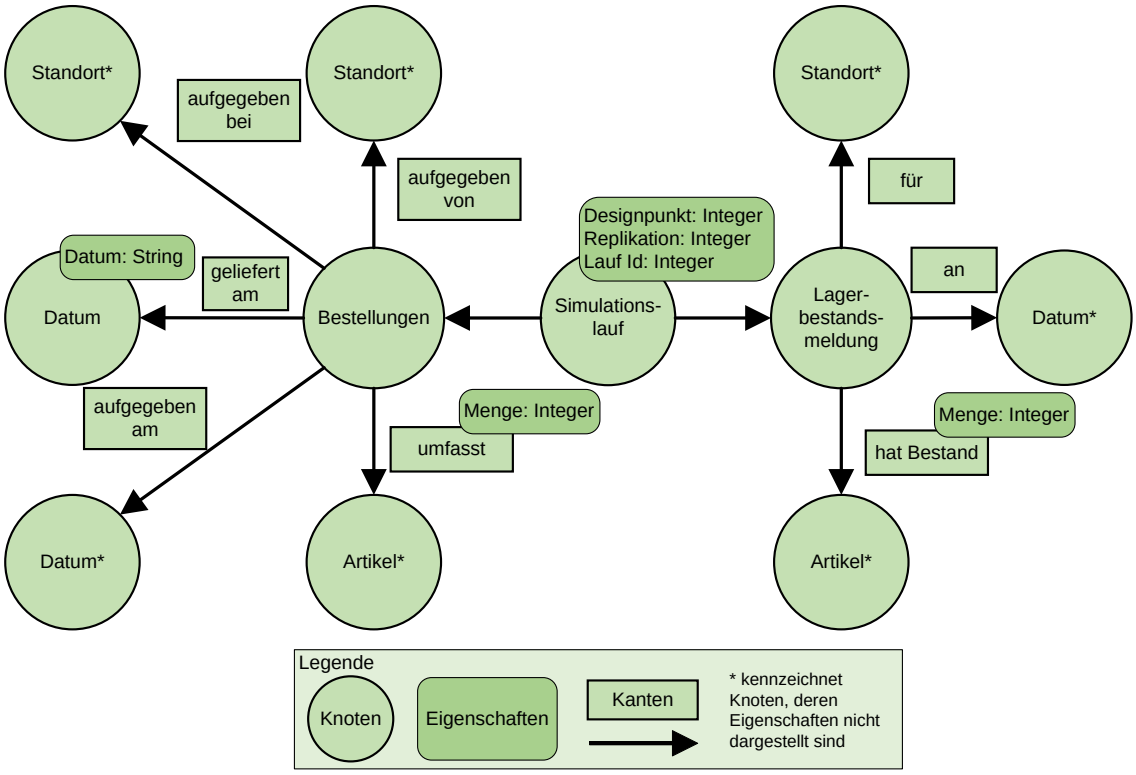


Abbildung 4.14: Eigenschaftsgraphen-Modell der generierten Daten

4.3.3 Softwarearchitektur

In diesem Abschnitt wird die Softwarearchitektur der Simulators erläutert. Eine Softwarearchitektur beschreibt, wie eine Anwendung organisiert wird, welche Softwarekomponenten sie aufweist und wie die Softwarekomponenten in Beziehung stehen (Sommerville 2016). Die Softwarekomponenten sind nicht zu verwechseln mit den im vorherigen Abschnitt vorgestellten Komponenten der Bausteine. LogFarms Softwarearchitektur ist in Abbildung 4.15 dargestellt.

Jedes der Elemente aus Abbildung 4.15 wird im Folgenden erläutert.

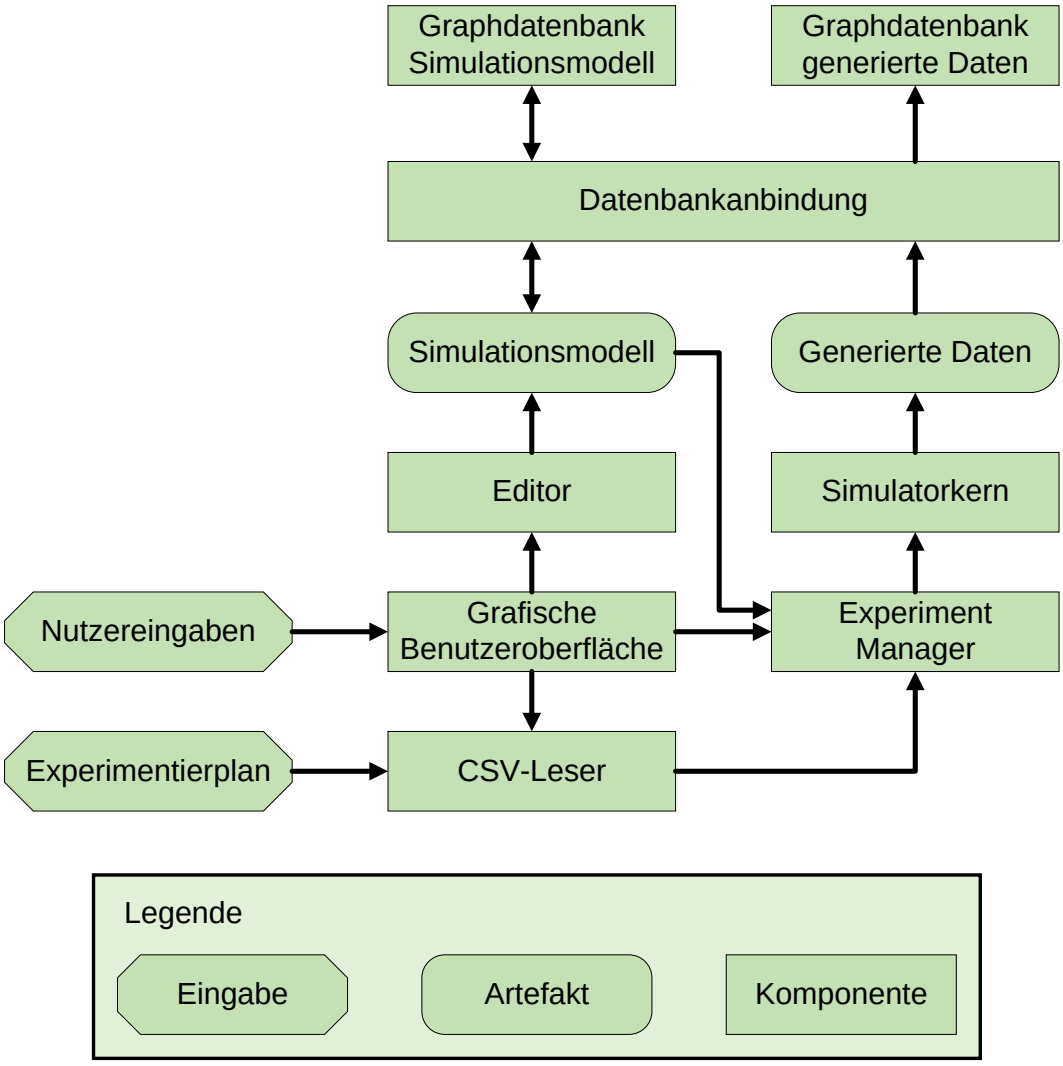


Abbildung 4.15: Softwarearchitektur des Simulators

- Nutzereingaben und grafische Oberfläche: Zur Nutzung des Simulators ist eine Interaktion mit dem Nutzer erforderlich. Die Nutzereingaben werden auf der grafischen Benutzeroberfläche getätigt und von dort an die Softwarekomponenten verteilt. Die drei direkt von Nutzereingaben abhängigen Softwarekomponenten sind der Editor, der Experiment Manager und der CSV-Leser.
- Experimentierplan und CSV-Leser: In Anforderung VII wird beschrieben, dass der Experimentierplan nicht im Simulator generiert, sondern dem Simulator im CSV-Format zugeführt wird. Der Nutzer kann über die grafische Oberfläche einen Experimentierplan im gültigen Format mithilfe eines Dialogs spezifizieren. Die Interpretation des Experimentierplans geschieht in der CSV-Leser Softwarekomponente, die die Daten in ein internes Datenformat überführt. Anschließend wird der Experimentierplan dem Experiment Manager zur Verfügung gestellt.
- Editor und Simulationsmodell: Aus Anforderung III ergibt sich, dass es einen Editor zur interaktiven Erstellung von Simulationsmodellen geben muss. Der Editor kann über die grafische Oberfläche bedient werden und erlaubt es, Bausteine, Beziehun-

gen und eine entsprechende Parametrisierung vorzunehmen. Der Editor wird später ausführlicher behandelt. Das Artefakt der interaktiven Modellerstellung ist das Simulationsmodell. Das Simulationsmodell wird in einer Graphdatenbank persistiert und durch den Experiment-Manager für das Data-Farming-Framework verwendet.

- Datenbankanbindung: Die Datenbankanbindung stellt die Schnittstelle der Anwendung zu den beiden Graphdatenbanken dar, die das Simulationsmodell und die generierten Daten speichern. Wie die Pfeile in der Abbildung andeuten, soll im Fall des Simulationsmodells ein bidirektionaler Austausch von Daten möglich sein, also sowohl Schreib- als auch Lesevorgänge. Zum Lesen der Daten aus der Graphdatenbank wird das OGM (vgl. Abschnitt 2.3.1) verwendet, um die Graphenstruktur auf interner Ebene abzubilden und keine weiteren Transformationsprozesse zu benötigen. Auch zum Schreiben der Daten in die Graphdatenbank wird das OGM verwendet.
- Graphdatenbank Simulationsmodell: Das gesamte Simulationsmodell wird in dieser Graphdatenbank gespeichert und über OGM in LogFarm übertragen. Der Simulator bietet vorerst keine Speicherung des Simulationsmodells in einer Datei an, jedoch kann über Funktionalitäten der Graphdatenbank ein Abbild erzeugt werden. Dieses Abbildung kann dann zur Speicherung oder Weitergabe des Modells genutzt werden.
- Experiment Manager: Der Experiment Manager ist für die Durchführung des Experiments zuständig. Dort laufen das Simulationsmodell, der Experimentplan und die durch den Nutzer spezifizierten Simulationsparameter zusammen. Die Simulationsläufe werden vorbereitet und alle für die Simulation nötigen Daten an den Simulatorkern geschickt. Der Experiment Manager ist auch dafür verantwortlich, dass eine möglichst gute Auslastung der zur Verfügung stehenden Hardware durch eine Verteilung von Simulationsläufen gewährleistet wird.
- Simulatorkern: Der Simulatorkern ist eine Softwarekomponente von Simulatoren und wurde allgemein in Abschnitt 2.1.3 beschrieben. Er ist für die Durchführung der Simulation zuständig und benötigt dazu ein Simulationsmodell und Simulationsparameter. Das Ergebnis der Simulation ist im Fall dieses Simulators eine Menge generierter Daten.
- Generierte Daten: Generierte Daten stellen ein Artefakt des Simulatorkern, bzw. der Simulation die durch den Simulatorkern durchgeführt wird, dar. Jeder Simulationslauf generiert Daten, die über die Datenbankanbindung in der dedizierten Graphdatenbank für generierte Daten gespeichert werden.
- Graphdatenbank generierte Daten: Diese graphbasierte Datenbank persistiert die Ergebnisse des Data-Farming-Vorgangs. Jeder Simulationslauf speichert die generierten Daten hier ab. Anders als bei der Graphdatenbank zur Speicherung des Simulationsmodells, ist die Verbindung lediglich unidirektional, besteht also ausschließlich aus Schreibprozessen. Eine Einsicht der generierten Daten ist nur über einen direkten Zugriff auf die Datenbank möglich, die Ergebnisse werden also nicht in der grafischen Benutzeroberfläche angezeigt. Die Graphdatenbank generierter Daten wird im Verlauf der Arbeit auch als Ergebnisdatenbank bezeichnet.

4.3.4 Simulatorkern

Der Simulatorkern stellt die zentrale Softwarekomponente des Simulators dar. In Anforderung IV wurden bereits Anforderungen an den Simulator gestellt. Dazu gehört, dass

die Simulation kontinuierlich ablaufen und agentenbasierte Modelle verarbeiten soll. Der Simulator ist für den zeitlichen Fortschritt der Simulation und die Orchestrierung der Agenten des Simulationsmodells verantwortlich. Die Einbettung in die Softwarearchitektur des Simulators wurde bereits in Abbildung 4.15 visuell dargestellt. Die Funktionsweise und Ablaufsteuerung des Simulatorkerns wird im folgenden Programmablaufplan, der in Abbildung 4.16 dargestellt ist, verdeutlicht.

Der Programmablaufplan des Simulatorkerns in Abbildung 4.16 zeigt in vereinfachter Weise den gesamten Ablauf eines Simulationslaufs. Die Variable t, die den aktuellen Tag der Simulation beschreibt, wird iterativ vom Start- bis zum Enddatum t_{ende} erhöht. Dazwischen werden alle Agenten des Simulationsmodells aktualisiert. Beim Aktualisieren der Agenten wird zwischen zwei Phasen unterschieden. Zunächst werden alle Komponenten der Agenten aktualisiert, die nicht die Distributionskomponente sind. Das bedeutet, dass zunächst kein Austausch von Artikeln zwischen Agenten stattfinden kann, bevor nicht alle anderen Komponenten aller Agenten vorher aktualisiert wurden. So kann sichergestellt werden, dass keine spezifische Reihenfolge bei der Reihenfolge der Abarbeitung der Agenten nötig ist. Die Details beim Ablauf des Updates eines Agenten in der ersten Phase können der Abbildung 4.17 entnommen werden.

Das Unterprogramm ruft die Aktualisieren-Methoden der Komponenten in der Reihenfolge Konsum-, Produktions-, Transformations-, Lagerungs- und Nachfragekomponente auf, vorausgesetzt der Agent weist die entsprechende Komponente auf. Bei den möglicherweise störungsbehafteten Produktions- und Transformationskomponenten wird eine Zufallszahl aus dem Zufallszahlenstrom gezogen und geschaut, ob eine Störung ausgelöst wird. Falls dies der Fall ist, wird die Rate der Komponente mit dem Störungsintensitätwert multipliziert, den der Modellierer hinterlegt hat. Wird in der Lagerungskomponente festgestellt, dass entsprechend der verwendeten Lagerhaltungspolitik eine Bestellung ausgelöst werden muss, werden die nötigen Schritte veranlasst. Ein passender Eintrag in der Graphdatenbank für generierte Daten erfolgt direkt im Anschluss. Dasselbe gilt für Bestellungen, die durch die Nachfragekomponente ausgelöst werden. Auch die Lagerbestandsberichte werden durch die Aktualisierung der Lagerungskomponente in der Datenbank für generierte Daten gespeichert. Das Unterprogramm endet, nachdem alle Komponenten, die der Agent aufweist, aktualisiert wurden.

Im weiteren Verlauf Programmablaufplans des Simulatorkerns in Abbildung 4.16 wird die Distributionskomponente der Agenten, falls sie vorhanden ist, aktualisiert. Zur Beschreibung des Unterprogramms ist der Pseudocode in Algorithmus 4.1 gegeben.

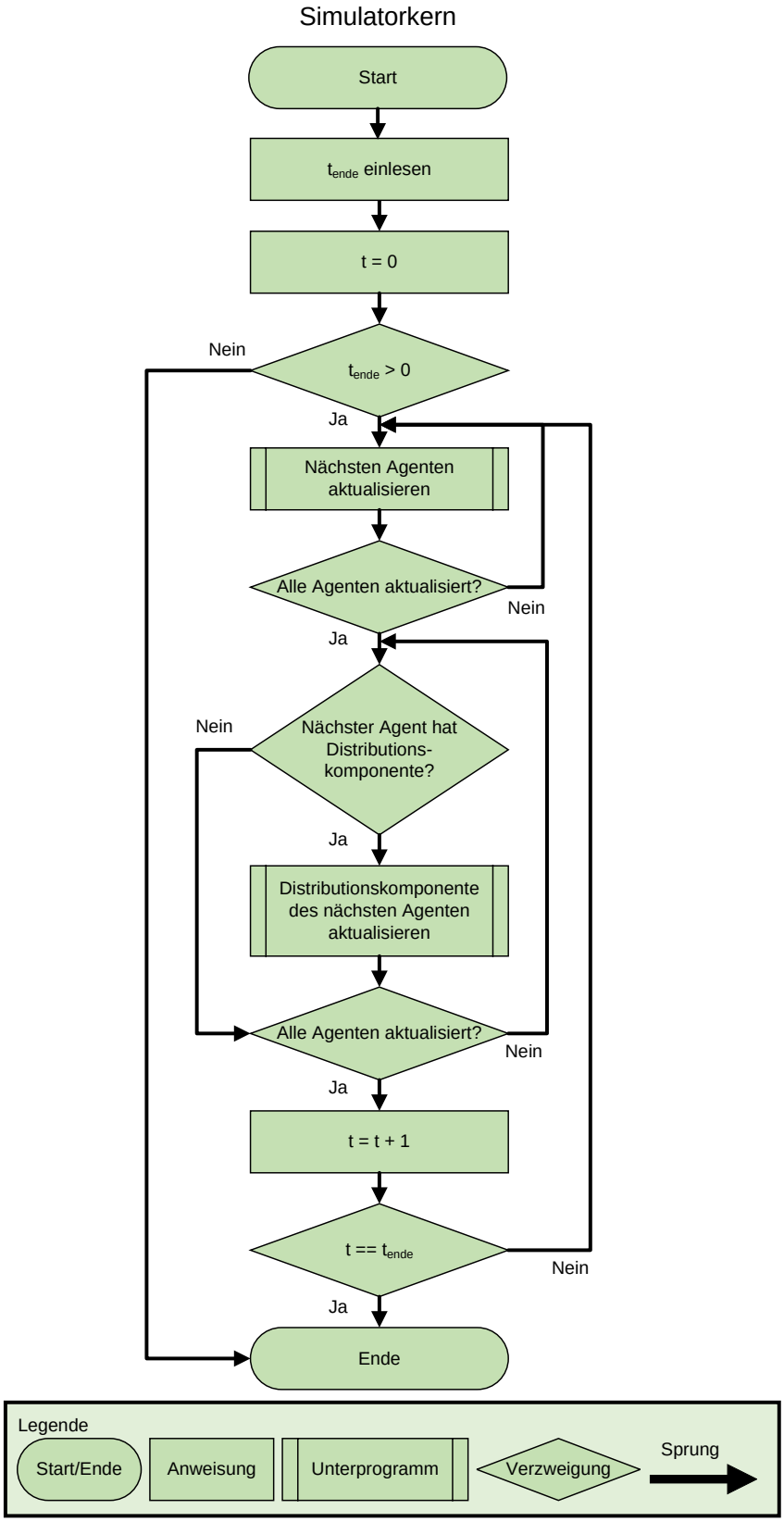


Abbildung 4.16: Programmablaufplan des Simulatorkerns

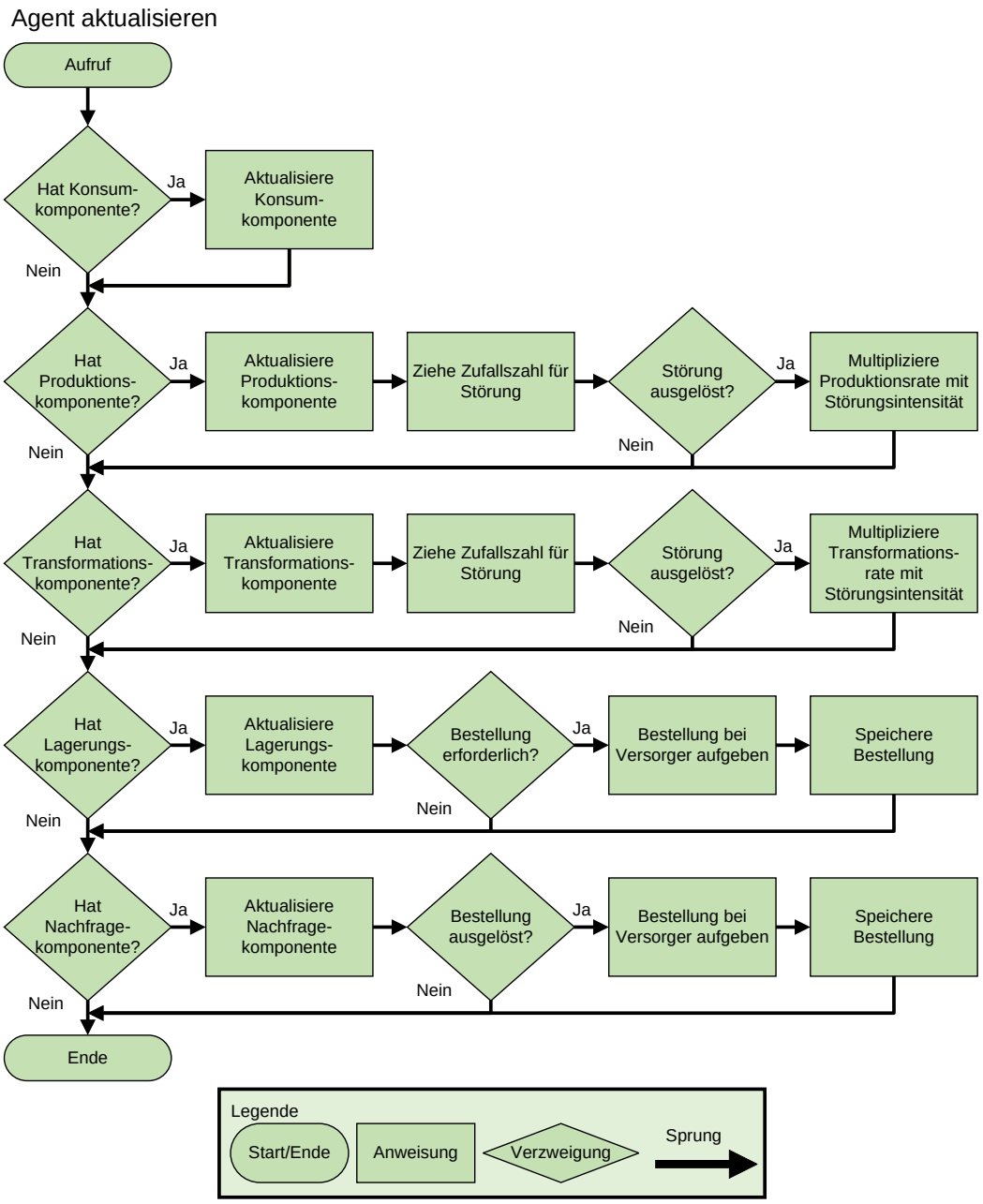


Abbildung 4.17: Programmablaufplan der Aktualisierung von Agenten

Algorithmus 4.1: Pseudocode Distributionskomponente

```
1 Erstelle Liste mit allen nicht erfüllten Bestellungen
2 Sortiere aufsteigend nach Bestelldatum
3 Erstelle Teilliste für statische Kunden
4 Erstelle Teilliste für dynamische Kunden
5 Weise nicht erfüllte Bestellungen der entsprechenden Teilliste zu
6 Erstelle leere Liste für zu bedienende Bestellungen
7 foreach Bestellung in Teilliste statischer Kunden
8   if Bestellmenge <= verfügbare Kapazität für den Kunden
```

```
9       if Fahrzeug noch nicht dem Kunden zugewiesen
10        Weise Fahrzeug dem Kunden zu
11      Weise Bestellung dem Fahrzeug zu
12      Füge Bestellung der Liste mit zu bedienenden Bestellungen hinzu
13  if Teilliste dynamischer Kunden hat mindestens einen Eintrag
14    Summiere Bestellmengen aller Bestellungen für alle dynamischen
        Kunden
15    Initialisiere CVRP-Modell mit dynamischen Kunden deren
        Bestellmenge > 0
16    Löse CVRP mit ICWA
17    Hänge Ergebnisliste des ICWA der Liste zu bedienender
        Bestellungen an
18  foreach Bestellung in Liste für zu bedienende Bestellungen
19    Führe Transport der Bestellung durch
20    Markiere Bestellung als ausgeliefert
```

Wie im Konzeptmodell (vgl. Abschnitt 4.2) beschrieben, bieten sich zwei Wege der Distribution. Alle Kunden sind entweder über statische oder dynamische Relationen mit dem Agenten verbunden. Im Pseudocode wird verkürzt von statischen und dynamischen Kunden gesprochen. Das Unterprogramm teilt zu Beginn alle offenen Bestellungen auf zwei Listen auf, die für Bestellungen von statisch beziehungsweise dynamisch angebundenen Kunden vorgesehen sind. Dabei muss sichergestellt werden, dass die Bestellungen hinsichtlich ihres Bestelldatums aufsteigend sortiert sind, sodass das erste Element der Listen die älteste offene Bestellung beinhaltet. Außerdem wird eine leere Liste vorbereitet, die alle zu bedienenden Bestellungen des Agenten für den betreffenden Tag beinhaltet.

