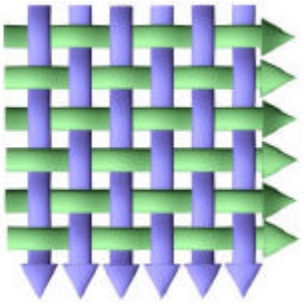
Sonderforschungsbereich 559
Modellierung großer Netze in der Logistik



Technical Report 03032

ISSN 1612-1376

Simulation eines Regionallagerkonzeptes

Teilprojekt A2:

Marco Motta
Iwo Riha
Stefan Weidt
Fraunhofer-Institut für Materialfluss und Logistik
Abteilung II
Joseph-von-Fraunhofer-Str. 2-4

44227 Dortmund

Dortmund, 15.01.2004

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Problemstellung

Unternehmen sehen sich im globalen Wettbewerb einem starken internationalen Konkurrenzdruck ausgesetzt. Um in diesem Umfeld bestehen zu können, muss ein Unternehmen zwei grundlegende Anforderungen erfüllen. Zum einen muss es in der Lage sein, dem hohen Preisdruck des internationalen Marktes standzuhalten, zum anderen muss es schnell auf Wünsche der Kunden reagieren können (/BULL94/, S. 1; /BICH79/, S. 3).

Zur Erfüllung der ersten der beiden grundlegenden Anforderungen ist es für ein Unternehmen unerlässlich Kosten zu senken. Hierzu müssen entlang der gesamten Wertschöpfungs- und Lieferkette Rationalisierungspotentiale aufgedeckt werden (/WILD87/, S. 1). In Verbindung mit der zweiten grundlegenden Anforderung an ein Unternehmen, der Kundenorientierung, ergibt sich jedoch ein Zielkonflikt. Dies zeigt sich darin, dass Maßnahmen zur Senkung von Kosten häufig zu einer Reduktion von Servicegraden führen, die ein Unternehmens seinen Kunden bietet.

Ein mögliches Rationalisierungspotential liegt in der Reduktion der Logistikkosten. Diese beinhalten dabei insbesondere die externen Transportkosten, externe und interne Lagerkosten, innerbetriebliche Transportkosten sowie die durch den Wareneinn- und -ausgang verursachten Kosten (/WEBE02/, S. 101). Die genannten Kosten fallen dabei entlang der gesamten Supply Chain, also in jeder Stufe des gesamten Wertschöpfungsprozesses vom Rohstofflieferanten bis zum Kunden (/HUGH00/, S. 17ff) an. Aufgrund der globalen Beschaffung von Rohstoffen und Gütern liegt ein hohes Optimierungspotential in der Beschaffung.

Auch die Beschaffung unterliegt dem oben beschriebenen Zielkonflikt. Eine Möglichkeit zur Senkung von Kosten besteht darin, vorhandene Bestände in einer Beschaffungskette zu minimieren. Hieraus ergibt sich ein Konflikt mit der Erfüllung der zweiten der beiden grundlegenden Anforderungen, der Minimierung der Reaktionszeit auf Kundenwünsche. Für eine Minimierung dieser Reaktionszeit ist eine Lagerhaltung unerlässlich. Die Bedeutung der Lagerhaltung zeigt sich insbesondere am Ende einer Wertschöpfungskette an der Schnittstelle zum Kunden. Hier dient die Lagerhaltung dazu, im Zeitraum zwischen zwei Wareneingängen die Befriedigung der Kundenwünsche zu gewährleisten (/SCHÄ72/, S. 29). Die Ausprägung dieses Servicegrades ist dabei abhängig von der Höhe der Lagerbestände (/BICH79/, S. 64), sowie von den zugrunde liegenden Beschaffungsstrategien und dem Kundenverhalten.

Aufgrund der gegenseitigen Beeinflussung der beiden Zielgrößen „Kosten“ und „Kundenorientierung“ und weiterer Einflussfaktoren ist die Untersuchung von

Beschaffungsketten mit einer komplexen Fragestellung verbunden (/PALU98/, S. 173ff).

Die Untersuchung einer gesamten Beschaffungskette stellt eine Untersuchung eines komplexen Gesamtsystems dar (/HUGH00/, S.17). Eine Beschaffungskette besteht aus mehrere Beschaffungsstufen, die selbst bereits ein hohes Maß an Komplexität aufweisen. Für die Untersuchung ist es daher sinnvoll das Hilfsmittel der Simulation einzusetzen, um eine Optimierung der Beschaffungskette im Rahmen des oben genannten Zielsystems vorzunehmen.

Für die Durchführung einer Untersuchung mit dem Hilfsmittel der Materialfluss-Simulation ist es erforderlich, eine geeignete Simulationssoftware für die Abbildung des zu untersuchenden Systems auszuwählen.

Die vorhandenen Simulationswerkzeuge zeigen einen deutlichen Trend zu spezialisierter Simulationssoftware für spezielle Anwendungsgebiete (/WENZ00a/, S. 423 – S. 432). Solche spezialisierte Simulationssoftware beinhaltet oft vorgefertigte Bausteine, mit welchen Systeme im Rahmen des jeweiligen Einsatzgebietes abgebildet werden können. Hierdurch sind jedoch die während der Modellierung verfügbaren Freiheitsgrade, z.B. hinsichtlich der Abbildungsgenauigkeit stark eingeschränkt (/NOCH01/, S. 27). Da insbesondere im Rahmen der Abbildung von Beschaffungsketten ein hohes Maß an verfügbaren Freiheitsgraden erforderlich ist, ist der Einsatz einer solchen spezialisierten Simulationssoftware nicht empfehlenswert.

Ein alternatives Konzept der Simulation ist das Sprachkonzept. Dieses sehr abstrakte Konzept basiert auf einer Programmiersprache mit simulationsspezifischen Erweiterungen (/WENZ00b/, S. 8). Ein solches Modellierungskonzept ermöglicht ein hohes Maß an Modellierungsfreiheit. Dieses Modellierungskonzept ist somit hinsichtlich der verfügbaren Freiheitsgrade für die Abbildung von Beschaffungsketten geeignet. Nachteile des Konzeptes zeigen sich jedoch in einem hohen Modellierungsaufwand und einem hohen Abstraktionsgrad der Modelle, so dass der Einsatz einer rein sprachorientierten Simulationssoftware für die Abbildung von Beschaffungsketten als ebenfalls nicht geeignet erscheint.

Für die Untersuchungen wurde die Simulationssoftware *promodel* ausgewählt. Diese Simulationssoftware zeichnet sich dadurch aus, dass sie einen Kompromiss zwischen den oben beschriebenen Modellierungskonzepten darstellt. Das Modellierungskonzept, welches von der Simulationssoftware *promodel* verfolgt wird, ist ein prozessorientiertes Modellierungskonzept. Dieses verbindet die beide Konzepte „Bausteinorientierung“ und „Sprachorientierung“. Die Simulationssoftware *promodel* basiert auf Bausteinen mit einem Minimum an Funktionalität. Zusätzlich können die Prozesse in und zwischen den Bausteinen mit einer Programmiersprache frei beschrieben werden (/PROM98a/, S. 87ff). Hierdurch erreicht das prozessorientierte Modellierungskonzept das für die

Abbildung von Beschaffungsketten erforderliche Maß an Freiheitsgraden bei einem vertretbaren Aufwand für die Modellierung.

Die Untersuchungen beziehen sich auf die Logistikkosten, welche in den letzten beiden Stufen einer Beschaffungskette verursacht werden. Als Beispiel dient die Beschaffung der Produkte eines Reifenherstellers durch den Handel. Ausgehend vom Kunden wird eine zweistufige Beschaffungskette untersucht. Dieses besteht aus drei regionalen Einzelhandelsverbänden, drei Regionallagern und zwei Produktionsstandorten. Eine detaillierte Beschreibung dieser Ausgangssituation erfolgt in Kapitel 3. Die Komplexität dieser Beschaffungskette zeigt sich unter anderem darin, dass sich aus Sicht des Handels die Beschaffung im Wesentlichen am Absatzmarkt orientiert und damit am Verhalten der Kunden (/MERT86/, S. 19). Der Bedarf kann daher verschiedenen Schwankungen, z.B. einer saisonalen Bedarfsschwankung oder einem regelmäßigen Trend, aber auch unberechenbaren Schwankungen unterliegen (/KLEI72/, S. 35ff).

1.2 Zielsetzung

Ziel dieser Untersuchung ist es, mit der Simulationssoftware *promodel* ein lauffähiges Modell einer beispielhaften Beschaffungskette zu erstellen. Das Modell soll dabei verschiedene denkbare Dispositionsstrategien für die Beschaffung beinhalten. Diese Strategien sollen für verschiedene Simulationsexperimente frei wählbar und beliebig kombinierbar sein. Eine weitere Anforderung, welche das zu erstellende Modell erfüllen muss, ist die Parametrierbarkeit verschiedener Einfluss- und Kostengrößen. Die Ergebnisdatenausgabe des Modells soll Rückschlüsse auf den mit dem jeweiligen Simulationslauf erzielten Kompromiss zwischen Kosten und Kundenorientierung (Lieferzeiten) zulassen.

Diese Modellierung und die durchzuführenden Simulationsexperimente bilden die Basis für eine abschließende Beurteilung der Simualtionssoftware *promodel* hinsichtlich ihrer Eignung zur Abbildung von Beschaffungsketten.

1.3 Aufbau der Untersuchung

Im Zentrum dieser Untersuchung steht das zu erstellende Simulationsmodell. Vorbereitend für die Modellierung erfolgt zunächst eine Beschreibung bzw. Analyse der verwendeten Simulationssoftware *promodel*.

In einem weiteren Schritt wird das zu modellierende System beschrieben. Hierbei handelt es sich um eine zweistufige Beschaffungskette, deren Ausgangspunkt für die Systemlast der Einzelhandel und damit der Kunde ist. Im Rahmen der Beschreibung des Systems werden die verschiedenen abzubildenden Prozesse aufgestellt. Ein weiterer Aspekt dieses Abschnitts der Studie ist die Aufstellung der im Modell umzusetzenden

Dispositionsstrategien für die Beschaffung, Herstellung und den Transport der Produkte.

Nach der Erstellung des Simulationsmodells werden mit diesem verschiedene Experimente durchgeführt. Hierfür werden in einem weiteren Schritt geeignete Experiment-Szenarien definiert. Diese Experimente basieren auf beispielhaften, nicht empirischen Daten, sollen aber geeignet sein, die Auswirkungen der Anwendung verschiedener Dispositionsstrategien auf die Zielgrößen Kosten und Kundenorientierung zu zeigen.

Nach Abschluss dieser Schritte erfolgt eine Beurteilung der Eignung der Simulationssoftware *promodel* zur Abbildung von Beschaffungsketten. In diese Beurteilung fließen die während der Modellierung gemachten Erfahrungen mit der Simulationssoftware ein. Ein weiterer wichtiger Beuteilungsaspekt sind die Schnittstellen zur Dateneingabe bzw. zur Ergebnisdatenausgabe, welche die Simulationssoftware zur Verfügung stellt. Hierbei soll zum einen der erforderliche Aufwand beurteilt werden, welcher bei der Aufbereitung empirische Daten für den Simulator entsteht. Zum anderen soll der Aufwand beurteilt werden, welcher für die Datenübernahme der vom Simulator bereitgestellten Ergebnisdaten in Standardsoftware wie z.B. MS-Excel erforderlich ist.

2 Grundlagen

Dieses Kapitel dient zur Erläuterung der im weiteren Verlauf der Studie verwendeten Hilfsmittel. Diese sind zum einen die Prozesskettendarstellung zur Abbildung der für das spätere Modell relevanten Prozesse in der untersuchten Beschaffungskette, sowie die Materialfluss-Simulation selbst. Den Abschluss dieses Kapitels bildet die detaillierte Beschreibung der verwendeten Simulations-Software *promodel*. Zunächst sollen jedoch verschiedene verwendete Begriffe erläutert werden:

2.1 Begriffsbestimmung

Im Folgenden werden die im weiteren Verlauf der Studie verwendeten Begriffe

- Logistik,
- Materialfluss,
- Kommissionierung,
- Beschaffung,
- Logistikkosten,
- Spedition sowie
- Disposition

anhand verschiedener Definitionen erläutert.

2.1.1 Logistik

Für den Begriff Logistik gibt es in der Literatur eine Vielzahl von Definitionen. Eine Definition ist, die Logistik als „die wissenschaftliche Lehre der Planung, Steuerung und Überwachung der Personen-, Material-, Energie- und Informationsflüsse in Systemen“ zu sehen (/JÜNE99/, S. 2).

Die weitere Betrachtung bezieht sich ausschließlich auf den Materialfluss und den hierfür erforderlichen Informationsfluss. Daher ist es sinnvoll, eine spezifischere Definition der Logistik zu verwenden. Im weiteren wird der Begriff Logistik als die „ganzheitliche und integrierte Planung, Steuerung, Durchführung und Kontrolle aller Güter- und dazugehörender Informationsströme von der Beschaffung der Roh- und Einsatzstoffe beim Lieferanten über die Produktion bis zur Verteilung der Erzeugnisse an den Kunden“ verstanden (/BULL94/, S. 3).

Das Ziel der Logistik ist es, Güter zur richtigen Zeit, in der benötigten Menge, Zusammensetzung und Qualität, kostengünstig und zuverlässig am richtigen Bedarfsort verfügbar zu machen (/JÜNE99/, S. 2f). Diese Ziele der Logistik sind in **Abbildung 1** dargestellt. Zur Zielerreichung bedient sich die Logistik einer ganzheitlichen Systembetrachtung, die durch Planung eine Abstimmung zwischen dem physischen

Materialfluss, dem benötigten Informationsfluss und dem Zusammenspiel der beteiligten Betriebsmittel und Personen ermöglicht.

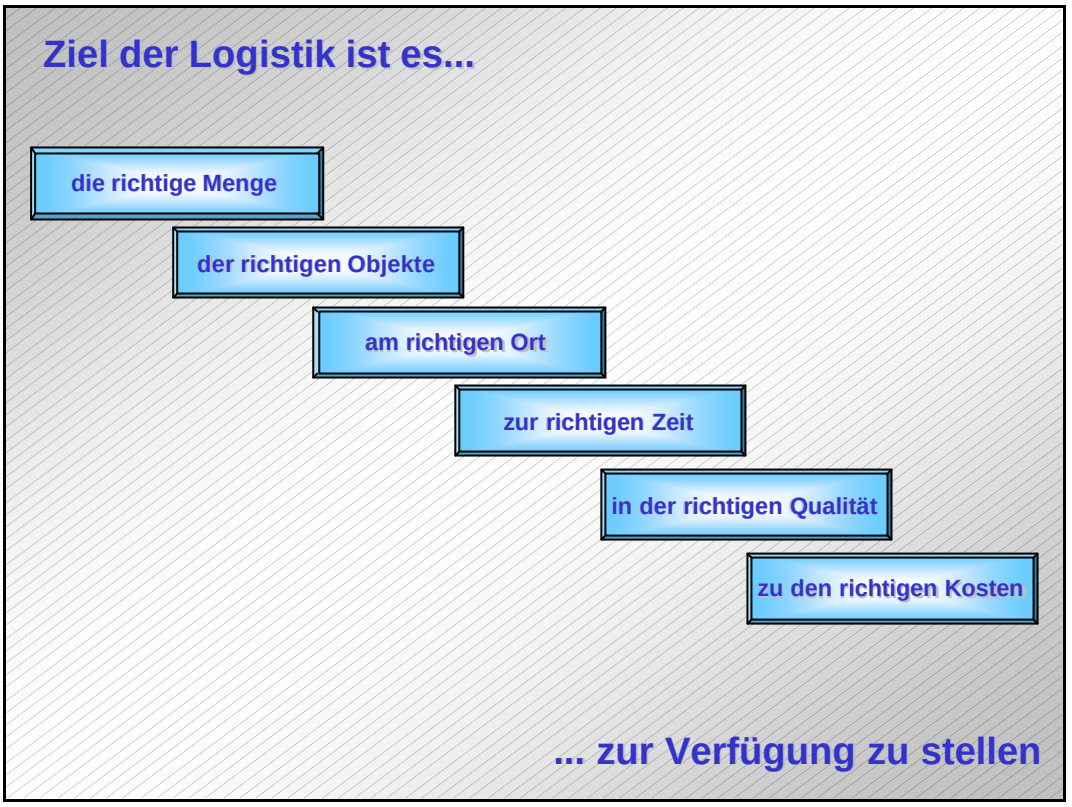


Abbildung 1: Aufgabe der Logistik (die 6R der Logistik), nach /JÜNE99/, S. 2f

Die Logistik kann sich auf vielfältige materielle und immaterielle Objekte beziehen. Wesentlich ist jedoch nur die Logistik physischer Güter und hier insbesondere die physischer Stückgüter. Stückgüter sind Objekte, die diskret im System auftreten.

Die Logistik als ganzheitliche Betrachtung basiert auf den drei Säulen Technik, Informatik sowie Betriebs- und Volkswirtschaft (/JÜNE89/, S. 10).

- Die Technik stellt die technischen Komponenten des Materialflusssystems dar.
- Die Informatik beinhaltet die Steuerung und Verarbeitung des Informationsflusses.
- Die Betriebs- und Volkswirtschaftslehre ermöglicht bei ganzheitlicher Anwendung der drei Säulen die Berücksichtigung von Kosten und Zeit bei der technisch-organisatorischen Umsetzung.

Diese drei Säulen der Logistik sind in **Abbildung 2** dargestellt.

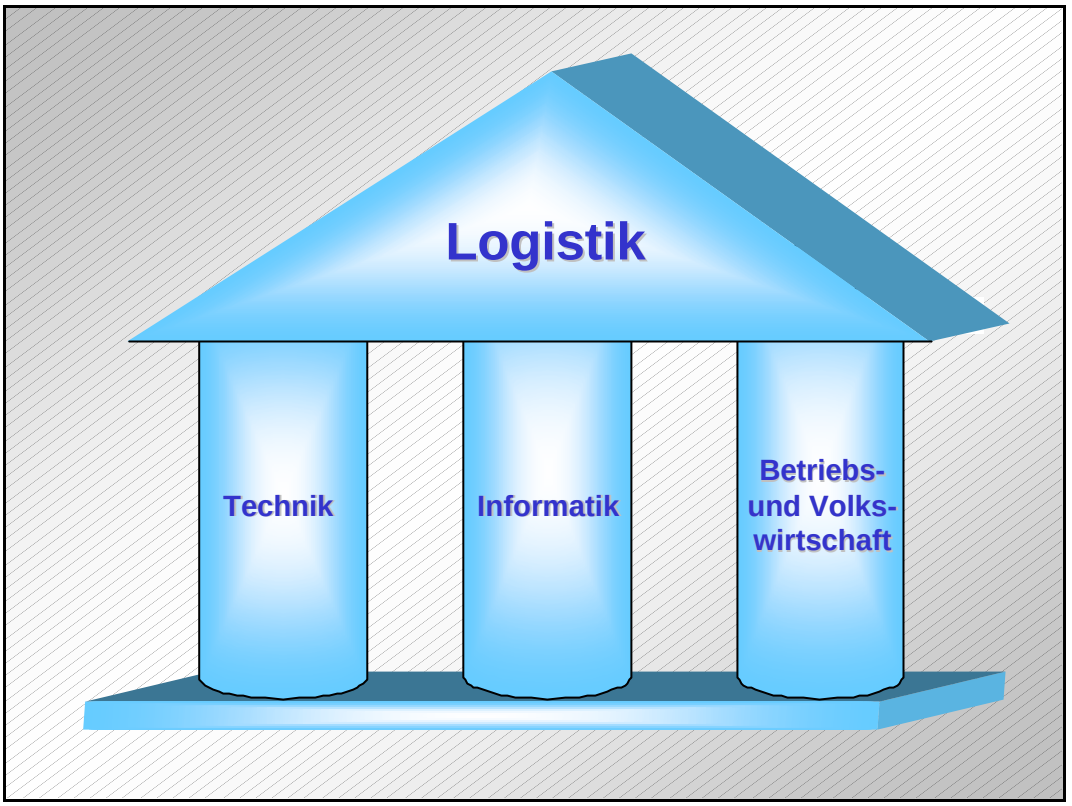


Abbildung 2: Die drei Säulen der Logistik, nach /JÜNE99/, S. 2

Aufgrund des großen Umfanges der Einsatzgebiete der Logistik ist es sinnvoll, die Gesamtfunktion Logistik hinsichtlich ihrer Schwerpunkte zu unterteilen. Hierbei ergeben sich drei wesentliche Einsatzgebiete für die Logistik innerhalb des Wertschöpfungsprozesses (/BULL94/, S. 4). Ein weiterer Schwerpunkt der Logistik liegt außerhalb des Wertschöpfungsprozesses, so dass die Logistik in vier Begriffe unterteilt werden kann:

- Beschaffungslogistik,
- Produktionslogistik,
- Distributionslogistik sowie
- Entsorgungslogistik.

Die Beschaffungslogistik ist der für diese Untersuchung wesentliche Teilbereich der Logistik. Sie befasst sich bezogen auf ein Unternehmen mit Gestaltung, Durchführung und Steuerung von Versorgungsprozessen mit betriebsfremden Bedarfsgütern. Bezogen auf den Handel gibt es nur betriebsfremde Bedarfsgüter, wobei die Bedarfsgüter hier die Endprodukte einer Wertschöpfungskette darstellen. Die Beschaffungslogistik beinhaltet Transformationsprozesse der Güter zwischen der Bereitstellung in einer Beschaffungsquelle (Warenausgang Zulieferer) und der Bereitstellung zur zweckbestimmten Verwendung im Unternehmen (Wareneingang) hinsichtlich Menge, Zusammensetzung, Ort und Zeit, die durch einen Informationsfluss zwischen dem Unternehmen, den Operatoren (z.B. Spedition) und dem Lieferanten gesteuert werden.

Die Beschaffungslogistik beinhaltet keine Materialflussprozesse im Unternehmen selbst. Sie endet mit der Bereitstellung zur Weiterverwendung (/BULL94/, S. 2ff; /JÜNE99/, S. 2ff).

Die Produktionslogistik umfasst die zielgerichtete, organisatorische und technische Gestaltung, Planung, Steuerung und Kontrolle des im Unternehmen anfallenden Materialflusses und des zugehörigen Informationsflusses (/BULL94/, S. 4).

Ziel der Produktionslogistik ist die Optimierung der innerbetrieblichen, auf das Material angewendeten, Transformationsprozesse (bezüglich Ort, Zeit, Zusammensetzung, Qualität, Gestalt und Wert) und der zugehörigen Informationsflüsse zwischen den Produktionsstellen (auch untereinander) und dem Ziel (Warenausgang).

Die Distributionslogistik befasst sich mit der Verteilung der erzeugten Güter an die Kunden. Sie beginnt am Warenausgang des Unternehmens und endet mit dem Wareneingang beim Kunden. Mittels Operatoren (z.B. Spedition) werden die erforderlichen logistischen Transformationsprozesse, die zur benötigten Änderung der Güter bezüglich Ort, Menge, Zusammensetzung und Zeit führen, auf die Güter angewendet.

2.1.2 Materialfluss

Der Materialfluss ist die Summe aller Transformationsprozesse, die angewendet auf physische Objekte zu einer Änderung hinsichtlich Ort, Zeit, Zusammensetzung, Menge und Qualität führen. Die Materialflussprozesse sind Bestandteil der logistischen Prozesse (/JÜNE99/, S. 3ff).

Dies bedeutet, dass der Materialfluss die Verkettung aller Vorgänge beim Gewinnen, Be- und Verarbeiten sowie der Verteilung von stofflichen Gütern innerhalb bestimmter, festgelegter Produktionsbereiche darstellt (/VDI70/; /VDI73/).

Die Kernprozesse des Materialflusses sind Transport-, Umschlag- und Lagerprozesse (/JÜNE99/, S. 3ff). Im Folgenden wird von Transformationsprozessen gesprochen, um neben den Transportvorgängen auch die Umschlags- und Lagerprozesse zu berücksichtigen.

Zu den Gegenständen im Materialfluss zählen die verschiedenen Objekte sowie die beteiligten physischen Operatoren. Objekte stellen in diesem Zusammenhang die physischen Güter dar, auf welche die Transformationsprozesse angewendet werden. Operatoren sind die technischen Einrichtungen oder Personen, welche die Transformationsprozesse durchführen. Die Kombination dieser „Gegenstände“ bildet ein Materialflusssystem, welches durch die Transformationsprozesse auch seinen Systemzustand bezüglich der oben genannten Eigenschaften ändert (/JÜNE99/, S. 4).

Der Materialfluss wird in 4 Stufen gegliedert (/BULL94/, S. 5ff):

- Materialfluss *erster Ordnung* beinhaltet alle Transformationsprozesse, die zwischen zwei Unternehmen auf die Objekte angewendet werden. Der Materialfluss der ersten Ordnung fällt in den Bereich der Beschaffungslogistik oder Distributionslogistik.
- Materialfluss *zweiter Ordnung* beinhaltet alle Transformationsprozesse, die auf dem Werksgelände einer Unternehmung zwischen verschiedenen Bereichen der Unternehmung auf die Objekte angewendet werden. Hierzu zählen z.B. Transporte vom Lager zum Montagesystem oder zwischen verschiedenen Montagesystemen.
- Materialfluss *dritter Ordnung* umfasst alle Transformationsprozesse, die zwischen Abteilungen einzelner Betriebsbereiche oder zwischen Betriebseinrichtungen innerhalb einer Abteilung auf die Objekte angewendet werden. Hierzu zählt z.B. die Weitergabe von Material von einem Arbeitsplatz zum nächsten.
- Materialfluss *vierter Ordnung* umfasst alle Transformationsprozesse, die an einem Arbeitsplatz auf die Objekte angewendet werden. Hierzu zählt die Versorgung mit bzw. die Annahme von Material am Arbeitsplatz und die Handhabung des Materials.

Die Materialflussarten zweiter, dritter und vierter Ordnung fallen in den Bereich der Produktionslogistik. Diese Materialflussarten werden als innerbetrieblicher Materialfluss bezeichnet. Für diese Studie sind demnach nur Materialflussvorgänge *erster Ordnung* relevant.

Materialflussvorgänge sind immer mit einem steuernden Informationsfluss verbunden, der zur Auslösung und Überwachung der Transformationsprozesse dient (/BULL94/, S. 7).

2.1.3 Kommissionierung

Kommissionierung hat laut Definition „das Ziel, aus einer Gesamtmenge von Gütern (Sortiment) Teilmengen aufgrund von Anforderungen (Aufträge) zusammenzustellen“ (/VDI94/).

Bekannt ist die Kommissionierung hauptsächlich aus ihrer Anwendung in Warenverteilzentren, deren Hauptaufgabe die Kommissionierung und Versendung von Waren aus einem großen Sortiment ist.

Die Kommissionierung erfolgt in einem Kommissioniersystem, welches dem Lager nachgeschaltet ist. Ein solches System besteht aus den nachfolgend genannten drei Elementen (/JÜNE99/, S. 211):

- Materialfluss,
- Informationsfluss sowie
- Organisation.

Die Organisation bestimmt hierbei die Abläufe des Material- und Informationsflusses.

Die weitere Betrachtung des Kommissionierens bezieht sich auf das Kommissionieren unter dem Einsatz von Menschen:

Die Gestaltung der Kommissioniersysteme kann auf zwei Arten erfolgen. Sie können entweder statisch oder dynamisch gestaltet sein (/SCHU99/, S. 114ff).

Ein statisches Kommissioniersystem bedeutet, dass sich der Kommissionierer zur Ware hin bewegt. Die Ware ist also statisch. In einem solchen System fallen für den Kommissionierer neben der Tätigkeit des Kommissionierens weitere Aufgaben an. Er muss für den Kommissionierauftrag klären, an welchem Lagerort sich das benötigte Material befindet und einen, nach Möglichkeit kurzen, Weg zu diesem Lagerort identifizieren. Danach muss er sich zu diesem Lagerort hin bewegen, das Lagerfach identifizieren und die benötigte Teilmenge entnehmen (/SCHU99/, S. 114ff).

Eine dynamische Ausführung eines Kommissioniersystems bedeutet, dass sich das Material zum Kommissionierer bewegt. In diesem Fall ist der Mensch das statische Element. Eine solche Organisationsform des Kommissioniersystems findet sich hauptsächlich in automatisierten Lagern (/SCHU99/, S. 114ff).

Tendenziell ist eine statische Kommissionierung vorteilhaft, wenn:

- sich ein Auftrag aus mehreren Teilen zusammensetzt, so dass mehrere Lagerorte angefahren werden müssen,
- es sich bei den benötigten Teilen um großvolumige Artikel handelt, so dass sich eine geringe Zugriffsdichte ergibt,
- eine Manipulation, d.h. eine Bewegung der Teile von Hand möglich ist sowie
- keine langen Entnahmezeiten am Lagerort benötigt werden.

Die Kommissionierung kann nun noch nach der Art der Auftragsabarbeitung in eine ein- und eine zweistufige Kommissionierung unterteilt werden.

Bei einer *einstufigen Kommissionierung* werden die benötigten Teile für jeden Auftrag einzeln aus dem Lager entnommen und am Übergabepunkt bereitgestellt. Hierdurch erhält das System eine hohe Flexibilität, die jedoch mit einem hohen Zeitbedarf für die Kommissionierung erkauft wird.

Eine *zweistufige Kommissionierung* ist dadurch gekennzeichnet, dass mehrere Aufträge zusammengefasst und die benötigten Teile gemeinsam aus dem Lager entnommen werden. Die Zuordnung zu den einzelnen Aufträgen erfolgt in einer zweiten Stufe am

Übergabeplatz. Diese Form der Kommissionierung bietet ein Rationalisierungspotential, da hier die zeitaufwendigen Prozesse der Bewegung zum Lagerort eingeschränkt werden.

2.1.4 Beschaffung

Für den Begriff „Beschaffung“ gibt es in der Literatur verschiedene Definitionen. In einer engen Sichtweise der Beschaffung umfasst sie alle Maßnahmen zur Erlangung der Verfügungsgewalt über Beschaffungsobjekte (/HAMM86/, S. 7). Diese Beschaffungsobjekte stellen für das betriebliche Handeln relevante Ressourcen dar (/PALU98/, S. 165). Die hierfür erforderlichen Materialflussoperationen werden bei dieser Auslegung der Logistik zugeordnet (/PION94/, S. 36).

Eine weiterreichende Definition der Beschaffung ordnet ihr neben der Erlangung der Verfügungsgewalt auch sämtliche hierfür erforderlichen Aktivitäten des Transports, der Lagerung und der Entsorgung zu (/MEYE86/, S. 18). In dieser Untersuchung wird, unter Ausnahme der Entsorgung, diese Definition der Beschaffung verwendet, damit bei der späteren Beurteilung der verwendeten Simulationssoftware für die Abbildung von Beschaffungsketten ein möglichst breites Anforderungsspektrum zugrunde liegen soll.

Zur Berücksichtigung der Kosten der Beschaffung kann die Hauptaufgabe der Beschaffung als die „wirtschaftliche Bereitstellung der Beschaffungsobjekte in der erforderlichen Qualität, zum günstigen Preis, in der ausreichenden Menge, zum richtigen Zeitpunkt, am nachgefragten Ort“ (/PION94/, S. 37) beschrieben werden. Hierbei meint die *wirtschaftliche* Bereitstellung eine Minimierung der anfallenden Beschaffungskosten. Eine kontinuierliche Optimierung der Beschaffung zeigt sich unter anderem in einer Schwerpunktsverlagerung der Beschaffung. Der Schwerpunkt der Beschaffung hat sich von der reinen Versorgung mit Gütern und Rohstoffen weg zu einer ganzheitlichen Integration der Beschaffung in die Unternehmensprozesse entwickelt (/WEID02/, S. 88). Im Rahmen dieses Supply Chain Management steht zunehmend der Kunde im Mittelpunkt des Beschaffungsprozesses (/WEID02/, S. 88). Ein weiteres Rationalisierungspotential der Beschaffung liegt in der Anwendung vernetzter DV-Systeme, dem Übergang zum so genannten E-Business. Hierbei werden die Lieferanten in ein Business-to-Business-Netzwerk integriert (/WEID02/, S. 88).

In der zugrunde liegenden Beschaffungskette wird von einer festen Händler-Lieferanten-Beziehung ausgegangen. Die Minimierung der Beschaffungskosten kann daher nur in den Kostenbereichen erfolgen, welche prinzipiell auch den Logistikkosten (vgl. Kapitel 2.1.5) zugerechnet werden können. Eine Minimierung der Beschaffungskosten z.B. durch Auswahl preisgünstiger Lieferanten kann in der untersuchten Beschaffungskette nicht erzielt werden.

Die Beschaffung gliedert sich in vier Phasen. Diese sind die Beschaffungsvorbereitung, die Beschaffungsanbahnung, der Beschaffungsabschluss und die Beschaffungsrealisation (/PION94/, S. 37ff). Die einzelnen Phasen der Beschaffung beinhalten dabei die folgenden Teilprozesse:

1. Beschaffungsvorbereitung:
    - Ermittlung des Beschaffungsbedarfs,
    - Spezifikation des Beschaffungsbedarfs.
2. Beschaffungsanbahnung:
    - Suche nach potentiellen Lieferanten,
    - Einholen von Angeboten,
    - Angebotsanalyse,
    - Lieferantenauswahl.
3. Beschaffungsabschluss
    - Vertragsverhandlungen,
    - Vertragsabschluss.
4. Beschaffungsrealisation
    - Überwachung der zeitlichen Vertragserfüllungen,
    - Raumüberbrückung zwischen dem Lieferanten und dem Empfänger,
    - Warenannahme,
    - Eingangslagerung.

Da in der dieser Studie zugrunde liegenden Beschaffungskette von einer festen Lieferanten-Empfänger Beziehung ausgegangen wird, sind nur die Phasen eins und vier der Beschaffung für die weiteren Untersuchungen relevant. Für die erste Phase der Beschaffung gilt im besonderen Fall des Handels, dass sich der Beschaffungsbedarf wesentlich am Absatzmarkt und damit am Verhalten der Kunden orientiert (/MERT86/, S. 19).

Mögliche Schwankungen, welchen der Beschaffungsbedarf unterliegen kann, sind gelegentliche (gleichmäßige) Schwankungen, eine saisonale Bedarfsschwankung, eine regelmäßiger, steigender oder fallender Bedarfstrend, aber auch unberechenbare Schwankungen (/KLEI72/, S. 35ff). Die sich für diese vier Schwankungsformen des Beschaffungsbedarfes ergebenden Bedarfslinien sind in der folgenden **Abbildung 3** dargestellt.
Die zu untersuchende Beschaffungskette bezieht sich auf die Beschaffung der Produkte eines Herstellers im Bereich des Reifenhandels. Hier ist für einzelne Produkte wie z.B. Sommer- und Winterreifen ein saisonal schwankender Beschaffungsbedarf anzunehmen.

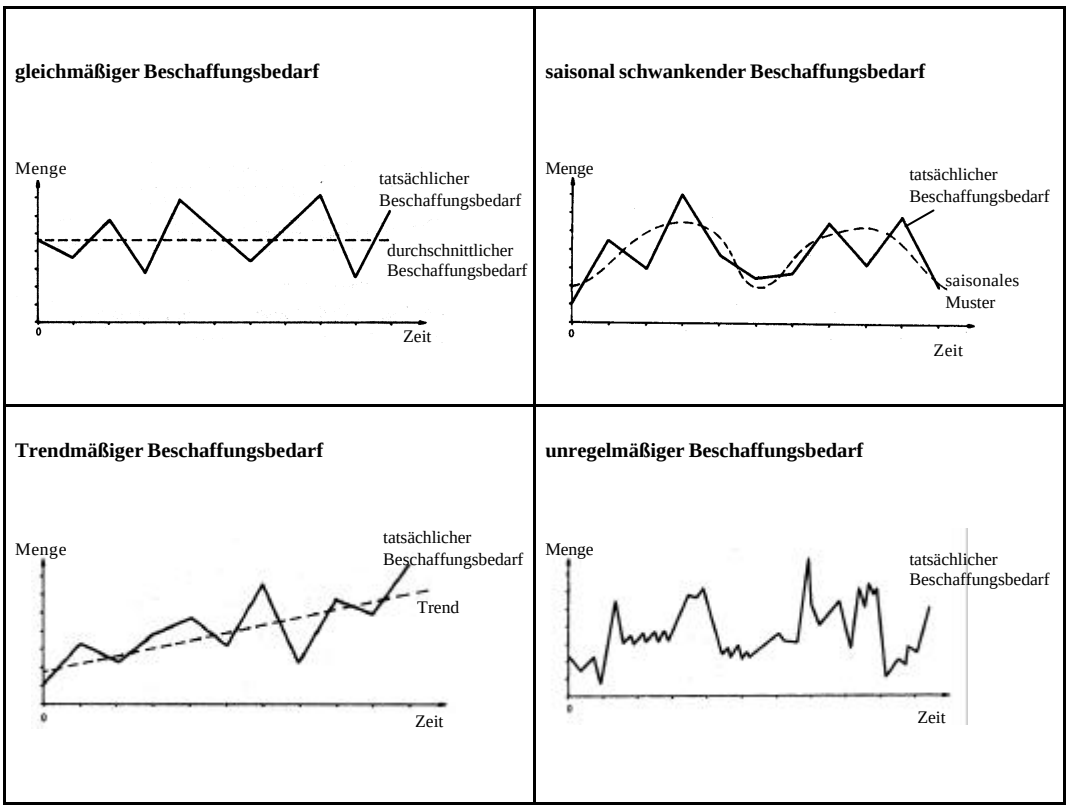


Abbildung 3: Bedarfslinien bei verschiedenen Bedarfsschwankungen, nach /KLEI72/, S. 35f

2.1.5 Logistikkosten

Der Umfang der Logistikkosten wird in der Praxis sehr unterschiedlich bewertet. Selbst in Unternehmen, welche in der selben Branche tätig sind, differiert der Kostenumfang, der den Logistikkosten zugerechnet wird, zum Teil erheblich. Ein Beispiel hierfür sind Zinsen und Abschreibungen auf vorhandene Bestände oder Kosten für externe Logistikdienstleister, welche von manchen Unternehmen nicht den Logistikkosten zugerechnet werden. Im anderen Extrem gibt es aber auch Unternehmen, welche selbst den Einkaufswert der beschafften Waren den Logistikkosten zurechnen (/GUDE00a/, S. 131).

Eine Übersicht der Bestandteile der Logistikkosten gibt die folgende **Abbildung 4**. Diese Abbildung zeigt das Ergebnis einer Umfrage zum Stellenwert der einzelnen Bestandteile an den Gesamtlogistikkosten.

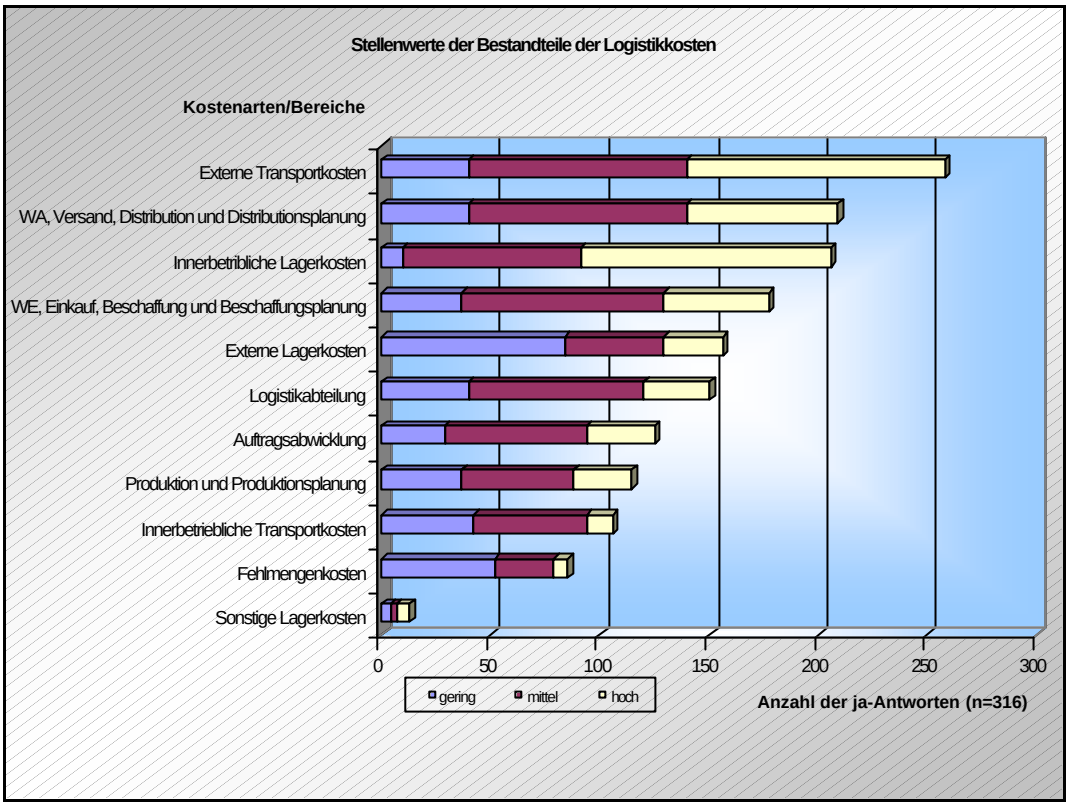


Abbildung 4: Bestandteile der Logistikkosten, nach /WEBE01/, S. 23

Sind die Logistikkosten einmal erfasst, so kann ihr Anteil an den Gesamtkosten eines Unternehmens als ein wichtiger Indikator für die Effizienz des Logistiksystems des Unternehmens angesehen werden (/WEBE02/, S. 101). Um über Unternehmensgrenzen hinweg vergleichbare Aussagen treffen zu können, ist es aber wichtig, eine gemeinsame Definition der Logistikkosten zu finden.

Eine Abgrenzung der Logistikkosten wirft jedoch verschiedene Probleme auf. Diese sind unter anderem darin begründet, dass die verschiedenen Definitionen der Logistik einen unterschiedlichen Leistungsumfang der Logistik zulassen (/WEBE02/, S. 135). Um die Komplexität der verschiedenen Logistikosten zu reduzieren, ist es sinnvoll, die Kosten in verschiedenen Kategorien zusammenzufassen. Diese Kategorien beziehen sich auf die beiden logistischen Hauptprozesse, den Transport und die Lagerung von Objekten. Sie beinhalten dabei jeweils fixe sowie variable Kostenanteile (/WEBE02/, S. 208).

2.1.5.1 Prozesse im Bereich der Transportkostenkategorie

Im Bereich der Transportkostenkategorie werden die Kosten angesetzt, welche entstehen, wenn Objekte im Raum bewegt werden. Ein solcher Transportprozess setzt sich aus fünf Teilprozessen zusammen (/WEBE02/, S. 208ff):

- Schaffung der Betriebsbereitschaft,
- Leerfahrt zur Objektaufnahme,

- Beladung,
- Nutzfahrt zum Objektabgabeort sowie
- Entladung.

Die Fahrt- und Ladeprozesse können noch weiter zusammengefasst werden. Die Inhalte der einzelnen Kategorien werden im Weiteren näher erläutert:

Schaffung der Betriebsbereitschaft

Die Schaffung der Betriebsbereitschaft eines Fördermittels steht am Anfang der Prozesse zur Erledigung eines Transportauftrags. Hiermit sind z.B. Rüstvorgänge zur Vorbereitung des Fördermittels auf die Objektaufnahme oder auch Betankung und Funktionsprüfung von Fördermitteln gemeint. Ein maßgeblicher Kostenfaktor in diesem Prozess sind die Personalkosten des bei der Durchführung der Prozesse gebundenen Personals (/WEBE02/, S. 208).

Nutz- und Leerfahrten

Ein für Transportprozesse typischer Fall ist, dass ladebereite Fördermittel zunächst an den Ort der Objektaufnahme verbracht werden müssen. Diese Ortsveränderung stellt den Leerfahrtanteil des Transportprozesses dar. Dieser Teilprozess ist aus Sicht der Kostenrechnung von der Nutzfahrt zu trennen, da die Leerfahrt zum Objektaufnahmeort keine Leistungserbringung darstellt, den verursachten Kosten somit keine erbrachten Leistungen gegenüberstehen (/WEBE02/, S. 209).

Die Leistungserbringung des Transportprozesses erfolgt während der Nutzfahrt vom Objektaufnahmeort hin zum Objektabgabeort (/WEBE02/, S. 209).

Im Rahmen dieser beiden Teilprozesse werden verschiedene Kosten verursacht. Wesentliche Kosten an dieser Stelle sind z.B. die Personalkosten für den Transportmitarbeiter, welcher das Fördermittel steuert, Treibstoff- und Energiekosten, nutzungsabhängige Instandhaltungskosten, eventuelle Straßenbenutzungsgebühren oder Grenzübertrittsgebühren (/WEBE02/, S. 211). Die Höhe dieser Kosten ist dabei abhängig von verschiedenen Einflussgrößen (z.B. Entfernung, etc.).

Be- und Entladen

Be- und Entladeprozesse können als eigenständige Transportprozesse aufgefasst werden. Zwar ist die absolute Ortsveränderung der Objekte bei diesen Prozessen zumeist sehr gering, dennoch setzen sie sich wiederum aus den oben aufgeführten Teilprozessen zusammen. Einzig die Prozesse des Be- und Entladens entfallen bei dieser Sichtweise. Die Be- und Entladeprozesse werden oft per Hand oder mit speziellen Fördermitteln (z.B. Krane etc.) durchgeführt (/WEBE02/, S. 212).

Die bei diesen Prozessen verursachten Kosten sind analog zu den oben bereits aufgeführten Kosten. Ihre Ausprägung ist hingegen oft abhängig von den Objekteigenschaften der Güter, wie z.B. Gewicht, Volumen oder Handlingseigenschaften (/WEBE02/, S. 212).

2.1.5.2 Prozesse im Bereich der Lagerkostenkategorie

In den Bereich der Lagerkostenkategorie fallen alle Teilprozesse, die direkt oder indirekt mit der Lagerung von Objekten in Zusammenhang stehen. Diese sind die Objektannahme, die Vorbereitung des Lagers, die Ein-, Um- und Auslagerung von Objekten, die Lagerung sowie die Lagernachbereitung und die Warenabgabe (/WEBE02/, S. 213ff). Die einzelnen Teilprozesse werden im Folgenden erläutert:

Objektannahme (Warenannahme)

Die Objektannahme bildet den ersten Teilprozess, welcher für die Lagerung von Objekten erforderlich ist. Die Teilprozesse, aus denen sich die Objektannahme zusammensetzt, können je nach Lager stark differieren. Neben dem Abladen der Objekte von einem Fördermittel können bei der Objektannahme weitere Teilprozesse, wie z.B. das Erfassen oder das Kontrollieren der Objekte, erforderlich sein. Als typische Teilprozesse hierfür sind z.B. das Wiegen oder Zählen der Objekte bzw. eine Sichtkontrolle oder eine Kontrolle von Lieferscheinen zu nennen. Zusätzlich können weitere Prozesse zur Entstapelung oder zum Entpacken der Objekte erforderlich sein. Die hier verursachten variablen Kosten sind hauptsächlich abhängig vom Volumen der Objekte. Der dominierende Kostenfaktor dieses Teilprozesses der Objektlagerung sind die Kosten des eingesetzten Personals (/WEBE02/, S. 213).

Lagervorbereitung

Auch die Prozesse der Lagervorbereitung variieren stark. Sie sind nicht nur abhängig vom jeweiligen Lager, sondern auch von Anforderungen der Objekte. Lagervorbereitungsprozesse sind z.B. das Umfüllen der Lagerobjekte in Lagerhilfsmittel, das Verpacken der Lagerobjekte oder das Anbringen von Schutzstoffen an die Lagerobjekte. Bei einer Lagerung in Tiefkühllagern ist auch das eventuell erforderliche Einfrieren der Lagerobjekte ein Teilprozess der Lagervorbereitung. Der dominierende Kostenfaktor ist auch hier das Personal. Ein weiterer wesentlicher Kostenfaktor können aber auch die verwendeten Schutzstoffe und Verpackungen sein (/WEBE02/, S. 213f).

Ein-, Um- und Auslagerung

Die Prozesse der Ein-, Um- und Auslagerung der Lagerobjekte können als kurze, lagerinterne Transportprozesse angesehen werden. Diese können entweder unter der

Zuhilfenahme von lagerinternen Transportmitteln (z.B. ein Regalbediengerät in einem automatischen Hochregallager) oder aber manuell erfolgen. Die Kostenaussagen sind ähnlich denen bei der Transportkostenkategorie. Die oben beschriebenen Transportprozesse finden alle lagerintern statt und sind hinsichtlich ihres Zweckes auf die Lagerung von Objekten ausgerichtet. Die anfallenden Kosten sind daher in die Lagerkostenkategorie einzuordnen (/WEBE02/, S. 214f).

Lagerung

Der Lagerungsprozess selbst dient der Zeitüberbrückung. Dieser Prozess bedarf zumeist keiner weiteren aktiv durchzuführenden Teilprozesse. Die Kosten dieses Teilprozesses werden maßgeblich durch von den Objekten losgelösten Ressourcen, wie z.B. dem Lagergebäude und den Lagereinrichtungen, verursacht. Diese Kosten stellen zumeist Fixkosten dar. Es entstehen aber auch variable Kosten, z.B. für die Lagerversicherung, durch das in den Lagerbeständen gebundene Kapital oder z.B. die Gewerbekapitalsteuer (/WEBE02/, S. 215f).

Lagernachbearbeitung und Objektabgabe (Warenausgabe)

Die Teilprozesse der Lagernachbearbeitung und der Objektabgabe sind analog zu denen der Objektannahme und der Lagervorbereitung. In dem nun betrachteten Fall ist jedoch die Ablaufreihenfolge entgegengesetzt der zu Beginn des Lagerprozesses. Wesentliche Teilprozesse der Lagernachbereitung und Objektabgabe sind das Kommissionieren, Verpacken und Verladen der Objekte (z.B. auf Förderhilfsmittel). Die Lagernachbereitung selbst umfasst z.B. das Rücklagern von nun leeren Lagerhilfsmitteln und die gegebenenfalls erforderliche Aktualisierung des Lagerverwaltungssystems. Die hier anfallenden wesentlichen Kosten setzen sich aus den zumeist dominierenden Personalkosten, sowie aus den Kosten für Verpackungsmaterial zusammen.

Die oben beschriebene Aufteilung der Logistikkosten in einzelne Kostenkategorien und die wesentlichen Prozesse innerhalb der beiden betrachteten Kostenkategorien sind in der folgenden **Abbildung 5** bildhaft dargestellt.

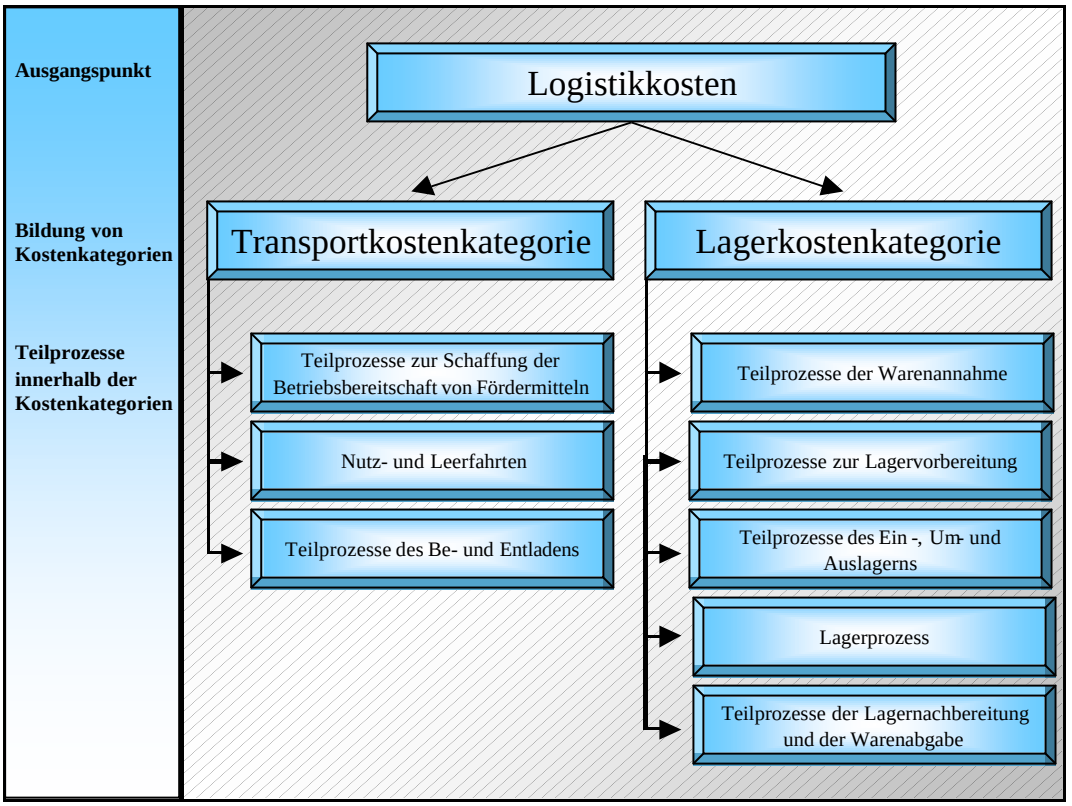


Abbildung 5: Zuordnung der Logistikkosten auf verdichtete Kostenkategorien

2.1.6 Spediteur

Eine Definition des Spediteurs findet sich im Handelsgesetzbuch. Ein Spediteur ist nach dieser Definition ein Kaufmann, der gewerbsmäßig Transportleistungen vermittelt, sie aber nicht selbst durchführt (/BISC95/, S. 22). Wird die Transportleistung, wie bei vielen Speditionen üblich, vom Speditionsunternehmen selbst mit eigenen Fahrzeugen vorgenommen, so wird das Unternehmen als Frachtführer tätig (/BISC95/, S. 22).

Die Aufgabe des Spediteurs ist demnach die Organisation einer Transportkette vom Absender der Waren bis hin zum Empfänger. Der Spediteur kombiniert dabei den Einsatz verschiedener Verkehrsträger, wie z.B. LKW, Bahn, Flugzeug, usw. so, dass unter Berücksichtigung des Leistungsvermögens der jeweiligen Verkehrsunternehmen ein durchgängiger und wirtschaftlicher Transportablauf entsteht (/BISC95/, S. 22).

Die Spedition als Glied der Logistikkette sieht sich heute neuen Herausforderungen gegenübergestellt. So entwickelt sich das Aufgabengebiet der Spedition von der reinen Vermittlung von Transportleistung weg. Die Speditionen übernehmen immer weiterreichende Aufgaben wie z.B. von Unternehmen ausgelagerte Wareneingangsaufgaben, Bündelung der Lieferanten eines Unternehmens bis hin zu Montageaufgaben. Die Spedition entwickelt sich somit immer mehr zu einem Logistikdienstleister (/AMBE93/, S. 6ff).

2.1.7 Disposition

Der Begriff Disposition bedeutet im hier verwendeten Sinn die Anordnung und Planung von Teilprozessen sowie die Verfügung über den Einsatz von Objekten (/DUDE90/, S. 192). Übertragen auf die untersuchte Beispiel-Beschaffungskette bedeutet die Disposition die Anordnung der Reihenfolge der zu bedienenden Aufträge sowie die Verfügung über den Einsatz der verwendeten Transportmittel.

Im Rahmen der operativen logistischen Tätigkeiten ist eine lang-, mittel-, und kurzfristige Planung zur fehlerfreien und wirtschaftlichen Durchführung der Tätigkeiten unerlässlich. Die Disposition bezeichnet dabei die kurzfristige operative Planung. Disponiert wird dabei der Transport, der Umschlag und die Lagerung von Gütern. Bezogen auf den Transport übernimmt die Disposition die Planung von Touren bzw. Routen und die Planung des Einsatzes sowie die Verfügung über die Transportmittel (/BUCH98/, S. 326ff).

2.2 Prozesskettendarstellung

Eine vollständige Prozesskette ist die Darstellung aller Tätigkeiten, die zur Abwicklung eines Auftrages durchgeführt werden müssen (/KUHN95/, S. 37). Dies bedeutet, dass die Prozesskettendarstellung immer auftragsorientiert erfolgt. Für die Analyse von Teilbereichen der Auftragsabwicklung ist es möglich, auch unvollständige Prozessketten zu bilden, welche nur einen Teil der erforderlichen Tätigkeiten abbilden. Eine solche unvollständige Prozesskette stellen auch die Prozessketten dar, welche in Kapitel 3 zur Verdeutlichung der zu modellierenden Prozesse in der betrachteten Beschaffungskette aufgestellt werden. Es handelt sich hierbei um indirekte Prozessketten, da sie nur Prozesse abbilden, welche nicht unmittelbar am wertschöpfenden Produktionsprozess beteiligt sind (/MIEH98/, S. 54).

Bei der Prozesskettendarstellung werden alle Material- und Informationsflüsse berücksichtigt (/KUHN95/, S. 37). Zur Aufstellung einer Prozesskette werden abgrenzbare Teilprozesse festgelegt und ihre logische Reihenfolge im Durchlauf der Informationen und des Materials durch ein Unternehmen festgelegt (/KUHN95/, S. 37). Diese einzelnen Prozesselemente werden nun mit ihren informations- und materialflusstechnischen Beziehungen über einer Zeitachse aufgetragen und bilden damit die Prozesskette (/KUHN95/, S. 47).

Jede Prozesskette verfügt über eine Quelle und eine Senke. Über die Quelle werden die Objekte generiert, welche die Prozesskette durchlaufen. Dies ist z.B. die Schnittstelle zwischen einem Spediteur und einem Zentrallager. Die generierten Elemente wären in diesem Fall die angelieferten und zu vereinnahmenden Produkte. Die Senke ist die

Schnittstelle der Prozesskette, durch welche die betrachteten Objekte die Kette wieder verlassen (/KUHN95/, S. 43). Neben der gesamten Prozesskette verfügt auch jedes Prozesskettenelement über eine Quelle und eine Senke. Über diese werden die Schnittstellen der einzelnen Prozesse abgebildet. Dadurch wird auch der Forderung nach einer Selbstähnlichkeit der Prozesskette Rechnung getragen, da jedes Element der Prozesskette durch eine weitere, verfeinerte Prozesskette ersetzt werden kann (/KUHN95/, S. 42).