Zunächst wird die statische Tourenplanung durchgeführt. Für jede Bestellung in der Liste von Bestellungen von Kunden, die statisch angebunden sind, wird geprüft, ob die Bestellmenge die verfügbare Kapazität für den Kunden nicht übersteigt. Die verfügbare Kapazität wird für jeden Kunden spezifisch berechnet, um Fahrzeuge, die bereits dem Kunden zugeordnet und nicht voll ausgelastet sind, zu berücksichtigen. Falls ausreichend Kapazitäten zur Verfügung stehen, um die Bestellung auszuführen, werden ein oder mehrere Fahrzeuge, je nach benötigter Kapazität, dem Kunden zugewiesen und der Auftrag der Liste zu bedienender Bestellungen hinzugefügt.

Nachdem die gesamte Liste von Bestellungen statisch angebundener Kunden abgearbeitet wurde, folgt die dynamische Tourenplanung für alle dynamisch angebundenen Kunden. Zur Lösung des Tourenplanungsproblems wird der ICWA verwendet, der in Abschnitt 2.3.2 beschrieben wurde. Um den Algorithmus anwenden zu können, muss zunächst die Gesamtbestellmenge für die Kunden ermittelt werden. Alle Kunden, für die mindestens eine Bestellung vorliegt, werden einem CVRP-Modell hinzugefügt. Anschließend berechnet der ICWA auf Basis des CVRP-Modells einen Tourenplan. Es sei angemerkt, dass es bei dieser Herangehensweise zu Problemen kommen kann, wenn die Gesamtbestellmenge eines Kunden die maximale Kapazität der ausliefernden Fahrzeugflotte übersteigt, da keine Teillieferungen möglich sind und somit in keinem Fall eine Lieferung möglich wäre. Aus dem berechneten Tourenplan kann abgeleitet werden, welche Bestellungen bedient werden sollen. Solche Bestellungen werden am Ende der dynamischen Routenplanung der Liste zu bedienender Bestellungen hinzugefügt.

Die Ausführung der Transporte findet nach der statischen und dynamischen Tourenplanung statt. Für jeden Auftrag, der in der Liste der zu bedienenden Bestellungen ist, wird der nötige Transport veranlasst und die bestellten Güter aus dem Lager des Agenten sofort an den Kunden transferiert. Die Bestellung wird anschließend als ausgeliefert mar-

kiert und mit dem Auslieferungsdatum versehen. Das Unterprogramm endet, nachdem alle Transporte durchgeführt wurden.

4.3.5 Experiment Manager

Eine weitere wichtige Softwarekomponente des Simulators ist der Experiment Manager. Die Aufgabe dieser Softwarekomponente besteht in der Durchführung des Experiments und effizienten Abarbeitung des Experimentierplans. Um der Forderung nach leistungsstarker Berechnung (siehe Anforderung VI) zu entsprechen, wendet der Experiment Manager das Konzept der parallelen Berechnung an. Die Berechnungen und somit die Simulation finden in so genannten Threads statt, die parallel ausgeführt werden können. Die Anzahl maximal parallel verfügbarer Threads kann durch den Nutzer vorgegeben werden und ist Teil der . Der Experiment Manager betreibt einen Pool von Threads mit der spezifizierten Größe. Der schematische Ablauf der Durchführung der Experimente durch den Experiment Manager mit der Thread-Pool Größe von zwei Threads kann Abbildung 4.18 entnommen werden.

Abbildung 4.18 zeigt, dass zur Durchführung eines Experiments die vier Artefakte Experimentplan, Experimentparameter, Simulationsmodell und Simulationsparameter nötig sind. Dem Experimentplan ist die Anzahl durchzuführender Simulationsläufe zu entnehmen. Die Reihenfolge bei der iterativen Abarbeitung des Experimentierplans entspricht derselben Reihenfolge, die in der CSV-Datei spezifiziert wurde, aus der der Experimentierplan extrahiert wurde. Die Experimentparameter umfassen die Nutzereingaben zur Spezifikation des Ablaufs des Experiments. Zu den Experimentparametern gehören die Größe des Threadpools, der Startwert der Simulation und die Anzahl der Replikationen, die für jeden Designpunkt durchgeführt werden sollen. Simulationsläufe benötigen zur Durchführung Simulationsparameter, die das Start-, End- sowie Warmlaufdatum festlegen, und ein Simulationsmodell. Aufgrund der geplanten Art der Implementierung des Simulationsmodells, die eine Manipulation der internen Daten beinhaltet, wird jedem Simulationslauf lediglich eine Arbeitskopie zur Verfügung gestellt, die dann durch den Simulatorkern manipuliert werden kann. Bei der Erstellung der Arbeitskopie stellt der Experiment Manager sicher, dass Parametern, die dynamisch durch den Experimentplan gesteuert werden sollen, die passenden Werte aus dem Experimentplan zugewiesen werden. Die Simulationsläufe werden anschließend durch den Experiment Manager einem Thread zugewiesen. Der Experiment Manager kann jederzeit die Anzahl aktiver bzw. inaktiver Threads im Thread-Pool ermitteln und inaktiven Threads einen bisher unbearbeiteten Simulationslauf zuweisen. Als aktive Threads im Kontext der Experiment-Manager-Softwarekomponente werden Threads bezeichnet, die mit einem Simulationslauf beauftragt wurden und diesen noch nicht vollkommen abgeschlossen haben. Inaktive Threads zeichnen sich dadurch aus, dass sie bisher entweder nicht verwendet wurden, oder den zugewiesenen Simulationslauf bereits abgeschlossen haben. Ein Simulationslauf umfasst neben der vollständigen Durchführung der Simulation auch das Schreiben der generierten Daten nach der Simulation in die Graphdatenbank der generierten Daten. Die Zugriffsverwaltung auf die Graphdatenbank wird von der Datenbankschnittstelle übernommen, wodurch sichergestellt werden kann, dass immer nur ein Thread gleichzeitig auf die Datenbank zugreifen kann. Ist die Datenbank bereits durch einen Schreibvorgang eines anderen Threads blockiert, bildet sich eine Warteschlange, die abgearbeitet wird. Da der Informationsfluss bei der Ansteuerung des Thread-Pools unidirektional ist, muss eine konstante Überwachung des Status der Threads seitens des Experiment Managers erfolgen, was in Abbildung 4.18 als Augen-Symbol visualisiert ist.

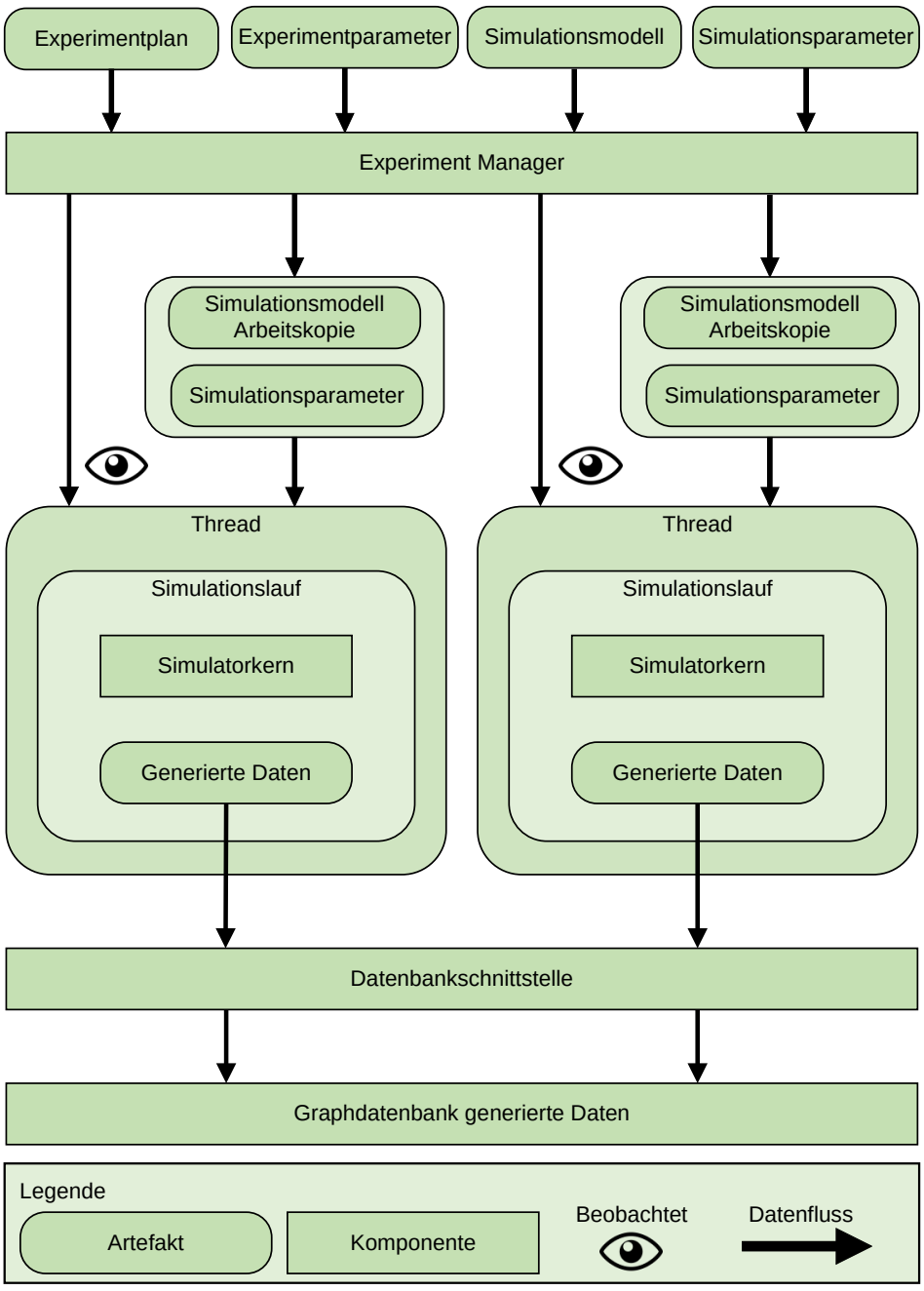


Abbildung 4.18: Funktionsweise Experiment Manager bei einer Thread-Pool Größe von Zwei

Bei der Auswahl einer Thread-Pool Größe sollte der Nutzer zwei Dinge beachten. Zum einen muss überprüft werden, welche Anzahl von Threads der Computer, auf dem die Simulation ausgeführt wird, effektiv verwenden kann. Das ist abhängig von der eingesetzten Hardware und muss im Einzelfall geprüft werden. Zum anderen ist die Entscheidung davon abhängig, wie das Verhältnis der Rechenzeit eines Simulationslaufs im Vergleich zur Dauer eines Schreibvorgangs in die Datenbank ist. Je höher die Rechenzeit der Simulation als die benötigte Zeit zum Schreiben der Ergebnisse ist, desto höher sollte die Anzahl der parallelen Threads ausfallen. Für den Fall, dass die Schreibvorgänge länger dauern als

die Simulation, kann durch eine parallele Ausführung die maximale Zeitersparnis durch parallele Berechnung durch eine Thread-Pool-Größe von zwei erreicht werden.

In Anforderung VI wurde beschrieben, dass LogFarm prinzipiell eine Clusterung von mehreren Computern unterstützen soll. Dies ist dann sinnvoll, wenn das Verhältnis von Rechenzeit der Simulation zur Dauer der Schreibvorgänge in die Datenbank besonders hoch ist. Im Fall einer Clusterung ist es möglich, die Orchestrierung des Experiment Managers nicht nur auf Threads des betreibenden Computers zu beschränken, sondern diese auch anderen Computern im Cluster zuzuweisen. Dazu ist lediglich eine weitere Schnittstelle nötig, über die die Arbeitskopie des Simulationsmodells und die Simulationsparameter ausgetauscht werden können. Zudem muss sichergestellt werden, dass alle Computer im Cluster Zugang zur Datenbank der generierten Daten haben. Eine detailliertere Erläuterung der Clusterung erfolgt in dieser Arbeit nicht.

4.3.6 Schnittstellen

Neben der grafischen Benutzeroberfläche, die die Schnittstelle zwischen dem Benutzer und dem Computer ist und im nächsten Abschnitt vorgestellt wird, gibt es zwei weitere relevante Schnittstellen, über die Informationen mit dem Simulator ausgetauscht werden können.

Eine der Schnittstellen stellt der CSV-Leser dar. Über ihn muss der Experimentierplan übermittelt werden (vgl. Anforderung VII). Dazu muss der Experimentierplan in einer Datei im CSV-Format vorliegen. Das CSV-Format ist ein einfaches und menschenlesbares Dateiformat, in dem Daten als Klartext gespeichert und durch Kommata getrennt werden (Shafranovich 2005). Neben der Trennung einzelner Daten, die auch als Datenfeld bezeichnet werden, durch ein Komma ist auch eine Trennung hinsichtlich Datensätze durch einen Zeilenumbruch möglich.

Für den Fall der Formatierung des Experimentierplans im CSV-Format entspricht jede Zeile der CSV-Datei den zu verwendenden Werten für jeden Faktor eines Simulationslaufs. Das bedeutet, dass jedes Datenfeld einem Faktor zugewiesen kann. Das erste Datenfeld einer Zeile ist dem ersten Faktor zugeordnet. Die weitere Zuordnung erfolgt in aufsteigender Reihenfolge. Wie im CSV-Format üblich, erwartet LogFarm eine Trennung von Datenfeldern mit Kommata und die Trennung von Datensätzen mit Zeilenumbrüchen. Abweichungen von exakt diesem Format können nicht verarbeitet werden.

Eine andere Schnittstelle stellt die direkte Kommunikation über die Graphdatenbank des Simulationsmodells dar. In Abbildung 4.15 wurde gezeigt, dass die Verbindung zu dieser Datenbank bidirektional erfolgt und somit auch Daten wieder in den Simulator gelesen werden können. Der Nutzer kann das Simulationsmodell direkt in der Datenbank manipulieren bzw. ein Abbild der Datenbank speichern oder einlesen. Da der Simulator in der geplanten Fassung keine Speicherung von Simulationsmodellen über die grafische Benutzeroberfläche vorsieht, stellt der direkte Zugriff auf die Datenbank derzeit die einzige Möglichkeit zur Speicherung und somit des Austauschs von Simulationsmodellen dar. Ein Nachteil der direkten Manipulation des Simulationsmodells in der Datenbank stellt die fehlende Integritäts- und Syntaxkontrolle dar, die nur bei einer Modellerstellung über den Editor des Simulators möglich sind. Somit wird von einer Manipulation des Simulationsmodells direkt auf Datenebene ohne umfassendes Wissen über das Datenbankschema und des Simulatorkerns abgeraten.

Wie eingangs beschrieben, stellt die grafische Benutzeroberfläche eine spezielle Schnittstelle dar, über die der Benutzer mit der Anwendung kommunizieren kann. Das Konzept der grafischen Benutzeroberfläche des Simulators wird im folgenden Abschnitt vorgestellt.

4.3.7 Konzept der grafischen Benutzeroberfläche

In Anforderung III wurde spezifiziert, dass der Simulator eine grafische Benutzeroberfläche aufweisen soll. Insbesondere soll es möglich sein, eine interaktive Modellbildung durchführen zu können. Als Sprache wurde Englisch spezifiziert, um den Simulator einer größeren Gruppe von Anwendern zugänglich zu machen.

Das grafische Konzept des Simulators sieht vor, dass das Anwendungsfenster in drei Bereiche eingeteilt werden soll, die in Abbildung 4.19 farblich gekennzeichnet sind.

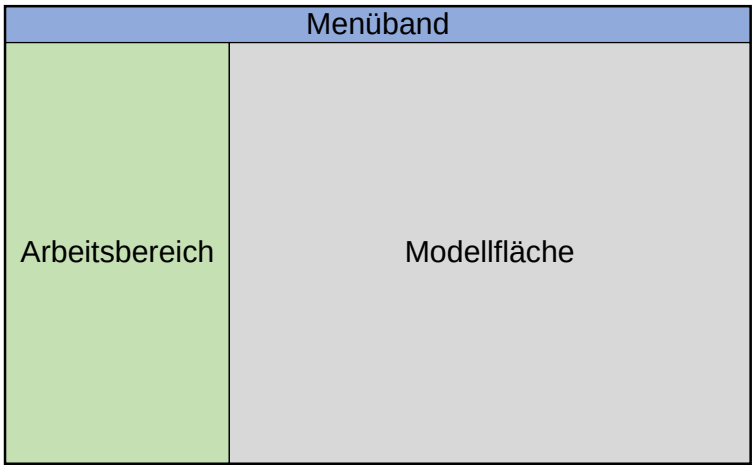


Abbildung 4.19: Schematischer Aufbau der grafischen Oberfläche

Über den Arbeitsbereich, der in Abbildung 4.19 als grün eingefärbte Fläche zu sehen ist, soll eine Vielzahl von Aufgaben erledigt werden können. Dazu gehören Aufgaben zur Modellbildung, wie die Auswahl von Bausteinen aus der Bausteinbibliothek von LogFarm, die Parametrisierung dieser Bausteine und das Anlegen jeweils eines Artikel- und Fahrzeugstamms. Weiterhin soll es über den Arbeitsbereich möglich sein, die Aufgaben zur Durchführung eines Experiments durchführen zu können. Dazu gehören die Spezifikation von Simulations- und Experimentparametern, das Spezifizieren eines externen Experimentplans, das Starten des Experiments und dessen Überwachung.

Die in Abbildung 4.19 grau eingefärbte Fläche soll die Modellfläche sein. Auf der Modellfläche sollen alle Bausteine, die die Akteure des Logistiknetzwerks darstellen, platziert werden können. Die Auswahl der Bausteine erfolgt im Arbeitsbereich. Zudem soll es möglich sein, die Beziehungen zwischen den Modellelementen durch Verbindungen zwischen Bausteinen modellieren zu können.

In Abbildung 4.19 stellt die blaue Fläche das Menüband dar. Hier sollen dem Nutzer weitere Funktionen zur Verfügung gestellt werden, die bisher nicht durch den Arbeitsbereich abgedeckt werden. Hierzu gehört beispielsweise das Laden und Speichern eines Modells, das Rückgängigmachen von Modellierungsschritten, oder das Aufrufen einer Hilfeseite.

Die grafische Benutzeroberfläche soll eine wichtige Rolle bei der Erstellung valider Modelle von Logistiknetzwerken gemäß des entwickelten generischen Konzeptmodells (vgl. Abschnitt 4.2) erfüllen, indem die Steuerelemente invalide Strukturen bereits vorzeitig bei der Modellbildung ausschließen. Beispielsweise soll sichergestellt werden, dass Produktionsraten von Gütern niemals negativ sein können, indem das betreffende Steuerelement zur Parametrisierung keine Eingabe negativer Zahlen erlaubt. Ein weiteres Beispiel ist, dass in den Bausteinen ausschließlich die Komponenten, gemäß der in Tabelle 4.1 dargestellten Gesetzmäßigkeiten, verwendbar sind.

4.4 Implementierung eines Prototyps

Der in diesem Kapitel vorgestellte Simulator soll in einem Prototypen umgesetzt werden, der als Vorführmodell dient (vgl. Anforderung VIII). Um dessen Entwicklungsaufwand gering zu halten, ist für die Implementierung des Simulators Drittsoftware notwendig. Die wichtigste Drittsoftware für den Simulator stellt das Graphdatenbanksystem dar, da es maßgeblich für die Leistungsfähigkeit des Simulators verantwortlich ist. Die Anforderungen an das Graphdatenbanksystem sind eine möglichst schnelle Lese- und Schreibgeschwindigkeit, die Umsetzung des Eigenschaftsgraphen-Modells, eine zuverlässige Zugriffsverwaltung bei paralleler Ansteuerung und die kostenfreie Nutzung für nicht-kommerzielle Zwecke. Eine Übersicht gängiger Graphdatenbanksysteme befindet sich in Besta et al. (2022). Nach dieser Übersicht erfüllen die Datenbanksysteme Neo4j, Sparksee/DEX, Memgraph, TigerGraph und Weaver die Anforderungen und weisen einen ähnlichen Funktionsumfang auf. Für den Simulator wird das populäre Graphdatenbanksystem Neo4j gewählt, da es die umfangreichste Dokumentation aufweist.

Neo4j bietet Schnittstellen zu verschiedenen Programmiersprachen an, jedoch ist laut Herstellerangaben die Verwendung der Java-Schnittstelle zu bevorzugen. Um die bevorzugte Java-Schnittstelle zu nutzen, bietet sich die Verwendung von Java zur Implementierung des Prototypen des Simulators an. Ohnehin ist Java eine der weit verbreitetsten Programmiersprachen mit einer Vielzahl von angebotenen Bibliotheken und Drittsoftware (Neumann 2019). Für die Erstellung grafischer Benutzeroberflächen in Java haben sich insbesondere zwei Bibliotheken etabliert, die Bibliotheken Swing und JavaFX (Neumann 2019). Für die Entwicklung der grafischen Benutzeroberfläche in LogFarm wurde JavaFX verwendet.

4.5 Grafische Benutzeroberfläche des Prototyps

In Abschnitt 4.2 wurde das grafische Konzeptmodell des Simulators vorgestellt, auf dem die grafische Benutzeroberfläche des Prototyps basiert. Da der Prototyp als Vorführmodell konzipiert ist, setzt die grafische Benutzeroberfläche nicht alle Aspekte des grafischen Konzepts um. Beispielsweise werden die Eingaben in den Steuerelementen nicht wie konzipiert auf Validität geprüft, sodass es leichter zu Fehlern in der Modellbildungen kommen kann. Eine beispielhafte Bildschirmaufnahme der grafischen Benutzeroberfläche des Prototypen mit einem Modell ist in Abbildung 4.20 dargestellt.

Die im Konzept der grafischen Benutzeroberfläche beschriebene Dreiteilung des Anwendungsfensters (vgl. Abbildung 4.19) ist in Abbildung 4.20 klar zu erkennen. Im oberen Bereich der Abbildung ist das Menüband umgesetzt worden. Für den Prototypen wurden im Menüband keine tatsächlichen Funktionen hinterlegt, lediglich über den Menüpunkt

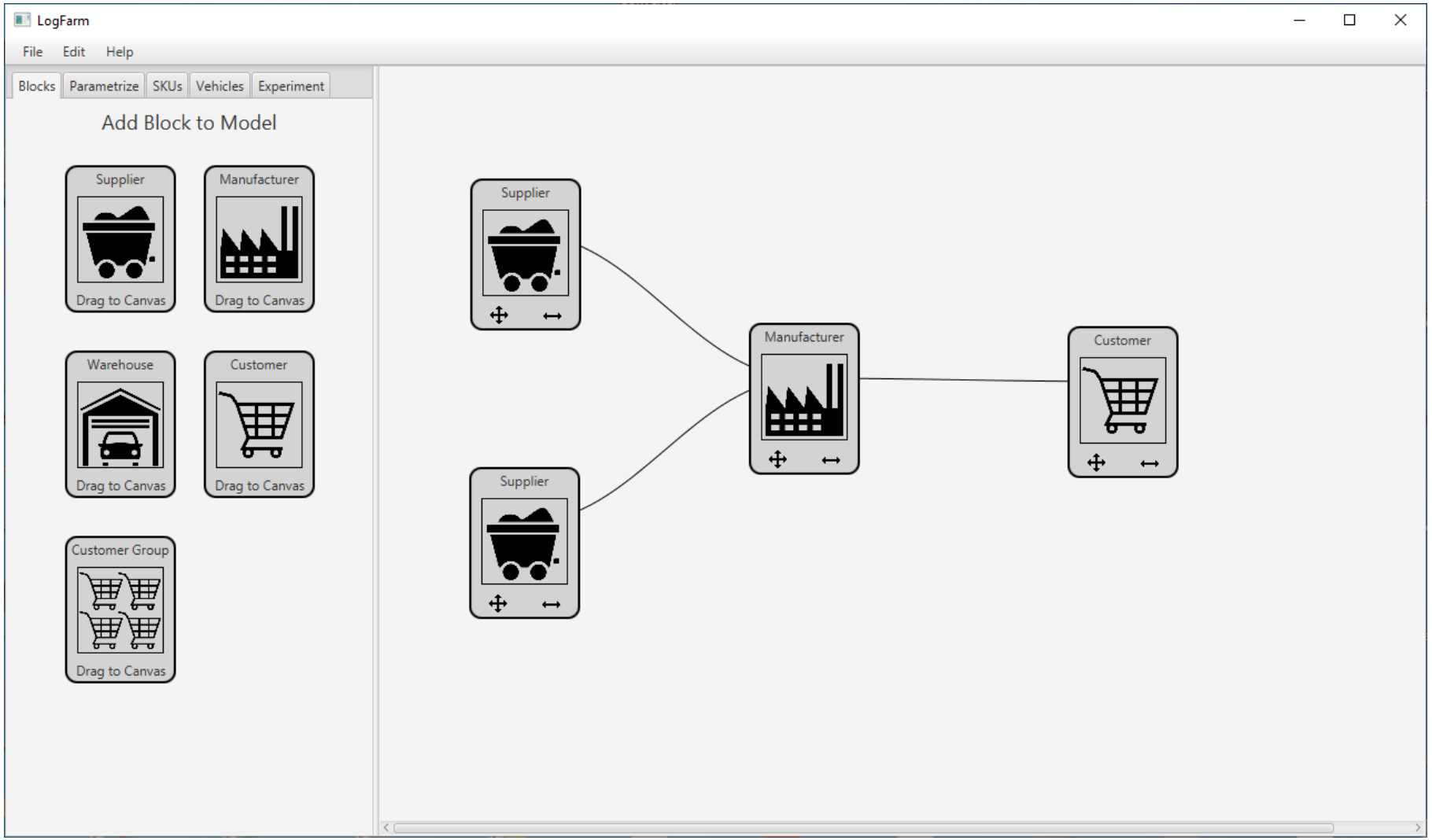


Abbildung 4.20: Bildschirmaufnahme des Anwendungsfensters des Prototyps mit einem beispielhaften Modell

Help können Informationen zum Prototypen über einen Dialog angezeigt werden. Der linke Bereich in der Abbildung ist der Arbeitsbereich. Um die Vielzahl der Aufgaben, die dort erledigt werden können, zu organisieren, wurden fünf Registerkarten verwendet. Zu den Registerkarten gehören die Bausteinbibliothek (*Blocks*), Parametrisierung (*Parametrize*), Artikel (*SKUs*), Fahrzeuge (*Vehicles*) und Experiment (*Experiment*). In der Registerkarte Parametrisierung können die Bausteine samt ihrer Komponenten parametrisiert werden. In den Registerkarten Artikel und Fahrzeuge können ein Artikel- bzw. Fahrzeugstamm hinterlegt werden. In der Registerkarte Experiment kann die Spezifikation der Simulations- und Experimentparameter vorgenommen werden, sowie das Experiment gestartet und überwacht werden. In der Abbildung ist zu erkennen, dass dort die Registerkarte der Bausteinbibliothek aktiv ist. Deshalb sind im Arbeitsbereich die fünf Bausteine zu erkennen, die in Abschnitt 4.3.1 identifiziert wurden. Die Bausteine sind durch Rechtecke mit abgerundeten Ecken dargestellt und weisen neben einem Namen ein Symbolbild auf, anhand dessen der Typ des Bausteins identifiziert werden kann. Die Bausteine können aus dem Arbeitsbereich auf die Modellfläche, die im rechten Bereich der Abbildung zu sehen ist, mit der Maus gezogen und abgelegt werden. In der Abbildung ist auf der Modellfläche ein beispielhaftes Modell dargestellt, bestehend aus Bausteinen von zwei Versorgern, einer Produktionsstätte und einem Kunden. Es ist zu erkennen, dass nach dem Platzieren eines Bausteins auf der Modellfläche im unteren Bereich der Bausteine zwei Symbole erscheinen. Durch erneutes Ziehen eines Bausteins an der Stelle des linken Symbols können die Bausteine verschoben werden. Durch Ziehen am rechten Symbol wird das Verknüpfungswerkzeug aktiviert und eine schwarze Linie erscheint am Mauszeiger. Durch das Ablegen der schwarzen Linie auf einem anderen Baustein wird eine Verknüpfung der beiden hergestellt. Verknüpfungen werden durch schwarze Linien visualisiert, wie in der Abbildung 4.20 zu sehen ist.