Die einzelnen Prozesse selbst werden in vier Typen unterschieden. Es gibt den wertschöpfenden Prozess des Bearbeitens und die nicht wertschöpfenden Prozesse Prüfen, Transportieren und Puffern sowie das Lagern von Material (/KUHN95/, S. 43). Bei den später betrachteten Prozessketten sind nur die nicht wertschöpfenden Prozesse Transportieren bzw. Puffern sowie das Lagern und das ebenfalls nicht wertschöpfende Bearbeiten von Informationen relevant.

Jedes Prozesselement, und damit jeder definierte Prozess, besteht aus drei Bereichen:

- Der erste Bereich ist die Lenkungsebene, welche die Regel und Steuerungsvorschriften repräsentiert, nach welchen der Prozess abläuft (/KUHN95/, S. 44).
- Der zweite Bereich sind die Ressourcen. Hierzu zählen die logistischen Betriebsmittel Personal, Bestände, Flächen, Arbeits-, Hilfs- und Organisationsmittel, welcher sich jeder Prozess bedient (/KUHN95, S. 45ff).
- Der dritte Bereich wird aus den Strukturen gebildet. Dies sind die topologischen Gegebenheiten der Umgebung, in welcher der Prozess abläuft (/KUHN95/, S. 46).

Die Darstellung eines Prozesskettenelementes mit einer Aufschlüsselung seiner vier Inhaltsbereiche wird in **Abbildung 6** gezeigt.

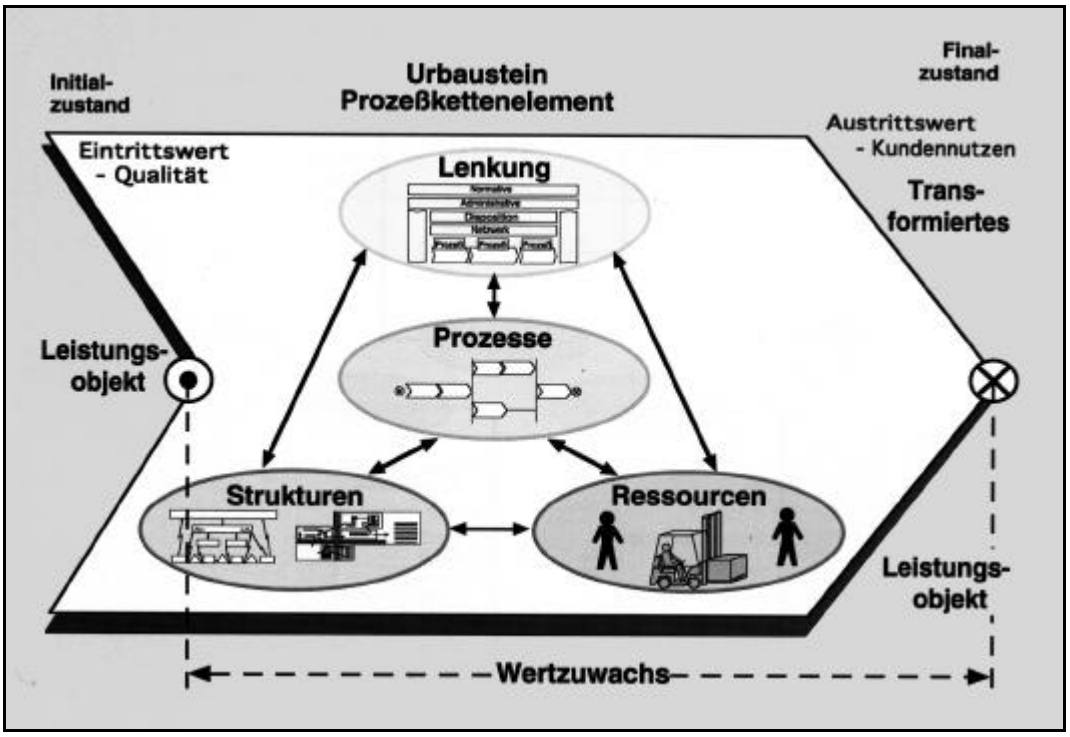


Abbildung 6: Darstellung eines Prozesskettenelementes mit Aufschlüsselung seiner Inhalte, aus /KUHN95/, S. 157

Zur Verdeutlichung der oben beschriebenen Prozesskettenmodellierung zeigt **Abbildung 7** die Darstellung einer beispielhaften Prozesskette. In dieser Abbildung lassen sich auch die Verbindungen zwischen den einzelnen Elementen der Prozesskette erkennen. Diese werden unterschieden in ablauflogische Verbindungen, Konnektoren und zeitliche Konnektoren.

Die ablauflogischen Verbindungen werden durch vertikale Verbindungen zwischen den Prozesselementen einer Unternehmensebene dargestellt. Sie symbolisieren auf der Materialflussebene den physischen Objektfluss zwischen zwei aufeinander folgenden Prozesselementen und auf der Informationsebene den materiellen oder immateriellen Informationsobjektfluss zwischen den jeweiligen aufeinander folgenden Prozesselementen.

Die Konnektoren werden durch eine vertikale durchgezogene Linie dargestellt. Sie symbolisieren eine Zusammenführung oder Verzweigung von Objektflüssen, stellen aber keinen Zeitverbrauch dar. Konnektoren werden gesetzt, wenn Prozesse parallel ausgeführt werden können oder wenn alternative Prozesse durchlaufen werden können.

Die zeitlichen Konnektoren werden durch eine gestrichelte Linie dargestellt. Sie kennzeichnen in senkrechter Richtung Bezugspunkte verschiedener paralleler Prozesskettenelemente. Diese werden von verschiedenen Objekten zeitgleich passiert.

Über die zeitlichen Konnektoren kann z.B. der Umstand abgebildet werden, dass eine Einlagerung von Produkten erst beginnen kann, wenn die Produkte erfasst wurden.

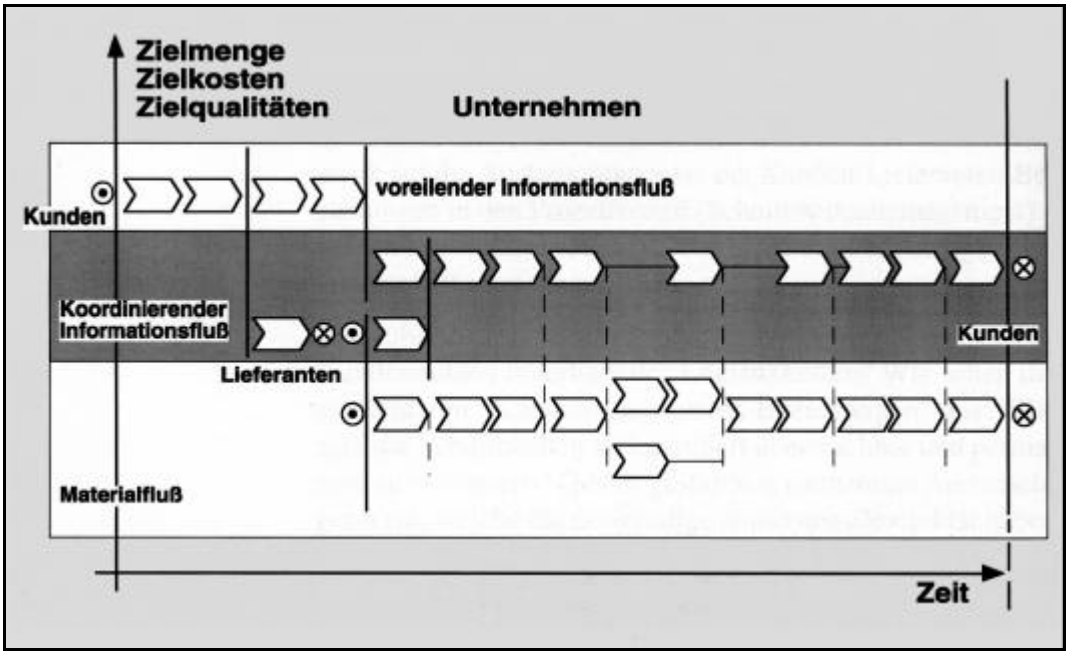


Abbildung 7: Darstellung einer beispielhaften Prozesskette, aus /KUHN95/, S. 16

2.3 Materialfluss-Simulation

Materialfluss-Systeme stellen komplexe technische Systeme dar. Die Untersuchung solcher Systeme ist mit mathematisch-analytischen Methoden oft nicht mehr möglich, da diese Verfahren hier an ihre Grenzen stoßen. Die Ursachen hierfür liegen in den vielen das komplexe System beschreibenden Systemgrößen begründet, da diese häufig auch voneinander abhängig sind (/WENZ00b/, S. 5). Ein weiterer Aspekt ist, dass viele Systemgrößen nicht kontinuierlich bzw. linear in ein solches System eingehen, sondern nur durch stochastische Verteilungen beschrieben werden können. Der Systembegriff erstreckt sich hierbei sowohl auf real bestehende Systeme, als auch auf geplante, also noch nicht physisch vorhanden Systeme.

2.3.1 Ablauf einer Simulationsstudie

Die Simulation solcher komplexer Systeme stellt ein Hilfsmittel dar, welches die Untersuchung mit einem wirtschaftlich vertretbaren Aufwand ermöglicht. Der Ablauf einer solchen Simulationsstudie zur Systemuntersuchung gliedert sich dabei in vier wesentliche Schritte:

Der *erste Schritt* einer Simulationsstudie ist die Analyse und Abstraktion des zu untersuchenden Systems (/WENZ00b/, S. 5). Die Analyse bezieht sich dabei auf die

räumlichen und vor allem die zeitlichen Abläufe und die im System vorhanden abhängigen Nebenläufe sowie auf die Relationen der Systemkomponenten untereinander. Ein einsetzbares Hilfsmittel zur Analyse eines Systems ist die in Kapitel 2.2 beschriebene Prozesskettendarstellung. Hierbei wird das System analysiert, indem alle später zu betrachtenden Prozesse aufgenommen werden. Ebenfalls zu diesem die Studie vorbereitenden Schritt gehört die Abstraktion des Systems. Diese ist erforderlich, da ein System niemals exakt in einem Modell abgebildet werden kann. Der Grad der Abstraktion, also die Detailgenauigkeit des späteren Modells richtet sich dabei nach den Untersuchungszielen. Der Abstraktionsgrad kann auch innerhalb eines Modells variieren. Das Ergebnis dieses ersten Schrittes ist die Datenbasis für das zu erstellende Modell. Diese Datenbasis mit ihren Informationen hinsichtlich der Struktur, des Verhaltens und der Kenngrößen des untersuchten Systems beeinflusst stark das spätere Modell. Dieses kann das System nur so gut wiedergeben, wie die zugrunde liegende Datenbasis das System beschreibt (/WENZ00/, S. 10).

Der *zweite Schritt* der Studie ist die Erstellung eines Simulationsmodells (/WENZ00b/, S. 5). Die Basis für die Modellerstellung bilden die im ersten Schritt aufgenommen bzw. ermittelten Daten. Häufig erkennt man erst bei der Modellierung eines Systems neuen Sichtweisen das System zu betrachten, so dass dieser Schritt eng mit der Analyse verknüpft ist. Hierdurch kann es auch während der Modellerstellung erforderlich werden, Teile des Systems erneut zu analysieren. Diese Phase der Simulationsstudie gliedert sich in vier Abschnitte:

- Zunächst ist es erforderlich, im Rahmen einer Konzeption des Modells, die vorhanden Systemkomponenten unter Berücksichtigung der Restriktionen der eingesetzten Simulationssoftware den zu erstellenden Modellkomponenten zuzuordnen (/WENZ00/, S. 10).
- Nach diesem Konzeptionierungsabschnitt erfolgt die eigentliche Modellierung des Systems. Hierbei wird unter dem Einsatz einer Simulationssoftware ein ablauf- und experimentierfähiges Softwaremodell des Systems erstellt (/WENZ00/, S. 10). Die Beschreibung dieses Softwaremodells ist dabei abhängig von der eingesetzten Simulationssoftware. Die verschiedenen Modellierungskonzepte werden in Kapitel 2.3.4 beschrieben.
- Das erstellte Modell muss in einem dritten Abschnitt der Modellierungsphase verifiziert werden. Die Verifizierung ist die Prüfung des Modells auf semantische bzw. syntaktische Fehler hinsichtlich der spezifischen Eigenschaften der verwendeten Simulationssoftware (/WENZ00/, S. 10). Dieser Abschnitt der Modellierungsphase wird häufig parallel zum Modellierungsabschnitt durchgeführt, wobei viele Programme zur Simulation den Modellierer bei der Verifizierung unterstützen, indem eine automatische Prüfung des Modells während des Modellierens durchgeführt wird.

- Den Abschluss der Modellierungsphase bildet die Validierung des Modells. Hierbei werden die errechneten Aussagen des Modells mit dem untersuchten System verglichen und nicht tolerierbare Abeichungen korrigiert (/WENZ00/, S. 10). Eine detaillierte Beschreibung des Validierungsabschnittes erfolgt in Kapitel 2.3.3. Das validierte experimentier- und ablauffähige Modell als Ergebnis dieses zweiten Schrittes der Simulationsstudie bildet die Basis für die zur Untersuchung des Systems durchzuführenden Versuche bzw. Experimente (/WENZ00/, S. 10).

Im *dritten Schritt* der Studie findet die eigentliche Systemuntersuchung statt. In diesem Schritt werden mit dem erstellten Simulationsmodell Experimente durchgeführt (/WENZ00b/, S. 5). Durch eine gezielte Variation von System- bzw. Modellparametern in verschiedenen Experimentläufen können Rückschlüsse auf das Systemverhalten gezogen werden. Das Ergebnis des dritten Schrittes sind zunächst Ergebnisse, welche das Verhalten des Simulationsmodells beschreiben. Da das Modell nur ein abstrahiertes Abbild des zu untersuchenden Systems darstellt, ist ein weiterer vierter Schritt erforderlich.

Der Inhalt dieses *vierten Schrittes* ist die Interpretation der formalen Ergebnisse der Simulationsexperimente. Die Interpretation erfolgt dabei dahingehend, dass die Ergebnisse unter Berücksichtigung der Abstraktion auf das zu untersuchende System übertragbar sind und Rückschlüsse für dieses System gezogen werden können (/WENZ00b/, S. 5). Diese Ergebnisinterpretation ist eng mit der Experimentdurchführung selbst verbunden, da hier auch die zu variierenden Modellparameter ermittelt werden.

Das Ergebnis einer solchen Simulationsstudie sind Erkenntnisse hinsichtlich des untersuchten Systems, welche das Verständnis dieses Systems erhöhen, bzw. sein mögliches Potential beschreiben, sowie Erkenntnisse über mögliche Schwachstellen des untersuchten Systems. Durch die Variation der Modellparameter bzw. durch eine Variation des Modells selbst können bei vorhanden Schwachstellen auch Lösungsansätze zur Behebung der Schwachstellen das Ergebnis der Simulationsstudie sein.

Der oben beschriebene typische Ablauf einer Simulationsstudie ist in der folgenden **Abbildung 8** dargestellt.

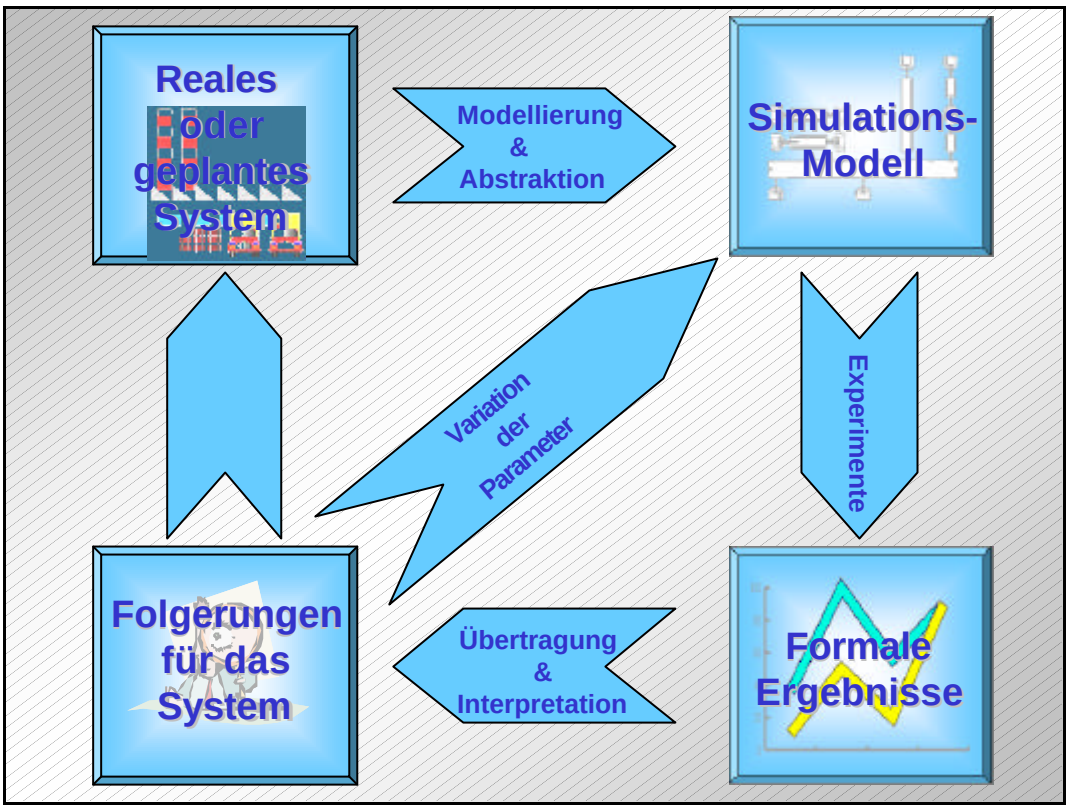


Abbildung 8: Ablauf einer Simulationsstudie, nach /ASIM97/, S. 3

2.3.2 Definition des Simulations- und Modellbegriffs

Der zuvor beschriebene Ablauf einer Simulationsstudie lässt sich direkt aus der Definition der Simulation ableiten. Der Verein Deutscher Ingenieure definiert Simulation wie folgt:

„Simulation ist das Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierfähigen Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind. Insbesondere werden die Prozesse über die Zeit entwickelt.“ (/VDI00/, S. 2)

In dieser Definition zeigt sich noch einmal, dass ein Hauptaugenmerk der Simulation auf der Untersuchung des zeitlichen Verhaltens eines Systems liegt. Ein weiteres Element der Definition ist die „Abbildung in einem experimentierfähigen Modell“. Um zu einem solchen experimentierfähigen Modell zu gelangen, ist es erforderlich die Aufgabenstellung der Simulationsstudie umfassend zu beschreiben (/WENZ00/, S. 6), da die Experimentierfähigkeit eines Modells eng mit der Aufgabenstellung verknüpft ist. Die in das Modell einzubringenden variierbaren Parameter und der Detaillierungsgrad des Modells sind dabei abhängig von der Aufgabenstellung.

Der Begriff Modell wird im Zusammenhang mit der Simulation in der VDI-Richtlinie 3633 wie folgt definiert:

„Ein Modell ist eine vereinfachte Nachbildung eines geplanten oder real existierenden Systems mit seinen Prozessen in einem anderen begrifflichen oder gegenständlichen System. Es unterscheidet sich hinsichtlich der untersuchungsrelevanten Eigenschaften nur innerhalb eines vom Untersuchungsziel abhängigen Toleranzrahmen vom Vorbild.“ (/VDI00/, S. 3)

2.3.3 Validierung eines Simulationsmodells

Während der Modellbildung ist es erforderlich, zu prüfen ob das Modell wie in der Modelldefinition gefordert, das zu untersuchende System innerhalb des definierten Toleranzrahmen beschreibt (/WENZ00/, S. 6). Diese Prüfung hinsichtlich des „Wahrheitsgehaltes“ des Modells ist die Modellvalidierung. Hinsichtlich von Modellparametern kann diese Validierung bereits während der Modellierung erfolgen, indem die Parameter verifiziert werden. Die ablauflogischen Eigenschaften eines Modells können erst validiert werden, wenn das Modell bereits zu einem experimentierfähigen Modell entwickelt wurde. Die Validierung dieser ablauflogischen Eigenschaften erfolgt mittels der Durchführung von ersten Validierungsexperimenten (/WENZ00/, S. 6). Bildet das Modell ein real existierendes System ab, so kann ein Validierungsexperiment auf der Grundlage real existierender, in der Vergangenheit ermittelter Daten erfolgen. Die Ergebnisse dieses Experiments können dann mit dem bekannten Verhalten des realen Systems verglichen werden. Bei nicht real existierenden geplanten Systemen besteht diese Möglichkeit der Validierung nicht. Eine erste Form der Validierung resultiert hier aus der Erfahrung des Validierenden, der die Ergebnisse mit denen ähnlicher Modell vergleichen kann, um einen ersten Anhaltspunkt über die Aussagekraft des modellierten Systems zu erhalten. Eine weitere Möglichkeit der Validierung bietet die Durchführung von Experimenten auf einer vereinfachten Datenbasis (z.B. ohne Berücksichtigung von stochastischen Einflüssen). Das Verhalten des Modells in Bezug auf diese vereinfachte Datenbasis kann parallel zu der Simulation mit mathematisch-analytischen Verfahren berechnet werden. Die Modellvalidierung kann dann unter Verwendung der ermittelten mathematisch-analytischen Ergebnisse durchgeführt werden.

Der Zeitpunkt, an welchem ein Modell ein experimentierfähiges Abbildungsstadium erreicht hat, differiert mit der eingesetzten Simulationssoftware. Einige Programme bieten die Möglichkeit bereits mit Teilsystemen Experimente durchzuführen, andere benötigen hierfür ein abgeschlossenes in sich konsistentes Modell. Je früher die Modellvalidierung begonnen werden kann, desto geringer ist der Anpassungsaufwand zur Bereinigung eventuell vorhandener Fehler im Modell. Die Simulationsprogramme, welche das Experimentieren mit Teilmodellen zulassen, bieten dem Anwender somit einen Vorteil gegenüber den Programmen, welche nur ein Experimentieren mit abgeschlossenen Modellen zulassen.

2.3.4 Simulationsmethoden und Modellierungskonzepte

Die Simulation basiert auf der Errechnung des Verhaltens von Systemen über der Zeit. Die Errechnung erfolgt dabei auf der Basis eines, das untersuchte System repräsentierenden, Softwaremodells. In diesem Modell ändert sich der Zustand der enthaltenden dynamischen Objekte während des Fortschreitens der Simulationszeit. Die Zustandsänderung der Modellobjekte über die Simulationszeit spiegelt dabei die Zustandsänderung der zugeordneten Systemobjekte während des Fortschreitens der (realen) Zeit im untersuchten System wieder. Die Zustandsänderungen der dynamischen Objekte sind dabei abhängig von den Wechselwirkungen aller Modellobjekte untereinander. Die Änderungen ergeben sich somit aus den Wechselwirkungen der dynamischen und statischen im Modell vorhandenen Objekte, sowie aus Wechselwirkungen der dynamischen Objekte untereinander (/WENZ00/, S. 7ff). Zur Errechnung des Systemverhaltens existieren zwei grundsätzliche Simulationsmethoden. Die Klassifizierung dieser Methoden ist in **Abbildung 9** dargestellt.

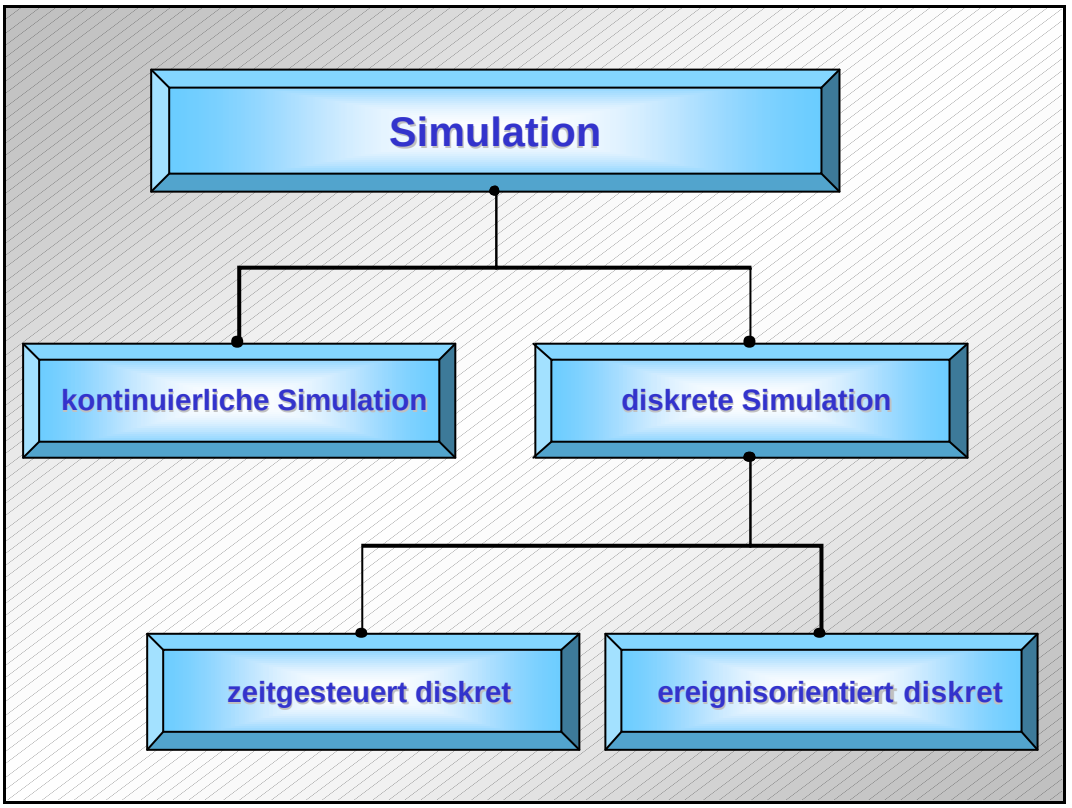


Abbildung 9: Klassifizierung der Simulationsmethoden, nach /WENZ00/, S. 8

Die beiden Methoden unterscheiden sich im Ansatz der Berechnung des Zeitverhaltens des Modells. Es existierten eine kontinuierliche Simulationsmethode und die diskrete Simulationsmethode. Die diskrete Methode kann weiter in eine zeitgesteuerte und eine ereignisgesteuerte Methode differenziert werden. Die einzelnen Methoden werden im Folgenden beschrieben:

Die *kontinuierliche Simulation* basiert auf dem Ansatz, dass sich die Objektzustände des Modells stetig ändern. Ist der Anfangszustand des Systems bekannt, so kann ein solches stetiges System durch miteinander gekoppelte Differentialgleichungen beschrieben werden. Das Lösen dieser Differentialgleichungssysteme ermöglicht die Systembeschreibung zu einem beliebigen Zeitpunkt. Diese Simulationsmethode wird häufig zur Simulation von Systemen des physikalisch-technischen Bereiches verwendet (/WENZ00/, S. 7f). Für die Materialflusssimulation hat diese Methode in der Praxis kaum Relevanz.

Die diskreten Simulationsmethoden sind dadurch gekennzeichnet, dass der Zustand des Modells nur zu diskreten Zeitpunkten berechnet wird. Hieraus resultiert, dass der Modellzustand zwischen zwei diskreten Zeitpunkten konstant ist und die Zustandsänderungen des Modells sprunghaft an den diskreten Zeitpunkten erfolgen (/WENZ00/, S. 8). Die diskreten Simulationsmethoden können in zeitgesteuerte und ereignisorientierte Methoden unterteilt werden.

Die *zeitgesteuerte diskrete Simulationsmethode* berechnet die Zustandsänderung des Modells im Abstand eines konstanten Zeitinkrements. Alle während der Periode eines Zeitinkrements aufgetretenen Zustandsänderungen des Modells werden dabei am Ende dieser Periode im Modell umgesetzt. Wird das zugrunde liegende Zeitinkrement genügend klein gewählt, so hat diese zeitgesteuerte Simulationsmethode einen quasi-kontinuierlichen Charakter. Mit zunehmender Reduzierung des Zeitinkrements steigt jedoch der für die Berechnung des Modells erforderliche Rechenaufwand (/WENZ00/, S. 8).

Die *ereignisorientierte diskrete Simulationsmethode* richtet den Zeitfortschritt der Simulation an dem Auftreten von Ereignissen, also am Auftreten von Zustandsänderungen im Modell aus. Die Berechnung der auftretenden Ereignisse erfolgt bei dieser Methode im Voraus, so dass beim Eintreten eines Ereignisses bereits feststeht, welches Ereignis als nächstes eintritt. Hierbei gilt, dass eine durch ein Anfangs- und Endereignis beschriebene Zustandsänderung im Modell Simulationszeit verbraucht, die Zustandsänderung aber keiner Rechenzeit bedarf. Umgekehrt benötigt die Ausführung der Zustandsänderung durch ein Ereignis Rechenzeit, verbraucht aber keine Simulationszeit (/WENZ00/, S. 8).

Das Beispiel eines Objektes auf einer Förderstrecke verdeutlicht den Unterschied der oben beschriebenen Simulationsmethoden:

Bei der kontinuierlichen Simulation wird dieses System durch die Bewegungsgleichungen des Objektes auf der Förderstrecke beschrieben. Die zeitgesteuerte diskrete Simulationsmethode berechnet zu festen Zeitpunkten die Position des Objektes auf der Förderstrecke, während die ereignisorientierte Simulationsmethode beim Eintrittsereignis des Objektes auf die Förderstrecke dessen Verweildauer auf der

Förderstrecke einmal berechnet. Zum Zeitpunkt der Summe aus der Simulationszeit beim Eintritt des Objektes und der errechneten Verweildauer auf der Förderstrecke wird das Austrittsereignis des Objektes aus der Förderstrecke in einer Ereignisliste des Simulationswerkzeuges eingetragen. Erst zu diesem Zeitpunkt wird der Systemzustand durch das Simulationswerkzeug aktualisiert. Die benötigte Rechenzeit ist bei dieser Simulationsmethode nur abhängig von der Anzahl der im Modell auftretenden Ereignisse, wohingegen sie bei der zeitgesteuerten Methode von der Dauer des gewählten Zeitinkrements abhängt und unabhängig von der Ereignisdichte ist.

Die folgende **Abbildung 10** verdeutlicht das oben beschriebene Beispiel:

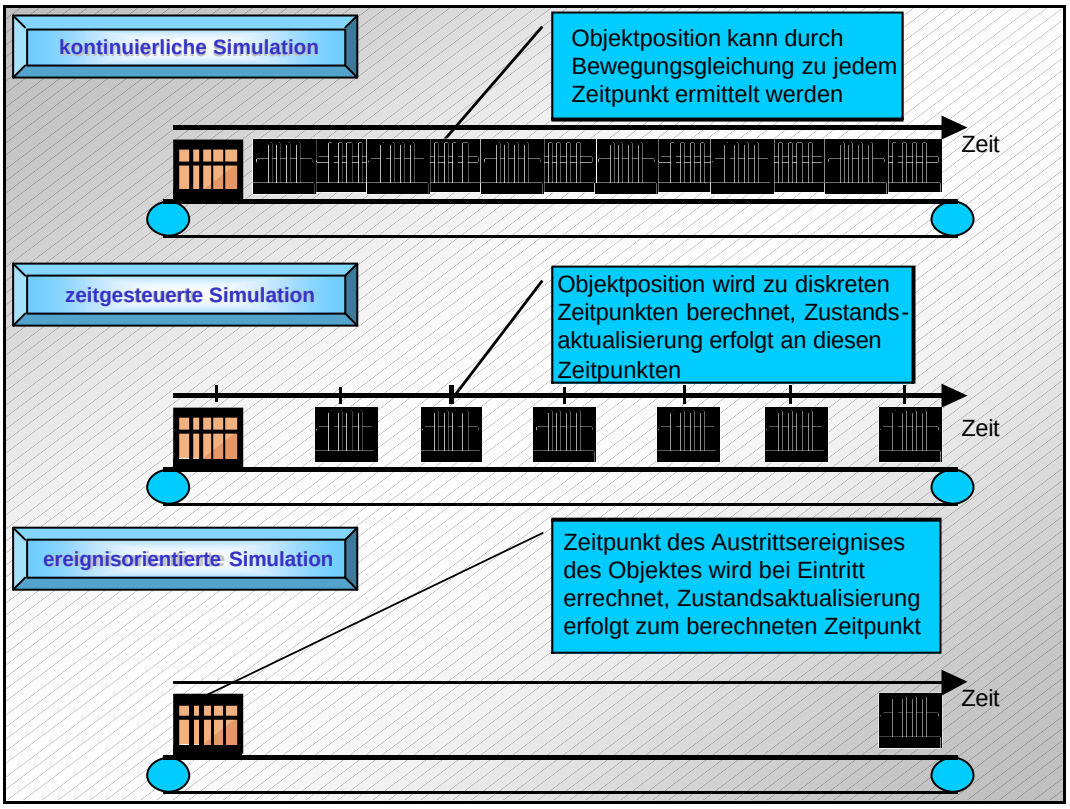


Abbildung 10: Verdeutlichung der Unterschiede der Simulationsmethoden am Beispiel eines Objektes auf einer Förderstrecke

Neben den oben erläuterten grundsätzlichen Simulationsmethoden verfolgen verschiedene Simulationswerkzeuge unterschiedliche Modellierungskonzepte zur Modellerstellung. Je nach Modellierungskonzept differiert auch der Abstraktionsgrad und der Anwendungsbezug der erstellten Modelle (/WENZ00/, S. 8). Im Folgenden werden fünf verbreitete Modellierungskonzepte vorgestellt:

Das *Sprachkonzept* stellt ein sehr abstraktes Modellierungskonzept dar (/NOCH93/. S. 267 ff). Bei diesem Modellierungskonzept wird das Modell durch eine einfache Programmiersprache beschrieben. Je nach Simulationswerkzeug kann die Modellbeschreibung auch auf der Basis einer um simulationsspezifische Befehle und Konzepte erweiterten Programmiersprache erfolgen (/WENZ00/, S. 8).

Eine weitere Modellierungskonzept-Kategorie bilden die *objektorientierten Modellierungskonzepte*. Hier erfolgt die Modellierung auf der Basis von gleichberechtigten Objekten die imstande sind miteinander zu kommunizieren. Die einzelnen Objekte sind wiederum Bestandteile verschiedener Objektklassen. Diese Modellierungskonzepte lehnen sich an die Philosophie der objektorientierten Programmierung an (/WENZ00/, S. 8).

Theoretische Modellierungskonzepte haben einen großen Anteil an den eingesetzten Modellierungskonzepten (/WENZ00/, S. 9). Diese Konzepte sind durch die ihnen zugrunde liegenden mathematischen Modelle geprägt. Sie basieren z.B. auf automatentheoretischen Konzepten, Petri-Netz Konzepten oder Warteschlangennetzen (/WENZ00/, S. 9).

Ebenfalls eingesetzt werden *anwendungsorientierte Modellierungskonzepte.* Als Beispiel eines solchen anwendungsorientierten Modellierungskonzeptes kann das listenorientierte Modellierungskonzept genannt werden (/NOCH93/. S. 267 ff). In diesem Modellierungskonzept werden Systeme auf der Basis applikationsspezifischer Listenstrukturen beschrieben. Der Einsatz solcher Modellierungskonzepte ist nur sinnvoll, wenn sehr abstrakte Strukturen modelliert werden (/WENZ00/, S. 9).

Ein eigenständiges Modellierungskonzept verfolgen *bausteinorientierte Simulatoren.* Die vorhanden Bausteine solcher Simulationswerkzeuge können je nach Simulationswerkzeug auf einem der oben beschriebenen Modellierungskonzepte basieren. Solche Simulationswerkzeuge haben den Vorteil, dass durch die vorhandenen Bausteine die für ein bestimmtes Anwendungsfeld einsetzbaren topologischen, organisatorischen und informatorischen Elemente durch das Simulationswerkzeug zur Verfügung gestellt werden. Hierdurch reduziert sich der Modellierungsaufwand, da die Funktionalität häufig verwendeter Elemente in Form der Bausteine bereits abgebildet wurde. Die Funktionalitäten dieser Bausteine müssen somit nicht mehr modelliert oder durch den Anwender hinterfragt werden (/WENZ00/, S. 9). Bieten bausteinorientierte Simulatoren nur einen festen Umfang an Bausteinen ohne die Möglichkeit in Form einer der zuvor beschriebenen Modellierungskonzepte neue Bausteine bzw. Elemente zu gestalten, so ist ihr Einsatzgebiet auf das durch die Bausteine repräsentierte Anwendungsfeld begrenzt. Treten im zu modellierenden System sehr spezifische Lösungen auf, so kann der Einsatz eines solchen Simulationswerkzeuges auch im eigenen Anwendungsfeld zu Problemen führen.

Eine Möglichkeit zur Klassifizierung von Simulationswerkzeugen hinsichtlich des zugrunde liegenden Modellierungskonzeptes und des Einsatzzieles ist in **Abbildung 11** dargestellt.

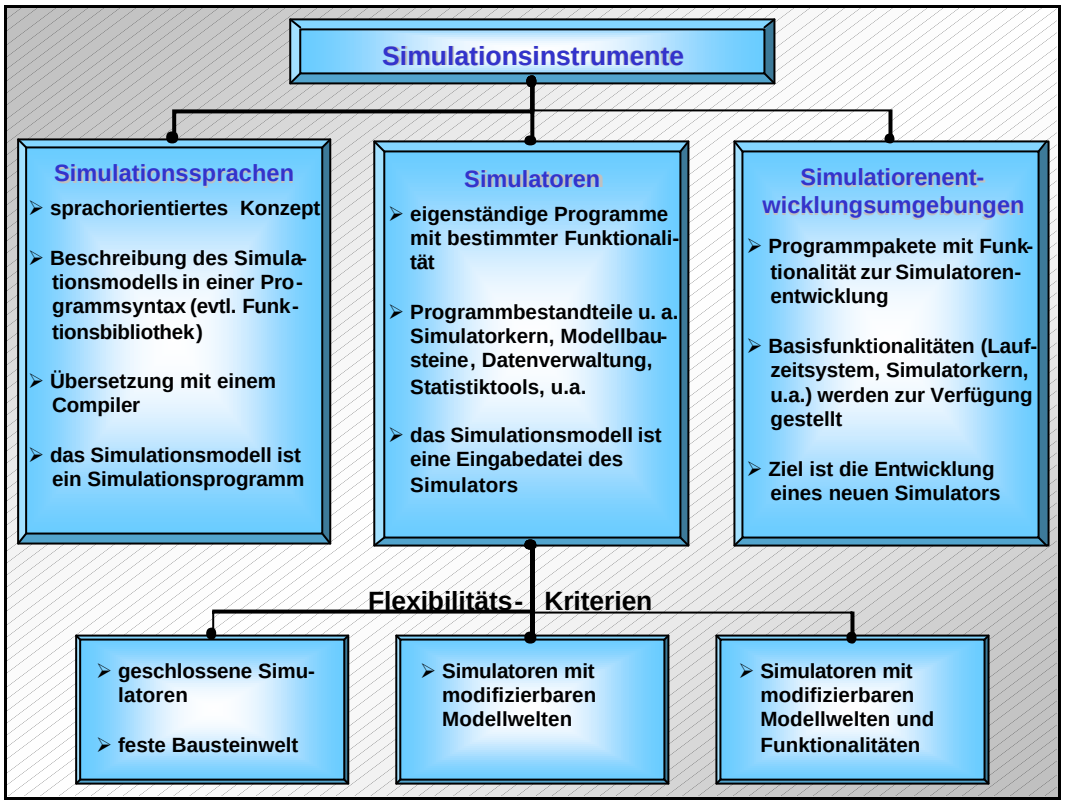


Abbildung 11: Klassifizierung von Simulationswerkzeugen, nach /HELL99/, Kap. 5, S. 8

2.4 Beschreibung des Simulationswerkzeuges *promodel*

Das Simulationswerkzeug *promodel* ist ein Programmpaket mit einem prozessorientierten Simulator. Der prozessorientierte Simulator verbindet das Konzept eines bausteinorientierten Simulators mit dem Sprachkonzept der Modellierung. Die Abbildung der Topologie eines Modells erfolgt mit Hilfe von Bausteinen, welche bei dem Simulationswerkzeug *promodel* als „Locations“ bezeichnet werden. Die Funktionalität dieser Bausteine in Bezug auf die Prozesse, welche auf die Objekte in und zwischen Bausteinen angewendet werden, muss mit Hilfe einer simulationsorientierten Programmiersprache modelliert werden. Die Bausteine bilden nur das Grundgerüst zur Modellierung der Prozesse. In Analogie zur Klassifizierung von Simulationswerkzeugen zeigt **Abbildung 12** die Einordnung des für die Modellerstellung verwendeten Simulationswerkzeuges *promodel* in die zuvor beschriebene Klassifizierung von Simulationswerkzeugen.

Im Folgenden wird ein kurzer Überblick über das Programmpaket des Simulationswerkzeuges *promodel* gegeben. Die einzelnen Komponenten werden im weiteren Verlauf dieses Abschnittes detailliert beschrieben.

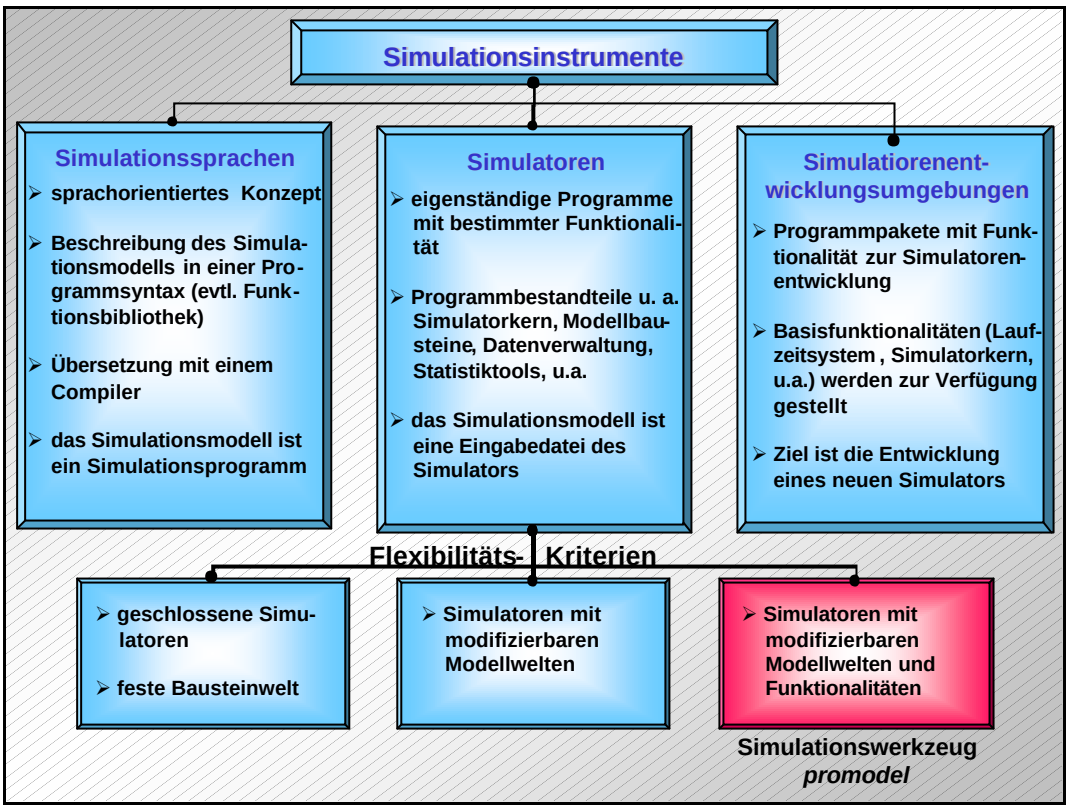


Abbildung 12: Einordnung des Simulationswerkzeuges *promodel*

Das Simulationswerkzeug *promodel* besteht aus einem Programmpaket verschiedener Programme und Hilfsmittel, welche den Anwender während der Modellerstellung, der Experimentdurchführung und der Auswertung der Ergebnisse unterstützen. Die Beschreibung der einzelnen Komponenten des Programmpaketes erfolgt auf Basis der bei der Modellierung gemachten Erfahrungen mit dem Simulationswerkzeug, sowie auf den Beschreibungen aus den Dokumentationen des Simulationswerkzeuges (/PROM98a/; /PROM98b/; /PROM98c/).

Der Modellierungsprozess wird durch die folgenden Werkzeuge unterstützt:

- *promodel*-Simulator,
- *promodel*-StatFit sowie
- *promodel*-Graphic Editor.

Der *promodel*-Simulator umfasst die Modellierungsumgebung sowie den Simulatorkern für die Durchführung von Experimenten. Für die Modellierung eines Systems stellt der Simulator verschiedene Elemente zur Verfügung, welche in Kapitel 2.4.1 näher beschrieben werden.

Das Hilfsprogramm *promodel*-StatFit unterstützt den Anwender dabei, empirisch ermittelte Daten in eine im Simulationsmodell verwendete Verteilungsfunktion zu überführen. Dieses Hilfsprogramm wird in Kapitel 2.4.1 beschrieben.

Mittels des *promodel*-Graphic Editors kann der Anwender beliebige Symbole für die im Modell verwendeten statischen und dynamischen Objekte erstellen. Hierdurch ist der Anwender in der Lage den Wiedererkennungswert des Modells in Bezug auf das abgebildete System zu erhöhen, indem z.B. Zeichnungen der Objekte des realen Systems für die Objektvisualisierung verwendet werden. Eine detaillierte Beschreibung dieser Komponente erfolgt in Kapitel 2.4.3.

Zur Unterstützung der Experimentdurchführung bietet das Simulationswerkzeug *promodel* das Hilfsprogramm *promodel*-SimRunner. Mittels dieses Hilfsprogramms können Experimentläufe automatisiert durchgeführt werden, wobei durch das Programm eine automatische Parametervariation zur Optimierung der Simulationsergebnisse in Bezug auf ein wählbares Zielsystem erfolgt. Dieses Hilfsprogramm wird in Kapitel 2.4.4 vorgestellt.

Die Datenauswertung wird durch das Hilfsprogramm *promodel*-OutputReport unterstützt. Mittels dieses Hilfsprogramms werden die Ergebnisdaten für den Anwender aufbereitet und dargestellt. Weitere Informationen zum *promodel*-OutputReport finden sich in Kapitel 2.4.5.

Der Umfang des Programmpaketes des Simulationswerkzeuges *promodel*, sowie die Relationen der einzelnen Programme untereinander sind in Abbildung 13 dargestellt.

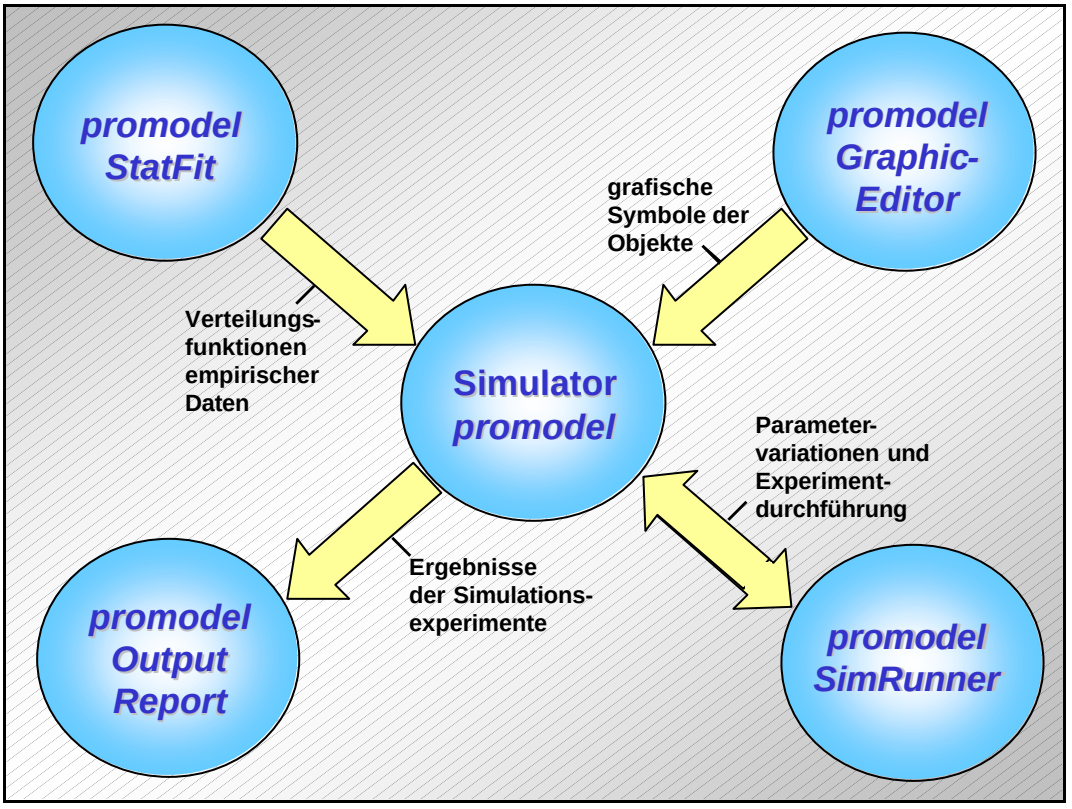


Abbildung 13: Komponenten und Relationen des Programmpaketes *promodel*

2.4.1 Vorstellung der Komponente *promodel*-Simulator

Der *promodel*-Simulator ist die zentrale Komponente des *promodel*-Programmpakets. Dieser stellt die Umgebung zur Modellierung der untersuchten Systeme zur Verfügung. Ebenfalls zum Simulator zählt der Simulator-Kern, durch den die Berechnung der einzelnen Experimentläufe mit einem Modell erfolgt.

Zur Modellierung stellt der Simulator dem Anwender verschiedene Modellierungselemente zur Verfügung, mittels welcher das Modell eines Systems abgebildet werden kann. Der Aufbau des Simulators mit seinen einzelnen Elementen ist in **Abbildung 14** dargestellt. Die einzelnen Modellierungselemente werden im Folgenden beschrieben.

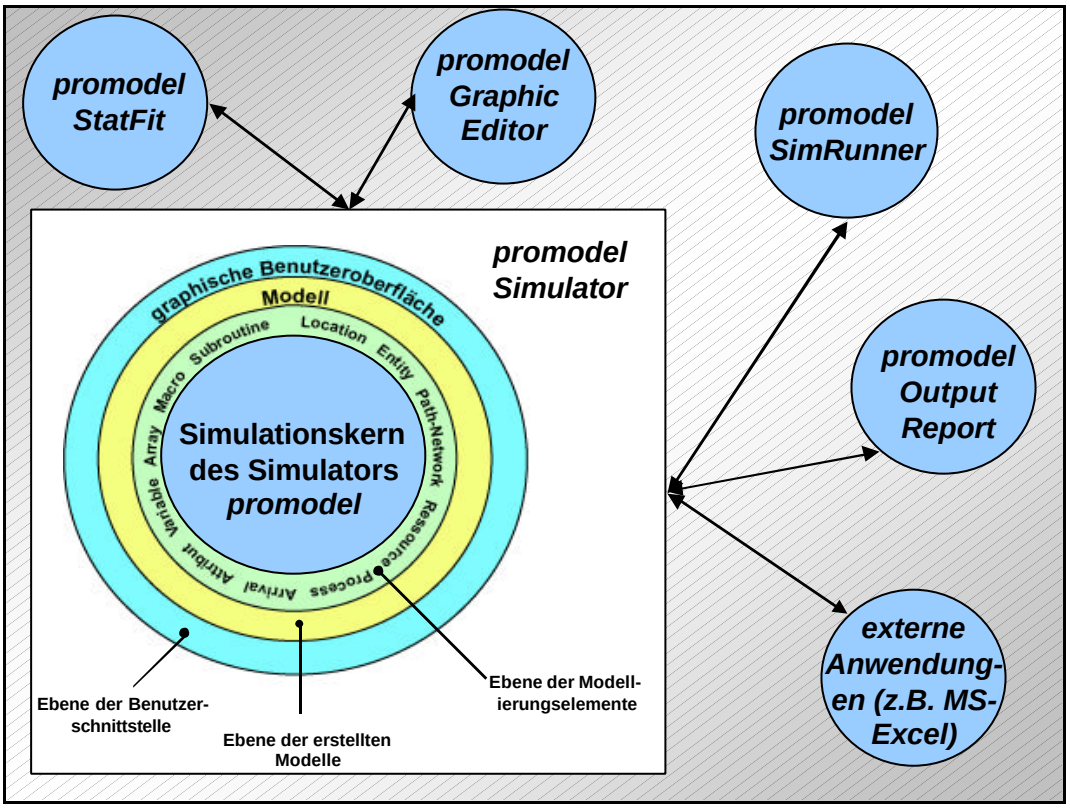


Abbildung 14: Modellierungselemente des *promodel*-Simulators

Die Modellerstellung erfolgt bei dem Simulationswerkzeug *promodel*-Simulator in einem zweistufigen Prozess. Zunächst werden von den durch den Simulator zur Verfügung gestellten Modellierungselementen verschiedene Instanzen dieser erstellt und parametriert. Eine Instanz zeichnet sich dadurch aus, dass alle Objekte des späteren Modells, welche aus derselben Instanz stammen, die Parameter dieser Instanz annehmen. Die eigentliche Modellerstellung erfolgt mit den aus den Instanzen erstellten Objekten („Units“).

2.4.1.1 Benutzeroberfläche des *promodel*-Simulators

Die graphische Benutzeroberfläche ist die Schnittstelle zwischen dem Anwender und dem Simulator. Die Benutzeroberfläche des *promodel*-Simulators orientiert sich am

Microsoft-Windows Standard. Das Simulationsmodell sowie die für die Bearbeitung der verschiedenen Modellelemente erforderlichen Benutzeroberflächen werden in eigenen Fenstern im Hauptfenster der Simulator-Anwendung dargestellt. Die Navigation in der Simulator-Anwendung erfolgt durch die Menüleiste des Hauptfensters. **Abbildung 15** zeigt das Hauptfenster der Simulator-Anwendung mit einem geöffneten Modell und der Nutzung der Menüleiste.

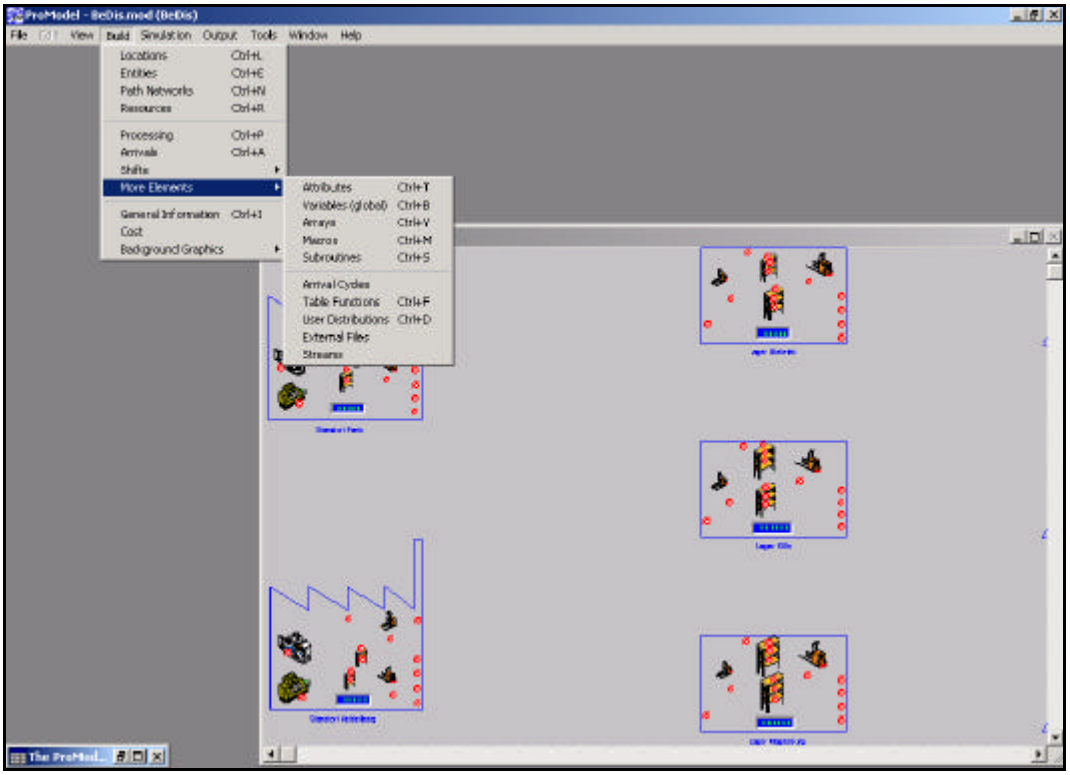


Abbildung 15: Benutzeroberfläche des *promodel*-Simulators

Die Bearbeitung der einzelnen Modellierungselemente erfolgt durch die Auswahl des Menüpunktes „Build“. Durch das unter diesem Punkt integrierte Menü können alle Modellierungselemente ausgewählt werden.

Für die Bearbeitung der Modellierungselemente werden zwei weitere Arbeitsfenster geöffnet. Diese Fenster werden automatisch um das Modellfenster herum angeordnet. In diesen beiden Fenstern werden die Funktionen zur Bearbeitung der Modellierungselemente sowie eine Übersicht der bereits erstellten Instanzen des aktuellen Modellierungselements dargestellt.

2.4.1.2 Modellierungselement „Location“

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 181ff).

Das Modellierungselement „Location“ repräsentiert die Bausteine des *promodel*-Simulators. Durch dieses Modellierungselement kann die Topologie eines Systems abgebildet werden. Die einzelnen „Locations“ besitzen jedoch im Gegensatz zu den meisten anderen verfügbaren bausteinorientierten Simulatoren keine integrierte Funktionalität. Sie bilden das Grundgerüst zur Erstellung von frei definierbaren Funktionalitäten. Diese werden durch die Definition von Prozessen in den Modellierungselementen „Location“ sowie zwischen den Elementen dieses Typs abgebildet. Bereits in das Grundgerüst integriert sind verschiedene Attribute der „Locations“ wie z.B. die Kapazität. Des weiteren verfügt jede Instanz dieses Modellelements bereits über die Fähigkeit Statistikdaten aufzunehmen.

Die folgende **Abbildung 16** zeigt die Arbeitsfenster zur Erstellung des Modellierungselements „Location“.