Auf eine detaillierte Beschreibung jeglicher Ansichten und Steuerelemente der grafischen Benutzeroberfläche des Prototyps wird in dieser Arbeit verzichtet. Exemplarisch wird die Parametrisierung einer Transformationskomponente betrachtet, die in Abbildung 4.21 dargestellt ist.

Die in Abbildung 4.21 abgebildete Ansicht in der Registerkarte Parametrisierung zur Parametrisierung einer Transformationskomponente gehört zu einem Produktionsstätten-Baustein. Aus diesem Grund ist die Kontrollbox gesperrt, da laut Tabelle 4.1 eine Produktionsstätte immer eine Transformationskomponente aufweisen muss. In der Abbildung wurde ein Transformationsprozess modelliert. Die Transformationsvorschrift besagt, dass für jede Ausführung des Prozesses zweimal *sku1* und dreimal *sku2* verbraucht werden, um einmal *sku3* herzustellen. Die Transformationsvorschrift kann um weitere Artikel erweitert bzw. gekürzt werden, indem die unbeschrifteten Schaltflächen mit dem Plus- oder Minus-Symbol verwendet werden. Die Anzahl der Transformationsschritte kann über die beschrifteten Schaltflächen *Add Process* und *Remove Process* variiert werden. In Abbildung 4.21a ist im obersten Eingabefeld eine feste Prozessrate von 2.5 spezifiziert. Wie in Abschnitt 4.2 beschrieben, sollen einige Parameter als variable Faktoren dienen, um diese mittels eines Experimentplans für jeden Simulationslauf eines Experiments zu spezifizieren. Um aus einem festen Parameter einen variablen Faktor zu machen, kann der Benutzer durch Klicken des betreffenden Eingabefelds unter gleichzeitigem Drücken der *STRG*-Taste die Umwandlung durchführen. Eingabefelder, die einen Faktor abbilden, werden Gelb markiert. Der in solchen Eingabefeldern hinterlegte Wert muss der Position des Faktors im Experimentplan entsprechen, der verwendet werden soll. Im Beispiel in Abbildung 4.21b ist der Wert sechs hinterlegt, das bedeutet, dass das sechste Datenfeld im Experimentplan die Prozessrate der modellierten Transformationskomponente steuert. Im

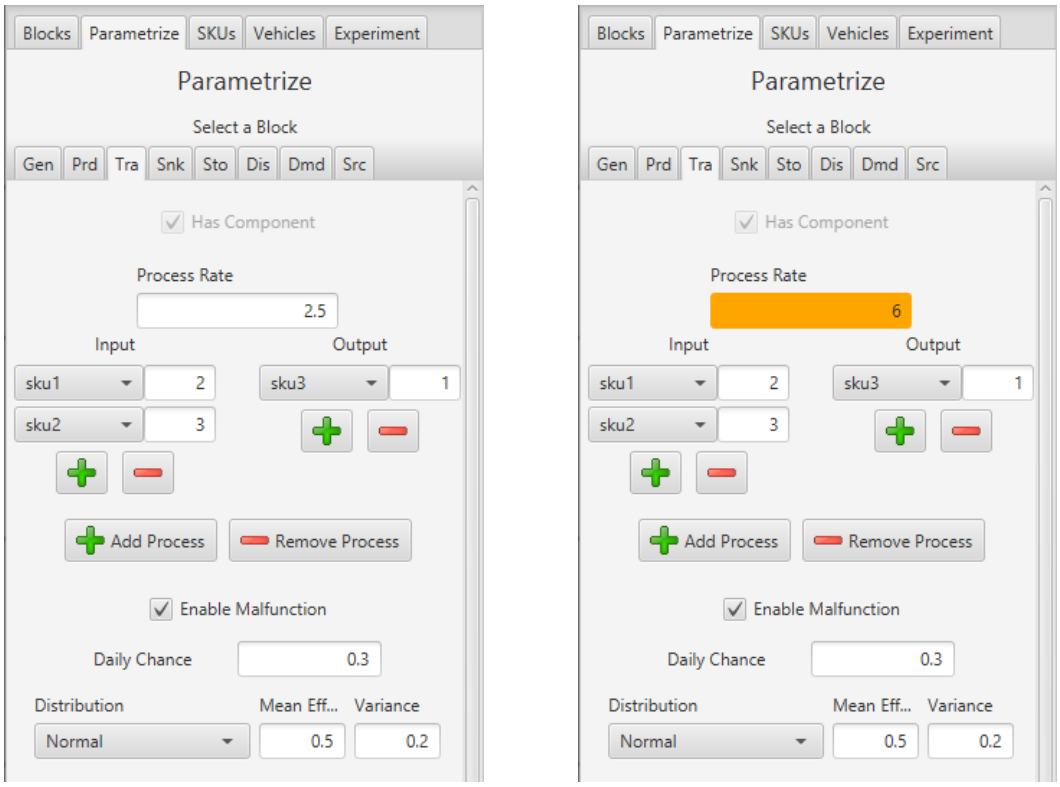


Abbildung 4.21: Beispielhafte Parametrisierung einer Transformationskomponente (a) mit fester Produktionsrate (b) mit variabler Produktionsrate

unteren Bereich der beispielhaften Parametrisierung ist eine weitere Kontrollbox zu sehen, welche dazu verwendet werden kann, die Komponente mit einer zusätzlichen Störungskomponente zu versehen. Da die Kontrollbox nicht gesperrt ist, handelt es sich hierbei um eine Option für den Modellierer. Im gezeigten Beispiel wurde eine Störung modelliert, die zu einer Wahrscheinlichkeit von 30% pro Tag auftreten kann und im Fall einer Störung die verbleibende Prozessrate auf durchschnittlich 50% der ursprünglichen Rate vermindert. Die verbleibende Prozessrate wird über eine Normalverteilung modelliert, für die als Erwartungswert 50% und als Varianz 20% spezifiziert wurde.

Ein weiteres Beispiel der grafischen Benutzeroberfläche ist in Abbildung 4.22 dargestellt, welche die Registerkarte *Experiment* zeigt.

In den oberen drei Eingabefeldern der Abbildung 4.22 können die Simulationsparameter, die aus dem Start-, Warmlauf- und Enddatum bestehen, mithilfe eines Kalender-Steuerelements spezifiziert werden. Nach Betätigung der Schaltfläche mit der Beschriftung *Experiment Plan* öffnet sich ein Dialog, über den der Benutzer einen externen Experimentplan im CSV-Format in die Anwendung laden kann. Bei erfolgreicher Hinterlegung eines Experimentplans ändert sich das Textfeld neben der Schaltfläche so, dass der Name der hinterlegten Datei angezeigt wird. In den drei übrigen Eingabefeldern können die Experimentparameter spezifiziert werden. Mit der Schaltfläche *Build Model* implementiert LogFarm ein ausführbares Modell, basierend auf dem im Editor erstellten Modell. Das Modell wird zudem in der Modelldatenbank hinterlegt. In der Abbildung ist zu sehen, dass die Schaltfläche *Start Experiment* gesperrt ist. Erst wenn ein ausführbares Modell erstellt wurde und zudem Simulationsparameter und ein Experimentplan hinterlegt wurden,

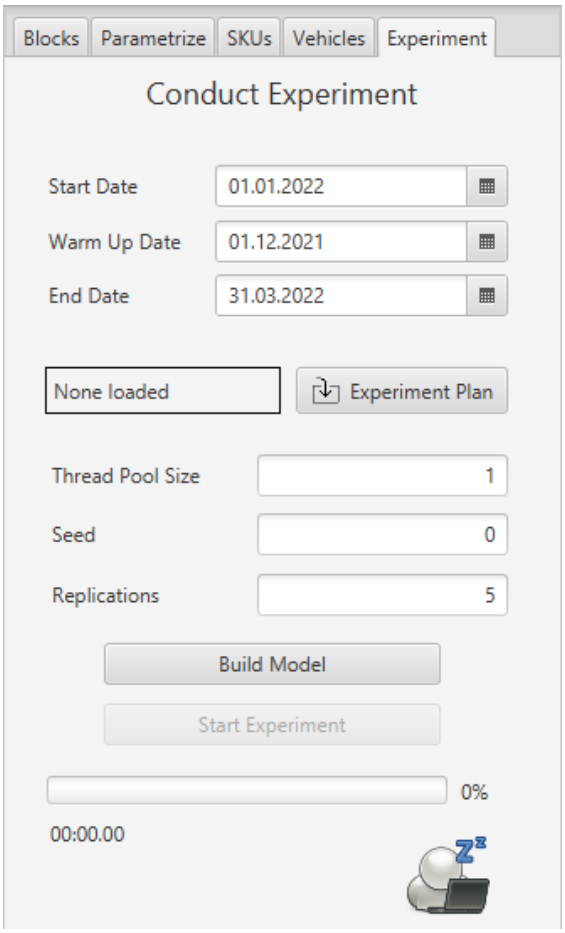


Abbildung 4.22: Bildschirmaufnahme der Registerkarte *Experiment*

wird die Sperre der Schaltfläche aufgehoben und ein Experiment kann gestartet werden. Der Fortschritt des Experiments kann über den Fortschrittsbalken im unteren Bereich der Abbildung eingesehen werden. Hier wird stets visuell der Prozentsatz angezeigt, wie viele Simulationsläufe anteilig an der Gesamtzahl der Simulationsläufe im Experiment bereits durchgeführt wurden. Das Textfeld unter dem Fortschrittsbalken zeigt die Zeit an, die seit Beginn des Experiments vergangen ist. Die Zeitmessung endet zeitgleich mit dem Experiment. Das Symbol in der rechten unteren Ecke der Abbildung visualisiert den Status des Experiments. Während der Ausführung des Experiments wird das Symbol einer Sanduhr angezeigt. Tritt ein Fehler, beispielsweise ein Abbruch der Verbindung zur Datenbank, im Experiment auf, wird ein rotes Kreuzsymbol angezeigt. Endete das Experiment ohne Fehler, wird das Symbol eines grünen Hakens dargestellt.

5 Exemplarische Anwendung des Simulators anhand von Fallbeispielen

Im vorangegangen Kapitel wurde der Simulator LogFarm vorgestellt. In diesem Kapitel wird die exemplarische Anwendung von LogFarm anhand zweier Fallbeispiele demonstriert. Dazu wird in Abschnitt 5.1 zunächst eine Planung der Fallbeispiele durchgeführt. In Abschnitt 5.2 und Abschnitt 5.3 werden die geplanten Fallbeispiele bearbeitet und anschließend in Abschnitt 5.4 ein Fazit hinsichtlich der Anwendung von LogFarm nach einer Diskussion gezogen. Die im Folgenden vorgestellten Fallbeispiele können mit dem Prototypen aus Abschnitt 4.4 nachvollzogen werden. Neben LogFarm werden dazu lediglich eine lokale Installation des Graphdatenbanksystems Neo4j (vgl. Abschnitt 2.3.1) und die entsprechenden Experimentpläne der Fallbeispiele, die zusammen mit LogFarm veröffentlicht werden, benötigt. Zudem wurden zwei Bedienelemente der Registerkarte *Experiment* der grafischen Benutzeroberfläche hinzugefügt, über die die Experimente der Fallbeispiele, in der in dieser Arbeit vorgestellten Form, durchgeführt werden können.

5.1 Planung der Fallbeispiele

Um Funktionsweise und -umfang des Prototyps von LogFarm zu veranschaulichen, werden zwei Fallbeispiele vorgestellt. Die beiden Fallbeispiele weisen unterschiedliche Szenarien auf, für die jeweils ein exemplarischer Data-Farming-Prozess mit LogFarm beschrieben wird.

Die Schritte zur Durchführung eines Data-Farming-Prozesses wurden in Abschnitt 2.3.3 beschrieben. Dazu gehören der schnelle Prototypenbau, die Modellentwicklung, die statistische Versuchsplanung, die leistungsstarken Berechnungen sowie die Analyse und Visualisierung. In den Fallbeispielen werden lediglich die Modellentwicklung mit LogFarm, die statistische Versuchsplanung mit anderen Werkzeugen und die leistungsstarke Berechnung, beleuchtet. Die Szenarien werden vorgegeben und werden nicht durch einen schnellen Szenario-Prototypenbau erhalten. Folgend der Argumentation von Hunker et al. (2021) und Feldkamp et al. (2018), die die Analyse und Visualisierung der Ergebnisse nicht zu den Kernaufgaben des Data-Farmings zählen, erfolgt keine tiefgehende Analyse der generierten Daten (vgl. Abschnitt 2.3.3). In Abschnitt 2.3.3 wurde zudem beschrieben, dass ein Data-Farming-Prozess iterativ durchgeführt werden kann. Für die Fallbeispiele wird sich lediglich auf eine Iteration beschränkt.

Die Szenarien der beiden Fallbeispiele behandeln Logistiknetzwerke und weisen eine Beschreibung der Akteure, deren Beziehungen untereinander und eine zu untersuchende Fragestellung auf. Zusätzlich wird spezifiziert, welche Parameter bei der statistischen Versuchsplanung als variabel berücksichtigt werden. Die Szenarien werden darauf folgend mithilfe des integrierten Editors von LogFarm in ein ausführbares Simulationsmodell überführt. Das entspricht der Phase der Modellentwicklung in einem Data-Farming-Prozess. Basierend auf den Szenarien werden anschließend die statistischen Versuchsplanungen

durchgeführt. Die Generierung der dazugehörigen Experimentierpläne erfolgt in externen Werkzeugen. Zum Abschluss werden alle geplanten Simulationsläufe mit LogFarm durchgeführt und dabei Daten generiert. Als quantifizierbares Ergebnis der Fallbeispiele sollen zudem Kennzahlen erhoben und im Folgenden spezifiziert werden.

Zu den Ergebnissen der Fallbeispiele gehören Daten über die erstellten Modelle, die anhand des generischen Schemas, das in Abschnitt 4.2 vorgestellt wurde, modelliert wurden. Dies umfasst die Anzahl der Knoten, Beziehungen und Eigenschaften, die für die Speicherung in Form eines Eigenschaftsgraphen nötig sind. Zusätzlich wird der Platzbedarf der Modelle hinsichtlich ihrer Dateigröße erfasst und die benötigte Zeit für Schreibvorgänge in die Graphdatenbank ermittelt.

Ein weiterer Teil der Ergebnisse ist die Untersuchung der zeitlichen Abläufe der Experimente. Neben der Erfassung der benötigten Gesamtzeit zur Durchführung des Experiments werden auch Zeiten zur Durchführung von Teilprozessen gemessen. Die zu messenden Zeiten sind:

- Gesamtlaufzeit des Experiments
- Dauer der Vorbereitungszeit eines Experiments
- Durchschnittliche Laufzeit eines Simulationslaufs
- Durchschnittliche Zeit zur Erstellung einer Arbeitskopie
- Durchschnittliche Simulationszeit
- Durchschnittliche Dauer eines Schreibvorgangs von generierten Daten eines Simulationslaufs in die Datenbank generierter Daten

Die durchschnittlichen Zeiten beziehen sich jeweils auf alle Simulationsläufe eines Experiments. Zusätzlich zum zeitlichen Ablauf wird auch die Auslastung der Computerressourcen erhoben. Dazu werden die Prozessor- und Arbeitsspeicherauslastung gemessen. Für die Messung wird die Windows-Leistungsüberwachung verwendet, die nativ zum Betriebssystem Windows 10 gehört. Mit diesem Werkzeug können in einem beliebigen Intervall vorher spezifizierte Leistungsindikatoren gemessen werden. Für die Fallbeispiele wird ein Messintervall von fünf Sekunden gewählt und als Leistungsindikatoren die Prozessorauslastung und der belegte Arbeitsspeicher verwendet.

Die Simulation der Fallbeispiele erfolgt auf einem einzelnen Computer. Wichtige Spezifikationen der Hard- und Software des Computers können Tabelle 5.1 entnommen werden. Insgesamt entspricht die Leistungsfähigkeit des Systems der eines Mittelklasse-Computers.

Tabelle 5.1: Hardware- und Softwarespezifikation des Computers auf dem die Fallbeispiele simuliert werden

Spezifikation	**Ausprägung**
Betriebssystem	Windows 10 Home (64-Bit, 21H1)
Prozessor	Intel(R) Core i5-3470 (3,20GHz, 4 Kerne)
Arbeitsspeicher	4 mal 4GB DDR3 (16GB insgesamt)
Festplatte	Samsung SSD 850 EVO 500GB (Lesen: 540MB/s, Schreiben: 520MB/s)
Mainboard	Gigabyte Technology H77-D3H

5.2 Fallbeispiel 1

Im ersten Fallbeispiel wird ein imaginäres Logistiknetzwerk betrachtet, das einem typischen Logistiknetzwerk nachempfunden ist. Nach Simchi-Levi et al. (2000) sind die typischen Teilnehmer in einem Logistiknetzwerk Versorger, Hersteller, Warenhäuser und Kunden (vgl. Abschnitt 3.3). Weiterhin würden Rohmaterialien beschafft und aus diesen Produkte erstellt, die dann zunächst im Warenhaus lagern und von dort an Kunden verteilt werden (vgl. Abschnitt 3.3).

5.2.1 Szenario

Das Unternehmen Engel möchte ein hochwertiges Lautsprechersystem in einem Holzgehäuse herstellen. Optional soll zu diesem Produkt eine passende Wandhalterung angeboten werden. Dazu wird eine innerbetriebliche Supply Chain mit mehreren Standorten in Deutschland geplant. Im Planungsprozess sollen mithilfe eines Farming-for-Mining-Frameworks (vgl. Abschnitt 2.3.3) die Planungsergebnisse hinsichtlich verborgenen Wissens untersucht werden.

Das Unternehmen will insgesamt fünf Standorte betreiben. Dazu gehören ein Sägewerk, ein Standort zur Herstellung von Halbfabrikaten, ein Standort zur Fertigung des Lautsprechersystems, ein Standort zur Herstellung von Wandhalterungen und ein Warenhaus. Die Lautsprechersysteme und Wandhalterungen werden in drei Verkaufsstellen vertrieben. In Abbildung 5.1 werden die Standorte und ihre Beziehungen untereinander visualisiert.

Wie in der Abbildung 5.1 zu sehen, werden das Sägewerk, die Standorte in denen Güter produziert werden und das Warenhaus in der Mitte Deutschlands verortet. Die drei Verkaufsstellen des Unternehmens befinden sich in Berlin, Dortmund und Frankfurt am Main. Der gerichtete Güterfluss der Supply Chain kann den Pfeilen der Abbildung 5.1 entnommen werden. Das Sägewerk und der Standort zur Fertigung der Halbfabrikate beliefern den Standort zur Fertigung des Lautsprechersystems. Von dort aus werden die Lautsprechersysteme an das Warenhaus geliefert. Vom Standort zur Herstellung der Wandhalterung existiert eine direkte Lieferbeziehung zum Warenhaus. Das Warenhaus liefert die Güter an die drei Verkaufsstellen des Unternehmens. Alle Transporte, die in der Supply Chain getätigt werden, können innerhalb eines Tages erfolgen.

Das Sägewerk verarbeitet lokal geschlagenes Rundholz zu Brettern. Es ist ein Lager vorhanden, in dem die Bretter zwischengelagert werden können. Der Transport der Bretter erfolgt mit unternehmenseigenen Fahrzeugen. Aufgrund der kompakten Form der Bretter können die im gesamten Unternehmen üblichen LKW zum Transport verwendet werden. Durch Erfahrungswerte ist bekannt, dass die Produktion der Bretter oftmals durch kurze Störungen beeinflusst wird, die durch eine Sägevorrichtung verursacht werden.

Im Standort zur Herstellung von Halbfabrikaten werden sowohl Lautsprecher gefertigt, als auch die Platinen des Lautsprechersystems bestückt. Die dazu nötigen Rohteile und Rohstoffe werden von Drittanbietern geliefert. Der Standort weist ein großzügig dimensioniertes Lager auf, in dem es zu keiner Zeit zur fehlendem Platz kommen sollte. Der Weitertransport der Waren kann mit dem üblichen LKW des Unternehmens erfolgen.

In dem Standort zur Herstellung der Lautsprechersysteme findet die Montage der Bestandteile statt. Zur Montage eines Lautsprechersystems werden ein Brett, eine Platine und zwei Lautsprecher benötigt. Der Produktionsschritt zur Verarbeitung eines Bretts zu einem Gehäuse soll nicht detailliert betrachtet werden. In einem Zwischenlager wer-

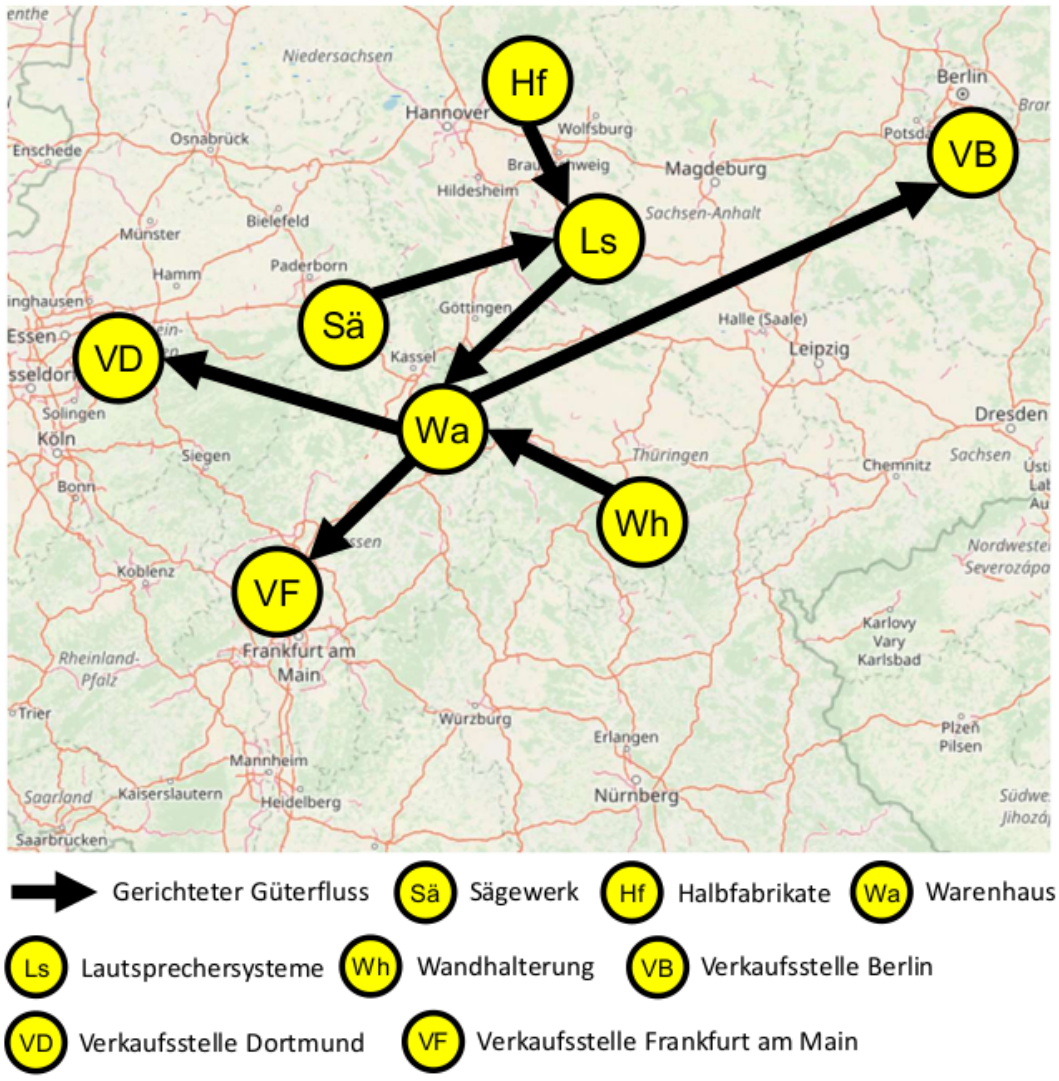


Abbildung 5.1: Standorte und Beziehungen Fallbeispiel 1 auf einer Landkarte

den die fertigen Lautsprechersysteme kurzzeitig gelagert, bevor sie mithilfe der Standard Auslieferungsfahrzeuge zum Warenhaus transportiert werden.