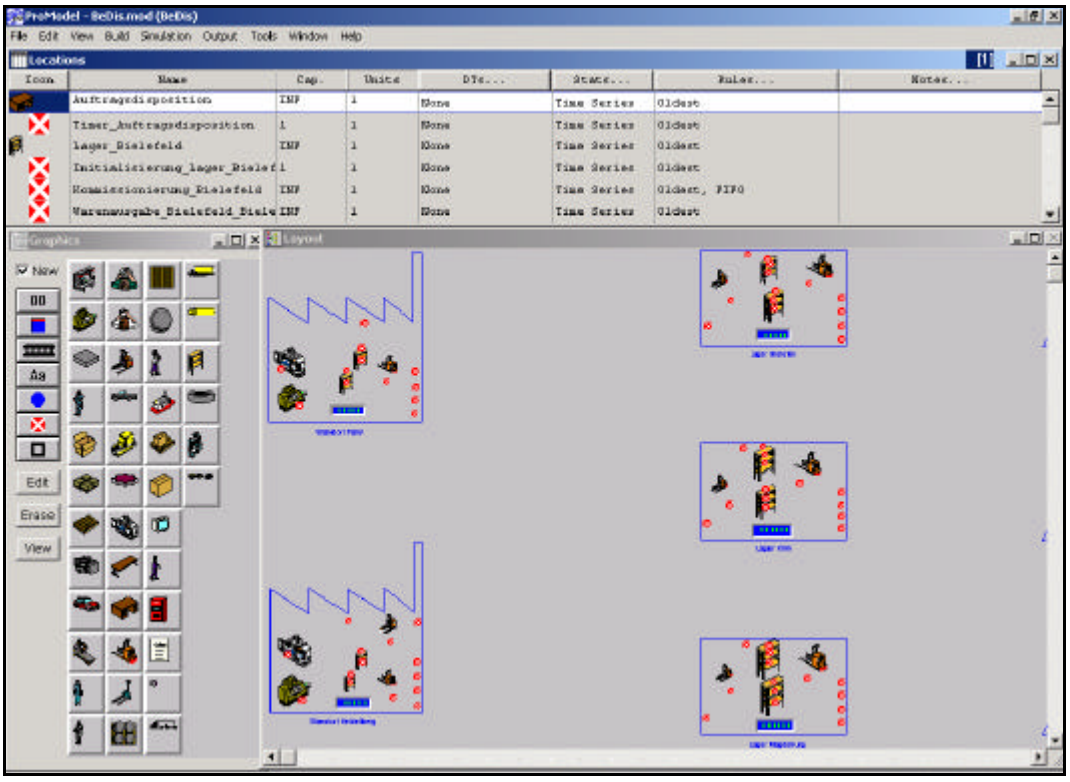


Abbildung 16: Erstellung des Modellierungselements „Location“

Bei der Erstellung einer Instanz des Modellierungselements „Location“ kann für jede Instanz eine beliebige Grafik zur Symbolisierung der „Location“ definiert werden. Diese Grafiken können aus einer Bibliothek ausgewählt oder mittels des Hilfsprogramms *promodel*-Graphic Editor erstellt werden.

Mit den einzelnen Instanzen können Statistikelemente wie z.B. „Counter“ oder „Statuslichter“ verknüpft werden. Durch diese kann der Zustand einer „Location“ visualisiert werden. Für die spätere Animation des Objektflusses können in den einzelnen „Locations“ Animationspunkte gesetzt werden. An diesen „Spots“ genannten Animationspunkten wird bei der späteren Visualisierung bzw. Animation der

Simulation ein sich in der „Location“ befindendes dynamisches Objekt, ein so genanntes „Entity“ (vgl. Kapitel 2.4.1.3), dargestellt.

Im Folgenden erfolgt eine Übersicht über die in das Modellierungselement „Location“ integrierten Attribute und deren Bedeutung:

- „Capacity“: Die Kapazität gibt an, wie viele dynamische Objekte („Entities“) sich gleichzeitig in einer „Location“ befinden dürfen bzw. können.
- „Units“: Die Anzahl der „Units“ gibt an, wie oft sich die beschriebene Instanz des Modellierungselements „Location“ im Modell befindet.
- „DTs“: Über die „Downtimes“ können Störungen und bzw. oder Pausen bezogen auf die aktuelle Instanz modelliert werden. Es können mehrere „Downtimes“ für eine Instanz des Elements „Location“ parametriert werden. Diese können durch die Definition einer Priorität gewichtet werden. Für jede „Downtime“ kann eine eigener Prozess (z.B. das Anfordern eines Maschinisten) definiert werden.
- „Stats“: Die Statistikeinstellung dient dazu festzulegen, wann und in welcher Form Statistikdaten für die Instanz des Elements „Location“ gesammelt werden.
- „Rules“: Die hier definierten Regeln dienen der Festlegung des Ein- und Austrittsverhaltens dynamischer Objekte in die bzw. aus der „Location“ (z.B. FiFo).
- „Notes“: Dieses Attribut bietet die Möglichkeit, Kommentare für die einzelnen Instanzen zu erstellen. Hierdurch kann der Anwender die Übersichtlichkeit des Modells steigern.

Ebenfalls zum Modellierungselement „Location“ zählt der einzige Baustein, der eine vordefinierte Funktionalität besitzt. Dieser Baustein „Conveyor“ repräsentiert ein Förderband bzw. eine Staustrecke. Das Verhalten des Bausteins, sowie weitere Parameter können über eine vordefinierte Parametermaske parametriert werden. Diese Parametermaske ist in **Abbildung 17** dargestellt.

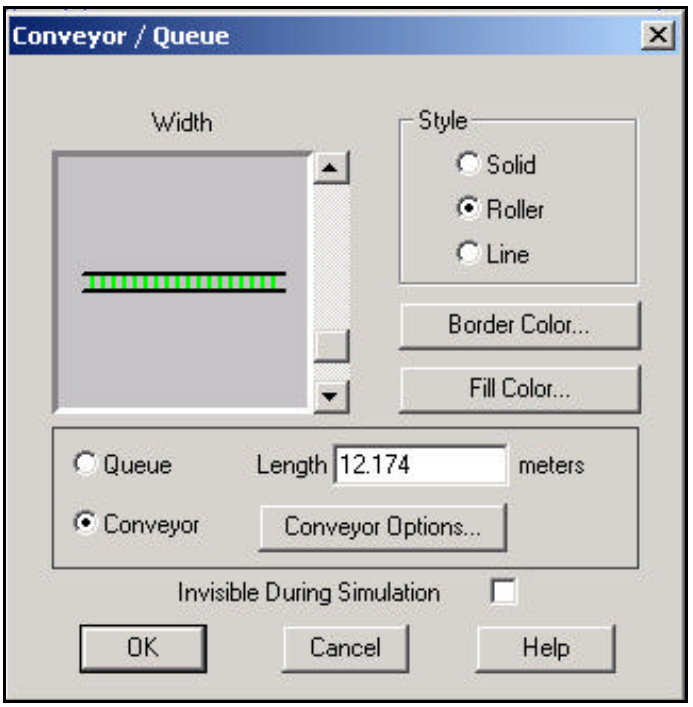


Abbildung 17: Parametermaske des Bausteins „Conveyor“

2.4.1.3 Modellierungselement „Entity”

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 217ff).

Das Modellierungselement „Entity“ repräsentiert die dynamischen Objekte des modellierten Systems. Diese Objekte durchlaufen während der Simulation die verschiedenen Instanzen des Modellierungselements „Location“. Dabei initialisieren sie die Ausführung der für die „Locations“ definierten Prozesse. Zur Erstellung von Instanzen dieses Modellierungselements muss der Menüpunkt „Build / Entities“ ausgewählt werden. Die zur Erstellung angezeigten Arbeitsfenster sind in **Abbildung 18** dargestellt.

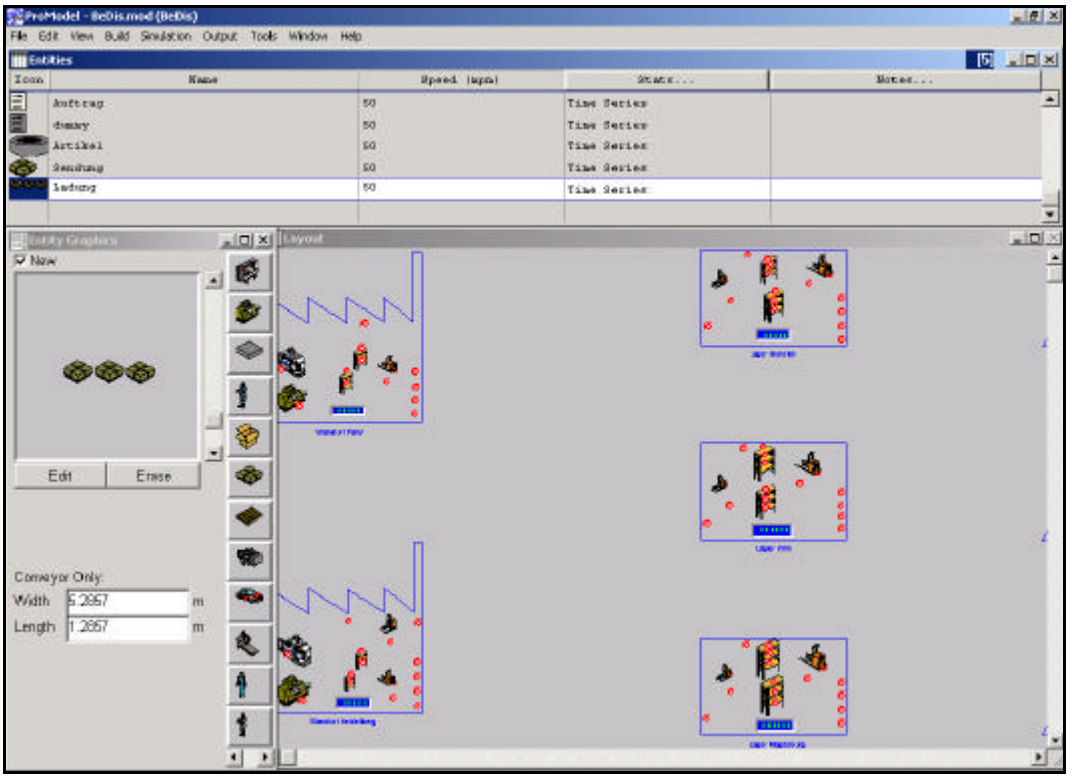


Abbildung 18: Erstellung des Modellierungselements „Entity“

Den erstellten Instanzen des Modellierungselements „Entity“ kann ebenfalls eine beliebige Grafik zugeordnet werden. Die Grafik kann auch hier aus einer Bibliothek stammen oder mittels des Hilfsprogramms *promodel*-Graphic Editor erstellt werden. Die zugeordnete Grafik dient der Symbolisierung der Objekte dieser Instanz des Modellierungselements während der Animation. Sie kann in Größe und Orientierung angepasst werden. Durch Deaktivierung der Schaltfläche „New“ können einer Instanz mehrere Grafiken zugeordnet werden, die sich während der Animation, etwa zur Darstellung einer Lackierung der Objekte des realen Systems, welche durch die Objekte der Instanz repräsentiert werden, wechseln lassen.

Dieses Modellierungselement verfügt ebenfalls über verschieden Attribute. Für die Berechnung der Kapazität eines Bausteins vom Typ „Conveyor“ (vgl. Kapitel 2.4.1.2) stehen die Attribute „Length“ und „Width“ zur Verfügung. Durch diese Attribute werden die Abmessungen des Objektes definiert.

Durch das Attribut „Speed“ wird die Geschwindigkeit definiert, mit welcher sich das Objekt im System bewegt, wenn es sich nicht in einem eine Modellierungselement vom Typ „Resource“ (vgl. Kapitel 2.4.1.5) oder auf einem Baustein vom Typ „Conveyor“ (vgl. Kapitel 2.4.1.2) befindet.

Durch das Attribut „Stats“ wird, wie bei dem Modellierungselement „Location“ festgelegt, wann und wie Statistikdaten für das Objekt aufgezeichnet werden. Ebenfalls wie bei dem Modellierungselement „Location“ existiert ein Attribut „Notes“ zur

Erstellung von Kommentaren bezüglich der Instanz des Modellierungselements „Entity“.

2.4.1.4 Modellierungselement „Path Network“

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 223ff).

Das Modellierungselement „Path Network“ ist für die Verwendung von Objekten des Modellierungselements „Resource“ (vgl. Kapitel 2.4.1.5) erforderlich. Das „Path Network“ bildet dabei ein Netzwerk von Wegen ab, die durch die Objekte des Modellierungselements „Resource“ verwendet werden. Während der Animation werden die „Resources“ auf diesen Wegen des „Path Network“ dargestellt. Die Entfernungen, welche die „Resources“ dabei zurücklegen, werden ebenfalls durch das Modellierungselement „Path Network“ definiert.

Die Arbeitsfenster der Modellierung von „Path Networks“ sind in der folgenden **Abbildung 19** dargestellt. In dieser Umgebung werden die Modellierungsobjekte „Path Network“ erstellt und parametriert.

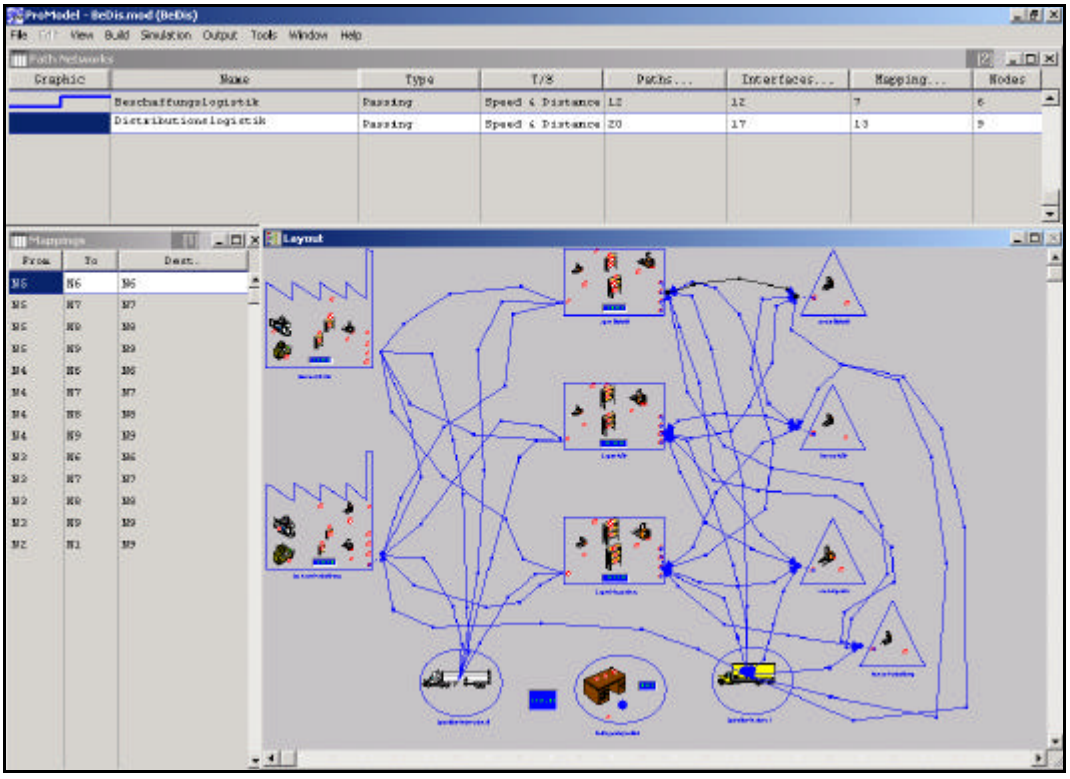


Abbildung 19: Erstellung des Modellierungselements „Path Network“

Ein Objekt des Modellierungselements „Path Network“ setzt sich aus mehreren verschiedenen Wegen, den „Paths“ zusammen. Diese werden durch einen Anfangs- und einen Endpunkt, die so genannten „Nodes“ definiert. Die „Nodes“, also die

Verknüpfungsknoten können dabei mehreren Wegen („Paths“) zugeordnet werden. Jeder der modellierten Wege kann mit einer definierten Länge parametriert werden.

Die Anfangs- bzw. Endknoten stellen die Verknüpfungspunkte der einzelnen Wege des „Path Networks“ dar. Von ihnen aus kann durch ein „Interface“ eine Schnittstelle zu einem Objekt des Modellierungselements „Location“ geschaffen werden. Durch diese Schnittstelle erfolgt während der Simulation die Kommunikation zwischen den Objekten der Modellierungselemente „Location“ und „Resource“. Ein Beispiel hierfür ist die Entladung eines Objektes „Resource“, welche ein Transportmittel abbildet, an einem Objekt „Location“, durch das ein Wareneingang (Laderampe) repräsentiert wird.

Die Definition der durch die „Resources“ über verschiedene Wegelemente hinweg zurückzulegenden Wege erfolgt durch das „Mapping“. Hierbei wird der Weg definiert, den ein Objekt „Resource“ zurücklegen muss, um von einem „Ausgangs–Node“ zu einem bestimmten „Ziel–Node“ zu gelangen. Mittels dieser Funktion können festgelegte Wege definieren werden, welche als Bewegungsvorschrift die normale automatische Wegfindung der Objekte „Resource“ überlagern. Die automatische Wegfindung wählt einen Weg nach dem Kriterium der kürzesten zurückzulegenden Wegstrecke.

Das Modellierungselement „Path Network“ beinhaltet verschiedene Attribute. Das Verhalten der Instanz des Modellierungselements „Path Network“ kann durch die Zuweisung eines Objekttyps beeinflusst werden. Mögliche Objekttypen für „Path Networks“ sind „Passing“, „Non-Passing“ und „Crane“.

Die Verweildauer der Objekte des Elements „Resource“ kann auf verschiedene Weisen definiert werden. Diese Festlegung erfolgt bei der Parametrierung des Objektes „Path Network“. Die Verweildauer kann zum einen als fester Wert parametriert werden (Einstellung „Time“) oder zum anderen vom Simulator aus dem Parameter „Länge“ des Teilweges des „Path Networks“ und dem Parameter „Geschwindigkeit“ der „Resource“ berechnet werden (Einstellung „Speed & Distance“).

Zur Reduktion des für die Simulation benötigten Rechenaufwandes ist es möglich, die grafische Darstellung der „Resource“-Objekte auf den modellierten „Path Networks“ zu deaktivieren.

2.4.1.5 Modellierungselement „Resource“

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 241ff).

Objekte des Modellierungselements „Resource“ sind dynamische Objekte. Diese können durch die ebenfalls dynamischen Objekte des Modellierungselements „Entity“

(vgl. Kapitel 2.4.1.3) während eines Prozesses oder zum Transport benutzt werden. Sie dienen zur Modellierung von Mitarbeitern oder Transportfahrzeugen. Die modellierten Objekte des Modellierungselements „Resource“ benötigen ein Objekt des Modellierungselements „Path Network“ um eine Bewegung ausführen zu können.

Die Arbeitsfenster zur Erstellung von Instanzen des Modellierungselements „Resource“ sind in der folgenden **Abbildung 20** dargestellt.

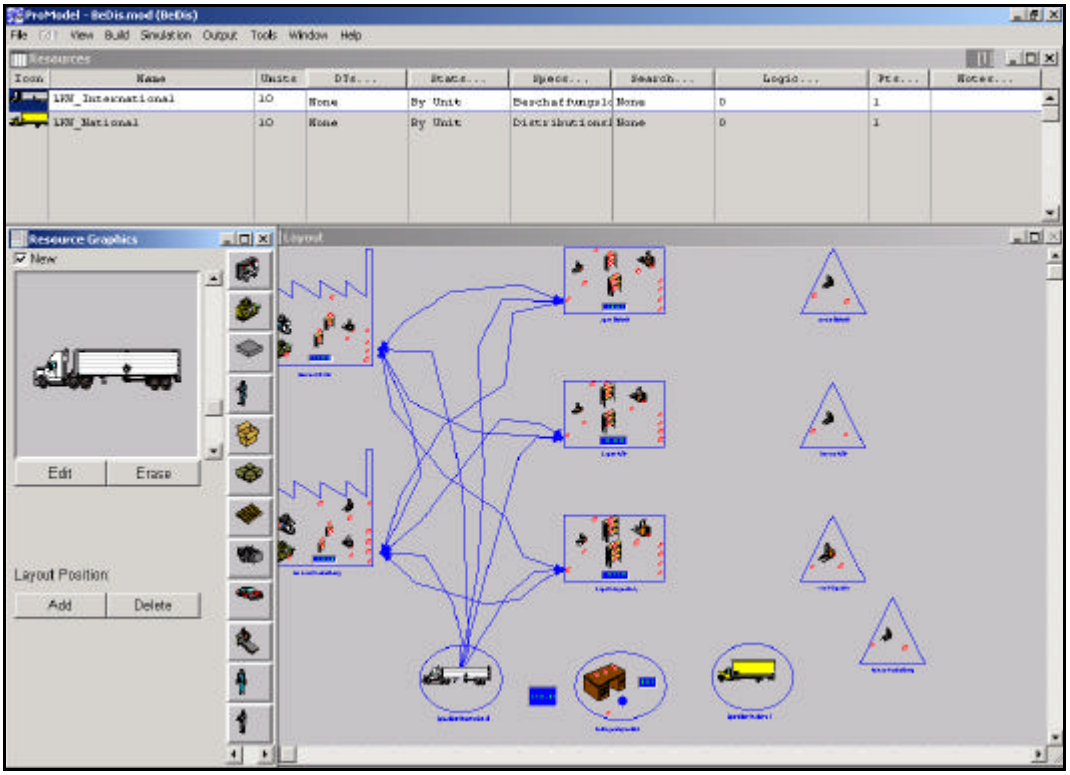


Abbildung 20: Erstellung des Modellierungselements „Resource“

Für eine Instanz des Modellierungselements „Resource“ kann eine beliebige Grafik zur Symbolisierung während der Animation gewählt werden. Diese kann entweder aus einer Bibliothek importiert oder mit dem Hilfsprogramm *promodel*-Graphic Editor erstellt werden. Die Größe und die Orientierung der verwendeten Grafik können der Modellumgebung angepasst werden. Durch Deaktivierung der Schaltfläche „New“ ist es möglich, einer Instanz des Modellierungselements „Resource“ mehrere Grafiken zuzuweisen. Diese Grafiken können während der Animation gewechselt werden. Hierdurch kann der Zustand des Objektes „Resource“ während der Simulation visualisiert werden.

Das Modellierungselement „Resource“ bietet verschiedene Attribute zur Parametrierung der erstellten Instanzen. Durch das Attribut „Units“ wird festgelegt, wie viele Objekte der gebildeten Instanz des Modellierungselements „Resource“ im Modell verfügbar sind. Die weiteren Parameter gelten für alle Objekte dieser Instanz.

Das Attribut „Downtimes“ ist in seiner Funktion identisch mit dem des Modellierungselements „Location“. Es können mehrere Störungen bzw. Pausen mit unterschiedlichen Prioritäten und durchzuführenden Prozessen definiert werden.

Das Attribut „Stats“ dient der Parametrierung der Aufnahme der Statistikdaten. Diese können wahlweise für alle Objekte der Instanz des Modellierungselements („Units“) getrennt erhoben werden, oder für die gebildete Instanz als Gesamtstatistik erhoben werden.

Das Attribut „Logic“ bezieht sich auf die „Nodes“ in einem modellierten „Path Network“. Für jeden „Node“ kann eine Eingangs- und eine Ausgangslogik programmiert werden.

Die weiteren Parameter des Modellierungselements „Resource“ werden in einer Parametermaske spezifiziert. Diese ist in der folgenden **Abbildung 21** dargestellt. Eine detaillierte Erläuterung der einzelnen Parameter findet sich in der Dokumentation des Programmpakets *promodel* (/PROM98a/, S. 241ff).

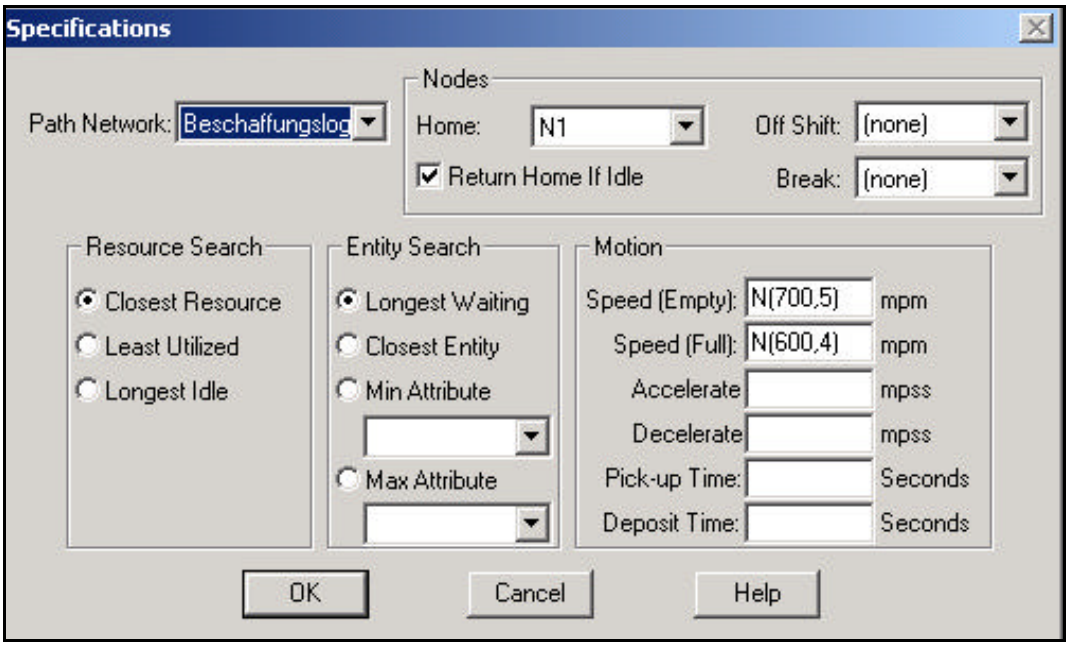


Abbildung 21: Parametermaske des Modellierungselements „Resource“

2.4.1.6 Modellierungselement „Processing“

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 275ff).

Durch das Modellierungselement „Processing“ werden die Funktionen der Objekte des Modellierungselements „Location“ erstellt. Die durch das Modellierungselement „Processing“ definierten Prozesse beziehen sich immer auf Objekte des

Modellierungselements „Entity“. Ein so definierter Prozess beginnt in einem Objekt des Modellierungselements „Location“ und endet in einem anderen Objekt des Modellierungselements „Location“. Der Prozess umfasst damit die Funktion der „Location“, in der er beginnt, und die Bewegung sowie die Disposition (das „Routing“) des „Entities“ zu der „Location“, in der er endet.

In der folgenden **Abbildung 22** sind die Arbeitsfenster zur Definition von Prozessen dargestellt.

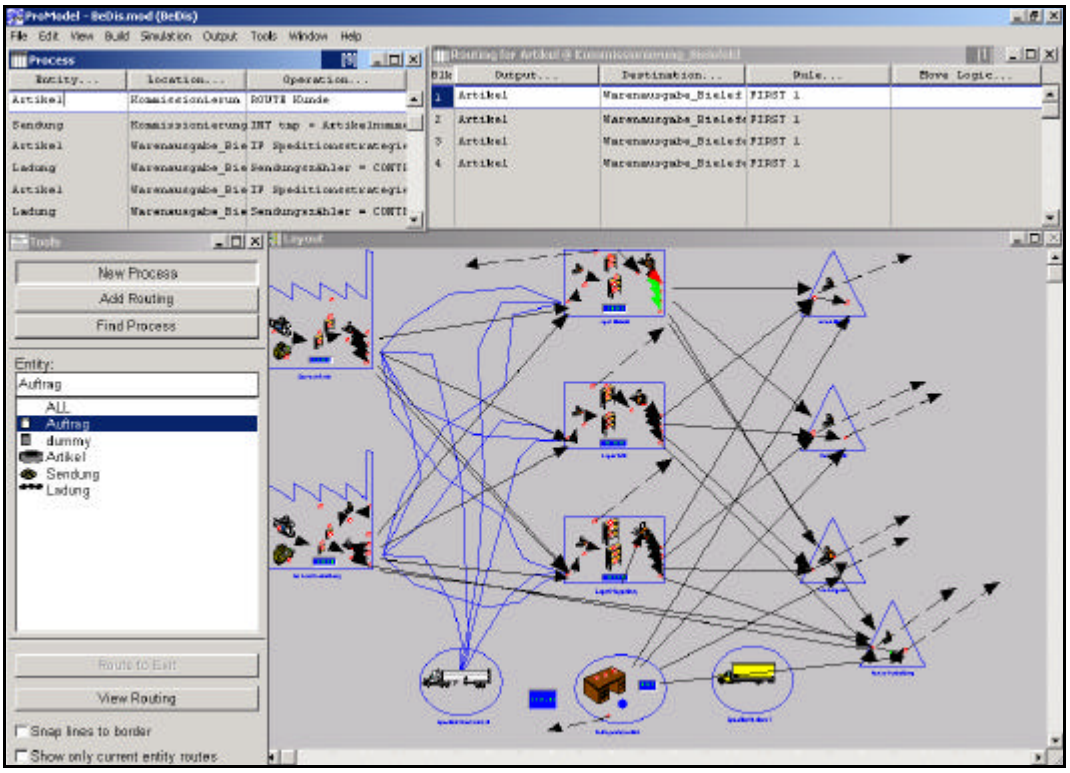


Abbildung 22: Erstellung von Prozessen (Modellierungselement „Processing“)

Ein definierter Prozess bezieht sich immer nur auf ein ihm zugeordnetes Objekt des Modellierungselements „Entity“. Für Objekte verschiedener Instanzen des Modellierungselements „Entity“ können daher in einer „Location“ verschiedene Prozesse definiert werden. Des weiteren ist ein Prozess an eine „Location“ gebunden.