An einem weiteren Standort des Unternehmens werden die Wandhalterungen gefertigt, die optional zum Lautsprechersystem erworben werden können. Die durch einen Drittanbieter gewährleistete Bereitstellung von Rohstoffen, die zur Fertigung der Wandhalterungen notwendig sind, werden nicht tiefgehend betrachtet. Es ist jedoch bekannt, dass bei der Fertigung in seltenen Fällen schwerwiegende Störungen auftreten können. Der Standort verfügt über ein kleines Lager und transportiert die Wandhalterungen mithilfe der Auslieferungsfahrzeuge des Unternehmens zum Warenhaus.

Das Warenhaus wird sowohl zum Lagern, als auch zum Umschlagen der Lautsprechersysteme und Wandhalterungen verwendet. Von hier werden Auslieferungsfahrzeuge zu den drei Verkaufsstellen geschickt. Die Auslieferungsfahrzeuge des Warenhauses unterscheiden sich nicht von den anderen im Unternehmen verwendeten Fahrzeugen. Ein Fahrzeug kann aufgrund der geografischen Lage der Verkaufsstellen immer nur eine Verkaufsstele pro Tag bedienen. Die Lagerhaltung des Warenhauses wird durch eine *(s,q)*-Politik bestimmt (vgl.

Abschnitt 3.2.1). Der Kundenentkopplungspunkt des Logistiknetzwerks soll im Warenhaus liegen (vgl. Abschnitt 3.3).

Die drei Verkaufsstellen befinden sich in verschiedenen Städten in Deutschland, wie in Abbildung 5.1 zu sehen. Die tägliche Nachfrage fällt abhängig vom Standort unterschiedlich aus und kann mitunter stark variieren. Mit der stärksten Nachfrage wird in Berlin gerechnet, gefolgt von Frankfurt am Main. In der Verkaufsstelle in Dortmund wird mit der niedrigsten Nachfrage geplant. Die Lagerhaltung der drei Verkaufsstellen muss nicht detailliert betrachtet werden. Es kann davon ausgegangen werden, dass jedes beim Warenhaus bestellte Lautsprechersystem und jede Wandhalterung verkauft wird. Aufgrund hoher Nachfrage und geringer Lagerkapazitäten vor Ort findet eine Nachbestellung der Güter beim Warenhaus regelmäßig alle paar Tage statt.

Einige Parameter der Simulationsstudie sollen für verschiedene Wertestufen untersucht werden. Dazu gehören die Produktionsraten der Bretter, Wandhalterungen, Lautsprecher und Platinen. Auch die Transformationsrate der Lautsprechersysteme doll dynamisch gehalten werden. Weiterhin sollen alle Parameter zur Beschreibung der Störungen im Sägewerk und der Fabrik für Halbfabrikate als Faktoren behandelt werden. Zuletzt ist auch die Nachfrage nach Lautsprechersystemen und Wandhalterungen in allen Verkaufsstellen dynamisch zu gestalten.

5.2.2 Modellierung und Durchführung

Das zuvor beschriebene Szenario wird nun mithilfe des Editors von LogFarm modelliert. Die verwendeten Werte der Eigenschaften werden in diesem Abschnitt nicht spezifiziert, können aber in Anhang eingesehen werden.

Zunächst werden alle Artikel in das Artikelverzeichnis eingetragen. Entsprechend dem Szenario werden die Artikel Brett, Platine, Lautsprecher, Lautsprechersystem und Wandhalterung verwendet. Auf die Verwendung von Rundholz als Artikel wird verzichtet, da dieser nicht explizit modelliert wird. Als nächstes werden alle verwendeten Fahrzeugklassen dem Fahrzeugklassenverzeichnis hinzugefügt. In diesem Fall gibt es lediglich die Fahrzeugklasse LKW. Im Folgenden werden die Bausteine zur Modellierung der Standorte beschrieben.

Das Sägewerk produziert Bretter aus Rundholz. Zwar ist dieser Vorgang ein Transformationsprozess, jedoch kann vereinfachend ein Produktionsprozess für Bretter modelliert werden. Da an dem Standort somit nur ein Produktions und kein Transformationsprozess stattfindet, wird ein Baustein der Rolle Zulieferer verwendet. Zusätzliche Komponenten sind die Lager- und Distributionskomponente. Zudem wird in der Produktionskomponente die Option aktiviert, die eine mögliche Störung der Komponente vorsieht. Entsprechend der Szenariobeschreibung wird eine hohe tägliche Wahrscheinlichkeit eines Ausfalls verwendet, der jedoch den Betrieb nur wenig beeinträchtigt.

Der Standort zur Herstellung von Platinen und Lautsprechern wird wie das Sägewerk als Zulieferer modelliert, da nur Produktionsprozesse ablaufen. Dem Baustein werden in der Produktionskomponente zwei Quellen hinzugefügt worden, um die Produktion von Lautsprechern und Platinen zu modellieren. Wie beim Sägewerk werden eine Lager- und eine Distributionskomponente verwendet. Auch in diesem Baustein wird eine mögliche Störung hinterlegt. Im Vergleich zum Sägewerk erfolgt diese zwar seltener, beeinträchtigt den Betrieb jedoch maßgeblich.

Im Standort zur Herstellung der Lautsprechersysteme sollen die Güterströme der Bretter, Platinen und Lautsprecher zusammengeführt und zu besagtem Lautsprechersystem verar-

beitet werden. Dieser Prozess stellt eine Transformation von Artikeln dar und wird somit mit einer Transformationskomponente dargestellt. Daher ist der ausgewählte Baustein ein Produktionsstätten-Baustein. Die Transformationsvorschrift der Transformationskomponente besagt, dass ein Brett, eine Platine und zwei Lautsprecher zu einem Lautsprechersystem transformiert werden können. Die Produktionsstätte Lautsprechersysteme weist weiterhin eine Lager- und eine Distributionskomponente auf.

Die Modellierung des Standorts zur Herstellung von Wandhalterungen geschieht identisch zu der Modellierung des Sägewerks. Es wird ein Zulieferer-Baustein gewählt mit einer Produktions-, einer Lager- und einer Distributionskomponente. Im Gegensatz zum Sägewerk wird die Produktionskomponente nicht mit einer möglichen Störung modelliert.

Das Warenhaus des Unternehmens wird mit einem Warenhaus-Baustein abgebildet. In dem Baustein werden die Lager-, und Distributionskomponente verwendet. Als Lagerhaltungspolitik der beiden Artikel wurde jeweils die *(s,q)*-Politik verwendet. Die drei Verkaufsstellen sind über statische Transportrelationen mit dem Warenhaus verbunden.

Eine Bildschirmaufnahme der Modellierung mit Bausteinen auf der Modellfläche des Editors ist in Abbildung 5.2 dargestellt.

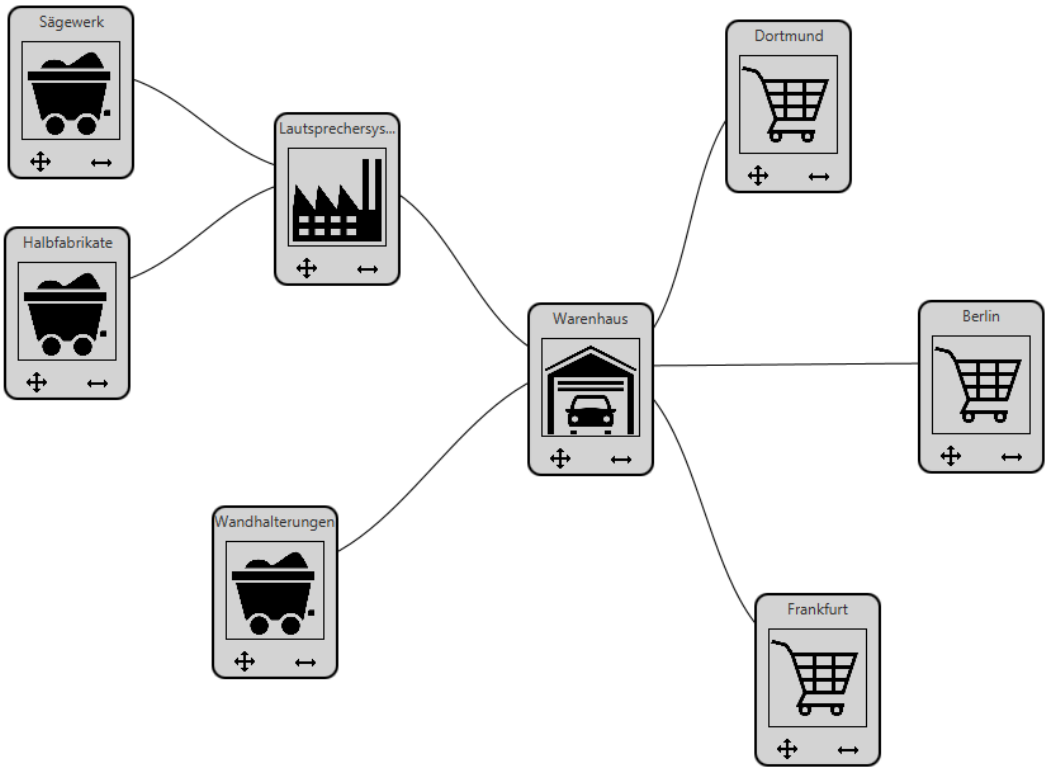


Abbildung 5.2: Bildschirmaufnahme der Modellfläche mit Modell von Fallbeispiel 1

In Abbildung 5.2 sind die acht Standorte des Logistiknetzwerks und deren Verbindungen untereinander wie in Abbildung 5.1 zu sehen. Die Namen der Bausteine wurden dem Szenario angepasst und die nötige Parametrisierung vorgenommen. Ein Abbild der Modelldatenbank, nach Überführung der Modellierung in ein ausführbares Modell durch den Editor, ist in Abbildung 5.3 dargestellt.

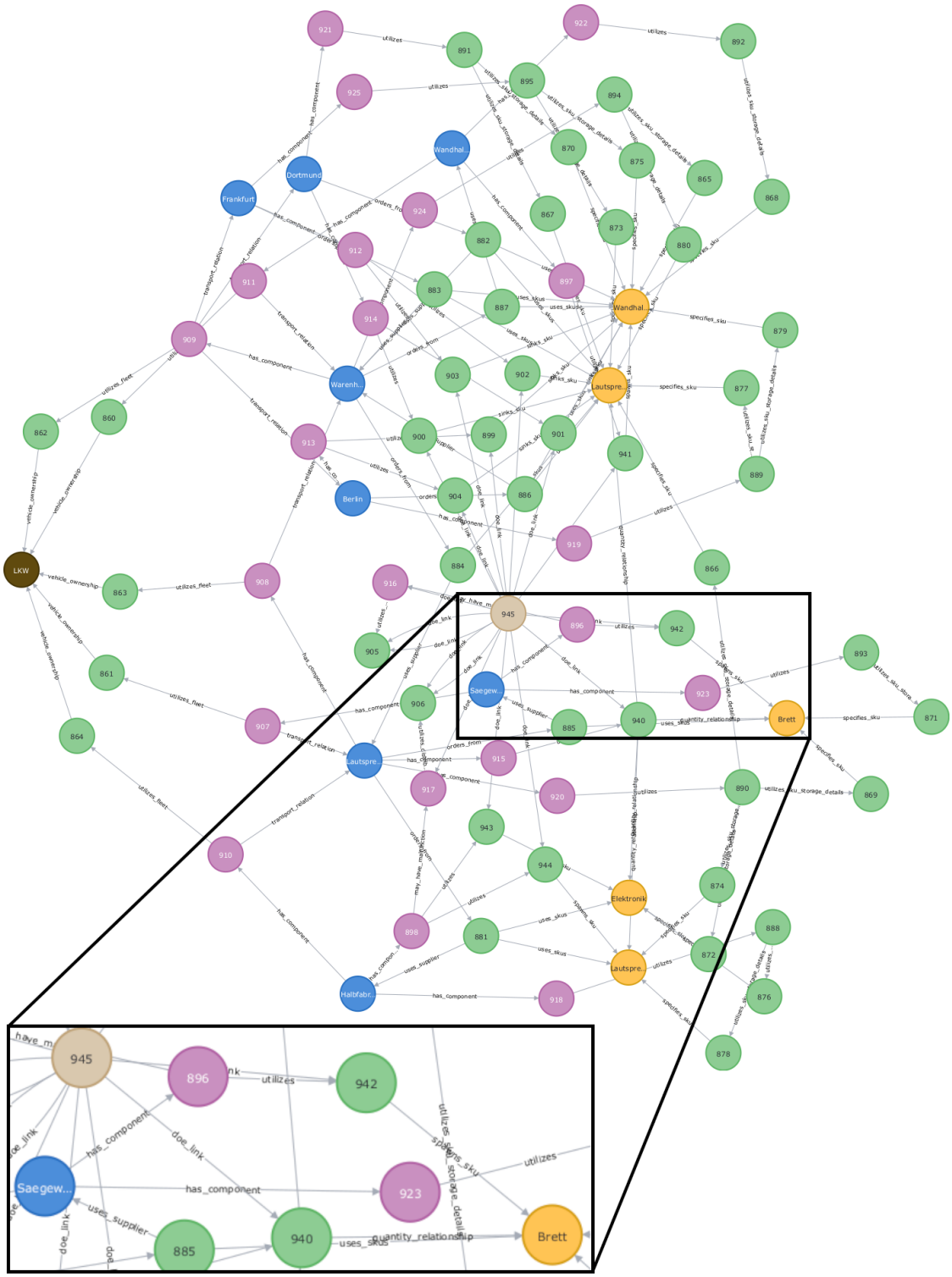


Abbildung 5.3: Visualisierung der Modelldatenbank von Fallbeispiel 1

Die Abbildung 5.3 zeigt die Vernetzung der Daten im Simulationsmodell auf, die selbst bei wenigen Teilnehmern im Logistiknetzwerk und einer sehr abstrahierten Modellierung stark ist. In der vergrößerten Ansicht im unteren linken Bereich der Abbildung kann der Aspekt der Modellierung, der die Produktion von Brettern im Sägewerk beschreibt, beispielhaft nachverfolgt werden. Der blaue Knoten mit der Aufschrift *Saegew* hat eine Komponente,

deren Knoten die Aufschrift *896* trägt. Diese Komponente ist die Produktionskomponente. Die Aufschrift des Knotens, wie bei den anderen Knoten die eine numerische Aufschrift aufweisen, entspricht der von Neo4j intern vergebenen Identifikationsnummer. Die Produktionskomponente ist mit einem grünen Knoten verbunden, dessen Aufschrift *942* ist. Dieser Knoten ist die Quelle. Die Produktionsrate ist durch eine Eigenschaft in diesem Knoten abgebildet. Durch die Verbindung zum gelben Knoten mit der Aufschrift *Brett* wird spezifiziert, welcher Artikel in der Quelle produziert wird. Einige weitere Informationen zur Beschreibung der Modelldatenbank sind in Tabelle 5.2 aufgeführt.

Tabelle 5.2: Messgrößen der Modelldatenbank in Fallbeispiel 1

Messgröße	Wert
Knoten in Modelldatenbank	86
Kanten in Modelldatenbank	148
Eigenschaften in Modelldatenbank	172
Größe der Modelldatenbank	2,55 MB

Aus den acht Bausteinen der Modellfläche wurde ein ausführbares Modell mit insgesamt 86 Knoten generiert. Zwischen den Knoten der Modelldatenbank befinden sich 148 Kanten. Die Anzahl der Eigenschaften, die entweder den Knoten oder Kanten zugewiesen sind, beträgt 172. Hier sei angemerkt, dass die internen Identifikationsnummern von Neo4j nicht einbezogen werden. Die Größe der Daten der Modelldatenbank beträgt 2,55 MB. Die Parameter der Bausteine und der Komponenten können Tabelle A.1 entnommen werden.

Das Szenario beschreibt insgesamt 17 Faktoren, die untersucht werden sollen. Zur Konstruktion eines geeigneten Experimentplans wird ein Nearly Orthogonal Latin Hypercube verwendet (vgl. Abschnitt 2.1.2). Das SEED Center for Data Farming (Sanchez 2011) bietet ein Werkzeug an, mit dem ein solcher Experimentplan erstellt werden kann. Das verwendete Design des Nearly Orthogonal Latin Hypercubes basiert auf einer Arbeit von Cioppa und Lucas (2007). Der Experimentplan wurde in ein mit LogFarm kompatibles CSV-Format überführt und umfasst 129 Simulationsläufe. Der Experimentierplan kann mithilfe des vorgestellten Tools und der in Tabelle A.2 spezifizierten Werte nachgebildet werden. Der Data-Farming-Vorgang wird für einen Beobachtungszeitraum von zwölf Monaten durchgeführt und hat eine zusätzliche Woche als Warmlaufphase eingeplant. Für jeden Simulationslauf werden vier Replikationen betrachtet, wodurch sich eine Gesamtzahl von auszuführenden Simulationsläufen von 516 ergibt. Für das Experiment wurde eine Thread-Pool-Größe von zwei verwendet.

5.2.3 Ergebnisse

Bei der Ausführung des Experiments wurden alle in Abschnitt 5.1 beschriebenen Größen erhoben. Die Ergebnisse der Erhebung können Tabelle 5.2 entnommen werden.

Durch die 516 durchgeführten Simulationsläufe wurde die Ergebnisdatenbank mit 1.894.831 Knoten und 10.282.736 Kanten befüllt. Die Anzahl der Eigenschaften in der Ergebnisdatenbank beträgt 2.973.295 Einträge. Der Umfang generierter Daten beträgt 512,59 MB was einer Rate von etwa 1,2 MB pro Sekunde entspricht, wenn diese Menge auf die Dauer des Experiments von 429,42 Sekunden (etwas über sieben Minuten) bezogen wird. Die Vorbe-

Tabelle 5.3: Ergebnisse des Experiments von Fallbeispiel 1

Messgröße	Wert
Knoten in Ergebnisdatenbank	1.894.831
Kanten in Ergebnisdatenbank	10.282.736
Eigenschaften in Ergebnisdatenbank	2.973.295
Größe der Ergebnisdatenbank	512,59 MB
Benötigte Zeit für das Experiment	429,42s
Vorbereitungszeit	0,826s
Durchschnittlich benötigte Zeit für einen Simulationslauf	1,657s
Durchschnittliche Kopierzeit	0,001s
Durchschnittliche Rechenzeit	0,007s
Durchschnittliche Schreibzeit	1,649s

reitungszeit des Experiments betrug 0,826s und weist somit nur einen geringen Anteil an der insgesamt benötigten Zeit auf.

Ein weiteres zu untersuchendes Ziel ist es, die durchschnittlich für einen Simulationslauf benötigte Zeit zu analysieren. Dazu bietet sich die Betrachtung der einzelnen Zeiten an, die summiert die Dauer eines Simulationslaufs ergeben. Die Dauer eines Simulationslaufs ergibt sich aus dem Addieren der Kopier-, Rechen- und Schreibzeit und betrug durchschnittlich 1,657 Sekunden. Tabelle 5.3 ist zu entnehmen, dass die Kopierzeit davon lediglich 0,001 Sekunden beansprucht, was einem Anteil von 0,06% entspricht. Die Rechenzeit der Simulation, die durchschnittlich 0,007 Sekunden benötigte, ist mit einem Anteil von 0,42% beteiligt. Zusammen ergibt der Anteil dieser beiden Zeiten nur 0,48% der Gesamtdauer und ist damit vernachlässigbar. Fast die gesamte Zeit, in diesem Fall 99,52%, wird für den Schreibvorgang in die Ergebnisdatenbank benötigt. Somit ist dieser Vorgang eindeutig der Engpass des Data-Farming-Vorgangs in Fallbeispiel 1. Die bisher analysierten Werte basieren auf den Durchschnittswerten, aus denen nicht ersichtlich ist, wie konstant die Werte erreicht werden. Dazu bietet sich die Verwendung von der Box-Plot Methode an, die in Abbildung 5.5 dargestellt sind.

In Abbildung 5.4 ist ersichtlich, dass die Interquartilsabstände der Kopier- und Rechenzeiten niedrig sind. Auch sind keine signifikanten Ausreißer zu entdecken. Der Interquartilsabstand der Schreibzeit ist auch gering. Auffällig ist jedoch, dass eine hohe Anzahl von Ausreißern zu beobachten ist. Die Ausreißer führen zu stark variierenden Ausführungszeiten von Simulationsläufen. Der Grund für die Ausreißer in der Schreibzeit ist wahrscheinlich auf das Datenbankmanagementsystem von Neo4j zurückzuführen. Hierbei kann möglicherweise die Anwendung mehrerer paralleler Threads zu Verzögerungen bei den Schreibvorgängen führen. Die Performanz des Simulators kann weiterhin durch den zeitlichen Verlauf der Hardware-Ressourcen Arbeitsspeicher und Prozessor bewertet werden, der Abbildung 5.6 zu entnehmen ist.

Die Messungen für Abbildung 5.5 wurden mit der Leistungsüberwachung von Windows vorgenommen (vgl. Abschnitt 5.1). Der Abstand zwischen zwei Messpunkten beträgt 15 Sekunden. Das Experiment wurde an Messpunkt vier gestartet und endete an Messpunkt 34. In Abbildung 5.5a ist der zeitliche Verlauf des freien Arbeitsspeichers dargestellt. Zu

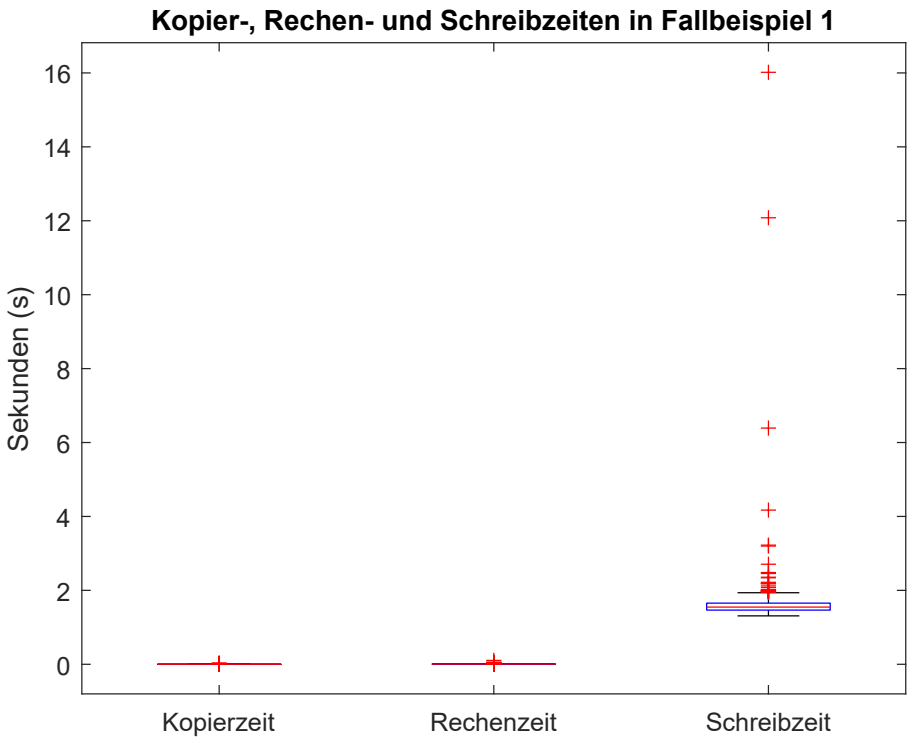
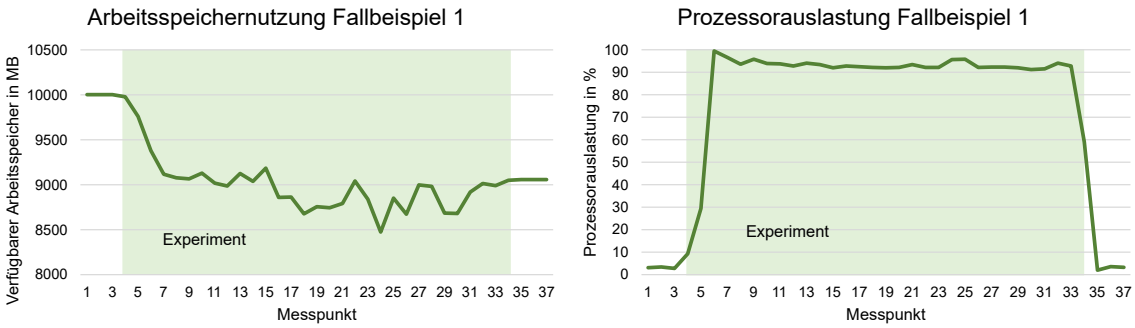


Abbildung 5.4: Box-Plot-Diagramme der gemessenen Kopier-, Rechen- und Schreibzeiten der Simulationsläufe in Fallbeispiel 1



(a) (b)

Abbildung 5.5: Zeitlicher Verlauf der (a) Arbeitsspeichernutzung und (b) Prozessorauslastung in Fallbeispiel 1

Beginn des Experiments kann festgestellt werden, dass der freie Arbeitsspeicher reduziert wird und somit durch LogFarm beansprucht wird. In der zweiten Hälfte des Experiments scheint die Arbeitsspeicherauslastung im Durchschnitt annähernd konstant zu verlaufen, jedoch sind einige Fluktuationen zu erkennen. Insgesamt beträgt die Belastung des Arbeitsspeichers während des Experiments im Mittel etwa 1.000 MB. Es ist zu beobachten, dass nach dem Experiment der verfügbare Arbeitsspeicher nicht auf das Vorniveau zurückkehrt. Dies ist damit zu begründen, dass der Prototyp von LogFarm die Ressourcen nach dem Experiment nicht wieder freigibt. Die Prozessorauslastung ist in Abbildung 5.5b dargestellt. Während des Experiments kann ein steiler Anstieg der Auslastung beobachtet werden, die dann während der gesamten Dauer des Experiments auf einem hohen

Niveau verbleibt. Daher kann gesagt werden, dass LogFarm die zur Verfügung stehenden Hardware-Ressourcen in hohem Maße ausnutzen kann. Kurz vor Ende des Experiments, an Messpunkt 34, fällt die Prozessorauslastung ab. Eine mögliche Erklärung dafür ist, dass zu diesem Zeitpunkt bereits ein Thread inaktiv geworden ist und kein neuer Simulationslauf zugewiesen wurde.

5.3 Fallbeispiel 2

Das Szenario des zweiten Fallbeispiels ist an eine Simulationsstudie von Rabe et al. (2018) angelehnt. Mit diesem Fallbeispiel soll untersucht werden, inwiefern der Einsatz von LogFarm für die Untersuchung realer Szenarien verwendet werden kann.

5.3.1 Szenario

Die Simulationsstudie von Rabe et al. (2018) untersucht den Einsatz von Konsolidierungszentren in der Metropolregion Athen in Griechenland. Die Untersuchung wurde anlässlich einer Beurteilung möglicher Konzepte horizontaler Kollaboration von Anbietern durchgeführt, die Konsumgüter in urbanen Gebiete Athens liefern. Das Ziel der Kooperation sollte sowohl eine gesamtheitliche als auch für jeden Kooperationspartner individuelle Verbesserung der ökologischen Aspekte, wie die Verringerung der Feinstaubbelastung in dicht besiedelten Gebieten, und ökonomischer Aspekte, wie eine verbesserte Auslastung und Tourenplanung von Auslieferungsfahrzeugen, herbeiführen.

In der Simulationsstudie von Rabe et al. (2018) wurden drei Szenarien simuliert. Das Fallbeispiel der vorliegenden Arbeit lehnt sich an das Szenario an, das lediglich ein Konsolidierungszentrum aufweist und von nun an als Referenzszenario bezeichnet wird. Die Topologie des Logistiknetzwerks kann als zweistufiges Netzwerk beschrieben werden, bei dem das Konsolidierungszentrum die zentrale Annahmestelle aller Artikel ist, die von den Kooperationspartnern geschickt werden. Vom Konsolidierungszentrum aus werden die Waren an die Kunden ausgeliefert. Bei der Beladung der Auslieferungsfahrzeuge können Artikel verschiedener Kooperationsteilnehmer verwendet werden, um eine optimale Tourenplanung zu gewährleisten. Die Anzahl der verfügbaren Fahrzeuge wird im Referenzszenario nicht beschränkt.

Es werden fünf Kooperationspartner betrachtet, die die Rolle der Zulieferer im Logistiknetzwerk übernehmen. Die geographischen Koordinaten der Standorte der Zulieferer und auch des Konsolidierungszentrums können dem Referenzszenario entnommen werden. Anders als im Referenzszenario, bei dem keine Unterscheidung von Artikeln unternommen, sondern lediglich ein generischer Artikel verwendet wurde, werden im ersten Fallbeispiel fünf verschiedene Artikel betrachtet. Jeder Zulieferer ist für die Bereitstellung genau eines Artikels zuständig. Es wird angenommen, dass die Zulieferer eine vollständige Verfügbarkeit der jeweiligen Artikel sicherstellen können. Die Transporte von den Zulieferern zum Konsolidierungszentrum werden von LKW durchgeführt. Die Kapazität dieser Fahrzeuge beträgt 1650 Artikel. Für die Auslieferung an den Kunden werden Kleintransporter mit einer Kapazität von 750 Artikeln verwendet. Im Referenzszenario werden 8.537 Kunden in der gesamten Metropolregion Athen betrachtet. Kunden sind im Referenzszenario Einzelhandelsgeschäfte. Die exakten geographischen Koordinaten der Kunden können nicht aus dem Referenzszenario ermittelt, sondern lediglich auf das Stadtgebiet Athens beschränkt

werden. Auch genaue Daten über die Höhe des Bedarfs der Kunden können dem Referenzszenario nicht entnommen werden.

Die folgende Fragestellung weicht von der Arbeit von Rabe et al. (2018) ab. In dem zuvor beschriebenen Szenario soll die Lagerhaltung des Konsolidierungszentrums untersucht werden. Es steht fest, dass eine (s,q)-Politik für alle Artikel verwendet werden soll. Jedoch gibt es für die Parameter Meldebestand und Bestellmenge der Lagerhaltungspolitik jeweils zwei Optionen. Zum einen soll untersucht werden, wie sich Meldebestände der Niveaus 2.000 und 3.000 auf das Modell auswirken. Zum anderen soll auch die Bestellmenge hinsichtlich der beiden Niveaus 750 und 1.650 untersucht werden. Die Niveaus der Bestellmenge sind so gewählt, dass entweder genau ein LKW oder ein Kleintransporter verwendet werden kann, um die Bestellmenge zu transportieren. Zusätzlich soll eine Untersuchung stattfinden, die eine erhöhte Nachfrage der Kunden von 20% berücksichtigt. Die Bestellmengen der Kunden sind starken Schwankungen unterworfen.

5.3.2 Modellierung und Durchführung

Für die Modellierung des Szenarios werden fünf Artikel im Artikelverzeichnis sowie die beiden Fahrzeugklassen LKW und Kleintransporter im Fahrzeugverzeichnis angelegt. Die Kapazitäten werden entsprechend dem Referenzszenario auf 1650 Artikel für LKW und auf 750 Artikel für Kleintransporter festgelegt.

Die fünf Standorte der Zulieferer werden mit jeweils einem Zulieferer-Baustein modelliert. Die Geo-Koordinaten werden entsprechend dem Referenzmodell festgelegt. Als Komponenten werden die Produktions-, Lagerungs- und Distributionskomponenten verwendet. In der Produktionskomponente wird jeweils nur der Artikel bereitgestellt, der exklusiv dem Zulieferer zuzuordnen ist. Die Produktionsrate wird auf einen unrealistisch hohen Wert gesetzt, um eine unendliche Verfügbarkeit zu realisieren. Aus demselben Grund die Kapazität der Lagerungskomponenten auf unrealistisch hohe Werte gesetzt. In den Distributionskomponenten werden jeweils eine statische Transportrelation zum Konsolidierungszentrum spezifiziert und eine unrealistisch große Fahrzeugflotte zur Verfügung gestellt, um keine Engpässe bei der Auslieferung zuzulassen. Das Referenzszenario liefert keine Angaben über die tatsächlichen Betriebszeiten der Zulieferer. Für die Modellierung wurde daher eine für Industrieunternehmen in Griechenland übliche Fünftagewoche (Montags bis Freitags) mit einer täglichen Arbeitszeit von acht Stunden angenommen.

Das Konsolidierungszentrum wird mit einem Warenhaus-Baustein modelliert. Die Komponenten des Bausteins sind die Lagerungs- und Distributionskomponente. Die Lagerhaltungspolitik der Lagerungskomponente wird für jeden Artikel auf die (s,q)-Politik gesetzt. Aufgrund der in der Szenariobeschreibung geforderten Untersuchung der Parameter der (s,q)-Politik, werden im Modell der Wert des Meldebestands und der Bestellmenge durch den Experimentierplan bestimmt. Die Niveaus werden entsprechend der Szenariobeschreibung festgelegt. Da im Szenario ein stets ausreichend großes Lager gefordert wird, wird für die Kapazität des Lagers ein unrealistisch hoher Wert angenommen. In der Distributionskomponente werden alle Kunden als dynamisch verbunden spezifiziert. Um den Zustand der unbegrenzten Größte der Flotte der Auslieferungsfahrzeuge abzubilden, wird wiederum ein unrealistisch hoher Wert von verfügbaren Fahrzeugen hinterlegt. Die Arbeitszeit unterliegt denselben Annahmen wie zuvor bei den Zulieferern.

Zur Modellierung der Kunden wird ein Kundengruppenbaustein verwendet. Mithilfe dieses Bausteins ist es möglich, Standorte von Kunden gleich verteilt in einem rechteckigen

Gebiet zu generieren. Mit dieser Methode ist es nicht möglich, die reale Verteilung der Kundenstandorte, wie sie im Referenzszenario auf Basis von Postleitzahlen gezeigt wird, abzubilden. Dies gilt insbesondere für Gebiete, die relativ weit vom Stadtzentrum entfernt sind. Für das Modell wird daher die Annahme getroffen, dass lediglich das Gebiet nah des Stadtzentrums betrachtet wird und somit auch nur ein Teil der im Referenzszenario genannten Anzahl von Kunden. Die Anzahl der zu generierenden Kunden durch den Kundengruppenbaustein wird auf 100 Kunden festgelegt. Die Begrenzungspunkte des Gebiets, in denen Kunden generiert werden können, wurden über einen Online-Kartendienst Koordinaten ermittelt. Die Begrenzungspunkte beschreiben in etwa ein Rechteck, das größtenteils das Stadtzentrum Athens abbildet. Die Arbeitszeit der Kunden, die wie im Szenario beschrieben Einzelhandelsgeschäfte sind, wird mit einer Sechstagewoche (Montag bis Samstag) und einer täglichen Betriebszeit von zwölf Stunden angenommen. Die Kunden der Kundengruppe weisen jeweils eine Nachfragekomponente auf. Die Parametrisierung der Nachfragekomponenten ist identisch. Die Kunden präferieren bei der Generierung der Nachfrage keinen der fünf Artikel, die Auswahl ist somit gleich verteilt. Im Szenario wurde beschrieben, dass die Nachfragekomponenten zwei unterschiedliche Niveaus abbilden sollen. Daher werden die Parameter der Nachfragekomponenten variabel mithilfe des Experimentierplans beschrieben. Dazu gehören neben den beiden Parametern zur Beschreibung der Häufigkeit einer Auslösung von Nachfrage als auch die beiden Parameter zur Beschreibung der nachgefragten Quantität. Da im Referenzszenario keine exakten Daten zur Höhe der Nachfragen spezifiziert wurden, müssen passende Parameter der Nachfragekomponente geschätzt werden. Die Schätzung des unteren Werts für den Erwartungswert der Quantität basiert auf den Ergebnisdaten des Referenzszenarios aus Rabe et al. (2018). Dort wird die durchschnittliche Anzahl von Kunden pro Route mit 16,7 angegeben. Bei einer Kapazität der Auslieferungsfahrzeuge von 750 Artikeln würden durchschnittlich bei jedem Kunden 45 Artikel abgeliefert. Dieser Wert wird im Experimentierplan als unterer Wert verwendet. Der obere Wert soll laut Szenario 20% über dem unteren Wert liegen. Das bedeutet, dass ein oberer Wert von 54 verwendet wird. Für den unteren und oberen Wert der Varianz der Quantität findet sich im Szenario lediglich der Hinweis, dass starke Schwankungen vorhanden sind. Somit unterliegt der Wahl die Werte der subjektiven Einschätzung des Modellierers. Für das vorliegende Modell wird als unterer Wert der Varianz der Quantität im Experimentierplan der Wert 25 und als oberer Wert 40 verwendet. Hinsichtlich der zeitlichen Abstände zwischen Nachfragen finden sich im Referenzszenario keine Hinweise. Es wird angenommen, dass das untere Niveau des Erwartungswerts drei beträgt und das obere Niveau vier. Die Werte für die Varianz werden auf drei und vier festgelegt. Eine Visualisierung der Standorte des Szenarios auf einer Landkarte sowie das Gebiet, in dem der Kundengruppenbaustein die Kunden generieren soll, sind in Abbildung 5.6 dargestellt.

Abbildung 5.6 zeigt eine Landkarte der Metropolregion Athen. Neben den fünf Zulieferern und dem Konsolidierungszentrum ist eine rote Fläche zu sehen. Diese Fläche entspricht dem Gebiet des Kundengruppenbausteins, in dem die Kunden zufällig generiert werden sollen. Das Modell wurde im Editor von LogFarm, wie in diesem Abschnitt beschrieben, erstellt. Eine Bildschirmaufnahme dessen ist in Abbildung 5.7 abgebildet.

Abbildung 5.7 zeigt die sieben verwendeten Bausteine. Durch die Verwendung des Kundengruppenbausteins mussten nicht 100 Kundenbausteine verwendet werden. Die Generierung eines ausführbaren Modells durch den Simulator wurde im Anschluss durchgeführt. In Tabelle 5.4 sind einige Kenngrößen der Modelldatenbank aufgeführt.

Tabelle 5.4 zeigt, dass die Modelldatenbank insgesamt 1.564 Knoten und 3.897 Kanten umfasst. Insgesamt wurden 5.211 Eigenschaften modelliert. In Fallbeispiel 1 war die Anzahl

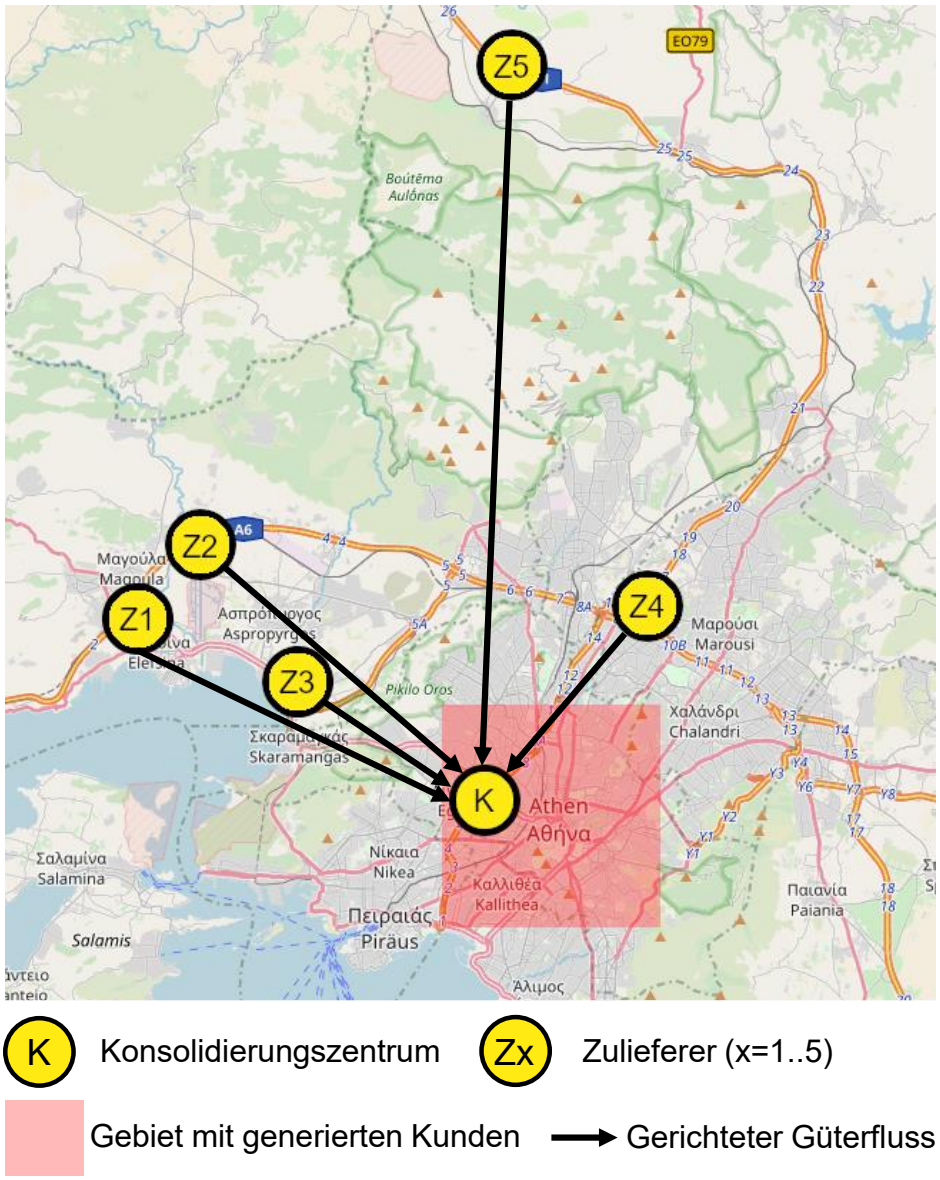


Abbildung 5.6: Standorte Fallbeispiel 2 auf einer Landkarte

Tabelle 5.4: Messgrößen der Modelldatenbank in Fallbeispiel 2

Messgröße	Wert
Knoten in Modelldatenbank	1.564
Kanten in Modelldatenbank	3.897
Eigenschaften in Modelldatenbank	5.211
Größe der Modelldatenbank	2,55 MB

der Eigenschaften etwas mehr als halb so groß wie die Summe der Knoten und Kanten (vgl. Tabelle 5.2). In Fallbeispiel 2 ist die Anzahl der Eigenschaften etwa gleich groß wie diese Summe. Eine Erklärung dafür ist, dass im zweiten Fallbeispiel mit deutlich mehr stochastischen Prozessen gearbeitet wird, deren Repräsentation auf Datenebene verhältnismäßig

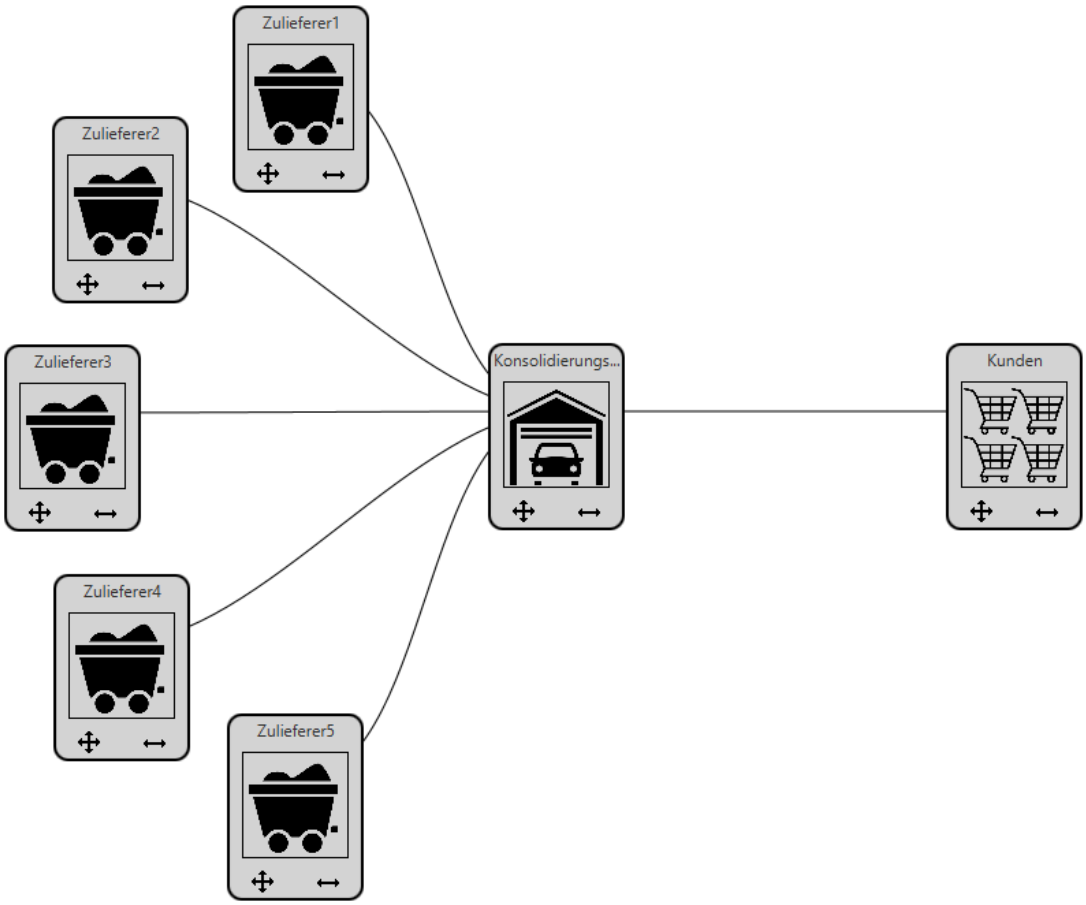


Abbildung 5.7: Bildschirmaufnahme der Modellfläche mit Modell von Fallbeispiel 2

viele Eigenschaften benötigt (vgl. Abbildung 4.4). Die Menge der Daten in Fallbeispiel 2 beträgt 2,55 MB. Eine Visualisierung des generierten ausführbaren Modells in der Modelldatenbank ist in Abbildung 5.8 dargestellt.

Der visualisierte Graph in Abbildung 5.8 zeigt die starke Vernetzung der Daten des Simulationsmodells. Auf der rechten Seite der Abbildung ist ein Cluster von grünen Knoten zu sehen, dass sich um einen hellbraunen Knoten gebildet hat. Die grünen Knoten modellieren Verteilungen und der hellbraune Knoten ist der Experimentplan-Koppler. Da die Nachfrage der 100 Kunden stochastisch über Verteilungen erfolgt und diese als Faktoren behandelt werden sollen, weisen sie alle eine Kante zum Experimentplan-Koppler auf, was zu der beschriebenen Clusterbildung führte.