In der „Location“, in welcher der Prozess beginnt, wird durch das Eintreten des Objektes „Entity“ in die „Location“ die Ausführung der für den Prozess programmierten „Operation Logic“ angestoßen. In dieser „Operation Logic“ wird die Funktionalität der „Location“ programmiert.

Zu jedem Prozess gehört ein anschließendes „Routing“. Das „Routing“ entspricht der Disposition des Objektes „Entity“, welches den Prozess angestoßen hat. An dieser Stelle wird definiert, welches „Entity“ den Prozess verlässt (Objekttransformation) und welche „Location“ sein Ziel ist. Gibt es verschiedene Ziele, so kann durch das „Routing“ ein Prozess mit mehreren „Locations“ verbunden werden. Für die verschiedenen Ziele

können Regeln für die Übergabe an die nachfolgende „Location“ definiert werden. Dies geschieht durch den „Routing Editor“. Das Dialogfenster des „Routing Editors“ ist in der folgenden **Abbildung 23** dargestellt.

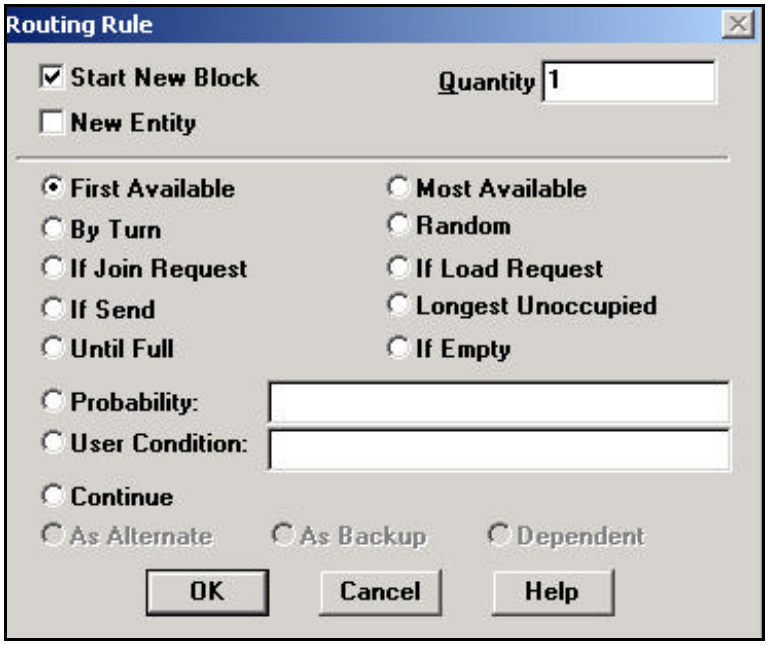


Abbildung 23: Dialogfenster des „Routing Editor“

2.4.1.7 Modellierungselement „Arrival”

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 295ff).

Instanzen des Modellierungselements „Arrival“ dienen dazu, Objekte des Modellierungselements „Entity“ im modellierten System zu erzeugen. Die Instanzen des Modellierungselements „Arrival“ bilden somit ein Quellverhalten im *promodel*-Simulator ab. Eine Instanz des Modellierungselements „Arrival“ bezieht sich dabei immer auf eine Instanz des Modellierungselements „Entity“ und ein Objekt einer Instanz des Modellierungselements „Location“. Die Instanzen des Modellierungselements „Arrival“ sind einem Objekt gleichzusetzen, da eine Instanz immer nur ein mal in einem Modell auftreten kann. Durch die Zuordnung der „Arrivals“ zu einzelnen „Locations“ ist es möglich, aus jeder „Location“ eine Quelle zu erstellen.

Jede Instanz des Modellierungselements „Arrival“ wird durch seine Erstankunft und die Wiederholfrequenz beschrieben. Des weiteren ist es möglich, die Anzahl der Objekte einer Instanz des Modellierungselements „Entity“, welche bei einer Ausführung des „Arrivals“ erzeugt werden, zu parametrieren. Ein weiterer Parameter dient der Beschränkung der insgesamt zu erzeugenden Objekte auf eine parametrierbare Maximalanzahl.

Das Zeitverhalten einer Instanz des Modellierungselements „Arrival“ kann sowohl kalendarisch als auch nach Simulationszeit erfolgen. Es ist auch eine Kombination der verschiedenen Zeittypen möglich.

Die durch ein „Arrival“ im Modell erzeugten Objekte („Entities“) können die Ausführung einer in der Instanz des Modellierungselements „Arrival“ frei programmierbaren „Logic“ anstoßen.

Während der Modellierung kann eine Instanz des Modellierungselements „Arrival“ deaktiviert werden, ohne das sie aus dem Modell entfernt werden muss. Diese Funktion ist für die Validierung des Modells nützlich, da so das Modell im Hinblick auf sein Verhalten in Bezug auf Objekte einzelner Instanzen des Modellierungselements „Entity“ isoliert untersucht werden kann.

Die Arbeitsfenster für die Erstellung von Instanzen des Modellierungselements „Arrival“ sind in **Abbildung 24** dargestellt.

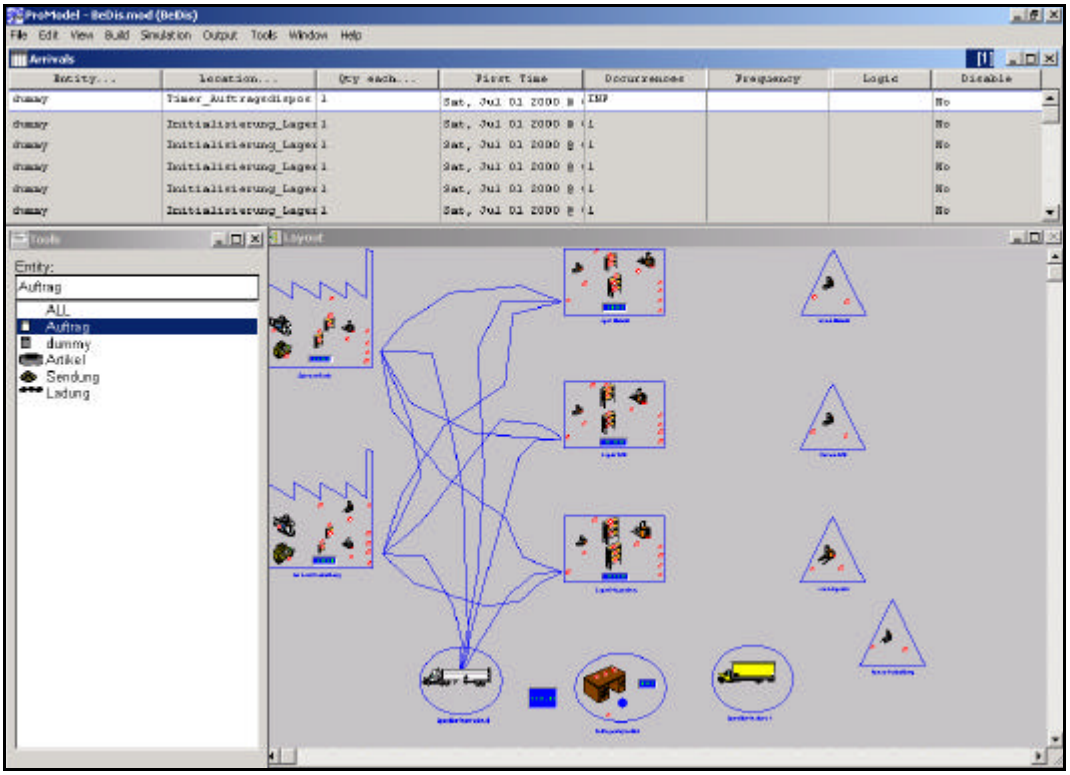


Abbildung 24: Erstellung des Modellierungselements „Arrival“

2.4.1.8 Modellierungselement „Attribute“

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 391ff).

Attribute sind Variablen, die entweder mit einer Instanz des Modellierungselements „Location“ oder einer Instanz des Modellierungselements „Entity“ verbunden sind. Die

Anzahl der Attribute einer Instanz ist im *promodel*-Simulator nicht begrenzt. Die zulässigen Datentypen für Attribute sind die Datentypen „Integer“ oder „Real“. Die Werte eines Attributes sind für die Programmierung eines Prozesses nur dann verfügbar, wenn das Attribut entweder der Instanz des Objektes „Entity“ zugewiesen ist, welches die Ausführung des Prozesses angestoßen hat, oder das Attribut der Instanz des Objektes „Location“ zugewiesen ist, auf welches sich der Prozess bezieht.

Das Dialogfenster zur Erstellung von Attributen ist in **Abbildung 25** dargestellt.

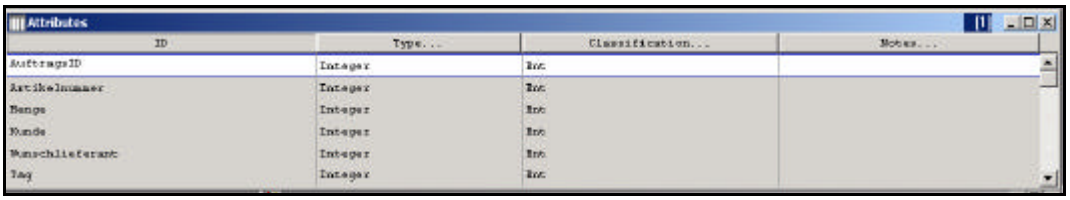


Abbildung 25: Erstellung von „Attributen“

Wurden mehrere Objekte von einer Instanz oder mehreren Instanzen des Modellierungselements „Entity“ durch eine Gruppierung zusammengefasst, oder befindet sich ein Entity auf einem Objekt einer Instanz des Modellierungselements „Resource“, so besteht keine Möglichkeit auf die zugeordneten Attribute zuzugreifen.

Führt ein definierter Prozess zu der Erzeugung eines neuen Objektes einer Instanz des Modellierungselements „Entity“, so werden die Attribute des Objektes „Entity“, welches den Prozess angestoßen hat, auf das erzeugte Objekt kopiert.

2.4.1.9 Modellierungselement „Variable“

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 401ff).

Der Zugriff auf die Werte definierter Variablen ist im *promodel*-Simulator global möglich. Zulässige Datentypen für Variablen sind die Datentypen „Integer“ und „Real“. Bei der Definition von Variablen kann für jede Variable ein Startwert vorgegeben werden. Während der Simulation kann der Wert der Variable durch die Programmierung eines Prozesses, die Programmierung eines Objektes des Modellierungselements „Arrival“, eine programmierte Subroutine oder jede andere Form von programmierter Logik im Modell verändert werden.

Das Dialogfenster zur Erstellung von Variablen ist in **Abbildung 26** dargestellt. Der Werteverlauf der definierten Variablen kann statistisch erfasst werden. Die Aktivierung dieser Statistikfunktionalität des Modellierungselements „Variable“ erfolgt unter dem Menüpunkt „Stats“.

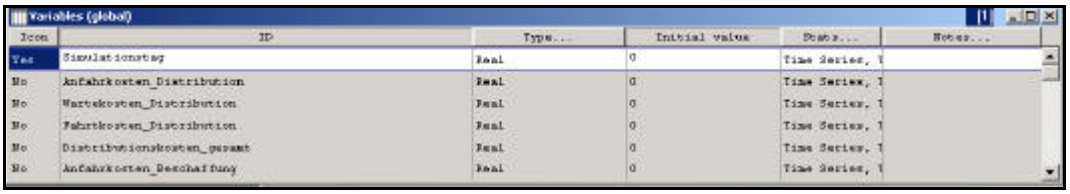


Abbildung 26: Erstellung von „Variablen“

2.4.1.10 Modellierungselement „Array“

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 409ff).

Ein Array ist eine n-dimensionale Matrix von Variablen. Die zulässigen Datentypen für Arrays sind die Datentypen „Integer“ und „Real“. Die definierten Arrays stehen im *promodel*-Simulator global zur Verfügung. Für die Änderung der Werte eines Arrays gelten dieselben Bedingungen wie für die Änderung des Wertes einer Variablen.

Das Dialogfenster zur Erstellung von Arrays ist in **Abbildung 27** dargestellt.

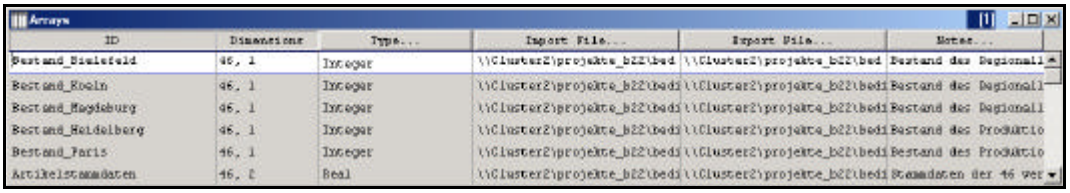


Abbildung 27: Erstellung von „Arrays“

Die Initialisierung eines definierten Arrays mit Startwerten kann während der Initialisierung eines Modells durch die Importierung von Daten aus MS-Excel-Tabellen realisiert werden. Hierbei ist ein Zugriff auf verschiedene Bereiche auf verschiedenen Blättern einer Arbeitsmappe möglich. Die Definition der zu importierenden Bereiche und Blätter einer Arbeitsmappe erfolgt durch das Setzen von Importparametern.

Der Datenexport zum Simulationsende ist ebenfalls möglich. Hierbei wird der Inhalt eines definierten Arrays in einer MS-Excel-Tabelle gespeichert.

Die Dialogmaske für die Angabe der Im- und Exportparameter ist in **Abbildung 28** dargestellt.

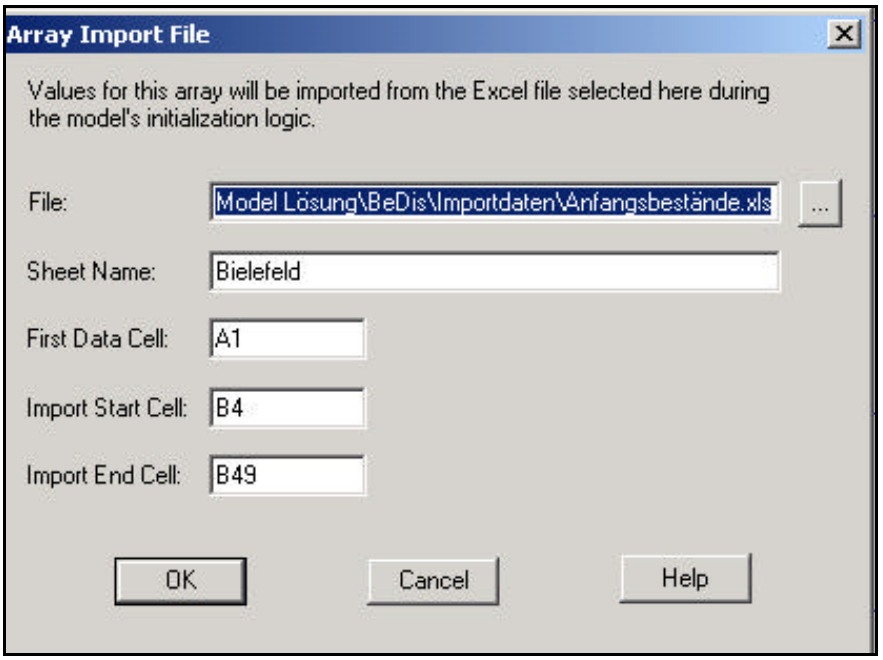


Abbildung 28: Importparameter für den Import von Excel-Tabellen

2.4.1.11 Modellierungselement „Macro“

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 415ff).

Mit Instanzen des Modellierungselements „Macro“ werden durch den Anwender variierbare Modell-Parameter abgebildet. Durch die Definition eines „Run-Time-Interface“ wird einer Instanz des Modellierungselements „Macro“ ein Dialogfenster zugewiesen, durch das der Anwender den repräsentierten Parameter des Modells parametrieren kann. Die Werte von „Macros“ stehen im *promodel*-Simulator global zur Verfügung.

Die folgende **Abbildung 29** zeigt das Dialogfenster zur Erstellung von Instanzen des Modellierungselements „Macro“. In **Abbildung 30** ist das Dialogfenster zur Erstellung von „Run-Time-Interfaces“ abgebildet.

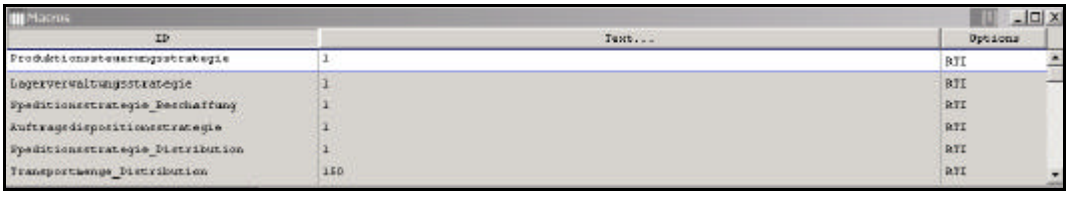


Abbildung 29: Erstellung von „Macros“

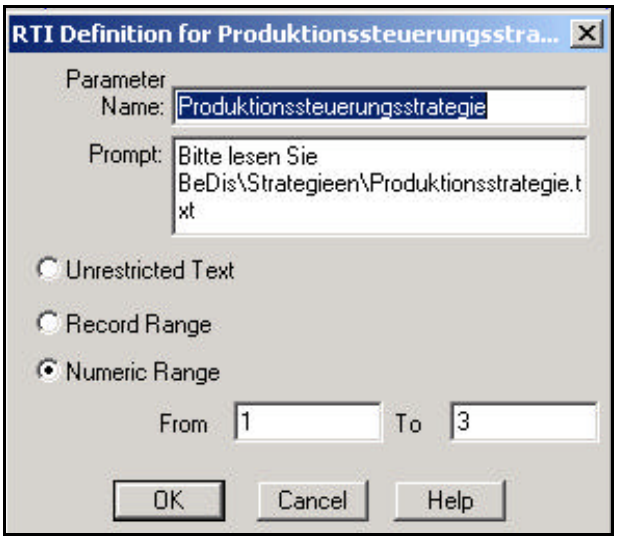


Abbildung 30: Dialogfenster des „RTI-Editors“

Jeder definierten Instanz des Modellierungselements „Macro“ kann ein Standardwert zugewiesen werden. Dieser Standrdwert wird durch den *promodel*-Simulator verwendet, wenn der Anwender vor Simulationsstart den Wert eines „Macros“ nicht parametriert hat und kein vorbestimmtes Szenario (vordefinierte Parameterfestlegung) simuliert wird.

Ein Szenario besteht aus den fixen Werten, welche den definierten Instanzen des Modellierungselements „Macro“ zugewiesen werden. Es besteht die Möglichkeit mehrere Szenarien zu definieren und zu speichern.

2.4.1.12 Modellierungselement „Subroutine“

Die Angaben zu diesem Modellierungselement basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Simulators (/PROM98a/, S. 423ff).

Durch die Verwendung von „Subroutines“ ist es möglich, einen modularen Modellaufbau zu erreichen. Häufig verwendete Funktionen können in einer „Subroutine“ programmiert werden, welche dann z. B. in der „Operation-Logic“ eines definierten Prozesses ausgeführt wird. Die Verwendung von Subroutinen zur Modellerstellung ist von Vorteil, da Änderungen nur an einer Stelle vorgenommen werden müssen.

Abbildung 31 zeigt das Dialogfenster zur Erstellung von „Subroutines“.

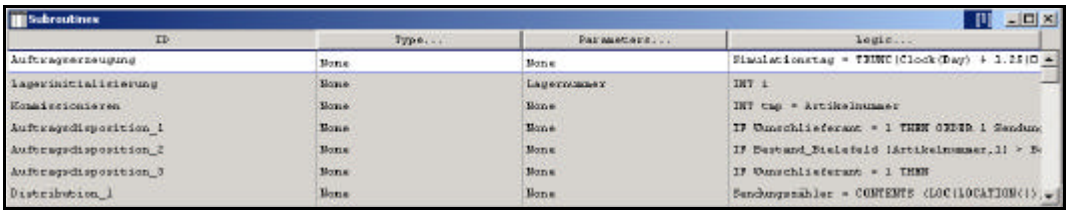


Abbildung 31: Erstellung von „Subroutines“

Promodel bietet die Möglichkeit „Subroutines“ als Funktion zu gestalten, die einen Wert des Datentyps „Integer“ oder „Real“ annehmen können.

Zur Programmierung der „Subroutines“ stellt der *promodel*-Simulator ein Assistenzprogramm zur Verfügung, den so genannten „Logic-Builder“ (/PROM98a/, S. 465ff). Dieses Assistenzprogramm erstellt die Befehlssyntax und erläutert die Funktion der Befehle. Der „Logic-Builder“ ist weiterhin ein nützliches Navigationshilfsmittel, da er auf alle „Variablen“, „Macros“ und „Arrays“ zugreifen kann.

Der „Logic-Builder“ ist ein hilfreicher Assistent, mit welchem eine Einarbeitung in die Syntax der Programmiersprache unterstützt wird. In Verbindung mit den gut dokumentierten und nützlichen Befehlen ist es auch für ungeübte Anwender möglich, in *promodel* zu programmieren. Das Dialogfenster des „Logic-Builders“ ist in **Abbildung 32** dargestellt.

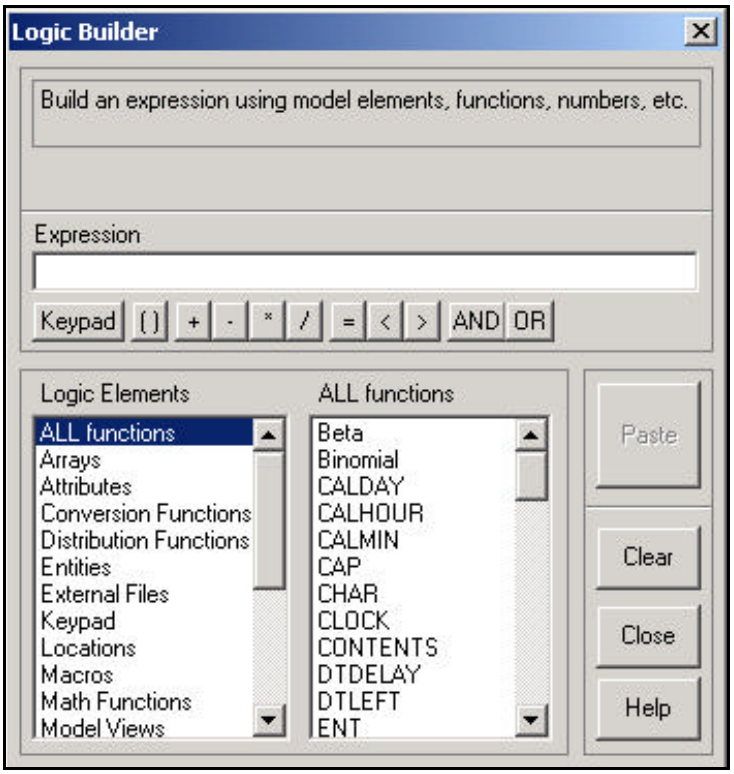


Abbildung 32: Dialogfenster des „Logic-Builders“

2.4.2 Vorstellung der Komponente *promodel*-Stat Fit

Die Angaben zu diesem Hilfsprogramm des *promodel*-Programmpakets basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Programmpakets (/PROM98a/, S. 490ff).

Das Hilfsprogramm *promodel*-Stat Fit unterstützt den Anwender dabei, empirisch ermittelte Daten in eine im Simulationsmodell verwendete Verteilungsfunktion zu überführen.

Die empirischen Daten können dabei aus externen Programmen wie z.B. MS-Excel ausgelesen werden. Die Daten werden durch das Hilfsprogramm automatisch analysiert und eine Verteilungsfunktion, welche die empirischen Daten repräsentiert, wird vorgeschlagen. Es besteht auch die Möglichkeit, dass der Anwender explizit eine Verteilungsfunktion auswählt. Durch die in das Hilfsprogramm integrierten Validierungsmechanismen unterstützt das Hilfsprogramm den Anwender dabei zu ermitteln, in wie weit eine ausgewählte Verteilungsfunktion die empirischen Daten wiedergibt.

Die mit dem Hilfsprogramm *promodel*-Stat Fit ermittelten Verteilungsfunktionen können bei der Modellierung im *promodel*-Simulator verwendet werden.

Die Benutzeroberfläche des Hilfsprogramms *promodel*-Stat Fit ist in **Abbildung 33** dargestellt.

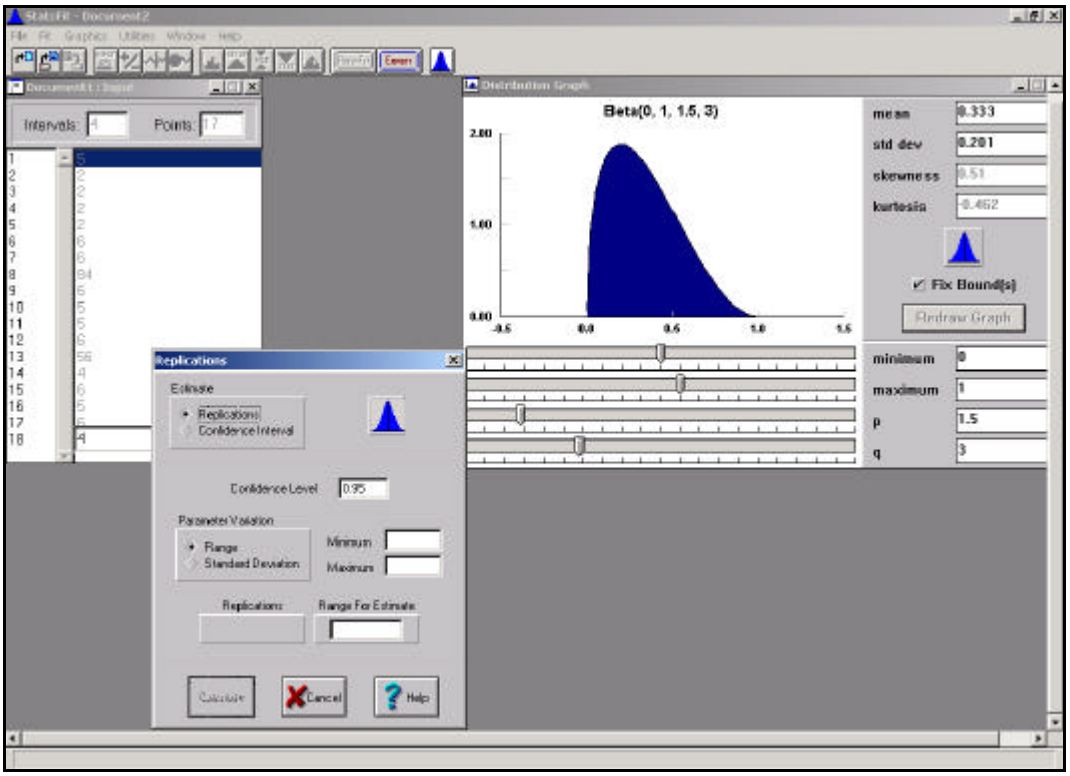


Abbildung 33: Benutzeroberfläche des Hilfsprogramms *promodel*-Stat Fit

2.4.3 Vorstellung der Komponente *promodel*-Graphic Editor

Die Angaben zu diesem Hilfsprogramm des *promodel*-Programmpakets basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Programmpakets (/PROM98a/, S. 491ff).

Das Hilfsprogramm *promodel*-Graphic Editor aus dem *promodel*-Programmpaket dient der Editierung bzw. Erstellung von Grafik-Bibliotheken für die Verwendung durch den *promodel*-Simulator.

Die Benutzeroberfläche des promodel-Grafic Editors ist in der folgenden **Abbildung 34** dargestellt.