Die Generierung des Experimentierplans wurde mithilfe der Software MATLAB des Unternehmens MathWorks vorgenommen (Schweizer 2022). MATLAB verwendet eine eigene Skriptsprache, über die der Anwender Programme schreiben kann, um die Funktionen der Software zu nutzen. Der Anwendungsbereich von MATLAB liegt in der Lösung mathematischer Probleme und der Darstellung der Ergebnisse solcher Problemlösungen. Eine weitere Anwendung ist die Generierung von Experimentierplänen. Für das vorliegende Modell sollen insgesamt sechs Faktoren in den Experimentierplan integriert werden. Zwei Faktoren werden benötigt, um die Lagerhaltungspolitik des Konsolidierungszentrums dynamisch zu untersuchen. Für die dynamische Modellierung der Nachfrage der Kunden werden durch die Verwendung des Kundengruppenbausteins lediglich vier Faktoren benötigt. Somit wird

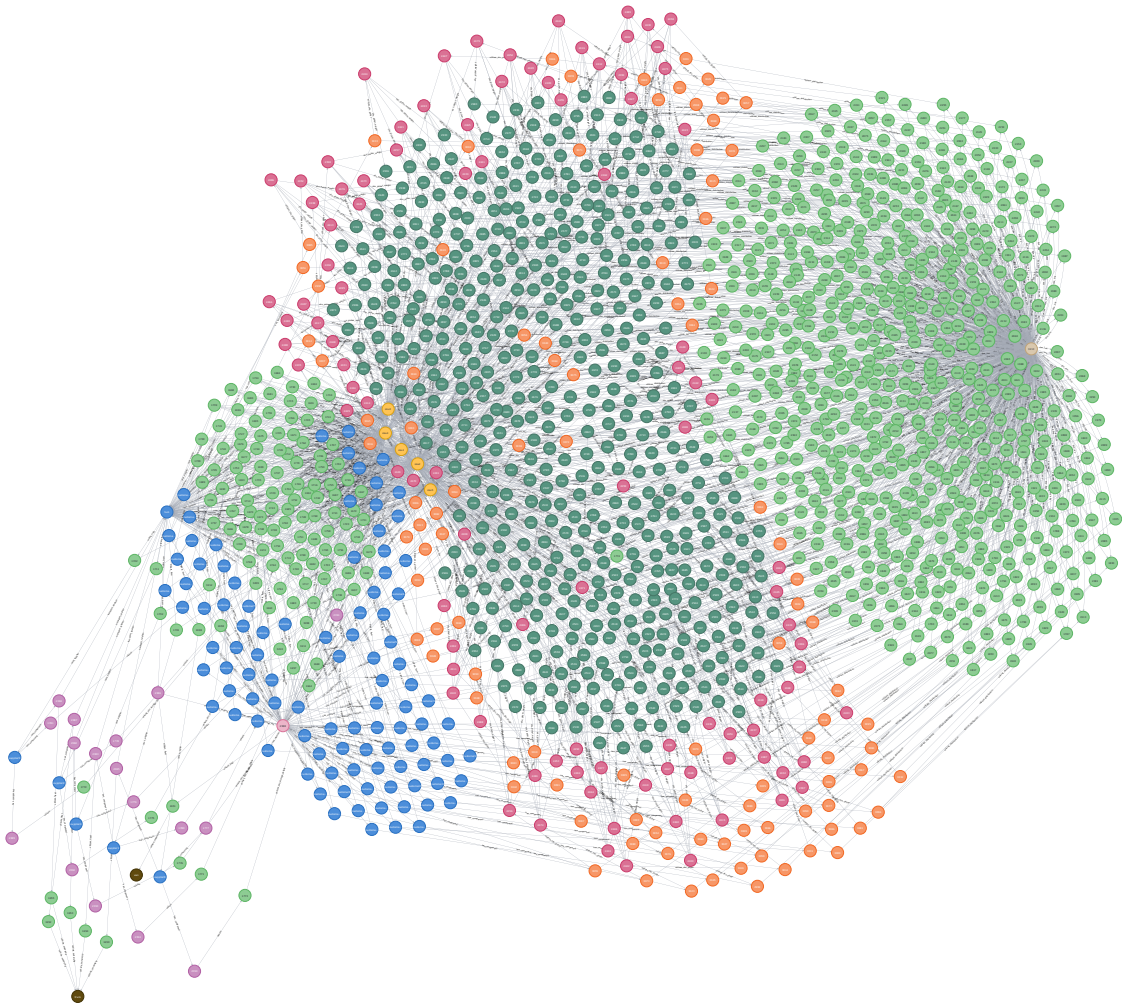


Abbildung 5.8: Visualisierung der Modelldatenbank von Fallbeispiel 2

die Nachfrage aller Kunden durch diese vier Faktoren gesteuert. Es werden zwei Faktoren für die Verteilung zur Steuerung der Frequenz des Auftretens der Nachfrage und zwei weitere Faktoren für die Verteilung zu Beschreibung der Nachfragemenge benötigt. Aufgrund der vergleichsweise niedrigen Anzahl von Faktoren bietet sich ein vollfaktorieller Experimentierplan an (vgl. Abschnitt 2.1.2). Insgesamt umfasst der Experimentierplan 64 Simulationsläufe, um alle Kombinationen von Wertestufen zu untersuchen. In Algorithmus A.1 ist das verwendete MATLAB-Skript zur Erstellung des Experimentierplans dargelegt.

Die Grenzen des Simulationszeitraums wurden, entsprechend der Simulationsstudie von Rabe et al. (2018), auf die Werte 1.7.2014 und 1.7.2015 gelegt. Der Zeitraum entspricht 365 Tagen. Zusätzlich wurde eine Warmlaufphase von sieben Tagen spezifiziert, sodass die Simulation bereits am 24.6.2014 beginnt. Um eine gründliche Untersuchung der Designpunkte zu gewährleisten, wird die Anzahl der Replikationen jedes Simulationslaufs auf acht gesetzt. Insgesamt ergibt sich somit die Anzahl auszuführender Simulationsläufe zu 512. Das Experiment wurde auf demselben Computer wie Szenario 1 ausgeführt (siehe Tabelle 5.1). Auch die Anzahl der parallel verfügbaren Threads wurde nicht verändert und beträgt zwei. Der verwendete Startwert des Experiments ist null. In Tabelle A.3 sind die verwendeten Parameter für die Bausteine und ihrer Komponenten dargestellt.

5.3.3 Ergebnisse

In einem beispielhaften Data-Farming-Vorgang wurden die in Abschnitt 5.1 beschrieben Größen erhoben. Die erhobenen Größen können Tabelle 5.5 entnommen werden.

Tabelle 5.5: Ergebnisse des Experiments von Fallbeispiel 2

Messgröße	Wert
Knoten in Ergebnisdatenbank	5.494.211
Kanten in Ergebnisdatenbank	31.876.925
Eigenschaften in Ergebnisdatenbank	6.030.081
Größe der Ergebnisdatenbank	1.449 MB
Benötigte Zeit für das Experiment	1.630,55s
Vorbereitungszeit	0,987s
Durchschnittlich benötigte Zeit für einen Simulationslauf	6,357s
Durchschnittliche Kopierzeit	0,007s
Durchschnittliche Rechenzeit	0,63s
Durchschnittliche Schreibzeit	5,72s

Die generierte Datenmenge umfasste 5.494.211 Knoten, 31.876.925 Kanten und insgesamt 6.030.081 Eigenschaften. Die niedrige Anzahl der Eigenschaften im Vergleich zu der Summe der Knoten und Kanten resultiert aus der Modellierung auf Datenebene der generierten Daten, die hauptsächlich über Vernetzung Zusammenhänge darstellt und auf Eigenschaften verzichtet (vgl. Abbildung 4.14). Die Größe der Datenmenge betrug 1.449 MB und wurde in 1.630,55 Sekunden generiert. Damit ergibt sich mit 0,89 MB pro Sekunde eine niedrigere Rate der Generierung von Daten im Vergleich zu Fallbeispiel 1.

Die Vorbereitungszeit des Experiments ist, wie bereits in Fallbeispiel 1, vernachlässigbar gering. Die durchschnittlich benötigte Zeit für einen Simulationslauf betrug 6,357 Sekunden. Bestandteile dieser Zeit sind die Kopierzeit, die 0,007 Sekunden betrug, die Simulationszeit, die 0,63 Sekunden betrug, und die Schreibzeit mit einem Wert von 5,72s. Im Verhältnis zur Gesamtdauer des Simulationslaufs ist die Kopierzeit ein vernachlässigbarer Faktor. Anders als in Fallbeispiel 1 ist in Fallbeispiel 2 die Simulationszeit nicht zu vernachlässigen. Eine Visualisierung der Verhältnisse der Zeiten ist im Kreisdiagramm der Abbildung 5.9 dargestellt.

Das Kreisdiagramm in Abbildung 5.9 zeigt, dass der Anteil der Rechenzeit im Vergleich zur Schreibzeit spürbar ist. Insgesamt macht die Rechenzeit etwa 10% der durchschnittlichen Dauer eines Simulationslaufs aus. Der im Vergleich zu Fallbeispiel hohe Anteil der Rechenzeit lässt sich auf die Verwendung der dynamischen Tourenplanung mit dem ICWA zurückführen. Dennoch ist der Engpass des Data-Farming-Vorgangs wiederum eindeutig als die Schreibzeit in die Ergebnisdatenbank zu identifizieren. Würde eine höhere Zahl von Kunden verwendet oder die Frequenz der Nachfrage erhöht und somit die Komplexität des CVRP gesteigert werden, könnte der Anteil der Rechenzeit vergrößert werden oder sogar die Schreibzeit übersteigen, sodass eine höhere Anzahl von Threads sinnvoll sein könnte. Um die Konstanz der Kopier-, Rechen- und Schreibzeit zu überprüfen, sind in Abbildung 5.10 entsprechende Box-Plot-Diagramme abgebildet.

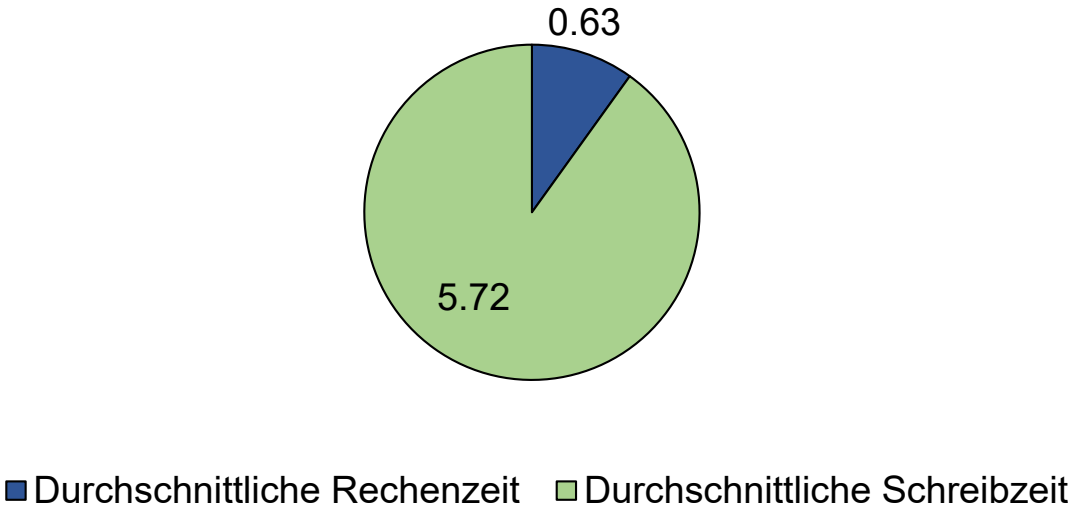


Abbildung 5.9: Kreisdiagramm der gemessenen Rechen- und Schreibzeiten der Simulationsläufe in Fallbeispiel 2

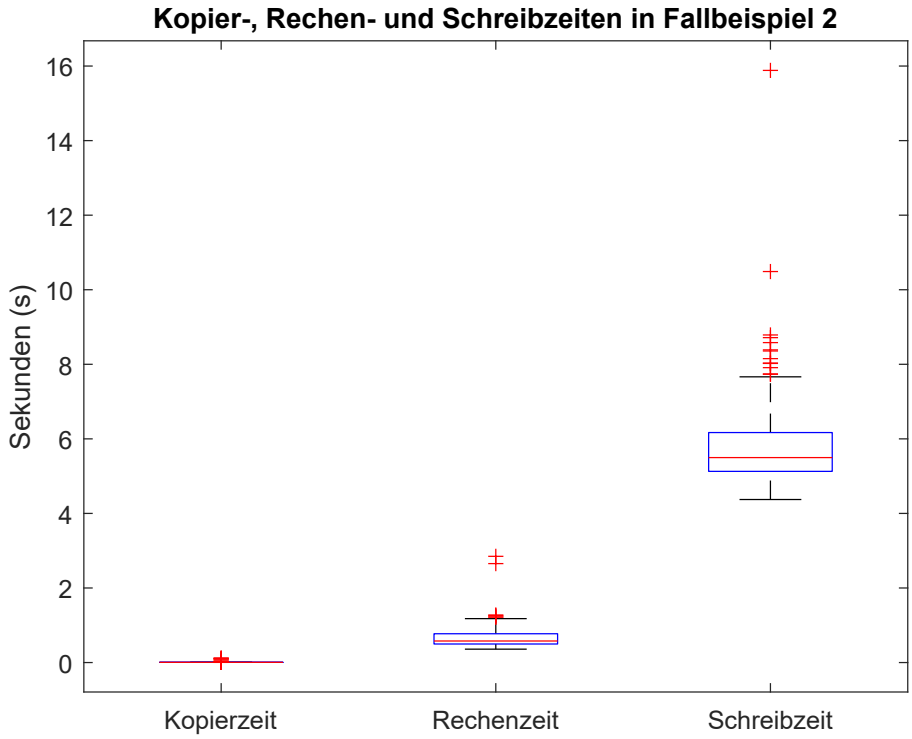
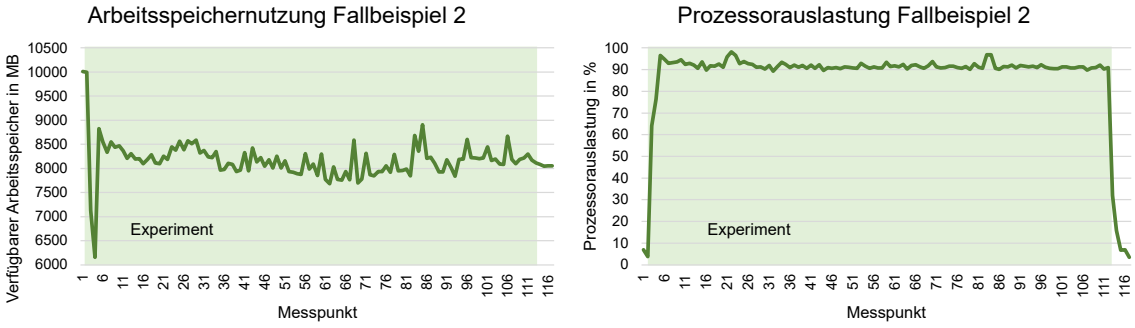


Abbildung 5.10: Box-Plot-Diagramme der gemessenen Kopier-, Rechen- und Schreibzeiten der Simulationsläufe in Fallbeispiel 2

Abbildung 5.10 zeigt, dass die Kopierzeit sehr konstant ist und keine Ausreißer aufweist. Die Rechenzeit in Fallbeispiel 2 zeigt ein moderates Maß an Variation in den Werten. Auffällig sind die zwei erkennbaren Ausreißer, bei denen die Rechenzeit fast doppelt so viel Zeit benötigte als im Mittel. Eine Erklärung ist, dass in diesen Simulationsläufen eine überdurchschnittlich hohe Nachfrage generiert wurde, und somit die Tourenplanung aufwändiger wurde. Die Schreibzeit in die Ergebnisdatenbank weist auch eine moderate Inkonsistenz auf. Zudem existiert ein Ausreißer, der weit entfernt vom Median ist. Dieser

hohe Wert lässt sich dadurch erklären, dass zu Beginn des Experiments zwei Threads gestartet werden und der Thread, der als zweites die Simulation abschließt, erst auf den Schreibvorgang des ersten Threads warten muss.

Zuletzt wird die Auslastung der Hardware betrachtet. Zeitliche Verläufe der Arbeitsspeichernutzung und der Prozessorauslastung sind in Abbildung 5.11 dargestellt.



(a) (b)

Abbildung 5.11: Zeitlicher Verlauf der (a) Arbeitsspeichernutzung und (b) Prozessorauslastung in Fallbeispiel 2

Der Verlauf in Abbildung 5.11a weist zwar Fluktuation in der Nutzung des Arbeitsspeichers auf, der Durchschnitt ist jedoch konstant. Durchschnittlich beträgt die zusätzliche Arbeitsspeichernutzung durch LogFarm im zweiten Fallbeispiel 1.844 MB. Zudem ist zu Beginn des Experiments eine kurzzeitig starke Belastung von fast 4.000 MB zu erkennen. Dieser Effekt scheint mit einem Vorgang zur Vorbereitung des Experiments, wie dem Vorbereiten der Datenbankverbindung oder dem Laden des Modells aus der Modelldatenbank, zusammen zu hängen. Die Prozessorauslastung beträgt durchschnittlich 90% und bewegt sich somit auf einem sehr hohem Level. Aufgrund der guten Prozessorauslastung auf konstant hohem Niveau kann davon ausgegangen werden, dass die verfügbaren Hardwareressourcen gut ausgenutzt werden.

5.4 Diskussion und Fazit

Die Entwicklung des vorgestellten Simulators LogFarm zielt darauf ab, eine Lücke in der Verarbeitung von Daten in Form von Graphen im Bereich der Simulation in der Logistik zu schließen. Logistische Systeme wurden in dieser Arbeit als komplexe Systeme mit vielfältigen Problemstellungen beschrieben. Hunker et al. (2020) haben aufgezeigt, dass die Verwendung von Graphdatenbanken zur Unterstützung der Lösung logistischer Problemstellungen großes Potential aufweist, jedoch bisher wenig Anwendung findet. Eine Möglichkeit zur Unterstützung des Entscheidungsprozesses in der Logistik bietet die Wissensentdeckung in Datenbanken an, die hauptsächlich auf Data Mining basiert. Hunker et al. (2020) zeigten, dass Graphdatenbanken, insbesondere für Data Mining in der Logistik, eine aussichtsreiche Basis darstellen. In einer anderen Arbeit stellten Hunker et al. (2021) einen Farming-for-Mining-Ansatz vor, mit dem eine Datenbasis für das Data Mining mithilfe von Data Farming generiert wird. Unter Anbetracht der Ergebnisse von Hunker et al. (2020) kann gefolgert werden, dass die generierten Daten des Data Farmings vorzugsweise in einer Graphdatenbank vorliegen sollten. Da das Data Farming sich eines Simulator bedient, um die Generierung der Daten zu vollziehen, sollte der Simulator die Möglichkeit besitzen, Daten in Form eines Graphen direkt, ohne weitere Transformations-

prozesse, zu generieren. Nach Wissen des Autors der vorliegenden Arbeit existiert solch ein Simulator jedoch nicht. Daher wurde mit LogFarm eine Anwendung entwickelt, um diese Lücke zu schließen. Im Gegensatz zu existierenden Lösungen, die hauptsächlich auf das relationale Datenmodell ausgerichtet sind und daher zusätzliche Transformationsprozesse für die Umwandlung in das Graphenmodell nötig sind, generiert LogFarm direkt solche Daten und verwendet auch intern eine graphbasierte Arbeitsweise, sowie Simulationsmodelle im Graphenmodell. Somit stellt LogFarm eine graphbasierte Alternative zu existierenden Simulatoren, wie PlantSimulation, AnyLogistix oder Arena, (vgl. Abschnitt 2.1.3) dar.

Um ein fundiertes Fazit ziehen zu können, inwiefern der entwickelte Simulator LogFarm die Zielsetzung dieser Arbeit erfüllt, erfolgt zunächst eine Diskussion hinsichtlich der Erfüllung der Anforderungen aus Abschnitt 4.1. Zur Beurteilung werden die in den Abschnitt 4.2 bis Abschnitt 4.4 beschriebenen Informationen über LogFarm und Erkenntnisse sowie Ergebnisse der beiden Fallbeispiele in Abschnitt 5.2 und Abschnitt 5.3 herangezogen. Der Ablauf der Diskussion der Anforderungen erfolgt entsprechend der Nummerierung der Anforderungen.

Damit Anforderung I als erfüllt gilt, muss der Simulator Daten eines Logistiknetzwerks in Form eines Graphen mittels Data Farming generieren können. Der Simulator muss somit in der Lage sein, ein Logistiknetzwerk als Simulationsgegenstand simulieren zu können. Da der Funktionsumfang des Simulators, der durch das Konzeptmodell bestimmt wird (vgl. Abschnitt 4.2), aus Logistiknetzwerken abgeleitet wurde, sind Logistiknetzwerke als Simulationsgegenstand folgerichtig. Jedoch müssen aufgrund des limitierten Funktionsumfangs von LogFarm Einschränkungen gemacht werden. Für Logistiknetzwerke, die mit den zur Verfügung stehenden Funktionen nicht beschrieben werden können, stellt LogFarm in der aktuellen Fassung keine geeignete Lösung dar. Aufgrund der fehlenden Möglichkeit zur Erstellung eigener Komponenten im Rahmen des Simulators können die Funktionen nicht durch den Nutzer ergänzt werden. Somit kann gesagt werden, dass LogFarm zwar für einige Logistiknetzwerke und dazugehörige Fragestellungen eingesetzt werden kann, jedoch nicht für alle. Durch die exemplarische Simulation eines imaginären und eines realen Logistiknetzwerks in den Fallbeispielen wurde die Eignung LogFarms zur Simulation von Logistiknetzwerken unter Beweis gestellt. Weiterhin wurde gezeigt, dass LogFarm in der Lage ist, einen Data-Farming-Vorgang durchführen zu können. Eingeschränkt wird die Verwendung von LogFarm als Data-Farming-Werkzeug durch die bisher geringe Heterogenität der generierten Daten, da lediglich Daten über Bestellungen und Transporte generiert werden. Zudem fehlt, der Definition von Horne und Seichter (2014) aus Abschnitt 2.3.3 folgend, die Analyse- und Visualisierungsmöglichkeit der Ergebnisse. Außerdem fehlt die Fähigkeit, einen Experimentierplan innerhalb von LogFarm zu generieren. Für diesen Schritt müssen andere Werkzeuge verwendet werden. Uneingeschränkt lässt sich die Forderung nach generierten Daten in Form eines Graphen bestätigen, da zur Speicherung der Daten ein Eigenschaftsgraph verwendet wird. Da die Kernforderungen umgesetzt wurden, wird Anforderung I als erfüllt angesehen.

Anforderung II fordert eine graphbasierte Arbeitsweise des Simulators. Zur Erfüllung dieser Aufgabe wurden die zu verwendenden Simulationsmodelle in Form von Eigenschaftsgraphen modelliert (vgl. Abschnitt 4.3.2). Diese Struktur wurde mithilfe des OGM direkt in die interne Funktionsweise des Simulatorkerns übertragen, sodass von einer graphbasierten Anwendung gesprochen werden kann. Zudem wurde in Anforderung II beschrieben, dass auf Graphen basierende Algorithmen eingesetzt werden sollen. Hierzu wurde beispielhaft der ICWA zur Tourenplanung verwendet. Zusammenfassend wird Anforderung II als erfüllt angesehen.

In Anforderung III wird von LogFarm eine grafische Benutzeroberfläche gefordert, die auch einen Editor zur interaktiven Modellerstellung umfasst. In Abschnitt 4.3.7 wurde die grafische Benutzeroberfläche bereits detailliert vorgestellt. Zudem wurde durch die Fallbeispiele die Funktionsfähigkeit der grafischen Benutzeroberfläche bestätigt. Dies umfasst zum einen die interaktive Modellerstellung mit einem Editor, der über Drag-and-Drop Bausteine platzieren und verbinden kann, und zum anderen die Spezifikation und Ausführung von Experimenten. Zwar ist es mit dem Editor zur Modellerstellung möglich, alle verfügbaren Komponenten zu verwenden und zu parametrisieren, jedoch fehlen in der vorgestellten Fassung einige Funktionen, wie beispielsweise das Löschen bereits platzierter Bausteine und Verbindungen. Zusammenfassend kann gesagt werden, dass die grafische Benutzeroberfläche die Kernfunktionalität anbietet und als erfüllt angesehen wird, jedoch einige Komfortfunktionen fehlen.

Um Anforderung IV zu erfüllen, muss der Simulator mit der kontinuierlichen Simulation arbeiten und agentenbasierte Modelle simulieren können. Wie in Abschnitt 4.3.4 beschrieben, arbeitet LogFarm mit Modellen, in denen Agenten modelliert wurden und simuliert diese mit der kontinuierlichen Simulation. Die handelnden Agenten in den agentenbasierten Modellen, die über LogFarms Editor erstellt werden können, sind die Akteure der zu modellierenden Logistiknetzwerke. Die Modellierung der Agenten erfolgt mithilfe der Bausteine aus Abschnitt 4.3.1. Da der Zeitfortschritt des Simulators, wie in Abschnitt 4.3.4 beschrieben, immer genau einen Tag beträgt und nicht von anderen Ereignissen abhängt, wird von einer zeitgesteuerten, kontinuierlichen Simulation gesprochen. Anforderung IV wurde somit erfüllt.

Zur Erfüllung der Anforderung V muss der Umgang mit großen Datenbeständen möglich sein. Die großen Datenbestände sind hauptsächlich in der Ergebnisdatenbank zu erwarten. Da die Ergebnisdatenbank mithilfe von Neo4j, also einer Graphdatenbank (vgl. Abschnitt 2.3.1), umgesetzt wurde, kann zunächst von einer Eignung für große Datenbestände ausgegangen werden. In Fallbeispiel 2, welches im Vergleich zu Fallbeispiel 1 einen größeren generierten Datenbestand in der Ergebnisdatenbank aufweist, wurde für das vorgestellte Ergebnis eine Datenmenge von etwa 1,5 GB generiert. Somit kann zumindest sichergestellt werden, dass es bei vergleichbaren Datenmengen zu keinen Fehlern kommt. Da keine Leistungsminimierung im Verlauf es Experiments festgestellt wurde (vgl. Abschnitt 5.3.3), kann davon ausgegangen werden, dass auch größere Datenmengen verarbeitet werden können. Anforderung V wird somit, unter Vorbehalt, als erfüllt angesehen.

Anforderung VI besagt, dass der Simulator leistungsstarke Berechnungen bzw. Simulationen ermöglichen muss. Die Effizienz des Programmcodes von LogFarm kann nicht quantifiziert und somit nicht beurteilt werden. Jedoch haben die beiden Fallbeispiele gezeigt, dass der Engpass bei der Datengenerierung nicht an der Simulation in LogFarm zu suchen ist, sondern bei dem Datenbanksystem. Aufgrund der implementierten Funktion zur parallelen Simulation können die Ressourcen eines Mehrkern-Computers optimal ausgenutzt werden. Dies wird auch durch die Ergebnisse der beiden Fallbeispiele untermauert, bei denen eine durchschnittliche CPU-Auslastung von über 90% erreicht werden konnte. Weiterhin wurde LogFarm so entworfen, dass auch verteilte Simulation möglich ist. Da parallele und verteilte Simulation möglich ist, kann der Nutzer Einfluss auf die verfügbaren Ressourcen nehmen. Aufgrund dessen wird Anforderung VI als erfüllt angesehen.

Damit Anforderung VII erfüllt ist, muss ein Experimentierplan, der mit einem anderen Werkzeug generiert wurde, in LogFarm nutzbar gemacht werden. In Abschnitt 4.3.6 wurde beschrieben, dass der Simulator einen Experimentierplan im CSV-Format verarbeiten kann. Über die grafische Nutzeroberfläche kann der Nutzer eine entsprechende Datei ein-

lesen. In den Fallbeispielen wurde dieser Vorgang zweimal erfolgreich durchgeführt. Auch diese Anforderung kann somit, vorausgesetzt das vorgegebene Dateiformat wird eingehalten, als erfüllt angesehen werden.

Die letzte zu überprüfende Anforderung ist Anforderung VIII. Sie besagt, dass eine Weiterentwicklung LogFarms möglich sein muss. In Abschnitt 4.4 wurde beschrieben, dass LogFarm als Open-Source-Anwendung frei verfügbar ist und in englischer Sprache programmiert wird. Somit ist es für eine große Gruppe von Interessierten möglich, jeden Aspekt der Anwendung zu modifizieren. Aufgrund des modularen Aufbaus der Softwarearchitektur (vgl. Abschnitt 4.3.3), sind Anpassungen an einzelnen Modulen möglich, ohne detaillierte Kenntnisse des übrigen Programmcodes zu besitzen. Zudem ist es durch das komponentenbasierte System der Agenten leicht möglich, weitere Komponenten hinzuzufügen oder Änderungen durchzuführen. Aufgrund der genannten Argumente kann gefolgert werden, dass Anforderung VIII erfüllt ist.

Basierend auf der vorangegangenen Diskussion wird nun ein Fazit gezogen. Von den acht abgeleiteten Anforderungen aus Abschnitt 4.1 konnten alle erfüllt werden. Bei zwei Anforderungen zeigten sich jedoch Einschränkungen. Nichtsdestotrotz ist die Entwicklung von LogFarm als erfolgreich anzusehen. Das Hauptziel konnte erreicht werden und beide Fallbeispiele zeigten aussichtsreiche Ergebnisse und Erkenntnisse auf. Die Einschränkungen bei den Anforderungen liegen in der Tatsache begründet, dass weiterer Entwicklungsaufwand für den Prototyp erforderlich ist. Zu erwartende Fehler im weiteren Entwicklungsverlauf können aus den Einschränkungen nicht abgeleitet werden. Durch die Möglichkeit der Weiterentwicklung der Open-Source-Anwendung kann das aufgezeigte Potential von LogFarm in der Zukunft weiter untersucht und erschlossen werden. Zudem hat sich in den Fallbeispielen gezeigt, dass der Engpass bei der Generierung von Daten nicht bei der Simulation in LogFarm zu suchen ist, sondern bei dem verwendeten Graphdatenbanksystem, beziehungsweise der Schnittstelle dazu. Optimierungen an diesen Stellen würden helfen, LogFarm als Gesamtsystem zu einer höheren Performanz zu verhelfen.

Die Möglichkeiten zur Weiterentwicklung des Funktionsumfangs des Simulators sind vielfältig. Beispielsweise wurde bisher bei der Berechnung des Abstands von zwei Standorten eine gerade Linie, basierend auf den geografischen Koordinaten, gewählt. Wie Rabe et al. (2020) gezeigt haben, ist dieses Vorgehen für Logistiknetzwerke, insbesondere in städtischen Gebieten, ungenau und kann auf verschiedene Arten realistischer gestaltet werden. Ein anderes Beispiel betrifft die Generierung der Nachfrage, die bisher über eine vergleichsweise einfache Modellierung erfolgt (vgl. Abschnitt 4.3.2). Im Zuge der Forschung des vorgestellten Farming-for-Mining-Frameworks von Hunker et al. (2021) haben Wuttke et al. (2022) einen Nachfragegenerator entwickelt, der synthetische Nachfrage eines Logistiknetzwerks für saisonale Artikel generieren kann und durch eine statistische Versuchsplanung gesteuert wird. Eine Integration eines solchen Ansatzes kann die Modellierung von LogFarm verbessern. Weiterhin sollte erprobt werden, ob mit LogFarm eine verteilte Simulation möglich ist und wie sich ein Netzwerk von Computern auf die Performanz des Graphdatenbanksystems auswirkt.

Ein weiterer Aspekt zur Weiterentwicklung von LogFarm bietet die Generierung von Daten. Bisher werden nur Bestellungen und Lagerbestandsmeldungen generiert. Um eine große heterogene Datenmenge zu generieren, sollten weitere Daten generierbar sein.

6 Zusammenfassung und Ausblick

Im Verlauf der vorliegenden Arbeit wurde LogFarm, ein graphbasierter Simulator zur Simulation von Logistiknetzwerken, entwickelt. LogFarm wurde spezifisch für den Einsatz in einem Data-Farming-Framework konzipiert, was bedeutet, dass Daten durch den Simulator generiert werden. Die Simulation erfolgt kontinuierlich und simuliert Modelle, die auf dem agentenbasierten Modellierungskonzept beruhen. Über eine grafische Benutzeroberfläche wird ein Editor angeboten, der zur interaktiven Modellerstellung benutzt werden kann. Die Veröffentlichung des Simulators erfolgt unter einer Open-Source-Lizenz, um eine Weiterentwicklung des Simulators für Interessierte zu ermöglichen.

Zu Beginn wurden die technischen Grundlagen zur Entwicklung von LogFarm dargelegt. Die technischen Grundlagen umfassen Grundlagen zur Simulation und Modellierung, Graphen, Datenbanken und Data Farming. Außerdem wurde in den logistischen Kontext des Simulators eingeführt, welcher die Themen Logistik, Logistiknetzwerke und Logistikaufgaben beinhaltet.

Anschließend wurde die Entwicklung von LogFarm anhand des Wasserfallmodells beschrieben. Diesem Modell folgend, sind zunächst acht Anforderungen an den zu entwickelnden Simulator abgeleitet und spezifiziert worden. Basierend auf den identifizierten Anforderungen folgte die Vorstellung eines Konzeptmodells generischer Logistiknetzwerke zur Modellierung im Simulator, auf dem die Simulationsmodelle basieren, die durch LogFarm verarbeitet werden können. Zudem wurde das Softwaredesign des Simulators erläutert. Dazu gehört neben der Softwarearchitektur und dem Simulatorkern auch die Bausteinbibliothek des Simulators. Mithilfe der Bausteine kann der Anwender auf einfacher Weise Simulationsmodelle erstellen, die im Simulatorkern verarbeitet werden können. Die Bausteine weisen ein komponentenbasiertes System auf, mit dem grundlegende logistische Funktionalitäten einem Baustein zur Verfügung gestellt werden können. Die Bausteinbibliothek umfasst fünf Bausteine zur Modellierung von Zulieferern, Produktionsstätten, Warenhäusern, Kunden und Kundengruppen. Nach der Vorstellung der Bausteine und deren Komponenten, wurde die Modellierung der Komponenten auf Datenebene mit dem Eigenschaftsgraphen-Modell vorgestellt. Es wurde zudem beschrieben, wie die Datenbankanbindung mithilfe des OGM erfolgt, um in der internen Datenstruktur von LogFarm stets eine Graphenrepräsentation zu verwenden und somit unnötige Transformationsprozesse zu sparen. Im Zuge des Softwaredesigns wurde auch der Experiment Manager als wichtige Softwarekomponente des Simulator vorgestellt. Er ist für die Realisierung einer parallelen und verteilten Berechnung der Simulation zuständig. Zudem werden die Werte des Experimentplans dort mit dem Simulationsmodell zusammen geführt. Weitere Bestandteile des Softwaredesign sind die nicht-grafischen Schnittstellen und die grafische Benutzeroberfläche. Im Zuge der grafischen Benutzeroberfläche wurde auch der Editor zur interaktiven Modellbildung vorgestellt. Gemäß des Wasserfallmodells ist nach der Einführung in das Softwaredesign von LogFarm die Implementierung beschrieben worden. Insbesondere ist hierbei die Veröffentlichung als Open-Source-Anwendung hervorzuheben.

Der in der Implementierung entwickelte Prototyp wurde daran anschließend getestet. Dazu wurden zwei Fallbeispiele für ein imaginäres und ein reales Logistiknetzwerk geplant und

ausgeführt. Basierend auf den Erkenntnissen aus den Fallbeispielen und des Softwaredesigns ist eine Diskussion geführt worden, inwiefern LogFarm die ursprünglich aufgestellten Anforderungen erfüllt. Die Ergebnisse der Diskussion wurden in einem Fazit zusammengefasst, welches die Entwicklung von LogFarm als erfolgreich und aussichtsreich bezeichnet.

Die Ergebnisse dieser Arbeit zeigen, dass graphbasierte Ansätze in der Domäne Simulation eine Alternative zu herkömmlichen relationalen Ansätzen bieten. Dem Autor ist kein anderer Simulator bekannt, der ein Simulationsmodell in Form eines Eigenschaftsgraphen verwendet oder Ergebnisdaten direkt in einer Graphdatenbank abspeichert. Somit konnte eine Lücke in der Verarbeitung von Daten in Form von Graphen in der Logistik geschlossen werden.

Der angebotene Funktionsumfang von LogFarm lässt zwar die Modellierung einiger Aspekte von Logistiknetzwerken zu, ist im Vergleich zu anderen kommerziellen Simulatoren auf Basis des Relationenmodells jedoch gering. Daher sollte der Funktionsumfang zunächst erhöht werden, um Logistiknetzwerke realistischer modellieren zu können. Durch die Veröffentlichung unter einer Open-Source-Lizenz und der Umsetzung des Prototyps in englischer Sprache ist diese Möglichkeit geboten. Weiterhin ist LogFarm derzeit dazu konzipiert Daten zu generieren. Jedoch werden bisher lediglich Daten zu Bestellungen und Bestandsmeldungen generiert. LogFarm sollte dahingehend erweitert werden, dass weitere Daten generiert werden können. Zudem könnte LogFarm nicht nur zur Generierung von Daten, sondern auch für die klassische Untersuchung eines Systems eingesetzt werden, indem eine geeignete Erfassung und Aufbereitung von interessanten Messgrößen erfolgt.

Aufgrund der aussichtsreichen Ergebnisse, die LogFarm in den Fallbeispielen dieser Arbeit erzielen konnte, und dem bisher nicht ausreichend erforschten Potential von Graphdatenbanken, sollte weitere Forschung auf dem Gebiet graphbasierter Ansätze in der Simulation durchgeführt werden. Beispielsweise kann untersucht werden, inwiefern die Integration von LogFarm in ein graphbasiertes Farming-for-Mining-Framework Vorteile gegenüber anderen Simulatoren bietet. Weiterhin kann Betrachtet werden, ob graphbasierte Ansätze in der Simulation und die Modellierung von Simulationsmodellen im Eigenschaftsgraphen-Modell gegenüber klassischen Ansätzen so viele Vorteile bieten, dass die Entwicklung neuer Werkzeuge in diesem Bereich forciert werden sollte und vielleicht sogar zu einem Paradigmenwechsel führen könnte. In dieser Arbeit wurde die Entwicklung des graphbasierten Simulators auf die Domäne der Logistik bezogen. Die Anwendung in anderen Domänen, wie beispielsweise der Produktion, Physik oder Chemie, in denen vernetzte Daten eine große Rolle spielen können, erscheint sinnvoll und könnte untersucht werden. Abschließend sind die Ergebnisse der Arbeit im Bezug auf LogFarm und allgemein graphbasierte Simulatoren als vielversprechend zu bezeichnen.

Literaturverzeichnis

Aigner, M.: Graphentheorie: Eine Einführung aus dem 4-Farben Problem. 2. Aufl. Lehrbuch. Wiesbaden: Springer Spektrum, 2015.

Andelfinger, V. P.; Hänisch, T.: Industrie 4.0. Wiesbaden: Springer Fachmedien, 2017.

Andrae, S.; Pobuda, P.: Agentenbasierte Modellierung. Wiesbaden: Springer Fachmedien, 2021.

ArangoDB Inc: ArangoDB. 2022. URL: https://www.arangodb.com/ (zuletzt geprüft am 09. 04. 2022).

ASIM: Leitfaden für Simulationsbenutzer in Produktion und Logistik. ASIM-Mitteilungen (1997) 58, S. 1–32.

Bangsow, S.: Tecnomatix Plant Simulation. Cham: Springer International Publishing, 2020.

Banks, J.: Principles of simulation. In: Banks, J. (Hg.): Handbook of Simulation. Hoboken: John Wiley & Sons, 1998, S. 3–30.

Banks, J.; John, C.; Barry, N.; David, N.: Discrete-event system simulation. 4. ed. Prentice-Hall international series in industrial and systems engineering. Upper Saddle River, NJ: Pearson/Prentice Hall, 2005.

Banks, J.; Carson II, J. S.; Nelson, B. L.; Nicol, D. M.: Discrete-event system simulation. 5. Aufl. Always learning. Harlow: Pearson, 2014.

Bause, F.; Beilner, H.; Kemper, P.: Modellierung und Analyse von Logistiknetzwerken mit Prozessketten. ASIM 2000, 14th Symposium Simulationstechnik (2000), S. 63–67.

Bernhard, J.; Dragan, M.: Bewertung der Informationsgüte in der Informationsgewinnung für die modellgestützte Analyse großer Netze der Logistik. Dortmund: Sonderforschungsbereich (2007) 559.

Besta, M.; Peter, E.; Gerstenberger, R.; Fischer, M.; Podstawski, M.; Barthels, C.; Alonso, G.; Hoefler, T.: Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries. 2022. URL: http://arxiv.org/pdf/1910.09017v5 (zuletzt geprüft am 19. 04. 2022).

Bloch, J.: Effective Java. 3. Aufl. Bloch, Joshua (VerfasserIn). Boston u. a.: Addison-Wesley, 2018.

Bodendorf, F.: Daten- und Wissensmanagement. 2. Aufl. Springer-Lehrbuch. Berlin: Springer, 2006.

Bousonville, T.: Logistik 4.0: Die digitale Transformation der Wertschöpfungskette. essentials. Wiesbaden: Springer Fachmedien, 2017.

Box, G. E. P.; Hunter, J. S.; Hunter, W. G.: Statistics for experimenters: Design, innovation, and discovery. 2. Aufl. Wiley series in probability and statistics. Hoboken, NJ: Wiley-Interscience, 2005.

Brandstein, A. G.; Horne, G. E.: Data Farming: A Meta-technique for Research in the 21st Century. Maneuver Warfare Science (1998), S. 93–99.

Bretzke, W.-R.: Logistische Netzwerke. Berlin: Springer, 2020.

Cioppa, T. M.; Lucas, T. W.: Efficient Nearly Orthogonal and Space-Filling Latin Hypercubes. Technometrics 49 (2007) 1, S. 45–55.

Clarke, G.; Wright, J. W.: Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Operations Research 12 (1964) 4, S. 568–581.

Cleve, J.; Lämmel, U.: Data Mining. 3. Auflage. De Gruyter Studium. Berlin: De Gruyter, 2020.

Cook, D. J.; Holder, L. B. (Hg.): Mining graph data. Hoboken, N.J: Wiley-Interscience, 2007.

Deininger, M.: Modellierungsmethode für die simulationsbasierte Optimierung rekonfigurierbarer Produktionssysteme. Göttingen: Cuvillier, Dissertation, 2019.

Dietze, F.; Karoff, J.; Calero Valdez, A.; Ziefle, M.; Greven, C.; Schroeder, U.: An Open-Source Object-Graph-Mapping Framework for Neo4j and Scala: Renesca. In: Buccafurri, F.; Holzinger, A.; Kieseberg, P.; Tjoa, A. M.; Weippl, E. (Hg.): Availability, Reliability, and Security in Information Systems. Bd. 9817. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, S. 204–218.

DIN IEC 60050-351: Internationales Elektrotechnisches Wörterbuch - Teil 351: Leittechnik. Berlin: Beuth, 2014.

Dobhan, A.: Internal Supply Chain Management: Entwicklung und experimentelle Analyse hybrider Losgrößenplanungsverfahren. Bd. 7. Schriftenreihe Logistik und Supply Chain Management. Bamberg: University of Bamberg Press, 2012.

Domschke, W.: Logistik: Rundreisen und Touren. 4. Aufl. Oldenbourgs Lehr- und Handbücher der Wirtschafts- u. Sozialwissenschaften. Berlin: Oldenbourg Wissenschaftsverlag, 2018.

Domschke, W.; Drexl, A.; Klein, R.; Scholl, A.: Einführung in Operations Research. Berlin: Springer, 2015.

Fairley, R. E.: Software engineering concepts. Internat. ed. McGraw-Hill series in software engineering and technology. New York: McGraw-Hill, 1985.

Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.: From Data Mining to Knowledge Discovery in Databases. AI Magazine 17 (1996) 3, S. 37–54.

Feldkamp, N.; Bergmann, S.; Strassburger, S.; Borsch, E.; Richter, M.; Souren, R.: Combining Data Farming and Data Envelopment Analysis for Measuring Productive Efficiency in Manufacturing Simulations. In: Rabe, M.; Juan, A. A.; Mustafee, A.; Skoogh, S. J.; Johansson, B. (Hg.): Proceedings of the 2018 Winter Simulation Conference (WSC). Piscataway, New Jersey: IEEE, 2018, S. 1440–1451.

Fernandes, D.; Bernardino, J.: Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. In: Proceedings of the 7th International Conference on Data Science, Technology and Applications. SCITEPRESS - Science and Technology Publications, 2018, S. 373–380.

Fleischmann, B.: Grundlagen: Begriffe der Logistik, logistische Systeme und Prozesse. In: Arnold, D.; Isermann, H.; Kuhn, A.; Tempelmeier, H.; Furmans, K. (Hg.): Handbuch Logistik. 3. Aufl. Berlin: Springer, 2008, S. 3–12.

Francis, N.; Green, A.; Guagliardo, P.; Libkin, L.; Lindaaker, T.; Marsault, V.; Plantikow, S.; Rydberg, M.; Selmer, P.; Taylor, A.: Cypher. In: Das, G.; Jermaine, C.; Bernstein, P. (Hg.): Proceedings of the 2018 International Conference on Management of Data. New York, NY, USA: ACM, 2018, S. 1433–1445.

Franz Inc.: AllegroGraph. 2022. URL: https://allegrograph.com/ (zuletzt geprüft am 09. 04. 2022).

Fujimoto, R.: Parallel and Distributed Simulation. In: Proceedings of the 2015 Winter Simulation Conference, December 6 - 9, 2015, Huntington Beach, CA. Piscataway, NJ und Madison, Wis.: IEEE und Omnipress, 2015, S. 45–59.

Gadatsch, A.; Landrock, H.: Big Data für Entscheider. Wiesbaden: Springer Fachmedien, 2017.

García, S.; Luengo, J.; Herrera, F.: Data Preprocessing in Data Mining. 1. Aufl. Intelligent Systems Reference Library. Berlin: Springer-Verlag, 2015.

Gleißner, H.; Femerling, J. C.: Logistik. Wiesbaden: Gabler Verlag, 2012.

GNU: GPLv3. 2022. URL: https://www.gnu.org/licenses/quick-guide-gplv3.html (zuletzt geprüft am 14. 04. 2022).

Goldberg, D. E.; Korb, B.; Deb, K.: Messy Genetic Algorithms Motivation, Analysis and First Results. Complex Systems (1989) 3 (5), S. 493–530.

Gosnell, D. K.; Broecheler, M.: The practitioners guide to graph data. Boston: O'Reilly, 2020.

Gruler, A.; Juan, A. A.; Klüter, A.; Rabe, M.: A Simulation-Optimization Approach for the Two-Echelon Location Routing Problem Arising in the Creation of Urban Consolidation Centres. In: Wenzel, S.; Peter, T. (Hg.): Simulation in Produktion und Logistik 2017. ASIM-Mitteilung. Kassel: Kassel University Press, 2017, S. 129–138.

Gudehus, T.: Logistik 2. Berlin: Springer, 2012.

Gutenschwager, K.; Rabe, M.; Spieckermann, S.; Wenzel, S.: Simulation in Produktion und Logistik. Berlin: Springer, 2017.

Hausladen, I.: IT-gestützte Logistik. Wiesbaden: Springer Fachmedien, 2016.

Hecht, R.; Jablonski, S.: NoSQL evaluation: A use case oriented survey. In: 2011 International Conference on Cloud and Service Computing. IEEE, 2011, S. 336–341.

Heger, J.; Hildebrandt, T.: Simulationsbasierte Optimierung der Reihenfolgeplanung am Beispiel eines Liniensorters in der Automobilindustrie. In: Rabe, M.; Clausen, U. (Hg.): Simulation in Production and Logistics 2015. Stuttgart: Fraunhofer, 2015, S. 11–20.

Heiserich, O.-E.; Helbig, K.; Ullmann, W.: Logistik: Eine praxisorientierte Einführung. 4. Aufl. Lehrbuch. Wiesbaden: Gabler, 2011.

Holland, J. H.: Adaptation in Natural and Artificial Systems. The MIT Press, 1992.

Horne, G.; Seichter, S.: Data Farming in Support of NATO Operations – Methodology and Proof-of-Concept. In: Tolk, A.; Diallo, S. Y.; Ryzhov, I. O.; Yilmaz, L.; Buckley, S.; Miller J. A. (Hg.): Proceedings of the 2014 Winter Simulation Conference. Piscataway, New Jersey: IEEE, 2014, S. 2355–2363.

Huber, D.; Kaiser, T.: Wie das Internet der Dinge neue Geschäftsmodelle ermöglicht. In: Reinheimer, S. (Hg.): Industrie 4.0. Wiesbaden: Springer Fachmedien, 2017, S. 17–28.

Hunker, J.; Scheidler, A. A.; Rabe, M.: A Systematic Classification of Database Solutions for Data Mining to Support Tasks in Supply Chains. In: Kersten, W.; Blecker, T.; Ringle, C. (Hg.): Data Science and Innovation in Supply Chain Management : How Data Transforms the Value Chain. 2020, S. 395–425.

Hunker, J.; Wuttke, A.; Scheidler, A. A.; Rabe, M.: A Farming-for-Mining-Framework to Gain Knowledge in Supply Chains. In: Kim, S.; Feng, B.; Smith, K.; Masoud, S.; Zheng, Z.; Szabo, C.; Loper, M. (Hg.): Proceedings of the 2021 Winter Simulation Conference. Piscataway, New Jersey: IEEE, 2021.

Inderfurth, K.: Lagerhaltungsmodelle. In: Kern, W. (Hg.): Handwörterbuch der Produktionswirtschaft. 2., völlig neu gestaltete Aufl. Enzyklopädie der Betriebswirtschaftslehre. Stuttgart: Schäffer-Poeschel, 1996, S. 1024–1037.

Jahangirian, M.; Eldabi, T.; Naseer, A.; Stergioulas, L. K.; Young, T.: Simulation in manufacturing and business: A review. European Journal of Operational Research 203 (2010) 1, S. 1–13.

Jockisch, M.; Rosendahl, J.: Klassifikation von Modellen. In: Bandow, G.; Holzmüller, H. H. (Hg.): Das ist gar kein Modell! Wiesbaden: Gabler, 2009, S. 23–52.

Johnson, M. E.; Moore, L. M.; Ylvisaker, D.: Minimax and maximin distance designs. Journal of Statistical Planning and Inference 26 (1990) 2, S. 131–148.

Juan, A. A.; Rabe, M.: Combining simulation with heuristics to solve stochastic routing and scheduling problems. In: Dangelmaier, W.; Laroque, C.; Klaas, A. (Hg.): Simulation in Produktion und Logistik 2013. ASIM-Mitteilung. Paderborn: Heinz-Nixdorf-Inst. Univ. Paderborn, 2013, S. 641–649.

Jungmann, T.; Uygun, Y.: Das Dortmunder Prozesskettenmodell in der Intralogistik. In: Bandow, G.; Holzmüller, H. H. (Hg.): Das ist gar kein Modell! Wiesbaden: Gabler, 2009, S. 357–382.

Kemper, A.; Eickler, A.: Datenbanksysteme: Eine Einführung. 10. Aufl. De Gruyter Oldenbourg Studium. Berlin: De Gruyter Oldenbourg, 2015.

Klaus, P.; Krieger, W.; Krupp, M.: Gabler Lexikon Logistik. Wiesbaden: Gabler Verlag, 2012.

Krcmar, H.: Einführung in das Informationsmanagement. 2. Aufl. Springer-Lehrbuch. Berlin: Springer, 2015.

Krischke, A.; Röpcke, H.: Graphen und Netzwerktheorie: Grundlagen - Methoden - Anwendungen. Quantitative Methoden. München: Hanser, 2015.