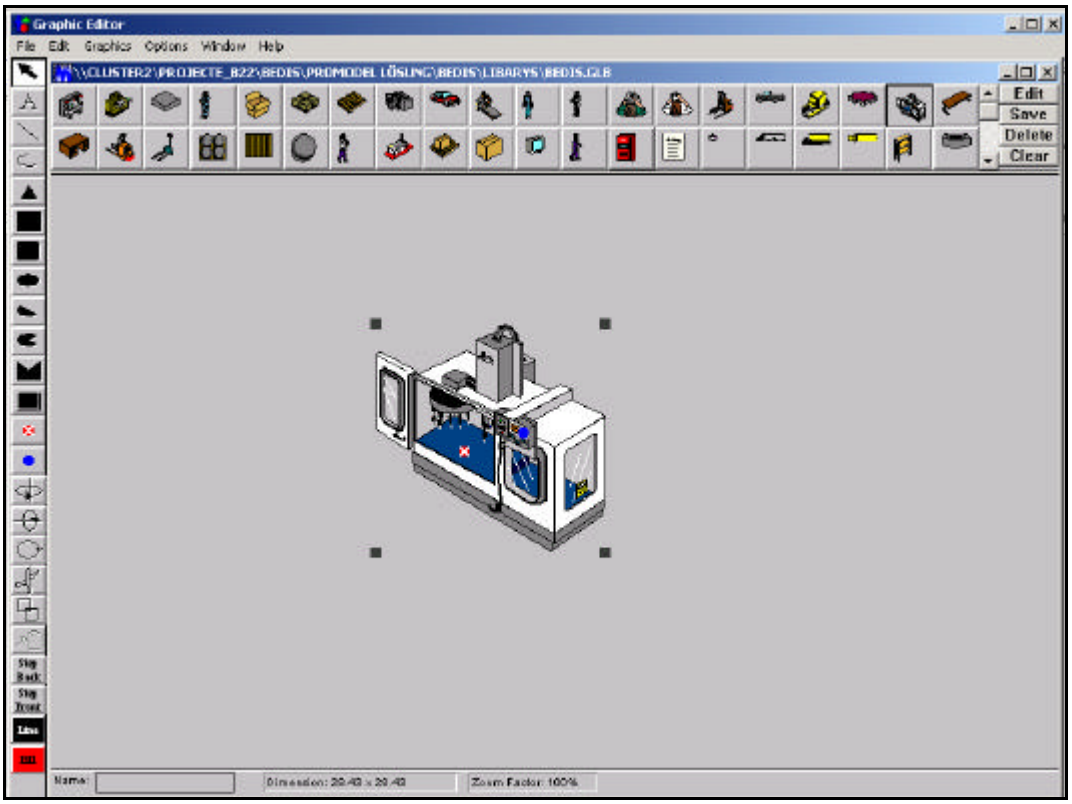


Abbildung 34: Benutzeroberfläche des Hilfsprogramms *promodel*-Graphic Editor

Das Hilfsprogramm *promodel*-Graphic Editor ermöglicht es, Grafiken verschiedener Bibliotheken auszutauschen und zu editieren. Mit externen Grafik-Programmen erstellte Grafiken können unter zu Hilfenahme des *promodel*-Graphic Editors ebenfalls für die Verwendung durch den *promodel*-Simulator importiert werden. Hierzu bietet das Hilfsprogramm eine Importfunktion auf der Basis einer Dateischnittstelle für Grafik-Dateien im Bitmap und PCX Datei-Format.

Für die Erstellung einfacher Grafiken stellt das Hilfsprogramm verschiedene Zeichentools zur Verfügung.

Mit dem *promodel*-Grafic Editor werden auch die Animationspunkte für die Darstellung dynamischer Objekte sowie die Ankerpunkte von dynamischen Objekten erstellt. Durch die Verwendung von Animationspunkten, den so genannten „Spots“, ist es möglich, im *promodel*-Graphic Editor zu bestimmen, an welcher Stelle des Symbols, z.B. eines Objektes einer Instanz des Modellierungselements „Location“, das Symbol eines in diesem Objekt befindlichen dynamischen Objektes während der Animation dargestellt wird.

Die erstellten Symbol-Grafiken können mit „Statuslichtern“ zur Zustandsvisualisierung ergänzt werden.

2.4.4 Vorstellung der Komponente *promodel*-SimRunner

Die Angaben zu diesem Hilfsprogramm des *promodel*-Programmpakets basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Programmpakets (/PROM98a/, S. 25ff).

Das Hilfsprogramm *promodel*-SimRunner unterstützt den Anwender bei der Experimentdurchführung mit dem *promodel*-Simulator. Mittels dieses Hilfsprogramms können Experimentläufe automatisiert durchgeführt werden, wobei durch das Programm eine automatische Parametervariation zur Optimierung der Simulationsergebnisse in Bezug auf ein wählbares Zielsystem erfolgt.

Eine Voraussetzung für die Verwendung des Hilfsprogramms *promodel*-SimRunner ist die Implementierung von Instanzen des Modellierungselements „Macro“ im Simulationsmodell, da durch dieses Modellierungselement die variierbaren Parameter definiert werden.

Das Hilfsprogramm *promodel*-SimRunner führt automatisiert eine parametrisierbare Anzahl von Simulationsexperimenten durch. Während dieser Experimente variiert das Hilfsprogramm ausgewählte Modellparameter bis die Ergebnisse der Experimente ein erreichbares Optimum im Hinblick auf ein definiertes Zielsystem darstellen.

Das definierte Zielsystem setzt sich aus frei wählbaren Ergebnisgrößen zusammen. Als Ergebnisgrößen können Variablen oder Attribute wie Belegungen, Kosten oder ähnliche gewählt werden, die je nach Zielsetzung maximiert oder minimiert werden. Die einzelnen Kriterien können zur Bildung des Zielsystems verschieden gewichtet werden.

Die Benutzeroberfläche des Hilfsprogramms *promodel*-SimRunner ist in der folgenden **Abbildung 35** dargestellt.

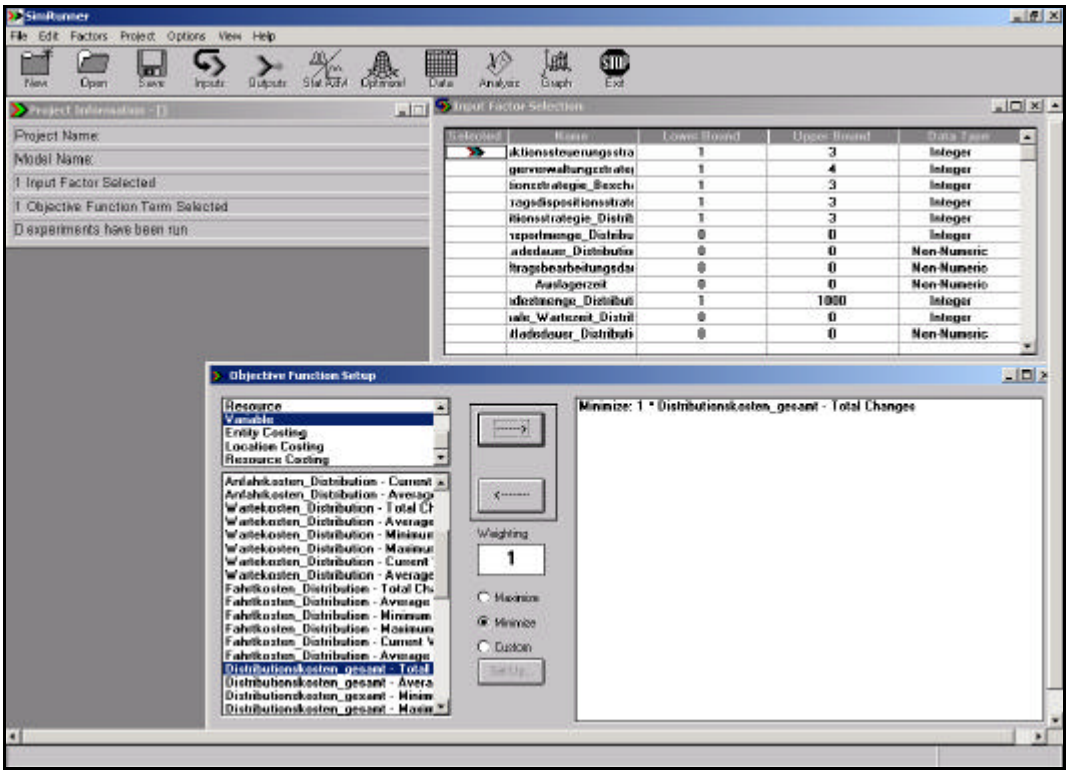


Abbildung 35: Benutzeroberfläche des Hilfsprogramms *promodel*-Simrunner

2.4.5 Vorstellung der Komponente *promodel*-Output Report

Die Angaben zu diesem Hilfsprogramm des *promodel*-Programmpakets basieren auf den bei der Modellerstellung gemachten Erfahrungen und der Dokumentation des *promodel*-Programmpakets (/PROM98a/, S. 585ff).

Die Programmkomponente *promodel*-Output Report unterstützt den Anwender bei der Auswertung der während eines Simulationslaufes durch den *promodel*-Simulator erfassten Ergebnisdaten. Die erfassten Daten liegen nach einem Simulationslauf in Form von Tabellen vor. Durch den Einsatz des Hilfsprogramms *promodel*-Output Report werden diese Ergebnisdaten in einer gegliederten Darstellung für den Anwender ausgegeben. Das Hilfsprogramm bietet dem Anwender weiter Funktionalitäten zur Auswertung der Daten. Dies können mit Hilfe des Programms automatisch in wählbare und parametrierbare Diagramme überführt werden. Somit wird dem Anwender die Möglichkeit gegeben, eine erste optisch leicht zu erfassende Auswertung der erhobenen Ergebnisdaten vorzunehmen. Das Hilfsprogramm bietet zusätzlich die Möglichkeit, die erhobenen Daten bzw. die erstellten Diagramme an eine externe Anwendung wie z.B. MS-Excel zu exportieren. Der Export kann dabei über eine Dateischnittstelle oder die Zwischenablage des Betriebssystems Microsoft-Windows erfolgen.

Die Benutzeroberfläche des Hilfsprogramms *promodel*-Output Report ist in der folgenden **Abbildung 36** dargestellt. In dieser Abbildung ist auch die gegliederte Darstellung der Ergebnisdaten sowie ein wählbarer Diagrammtyp für die Darstellung

der Auslastung der im Simulationsmodell eingesetzten Objekte der gebildeten Instanzen des Modellierungselements „Entity“ dargestellt.

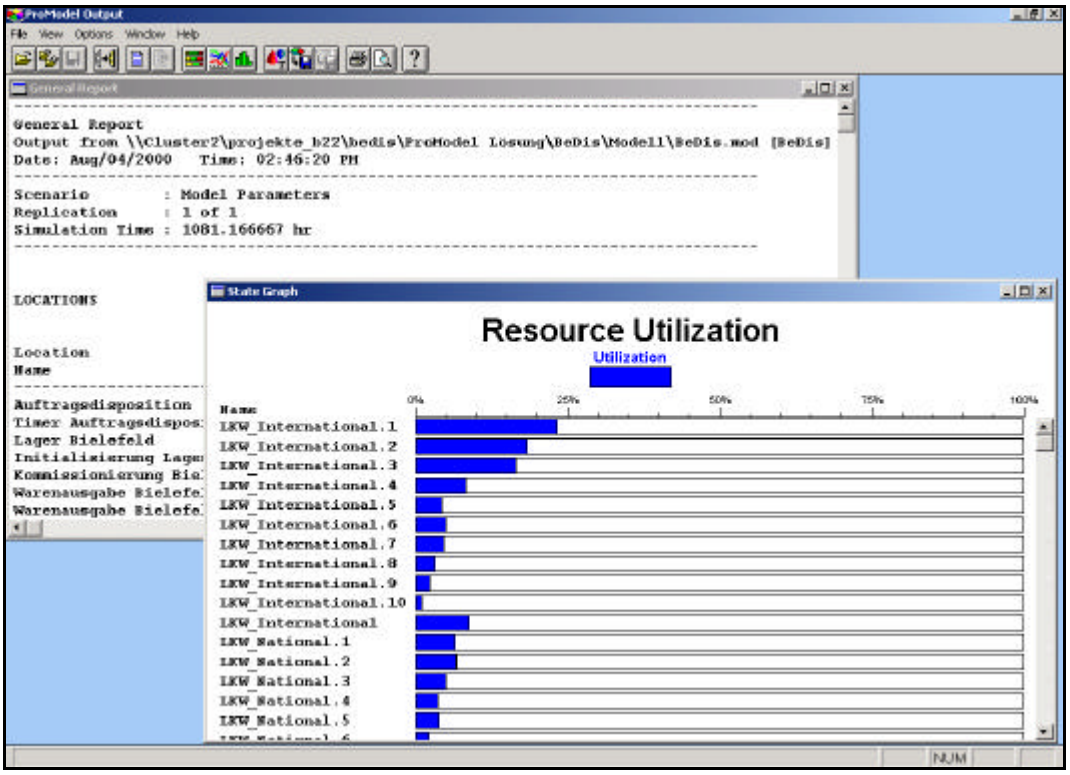


Abbildung 36: Benutzeroberfläche des Hilfsprogramms *promodel*-Output Report

3 Systembeschreibung der Beschaffungskette

Mit dem zu erstellenden Modell soll die Eignung des prozessorientierten *promodel*-Simulators zur Abbildung von Beschaffungsketten beurteilt werden sowie die Auswirkung verschiedener Dispositionsstrategien in der Beschaffungskette auf die Entwicklung von Logistikkosten und Lieferbereitschaft untersucht werden. Um diese Ziele zu erreichen, soll eine Beschaffungskette abgebildet werden, welche typische Elemente der Beschaffungslogistik enthält. Die zu modellierende Beschaffungskette soll dabei über die Abbildung nur eines speziellen realen Systems hinausgehen, um die Modellierbarkeit verschiedener Aspekte nachzuvollziehen.

Das System, welches als Grundlage für das Modell dient, ist an das Beispiel einer zweistufigen Beschaffungskette der Einzelhändler von Reifenprodukten eines namenhaften deutschen Herstellers angelehnt. Für die Realisierung der Beschaffung basiert dieses System auf einem Regionallagerkonzept mit zwei vorgelagerten Produktionsstandorten. Das spätere Modell bildet aber nicht nur dieses reale System im Detail ab, sondern orientiert sich an dessen konzeptionellen Schwerpunkten. Das Modell des bestehenden Systems wird bei der Modellierung um verschiedene zusätzliche Dispositionsstrategien in den einzelnen Ebenen der realen Beschaffungskette erweitert, um ein möglichst breites Spektrum von Anforderungen an die Modellierungsumgebung abzubilden.

Im Folgenden wird ein Überblick über die zugrunde liegende Beschaffungskette, die Abläufe in den einzelnen Elementen und die implementierten Dispositionsstrategien in den verschiedenen Ebenen und Elementen dieser Beschaffungskette gegeben.

3.1 Überblick über das zugrunde liegende Regionallagerkonzept

Das real existierende Regionallagerkonzept, welches dem modellierten System zugrunde liegt, ist schematisch in der folgenden **Abbildung 37** dargestellt. Die zugrunde liegende Topologie besteht aus den beiden Produktionsstandorten in Paris und Heidelberg, den drei Regionallagern für die Regionen Nord, West und Ost mit Standorten in Bielefeld, Köln und Magdeburg, sowie den Kundenregionen Nord, West, Ost und Süd, welche für die Darstellung der Entfernungen auf die Regionen Bielefeld, Köln, Magdeburg und Heidelberg beschränkt werden. Im bestehenden System verfügt der Produktionsstandort Heidelberg über ein internes Auslieferungslager, über welches die Produktbeschaffung der Kundenregion Süd (Bereich Heidelberg) realisiert wird. Die beiden Produktionsstandorte bieten jeweils ein bestimmtes, untereinander verschiedenes Produktspektrum an.

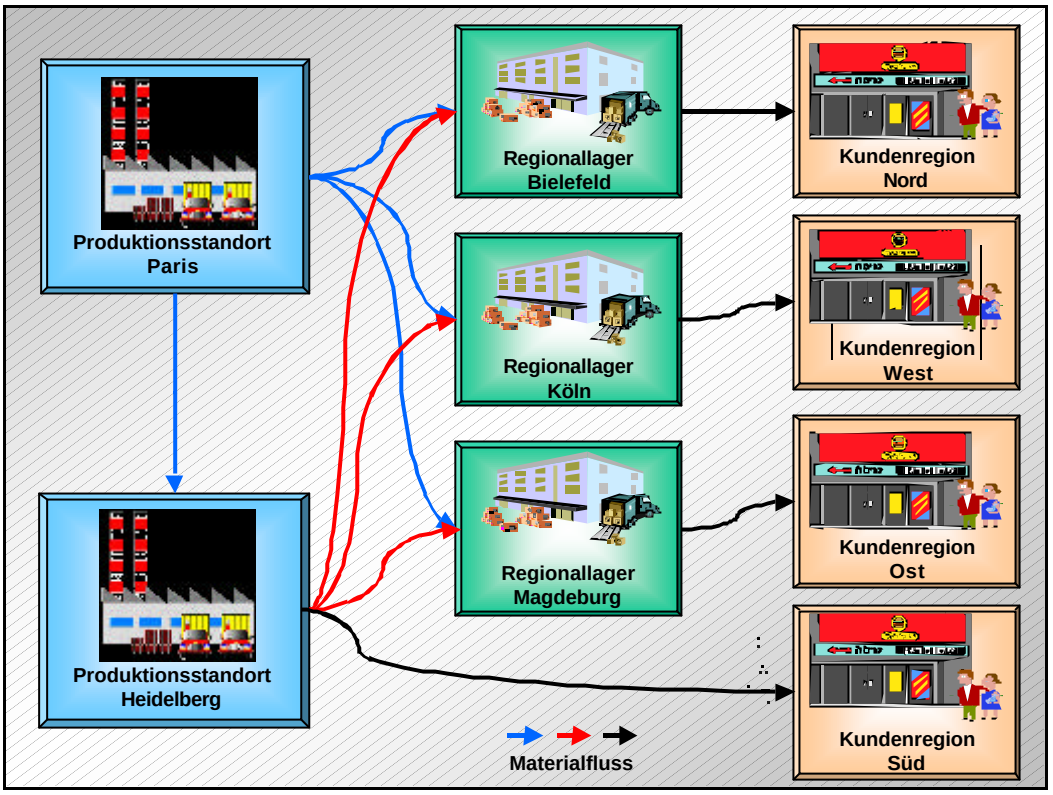


Abbildung 37: schematische Darstellung des zugrunde liegenden Regionallagerkonzepts

Die ursprüngliche Auftragsdisposition des Regionallagerkonzeptes ist dadurch gekennzeichnet, dass die Produktbeschaffung für die Kunden immer nur über das Regionallager der entsprechenden Region realisiert wird. Die wesentlichen Prozesse zur Durchführung der Materialbeschaffung in diesem Konzept und in Verbindung mit einer bedarfsorientierten Beschaffung der Regionallager sind in der folgenden **Abbildung 38** dargestellt. Diese Prozesskettendarstellung ist stark vereinfacht.

Für die Implementierung variabler Dispositionsstrategien muss das System um eine zentrale Auftragsdisposition erweitert werden. Im Fall der ursprünglichen Strategie leitet dies die Aufträge ohne Zeitverlust und ohne Kosten zu verursachen an die zuständigen Regionallager weiter, so dass das ursprüngliche Konzept durch diese Erweiterung nicht beeinträchtigt wird.

Die Ausgangsdaten der späteren Simulationsexperimente basieren auf empirischen Daten des oben beschriebenen Konzeptes. Es wird jedoch erforderlich sein, einige so vorgegebenen Daten, wie z.B. das Produktionsprogramm der beiden Produktionsstandorte, zu flexibilisieren, da sich das Zeitverhalten der Beschaffungskette durch die Strategievariationen ändern wird. Konstant für alle Strategien bleiben die empirischen Daten der Kundenbestellungen.

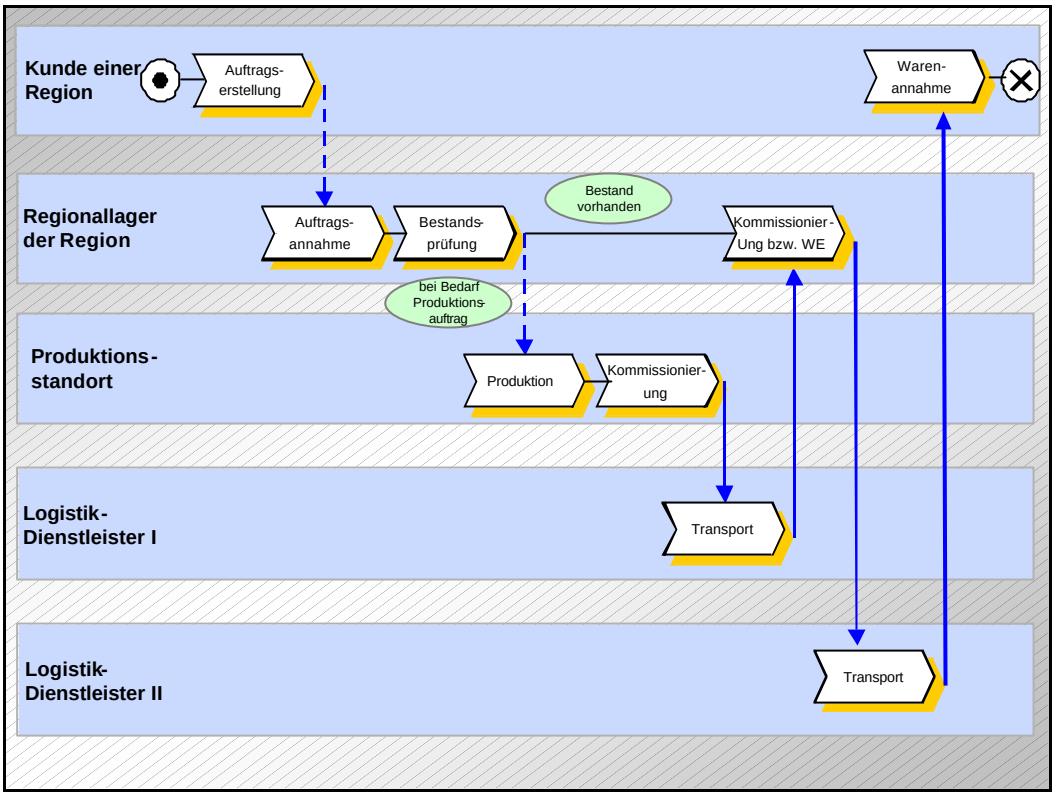


Abbildung 38: Prozesskette der Beschaffung mittels des Regionallagerkonzeptes

Das beschriebene System besitzt in verschieden Elementen Steuerungs- bzw. Dispositionsebenen, für welche variable Strategien implementiert werden können. Die so zu erweiternden Elemente sind:

- die zusätzlich zu implementierende Auftragsdisposition zur Disponierung der eingehenden Kundenbestellungen auf die verschiedenen Regionallager,
- die Lagerverwaltungssysteme der Regionallager zur Implementierung verschiedener Beschaffungsstrategien der Lager,
- die Dispositionsstrategien der Regionallager für die Sendungskommissionierung und die Transportanforderung,
- die Dispositionsstrategien der Produktionsstandorte für die Sendungskommissionierung und die Transportanforderung sowie
- die Produktionssteuerung der Produktionsstandorte, um auf das geänderte Zeitverhalten der Beschaffungskette bei verschiedenen Dispositionsstrategien zu reagieren.

Die für die oben aufgezählten Elemente implementierten Dispositionsstrategien werden im folgenden Abschnitt beschrieben. Die folgende **Abbildung 39** stellt die erweiterten Dispositionselemente in der Topologie der betrachteten Beschaffungskette dar.

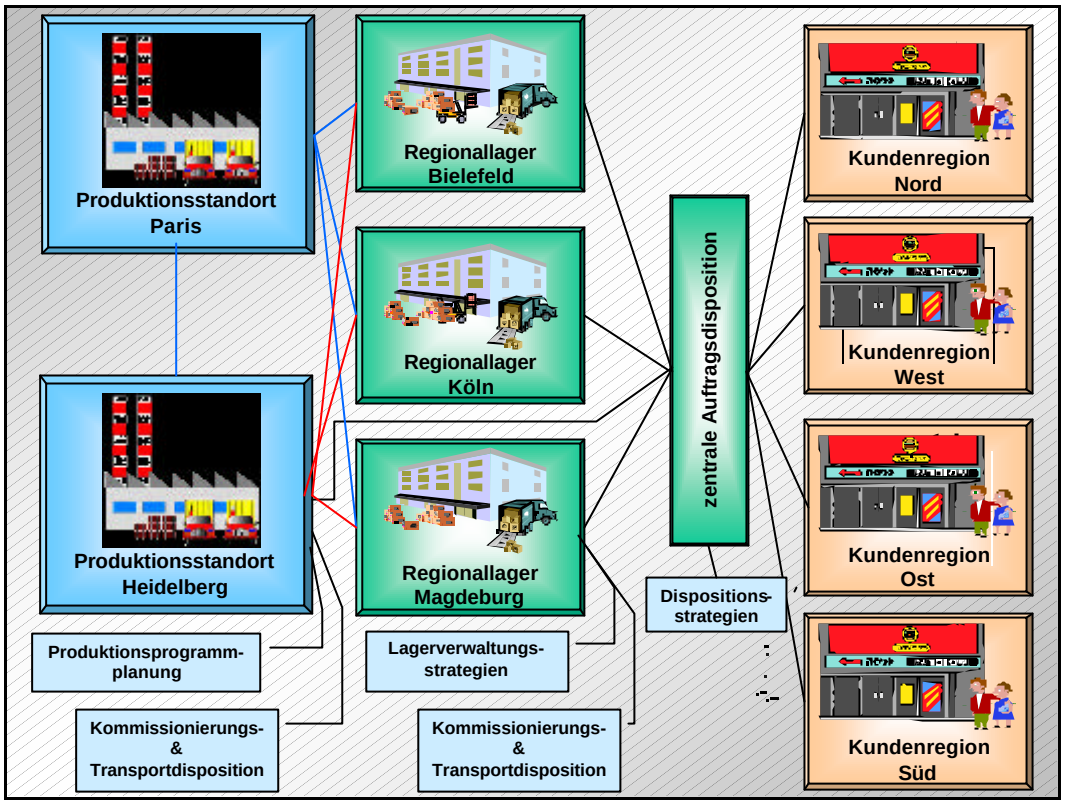


Abbildung 39: Elemente des Regionallagerkonzepts für die eine Strategieerweiterung durchgeführt wurde

3.2 Erweiterte Strategien für die Modellierung des Regionallagerkonzepts

Für die in der vorhergehenden Abbildung dargestellten Systemelemente werden nun die Strategien vorgestellt, um welche die Systemelemente zur Erstellung des Simulationsmodells erweitert wurden. Eine Übersicht über alle implementierten Strategien erfolgt am Ende dieses Kapitels in **Abbildung 41**.

3.2.1 Dispositionsstrategien des Elements Auftragsdisposition

Das zusätzlich implementierte Systemelement Auftragsdisposition wird mit drei integrierten Dispositionsstrategien im Modell abgebildet. Die einzelnen Strategien für die Auftragsdisposition sind:

1. Die Kundenbestellungen werden zur Befriedigung des Bedarfs nur an das für die jeweilige Kundenregion zuständige Regionallager weitergeleitet.
2. Die Kundenbestellungen werden zur Befriedigung des Bedarfs an das Regionallager weitergeleitet, welches zum Zeitpunkt der Auftragsdisposition über den höchsten Bestand des bestellten Artikels verfügt.
3. Ist eine Bedarfsbefriedigung durch das für eine Kundenregion zuständige Regionallager möglich, so werden die Kundenbestellungen der Region an dieses

Regionallager disponiert. Reicht der verfügbare Bestand des bestellten Artikels im zuständigen Regionallager nicht aus, um den Bedarf zu befriedigen, so wird die Kundenbestellung an das Regionallager weitergeleitet, welches zum Zeitpunkt der Auftragsdisposition über den höchsten Bestand des bestellten Artikels verfügt.

Die erste Dispositionsstrategie des Elements Auftragsdisposition entspricht dem ursprünglichen Regionallagerkonzept. Bei der Disponierung der Aufträge mittels dieser Strategie verhält sich die Auftragsdisposition für das System kosten- und zeitneutral.

Die zweite und dritte Strategie der Auftragsdisposition umfassen zur Behandlung von Ausnahmefällen noch den Aspekt, dass bei identischen Beständen in mehreren Regionallagern die Kundenbestellung an das geographisch näher an der Kundenregion liegende Regionallager disponiert wird.

Die Prozesse der Auftragsdisposition nach der zweiten und dritten beschriebenen Dispositionsstrategie sind in der folgenden **Abbildung 40** dargestellt. Die beiden Strategien unterscheiden sich nur in der Art der Bestandsprüfung voneinander. Zur realen Umsetzung dieser Strategien ist es erforderlich, dass zwischen der zentralen Auftragsdisposition und den einzelnen Regionallagern ein informatorische Verbindung geschaffen wird. Es muss gewährleistet sein, dass die Auftragsdisposition per EDV online auf die Bestände in den einzelnen Regionallagern zugreifen kann.

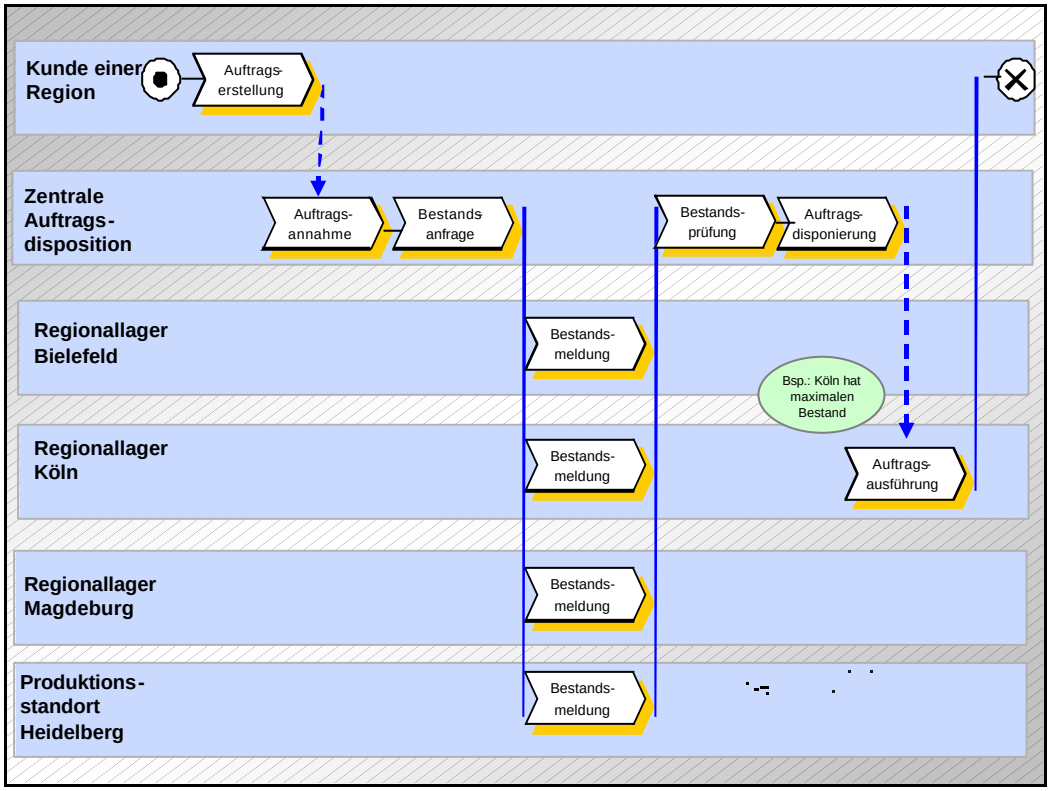


Abbildung 40: Prozesse der Auftragsdisposition der zweiten und dritten Dispositionsstrategie

3.2.2 Dispositionsstrategien des Elements Regionallager

Für das Element Regionallager wurden in zwei Bereichen erweiterte Strategien implementiert. Zum einen ist dies die Lagerverwaltung mit der Disponierung der Nachbestellungen eines Regionallagers, zum anderen wurden für die Kommissionierung und die Transportdisponierung erweiterte Strategien implementiert.