Kuhn, A.; Wenzel, S.: Simulation logistischer Systeme. In: Arnold, D.; Isermann, H.; Kuhn, A.; Tempelmeier, H.; Furmans, K. (Hg.): Handbuch Logistik. 3. Aufl. Berlin: Springer, 2008, S. 73–94.

Landry, M.; Oral, M.: In search of a valid view of model validation for operations research. European Journal of Operational Research 66 (1993) 2, S. 161–167.

Laney, D.: 3D Data Management: Controlling Data Volume, Velocity, and Variety. Application Delivery Strategies (2001), S. 1–4.

Law, A. M.: Simulation Modeling and Analysis. 5. Aufl. New York: McGraw-Hill Education, 2015.

Liu, Y.; Folz, P.; Pan, S.; Ramparany, F.; Bolle, S.; Ballot, E.; Coupaye, T.: Digital Twin-Driven Approach for Smart City Logistics: The Case of Freight Parking Management. In: Dolgui, A.; Bernard, A.; Lemoine, D.; Cieminski, G. von; Romero, D. (Hg.): Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems. Bd. 633. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, 2021, S. 237–246.

Lunze, J.: Ereignisdiskrete Systeme. Berlin: De Gruyter, 2012.

Lutz, W.-G.: Das objektorientierte Paradigma. Wiesbaden: Deutscher Universitätsverlag, 1997.

Manitz, M.: Lagerhaltungspolitiken. In: Claus, T.; Herrmann, F.; Manitz, M. (Hg.): Produktionsplanung und –steuerung. Berlin: Springer, 2015, S. 179–208.

Manivannan, M. S.: Simulation of Logistics and Transportation Systems. In: Banks, J. (Hg.): Handbook of Simulation. Hoboken: John Wiley & Sons, 1998, S. 571–604.

MariaDB Foundation: MariaDB. 2022. URL: https://mariadb.org/ (zuletzt geprüft am 09. 04. 2022).

März, L.; Krug, W.: Kopplung von Simulation und Optimierung. In: März, L.; Krug, W.; Rose, O.; Weigert, G. (Hg.): Simulation und Optimierung in Produktion und Logistik. VDI-Buch. Berlin: Springer, 2011, S. 41–46.

Mattfeld, D.; Vahrenkamp, R.: Logistiknetzwerke. Wiesbaden: Springer Fachmedien, 2014.

Mayo, C. S.; Kessler, M. L.; Eisbruch, A.; Weyburne, G.; Feng, M.; Hayman, J. A.; Jolly, S.; El Naqa, I.; Moran, J. M.; Matuszak, M. M.; Anderson, C. J.; Holevinski, L. P.; McShan, D. L.; Merkel, S. M.; Machnak, S. L.; Lawrence, T. S.; Haken, R. K. ten: The Big Data Effort in Radiation Oncology: Data Mining or Data Farming? Advances in Radiation Oncology 1 (2016) 4, S. 260–271.

McDermott, D.: An Alternative Framework for Urban Goods Distribution: Consolidation. Transportation Journal 15 (1975), S. 29–39.

Meier, A.; Kaufmann, M.: SQL- & NoSQL-Datenbanken. 8. Aufl. eXamen.press. Meier, Andreas (VerfasserIn) Kaufmann, Michael (VerfasserIn). Berlin: Springer, 2016.

Neo4j: Neo4j. 2022. URL: https://neo4j.com/ (zuletzt geprüft am 09. 04. 2022).

Neumann, M.: Java-Kompendium: Professionell Java programmieren lernen. Neumann, Markus (VerfasserIn). Landshut: BMU Verlag, 2019.

Noche, B.; Wenzel, S.: Marktspiegel Simulationstechnik in Produktion und Logistik. Praxiswissen aktuell. Köln: TÜV Rheinland, 1991.

North, K.: Wissensorientierte Unternehmensführung. Wiesbaden: Springer Fachmedien, 2016.

Oracle Corporation: MySQL. 2022a. URL: https://www.mysql.com/de/ (zuletzt geprüft am 09. 04. 2022).

Oracle Corporation: Oracle Database. 2022b. URL: www.oracle.com/database/ (zuletzt geprüft am 09. 04. 2022).

Otto, B.; Österle, H.: Corporate Data Quality. Berlin: Springer, 2016.

Peters, W.: Zur Theorie der Modellierung von Natur und Umwelt: Dissertation. Berlin, 1998.

Pfohl, H.-C.: Logistiksysteme. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018.

Pichpibul, T.; Kawtummachai, R.: An improved Clarke and Wright savings algorithm for the capacitated vehicle routing problem. ScienceAsia 38 (2012) 3, S. 307.

Plowman, E. G.: Lectures on Elements of Business Logistics. Stanford: Stanford University Press, 1964.

Pohl, K.; Rupp, C.: Basiswissen requirements engineering: Aus- und Weiterbildung zum Certified Professional for Requirements Engineering : foundation level nach IREB-Standard. 4. Aufl. Heidelberg: dpunkt.verlag, 2015.

Probst, G.; Raub, S.; Romhardt, K.: Wissen managen. Wiesbaden: Gabler Verlag, 2012.

Rabe, M.; Spieckermann, S.; Wenzel, S.: Verifikation und Validierung für die Simulation in Produktion und Logistik. Berlin: Springer, 2008.

Rabe, M.; Dross, F.; Wuttke, A.: Combining a discrete-event simulation model of a logistics network with deep reinforcement learning. In: Duarte, A.; Juan, A. A.; Lourenco, H. R. (Hg.): Proceedings of the 12th Metaheuristics International Conference (MIC). Barcelona, 2017a, S. 438–447.

Rabe, M.; Dross, F.; Schmitt, D.; Ammouriova, M.: Decision Support for Logistics Networks in Materials Trading Using a Simheuristic Framework and User-generated Action Types. In: Wenzel, S.; Peter, T. (Hg.): Simulation in Produktion und Logistik 2017. ASIM-Mitteilung. Kassel: Kassel University Press, 2017b, S. 109–118.

Rabe, M.; Klüter, A.; Wuttke, A.: Evaluating the Consolidation of Distribution Flows Using a Discrete Event Supply Chain Simulation Tool: Application to a Case Study in Greece. In: Rabe, M.; Juan, A. A.; Mustafee, A.; Skoogh, S. J.; Johansson, B. (Hg.): Proceedings of the 2018 Winter Simulation Conference (WSC). Piscataway, New Jersey: IEEE, 2018, S. 2815–2826.

Rabe, M.; Klueter, A.; Raps, J.: Evaluating different distance metrics for calculating distances of last mile deliveries in urban areas for integration into supply chain simulation. Journal of Simulation 14 (2020) 1, S. 41–52.

Riha, I. V.: Kosten- und leistungsoptimierter Betrieb kooperativer Logistiknetzwerke. In: Buchholz, P.; Clausen, U. (Hg.): Große Netze der Logistik. Berlin: Springer, 2009, S. 75–99.

Robinson, I.; Webber, J.; Eifrem, E.: Graph databases - new opportunities for connected data. 2. Aufl. Sebastopol, CA: O'Reilly, 2015.

Robinson, S.: Simulation: The Practice of Model Development and Use. Chichester, Eng und Hoboken, N.J: Wiley, 2004.

Rockwell Automation: Arena. 2022. URL: https://www.rockwellautomation.com/de-de/products/software/arena-simulation.html (zuletzt geprüft am 08. 04. 2022).

Rose, O.; März, L.: Simulation. In: März, L.; Krug, W.; Rose, O.; Weigert, G. (Hg.): Simulation und Optimierung in Produktion und Logistik. VDI-Buch. Berlin: Springer, 2011, S. 13–20.

Royce, W. W.: Managing the development of large software systems. Proceedings of IEEE WESCON 26 (1970), S. 1–9.

Sanchez, S. M.: NOLHdesigns spreadsheet. 2011. URL: http://harvest.nps.edu (zuletzt geprüft am 16. 04. 2022).

Sanchez, S. M.: Data Farming: Better Data, Not Just Big Data. In: Rabe, M.; Juan, A. A.; Mustafee, A.; Skoogh, S. J.; Johansson, B. (Hg.): Proceedings of the 2018 Winter Simulation Conference (WSC). Piscataway, New Jersey: IEEE, 2018, S. 425–439.

Savelsbergh, M.; van Woensel, T.: 50th Anniversary Invited Article - City Logistics: Challenges and Opportunities. Transportation Science 50 (2016) 2, S. 579–590.

Scheidler, A. A.: Methode zur Erschließung von Wissen aus Datenmustern in Supply-Chain-Datenbanken. 1. Aufl. Bd. 1. Schriftenreihe Fortschritte in der IT in Produktion und Logistik. Göttingen: Cuvillier, 2017.

Schmuck, T.: Ein Beitrag zur effizienten Gestaltung globaler Produktions- und Logistiknetzwerke mittels Simulation: Zugl.: Erlangen-Nürnberg, Univ., Diss., 2013. Bd. 257. Bericht aus dem Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik. Bamberg: Meisenbach, 2014.

Schuh, G.; Hering, N.; Brunner, A.: Einführung in das Logistikmanagement. In: Schuh, G.; Stich, V. (Hg.): Logistikmanagement. Berlin: Springer, 2013, S. 1–34.

Schulte, C.: Logistik. München: Verlag Franz Vahlen GmbH, 2016.

Schweizer, W.: MATLAB Kompakt. Heidelberg: De Gruyter, 2022.

SDZ GmbH: DOSIMIS-3. 2022. URL: https://www.sdz.de/software/dosimis-3/ (zuletzt geprüft am 08. 04. 2022).

Seeck, S.: Erfolgsfaktor Logistik: Klassische Fehler erkennen und vermeiden. 1. Aufl. Wiesbaden: Gabler, 2010.

Shafranovich, Y.: Common Format and MIME Type for Comma-Separated Values (CSV) Files. RFC Editor, 2005.

Siebertz, K.; van Bebber, D.; Hochkirchen, T.: Statistische Versuchsplanung. Berlin: Springer, 2017.

Simchi-Levi, D.; Kaminsky, P.; Simchi-Levi, E.: Designing and managing the supply chain: Concepts, strategies, and case studies. The Irwin/McGraw-Hill series Operations and decision sciences Operations Management. Boston: Irwin/McGraw-Hill, 2000.

Sinnott, R.: Virtues of the Haversine. Sky and Telescope 68 (1984) 2, S. 158–159.

Smith, J. S.: Survey on the use of simulation for manufacturing system design and operation. Journal of Manufacturing Systems 22 (2003) 2, S. 157–171.

solid IT: DB-Engines. solid IT (Hg.). 2022. URL: https://db-engines.com/de/ (zuletzt geprüft am 09. 04. 2022).

Sommerville, I.: Software engineering. 10. Aufl. Always learning. Sommerville, Ian (VerfasserIn). Boston: Pearson, 2016.

Sörensen, K.; Arnold, F.; Palhazi Cuervo, D.: A critical analysis of the improved Clarke and Wright savings algorithm. International Transactions in Operational Research 26 (2019) 1, S. 54–63.

Spieckermann, S.: Diskrete, ereignisorientierte Simulation in Produktion und Logistik - Herausforderungen und Trends. Conference: Simulation und Visualisierung 2005 (SimVis 2005), 3-4 März: Magdeburg, 2005.

Stachowiak, H.: Allgemeine Modelltheorie. Berlin: Springer, 1973.

Staritz, J.; Landwehr, M. auf der; Trott, M.; Viebahn, C. von: Entwicklung eines anwendungsorientierten Bausteinkastens zur Simulation kombinierter Transportmodelle mittels autonomer Fahrzeuge. In: Franke, J.; Schuderer, P. (Hg.): Simulation in Produktion und Logistik 2021. 1. Auflage. ASIM-Mitteilung. Göttingen: Cuvillier, 2021, S. 525–534.

Stich, V.; Hering, N.; Brosze, T.: Beschaffungslogistik. In: Schuh, G.; Stich, V. (Hg.): Logistikmanagement. Berlin: Springer, 2013, S. 77–114.

Sucky, E.: Netzwerkmanagement. In: Arnold, D.; Isermann, H.; Kuhn, A.; Tempelmeier, H.; Furmans, K. (Hg.): Handbuch Logistik. 3. Aufl. Berlin: Springer, 2008, S. 934–945.

Sucky, E.: Supply Chain Management. Sucky, Eric (VerfasserIn). Stuttgart: Kohlhammer Verlag, 2021.

Tempelmeier, H.: Bestandsmanagement in Supply Chains. 5. Aufl. Norderstedt: Books on Demand, 2015.

The AnyLogic Company: AnyLogic. 2022a. URL: https://www.anylogic.de/ (zuletzt geprüft am 10. 04. 2022).

The AnyLogic Company: AnyLogistix. 2022b. URL: https://www.anylogistix.com/ (zuletzt geprüft am 10. 04. 2022).

TigerGraph: TigerGraph. 2022. URL: https://www.tigergraph.com/ (zuletzt geprüft am 09. 04. 2022).

Tittmann, P.: Graphentheorie: Eine anwendungsorientierte Einführung. 4. Aufl. Hanser eLibrary. München: Carl Hanser Verlag, 2021.

2411: Begriffe und Erläuterungen im Förderwesen. Beuth, 1970.

VDI-Richtlinie 3633, Blatt 1: Simulation von Logistik-, Materialfluss- und Produktionssystemen - Grundlagen. Beuth, 2014.

Verbraeck, A.; Valentin, E. C.: Design Guidelines for Simulation Building Blocks. In: Proceedings of the 2008 Winter Simulation Conference. Piscataway, NJ: IEEE Service Center, 2008, S. 923–932.

Vukotic, A.; Watt, N.: Neo4j in action. Shelter Island, NY: Manning, 2015.

Waters, C. D. J.: Supply chain risk management: Vulnerability and resilience in logistics. 2. Aufl. London: Kogan Page, 2011.

Wenger, W.: Multikriterielle Tourenplanung: Dissertation. Gabler Research Produktion und Logistik. Wiesbaden: Gabler, 2009.

Wenzel, S.: Simulation logistischer Systeme. In: Tempelmeier, H. (Hg.): Modellierung logistischer Systeme. Berlin: Springer, 2018, S. 1–34.

Wenzel, S.; Bernhard, J.: Definition und Modellierung von Systemlasten für die Simulation logistischer Systeme. In: Nyhuis, P. (Hg.): Beiträge zu einer Theorie der Logistik. Berlin: Springer, 2008, S. 487–509.

Wunsch, G.; Schreiber, H.: Stochastische Systeme. 4. Aufl. Berlin: Springer, 2005.

Wuttke, A.; Hunker, J.; Scheidler, A. A.; Rabe, M.: Synthetic Demand Generation with Seasonality for Data Mining on a Data-Farmed Data Basis of a Two-Echelon Supply Chain. In: Proceedings of the International Conference on Industry Sciences and Computer Sciences Innovation. Elsevier, 2022.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACID	Atomarität, Konsistenz, Isolation, Dauerhaftigkeit
CSV	Comma Separated Values
CVRP	Capacitated Vehicle Routing Problem
ICWA	Improved Clarke and Wright Savings Algorithm
LKW	Lastkraftwagen
OGM	Object-Graph-Mapping
TUL-Prozesse	Transport-, Umschlags-, und Lagerprozesse
V&V	Verifikation und Validierung

Algorithmenverzeichnis

Anhang A: Anhang

A.1 Anhang zu Fallbeispiel 1

Tabelle A.1: Parameter des Simulationsmodells von Fallbeispiel 1

Baustein	Komponente	Eigenschaft	Faktor	Wert
Sägewerk	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Sägewerk	Standort	Breitengrad	○	51,4539
Sägewerk	Standort	Längengrad	○	9,1159
Sägewerk	Produktion	Produktionsrate	●	10
Sägewerk	Lagerung	Initialbestandsfaktor	○	0
Sägewerk	Lagerung	Politik	○	(s,q)
Sägewerk	Lagerung	Parameter1	○	200.000
Sägewerk	Lagerung	Parameter2	○	0
Sägewerk	Distribution	Fahrzeugflotte	○	50 LKW
Sägewerk	Störung	Tägliche Wahrscheinlichkeit	●	7
Sägewerk	Störung	Intensität Erwartungswert	●	8
Sägewerk	Störung	Intensität Abweichung	●	9
Wandhalterung	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Wandhalterung	Standort	Breitengrad	○	50,6717
Wandhalterung	Standort	Längengrad	○	10,9799
Wandhalterung	Produktion	Produktionsrate	●	11
Wandhalterung	Lagerung	Politik	○	(s,q)
Wandhalterung	Lagerung	Parameter1	○	200.000
Wandhalterung	Lagerung	Parameter2	○	0
Wandhalterung	Distribution	Fahrzeugflotte	○	50 LKW
Halbfabrikate	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Halbfabrikate	Standort	Breitengrad	○	52,4760
Halbfabrikate	Standort	Längengrad	○	10,1021
Halbfabrikate	Produktion	Produktionsrate	●	12 (Platine)
Halbfabrikate	Produktion	Produktionsrate	●	13 (Lautsprecher)
Halbfabrikate	Lagerung	Initialbestandsfaktor	○	0

Tabelle A.1: Parameter des Simulationsmodells von Fallbeispiel 1 (Fortsetzung)

Baustein	Komponente	Eigenschaft	Faktor	Wert
Halbfabrikate	Lagerung	Politik	○	(s,q) (beide)
Halbfabrikate	Lagerung	Parameter1	○	200.000 (beide)
Halbfabrikate	Lagerung	Parameter2	○	0 (beide)
Halbfabrikate	Distribution	Fahrzeugflotte	○	50 LKW
Halbfabrikate	Störung	Tägliche Wahrscheinlichkeit	●	14
Halbfabrikate	Störung	Intensität Erwartungswert	●	15
Halbfabrikate	Störung	Intensität Abweichung	●	16
Lautsprechersys.	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Lautsprechersys.	Standort	Breitengrad	○	51,8532
Lautsprechersys.	Standort	Längengrad	○	10,7905
Lautsprechersys.	Transformation	Transformationsrate	●	17
Lautsprechersys.	Transformation	Vorschrift	○	-2 Lautsprecher, -1 Platine, -1 Brett, +1 Lautsprechersys.
Lautsprechersys.	Lagerung	Politik	○	(s,q) (alle)
Lautsprechersys.	Lagerung	Parameter1	○	200.000 (alle)
Lautsprechersys.	Lagerung	Parameter2	○	0 (alle)
Lautsprechersys.	Distribution	Fahrzeugflotte	○	50 LKW
Warenhaus	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Warenhaus	Standort	Breitengrad	○	51,1459
Warenhaus	Standort	Längengrad	○	9,5630
Warenhaus	Lagerung	Initialbestandsfaktor	○	0
Warenhaus	Lagerung	Politik	○	(s,q) (alle)
Warenhaus	Lagerung	Parameter1	○	2.000 (alle)
Warenhaus	Lagerung	Parameter2	○	500 (alle)
Warenhaus	Distribution	Fahrzeugflotte	○	50 LKW
Dortmund	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Dortmund	Standort	Breitengrad	○	51,4920
Dortmund	Standort	Längengrad	○	7,3735
Dortmund	Konsum	Konsumrate	●	5 (Lautsprechersys.)
Dortmund	Konsum	Konsumrate	●	6 (Wandhalterung)

Tabelle A.1: Parameter des Simulationsmodells von Fallbeispiel 1 (Fortsetzung)

Baustein	Komponente	Eigenschaft	Faktor	Wert
Dortmund	Lagerung	Initialbestandsfaktor	○	0
Dortmund	Lagerung	Politik	○	(s,q) (beide)
Dortmund	Lagerung	Parameter1	○	900 (Lautsprechersys.)
Dortmund	Lagerung	Parameter2	○	180 (Wandhalterung)
Dortmund	Lagerung	Parameter2	○	250 (beide)
Frankfurt	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Frankfurt	Standort	Breitengrad	○	51,4920
Frankfurt	Standort	Längengrad	○	7,3735
Frankfurt	Konsum	Konsumrate	•	3 (Lautsprechersys.)
Frankfurt	Konsum	Konsumrate	•	4 (Wandhalterung)
Frankfurt	Lagerung	Initialbestandsfaktor	○	0
Frankfurt	Lagerung	Politik	○	(s,q) (beide)
Frankfurt	Lagerung	Parameter1	○	900 (Lautsprechersys.)
Frankfurt	Lagerung	Parameter2	○	180 (Wandhalterung)
Frankfurt	Lagerung	Parameter2	○	250 (beide)
Berlin	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Berlin	Standort	Breitengrad	○	52,4152
Berlin	Standort	Längengrad	○	13,4018
Berlin	Konsum	Konsumrate	•	1 (Lautsprechersys.)
Berlin	Konsum	Konsumrate	•	2 (Wandhalterung)
Berlin	Lagerung	Initialbestandsfaktor	○	0
Berlin	Lagerung	Politik	○	(s,q) (beide)
Berlin	Lagerung	Parameter1	○	900 (Lautsprechersys.)
Berlin	Lagerung	Parameter2	○	180 (Wandhalterung)
Berlin	Lagerung	Parameter2	○	250 (beide)
LKW		Kapazität	○	500

Legende:
Lautsprechersys. = Lautsprechersysteme ○ = kein Faktor • = Faktor

Tabelle A.2: Kopfzeile für das Nearly-Orthogonal-Latin-Hypercube-Tool von Sanchez (2011)

Spalte	**F.1**	**F.2**	**F.3**	**F.4**	**F.5**	**F.6**	**F.7**	**F.8**	**F.9**	**F.10**	**F.11**	**F.12**	**F.13**	**F.14**	**F.15**	**F.16**	**F.17**
Unterer Wert	20,25	4,05	13,5	2,7	11,25	2,25	0,6	0,3	0,1	45	9	45	90	0,005	0,7	0,2	45
Oberer Wert	24,75	4,95	16,5	3,3	13,75	2,75	0,8	0,5	0,3	55	11	55	110	0,1	0,8	0,3	55
Dezimalstellen	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

A.2 Anhang zu Fallbeispiel 2

Tabelle A.3: Parameter des Simulationsmodells von Fallbeispiel 2

Baustein	Komponente	Eigenschaft	Faktor	Wert
Zulieferer 1-5	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Zulieferer 1-5	Produktion	Produktionsrate	○	10.000
Zulieferer 1-5	Lagerung	Initialbestandsfaktor	○	0
Zulieferer 1-5	Lagerung	Politik	○	(s,q)
Zulieferer 1-5	Lagerung	Parameter1	●	1
Zulieferer 1-5	Lagerung	Parameter2	●	2
Zulieferer 1-5	Distribution	Fahrzeugflotte	○	500 LKW
Zulieferer 1	Standort	Breitengrad	○	38,059
Zulieferer 1	Standort	Längengrad	○	23,522
Zulieferer 2	Standort	Breitengrad	○	38,090
Zulieferer 2	Standort	Längengrad	○	23,562
Zulieferer 3	Standort	Breitengrad	○	38,031
Zulieferer 3	Standort	Längengrad	○	23,609
Zulieferer 4	Standort	Breitengrad	○	38,065
Zulieferer 4	Standort	Längengrad	○	23,762
Zulieferer 5	Standort	Breitengrad	○	38,284
Zulieferer 5	Standort	Längengrad	○	23,679
Konsol.	Standort	Arbeitszeit	○	[8,8,8,8,8,0,0]
Konsol.	Standort	Breitengrad	○	37,991
Konsol.	Standort	Längengrad	○	23,692
Konsol.	Lagerung	Initialbestandsfaktor	○	0
Konsol.	Lagerung	Politik	○	(s,q) (alle)
Konsol.	Lagerung	Parameter1	○	20.000 (alle)
Konsol.	Lagerung	Parameter2	○	500 (alle)
Konsol.	Distribution	Fahrzeugflotte	○	50 Van
Kunden	Standort	Arbeitszeit	○	[12,12,12-,12,12,12,0]
Kunden	Standort	Breitengrad min.	○	37,9421
Kunden	Standort	Breitengrad max.	○	38,0237
Kunden	Standort	Längengrad min.	○	23,6730
Kunden	Standort	Längengrad max.	○	23,7702
Kunden	Standort	Anzahl Kunden	○	100
Kunden	Nachfrage	Frequenz Erwartungswert	●	3

Tabelle A.3: Parameter des Simulationsmodells von Fallbeispiel 2 (Fortsetzung)

Baustein	Komponente	Eigenschaft	Faktor	Wert
Kunden	Nachfrage	Frequenz Abweichung	•	4
Kunden	Nachfrage	Gewichtung Einträge	○	0,2 (alle)
Kunden	Nachfrage	Menge Erwartungswert	•	5
Kunden	Nachfrage	Menge Abweichung	•	6
LKW		Kapazität	○	1650
Van		Kapazität	○	750

Legende:
Konsol. = Konsolidierungszentrum min. = Minimum max. = Maximum ○ = kein Faktor
• = Faktor

Algorithmus A.1: MATLAB-Skript zur Generierung des Experimentierplans von Fallbeispiel 2

```
plan=ff2n(6);
plan=transpose(plan);
plan=bsxfun(@times, plan, [1000;900;1;1;9;15]);
plan=bsxfun(@plus, plan, [2000;750;3;2;45;25]);
plan=transpose(plan);
writematrix(plan,'scenario2plan.csv');
```