Für die Lagerverwaltung wurden die folgenden vier Strategien implementiert:

1. Die Nachbestellung der Regionallager erfolgt zu statischen Zeitpunkten und mit statischen Bestellmengen der einzelnen Artikel. Nach einer vorgegebenen Zykluszeit wird für jeden Artikel eine statische Menge des Artikels nachbestellt.
2. Die Nachbestellung der Regionallager erfolgt zu statischen Zeitpunkten und mit variablen Bestellmengen der einzelnen Artikel. Nach einer vorgegebenen Zykluszeit wird für jeden Artikel die Menge nachbestellt, die seit dem letzten Bestellzeitpunkt aus dem Lager abgezogen wurde.
3. Die Nachbestellung der Regionallager erfolgt zu variablen Zeitpunkten und mit statischen Bestellmengen der einzelnen Artikel. Sobald der vorhanden Bestand eines Artikels einen vorgegebenen Meldebestand unterschreitet, wird eine statische Menge des Artikels nachbestellt.
4. Die Nachbestellung der Regionallager erfolgt zu variablen Zeitpunkten und mit variablen Bestellmengen der einzelnen Artikel. Sobald der vorhandene Bestand eines Artikels einen vorgegebenen Meldebestand unterschreitet, wird die Differenz zwischen dem vorhandenen Bestand und einer vorgegebenen Bestellgrenze nachbestellt.

Für die Kommissionierung und die Transportdisponierung von den Regionallagern zu den Kunden wurden die folgenden Strategien implementiert:

1. Die Kommissionierung der Sendung für einen Kunden wird erst dann abgeschlossen und ein Transport zum Kunden angefordert, wenn das bereits kommissionierte Sendungsvolumen für den Kunden dem gesamten Ladevolumen eines LKW entspricht.
2. Die Kommissionierung der Sendung für einen Kunden wird dann abgeschlossen und ein Transport zum Kunden angefordert, wenn das bereits kommissionierte Sendungsvolumen ein parametrierbares Mindestvolumen überschritten hat, keine weiteren Aufträge des Kunden vorliegen und seit dem Beginn der Sendungskommissionierung eine parametrierbare maximale Wartezeit abgelaufen ist.
3. Die Kommissionierung der Sendung für einen Kunden wird dann abgeschlossen und ein Transport zum Kunden angefordert, wenn keine weiteren Aufträge des Kunden vorliegen und seit dem Beginn der Sendungskommissionierung eine parametrierbare maximale Wartezeit abgelaufen ist.

3.2.3 Dispositionsstrategien des Elements Produktionsstandort

Die Produktion der Artikel selbst liegt außerhalb der für die Untersuchung betrachteten Systemgrenzen. Die für die Simulation verfügbaren Eingabedaten des Produktionsprogramms der Produktionsstandorte bauen auf dem eingangs beschriebenen ursprünglichen Regionallagerkonzept auf. Da die untersuchte Beschaffungskette durch den Einsatz verschiedener Strategien ihr Zeitverhalten ändert, ist es erforderlich, auch für die Erstellung des Produktionsprogramms verschieden einfache Strategien zu implementieren. Durch dies kann das Produktionsprogramm der beiden Produktionsstandorte dem geänderten Zeitverhalten der Beschaffungskette angepasst werden. Die folgenden Strategien wurden hierfür implementiert:

1. Das Produktionsprogramm basiert auf den vorgegebenen Eingangsdaten. Die an einem Werktag produzierten Artikel stehen ab dem Morgen um 6:00 Uhr des Folgetages vollständig zur Verfügung.
2. Das Produktionsprogramm basiert auf den vorgegebenen Eingangsdaten. Die an einem Werktag produzierten Artikel werden mit einem berechneten Takt über den Tag verteilt produziert. Der Takt berechnet sich aus dem Quotienten (Artikelanzahl) / (Produktionszeit).
3. Das Produktionsprogramm ist flexibel. Produziert werden nur die Artikel, die am Vortag von den Regionallagern bestellt wurden. Die produzierten Artikel werden mit einem berechneten Takt über den Tag verteilt produziert Der Takt berechnet sich aus dem Quotient (Zu produzierende Artikel) / (Produktionszeit).

Die Differenzierung der ersten beiden Strategien dient der Beeinflussung der Rechenzeit während der Simulation. Durch den Einsatz der zweiten Strategie wird während der Simulation eine gleichmäßigere Ereignisdichte erzielt. Aus logistischer Sicht sind die beiden Strategien gleichwertig.

Für die Kommissionierung und die Transportdisponierung von den Produktionsstandorten zu den Regionallagern wurden die folgenden Strategien implementiert:

1. Die Kommissionierung der Sendung für einen Kunden wird erst dann abgeschlossen und ein Transport zum Kunden angefordert, wenn das bereits kommissionierte Sendungsvolumen für den Kunden dem gesamten Ladevolumen eines LKW entspricht.
2. Die Kommissionierung der Sendung für einen Kunden wird dann abgeschlossen und ein Transport zum Kunden angefordert, wenn das bereits kommissionierte Sendungsvolumen ein parametrierbares Mindestvolumen überschritten hat, keine weiteren Aufträge des Kunden vorliegen und seit dem Beginn der Sendungskommissionierung eine parametrierbare maximale Wartezeit abgelaufen ist.
3. Die Kommissionierung der Sendung für einen Kunden wird dann abgeschlossen und ein Transport zum Kunden angefordert, wenn keine weiteren Aufträge des Kunden

vorliegen und seit dem Beginn der Sendungskommissionierung eine parametrierbare maximale Wartezeit abgelaufen ist.

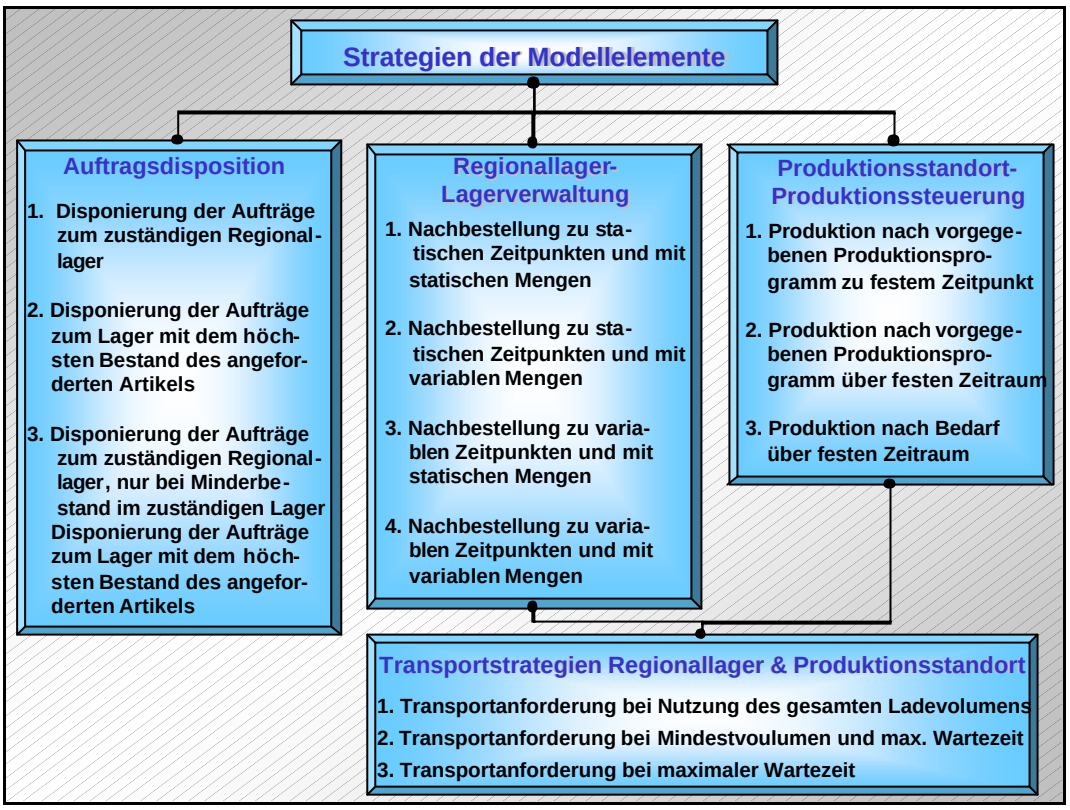


Abbildung 41: Übersicht der implementierten Strategien

4 Vorstellung des erstellten Modells

Die Abbildung des in Kapitel 3 beschriebenen Systems im *promodel*-Simulator erfolgte durch die Abbildung der einzelnen Systemelemente in abgeschlossenen Funktionseinheiten. Die Modellierung der Funktionseinheiten erfolgt, wie in Kapitel 2.4.1 beschrieben, durch das Einsetzen von Objekten aus abgeleiteten Instanzen der Modellierungselemente in das Modell. Die Ablauflogik einer Funktionseinheit wird durch die Programmierung von Prozessen abgebildet. Eine Funktionseinheit besteht immer mindestens aus Objekten einer Instanz des Modellierungselement „Location“ und einer den „Location“-Objekten zugeordneten Instanz des Modellierungselements „Process“.

Dies bedeutet, dass die modellierten Funktionseinheiten keine Bestandteile des *promodel*-Simulators sind. Eine Funktionseinheit ist nur eine Bezeichnung für die Gruppierung mehrerer „Processes“ und „Locations“, die in einem logischen Zusammenhang stehen. Die einzelnen Funktionseinheiten sind untereinander nicht hierarchisierbar.

4.1 Beschreibung der Gesamtstruktur des Modells

Die Gesamtstruktur des Modells bildet die Elemente der in **Abbildung 37** dargestellten Beschaffungskette auf der Basis eines Regionallagerkonzepts ab. Für die weiteren Betrachtungen werden die Regionallager als Ausgangspunkt gewählt. Die Beschaffung der Artikel durch die Regionallager stellt somit die Beschaffungsseite des Modells dar. In dieser Sichtweise wird die Beschaffung der Artikel durch die Händler zu einer Auslieferung von Artikeln von den Regionallagern weg zu den Händlern. Diese Seite der Beschaffungskette wird daher in den weiteren Betrachtungen als Distribution bezeichnet. Die Einführung dieser Sichtweise dient der einfacheren sprachlichen Bezeichnung der beiden Stufen der Beschaffungskette im Modell.

Die Gesamtstruktur des Modells ist in der folgenden **Abbildung 42** dargestellt. In dieser Darstellung wurde auf die Abbildung der modellierten Prozesse verzichtet, da eine Darstellung dieser die Abbildung zu unübersichtlich machen würde. Dargestellt sind die Symbole der Objekte aus Instanzen des Modellierungselements „Location“, aus welchen die Funktionseinheiten gebildet werden. Außerdem sind die Symbole der eingesetzten Objekte aus Instanzen des Modellierungselements „Resource“ dargestellt. Diese Objekte repräsentieren die für den Gütertransport eingesetzten LKW. Hierbei wird durch den Einsatz von zwei Instanzen des Modellierungselements „Resource“ eine Trennung der Transportmittel für die Beschaffungs- und die Distributionsseite geschaffen. Dies ermöglicht, die in den einzelnen Stufen der Beschaffungskette durchgeführten Transporte getrennt zu analysieren. Die Gesamtheit dieser Objekte wird

im Modell durch die Funktionseinheiten „Spedition international“ und „Spedition national“ abgebildet.

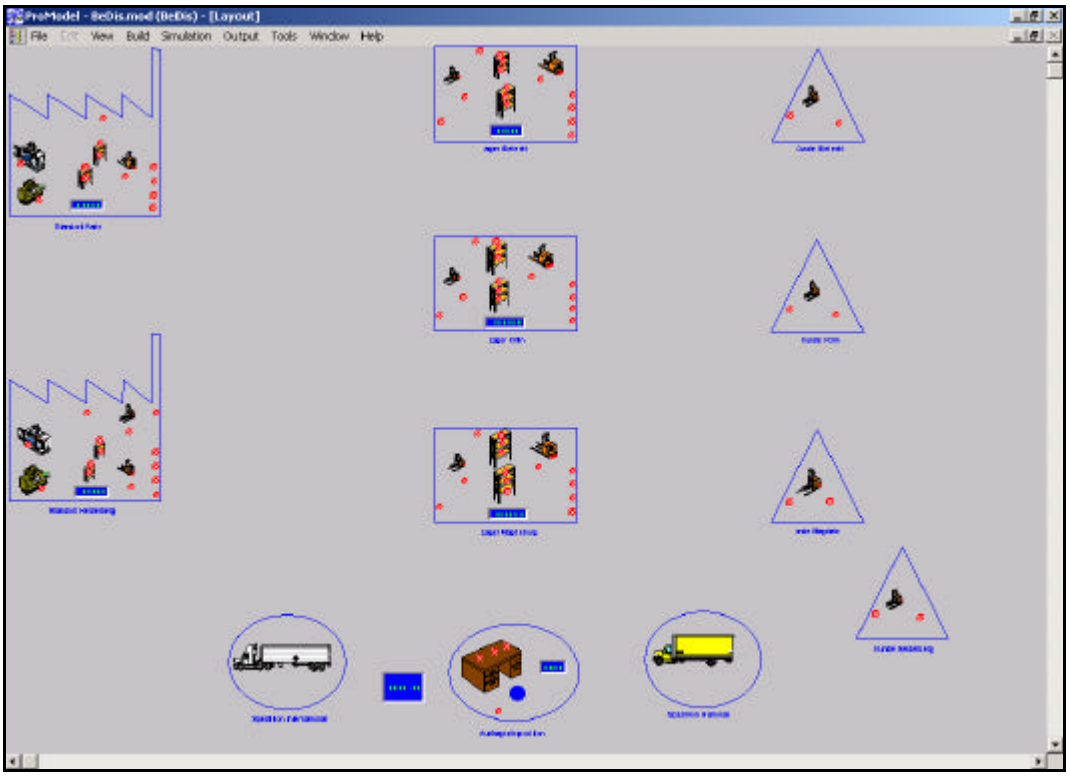


Abbildung 42: Gesamtstruktur des erstellten Modells

4.2 Beschreibung der Funktionseinheiten

Im Folgenden werden die Aufgaben und Abläufe der einzelnen modellierten Funktionseinheiten näher erläutert.

Das Modell setzt sich aus den modellierten Funktionseinheiten Auftragsdisposition, Regionallager, Produktion, Spedition und Kunde zusammen. Diese Funktionseinheiten umfassen die zur Abbildung des entsprechenden Systemelements eingesetzten und in einem logischen Zusammenhang stehenden Objekte.

Die Abläufe in den Funktionseinheiten werden durch die Abbildung von Flussdiagrammen verdeutlicht. Die in diesen Flussdiagrammen verwendeten Symbole sind in der folgenden **Abbildung 43** dargestellt.

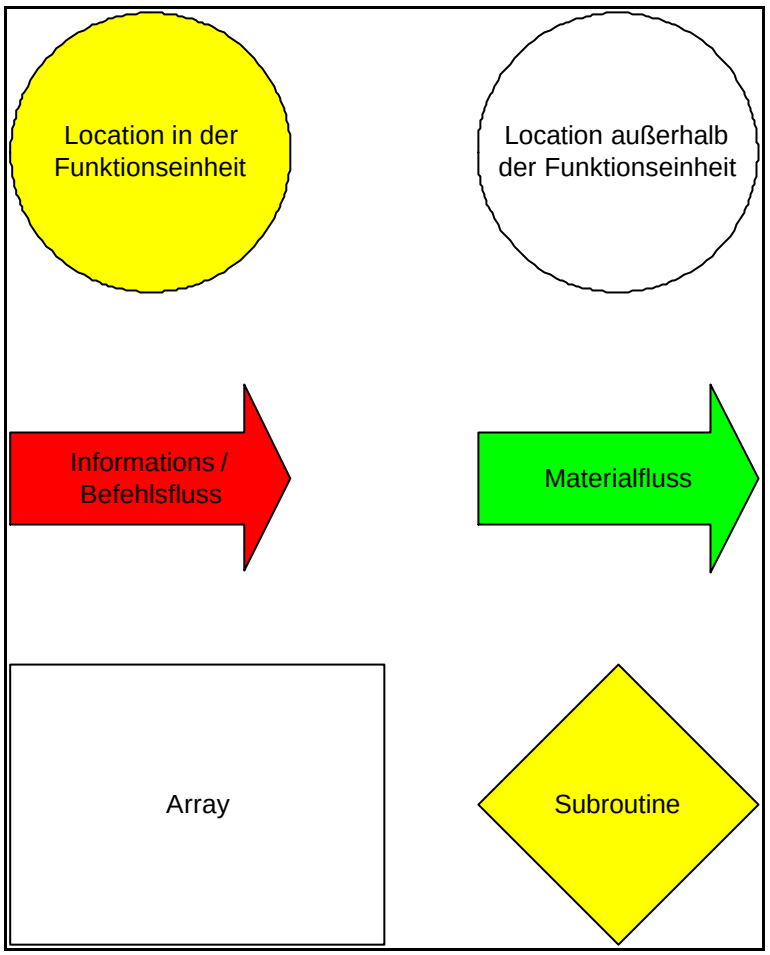


Abbildung 43: Übersicht der verwendeten Symbole

Das Element „Location innerhalb der Funktionseinheit“ repräsentiert ein Objekt einer Instanz des Modellierungselements „Location“, welches logisch der jeweiligen Funktionseinheit zugeordnet ist.

Das Element „Location außerhalb der Funktionseinheit“ repräsentiert ein Objekt einer Instanz des Modellierungselements „Location“, welches der jeweiligen Funktionseinheit nicht zugeordnet ist.

Das Element „Informations- / Befehlsfluss“ repräsentiert den Datenaustausch innerhalb der ausgeführten Prozesse bzw. den Aufruf von programmierten Subroutinen oder ähnlichem.

Das Element „Materialfluss“ repräsentiert die Bewegung von Objekten einer Instanz des Modellierungselements „Entity“.

Das Element „Array“ repräsentiert ein modelliertes Datenfeld, aus welchem Daten ausgelesen bzw. in das Daten geschrieben werden.

Das Element „Subroutine“ repräsentiert die Programmierung einer erstellten Instanz des Modellierungselements „Subroutine“.

Im Folgenden werden die Aufgaben und Abläufe der eingangs bezeichneten modellierten Funktionseinheiten beschrieben.

4.2.1 Funktionseinheit Auftragsdisposition

Die Funktionseinheit „Auftragsdisposition“ übernimmt im Modell die Aufgaben der Auftragsverwaltung. In dieser Funktionseinheit gehen die verschiedenen Kundenaufträge ein und werden durch diese disponiert. Hierbei hat der Anwender die Möglichkeit, zwischen den in Kapitel 3.2.1 beschriebenen Dispositionsstrategien zu wählen.

Die Symbole der in der Funktionseinheit „Auftragsdisposition“ zusammengefassten Objekte von Instanzen des Modellierungselements „Location“ sowie die grafischen Symbole zur Abgrenzung der Funktionseinheit sind in der folgenden **Abbildung 44** dargestellt.

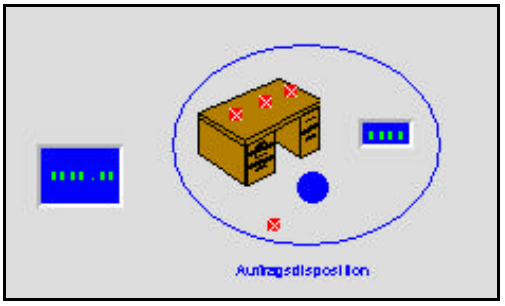


Abbildung 44: Symbol der Auftragsdisposition

Die Funktionseinheit „Auftragsdisposition“ umfasst die „Locations“ *Auftragsdisposition* und *Initialisierung Auftragsdisposition*. Der modellierte Counter dient zur Visualisierung der Auftragsanzahl, welche sich in der Bearbeitungsqueue der „Auftragsdisposition“ befinden. Das modellierte Anzeigeelement neben der Funktionseinheit visualisiert das Datum des aktuellen Simulationszeitpunktes.

Der logische Ablauf in der Funktionseinheit „Auftragsdisposition“ ist in **Abbildung 45** dargestellt.

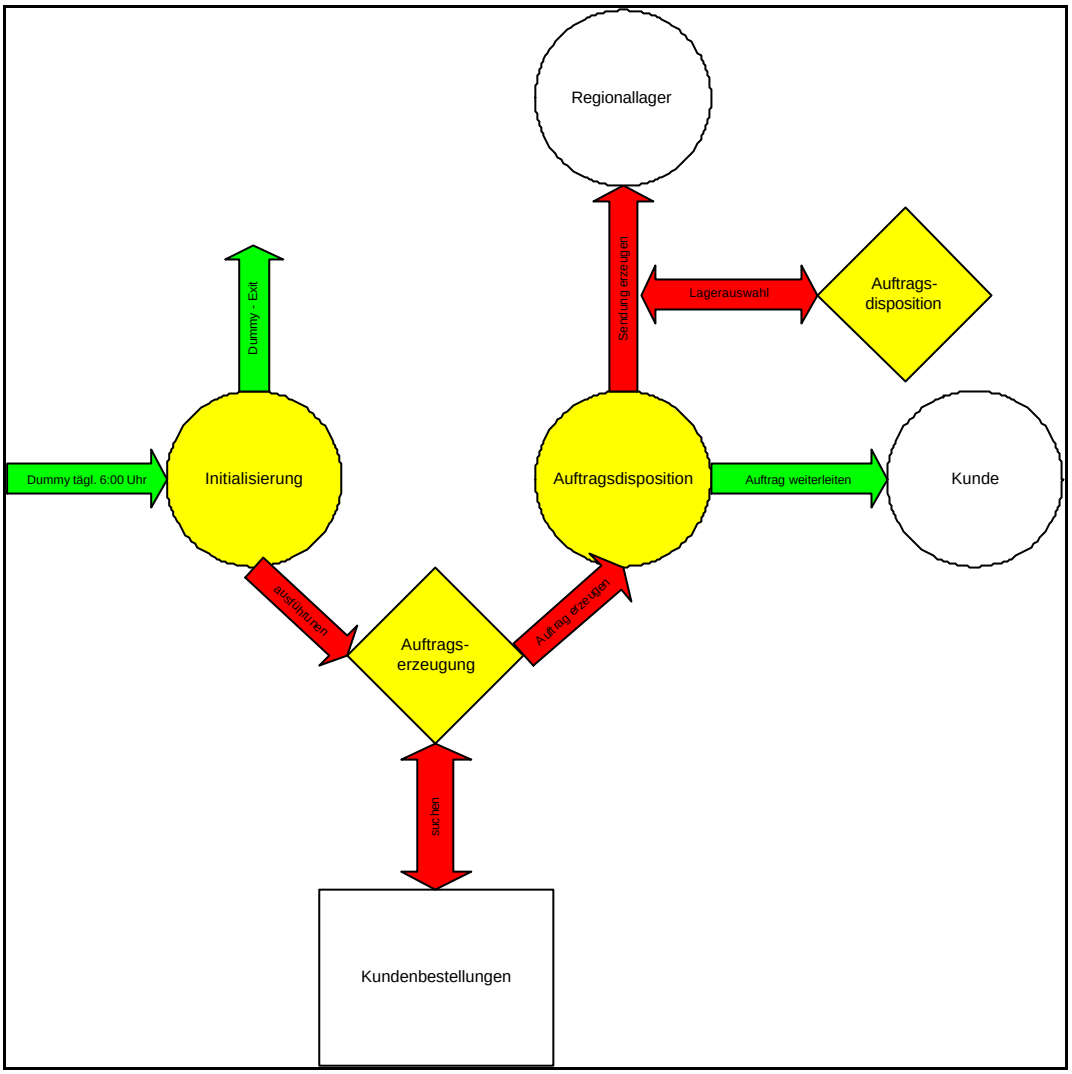


Abbildung 45: Ablauf in der Funktionseinheit „Auftragsdisposition“

Täglich (bezogen auf die Simulationszeit) um 6:00 Uhr wird in der „Location“ *Initialisierung* ein Objekt der Instanz *Dummy* des Modellierungselements „Entity“ erzeugt. Die Erzeugung dieses Objekts bewirkt die Ausführung der „Subroutine“ *Auftragserzeugung*. Die programmierte Logik dieser „Subroutine“ durchsucht das „Array“ *Kundenbestellungen* nach Kundenaufträgen, die am aktuellen Simulationstag eingehen, und erzeugt diese Aufträge mit den benötigten Attributen in der „Location“ *Auftragsdisposition*.

Der Prozess, welcher der „Location“ *Auftragsdisposition* zugeordnet ist, speichert im „Array“ *Kundenbestellungen* die Ankunftszeit (Erzeugungszeitpunkt) des erzeugten Auftrages ab. Danach wird die Prozessausführung für die Dauer der Auftragsbearbeitung unterbrochen. Diese Auftragsbearbeitungszeit ist frei parametrierbar. Nach Ablauf der Auftragsbearbeitungszeit stößt der Prozess - je nach parametrierter Auftragsdispositionsstrategie - die Ausführung einer der Subroutinen *Auftragsdisposition 1-3* an. Durch die programmierte Logik dieser Subroutinen wird das Regionallager ausgewählt, durch welches die Befriedigung des aktuellen Kundenauftrags erfolgen soll. Nach der

Auswahl des Lagers wird im Warenausgang des Lagers ein Objekt der Instanz *Sendung* des Modellierungselements „Entity“ erzeugt. Auf dieses Objekt werden die Attribute des Kundenauftrags kopiert. Das Auftragsobjekt wird nun an den Kunden, der den Auftrag erteilt hat, weitergeleitet. Dies dient dazu, die Auftragsdurchlaufzeit zu ermitteln (weiteres in der Beschreibung der Funktionseinheit „Kunde“).

4.2.2 Funktionseinheit Regionallager

Die Funktionseinheit „Regionallager“ bildet im Modell die Regionallager der in Kapitel 3 beschriebenen Beschaffungskette ab. Die Funktionseinheit dient der Bevorratung von Artikeln, welche durch die Händler über die Regionallager beschafft werden. Die Funktionseinheit umfasst neben der Bevorratung der Artikel auch die Strategien zur Beschaffung dieser über die beiden Produktionsstandorte sowie die Strategien zur Sendungskommissionierung und Transportinitialisierung für die Lieferung der bestellten Artikel. Der Anwender hat für die oben genannten Strategien die Möglichkeit, zwischen den in Kapitel 3.2.2 beschriebenen Lagerverwaltungs- und Dispositionsstrategien zu wählen.

Die Symbole der in der Funktionseinheit „Regionallager“ zusammengefassten Objekte von Instanzen des Modellierungselements „Location“ sowie die grafischen Symbole zur Abgrenzung der Funktionseinheit sind in der folgenden **Abbildung 46** dargestellt.

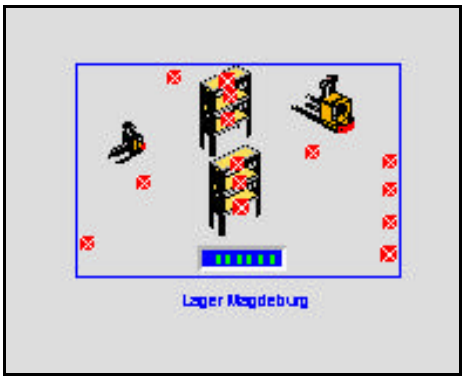


Abbildung 46: Symbol des Regionallagers

Die modellierten Funktionseinheiten „Regionallager“ setzen sich aus Objekten der Instanzen *Initialisierung*, *Lager*, *Warenannahme*, *Einlagerung*, *Kommissionierung* und *Warenausgabe* des Modellierungselements „Location“ zusammen. Aus der Instanz *Warenausgabe* sind vier Objekte für die vier betrachteten Kundenregionen eingesetzt. Der in das Symbol integrierte Counter dient der Visualisierung der Artikelanzahl bzw. der Gesamtbestandshöhe der Funktionseinheit „Regionallager“.

Der logische Ablauf der Prozesse in der Funktionseinheit „Regionallager“ ist in **Abbildung 47** dargestellt.

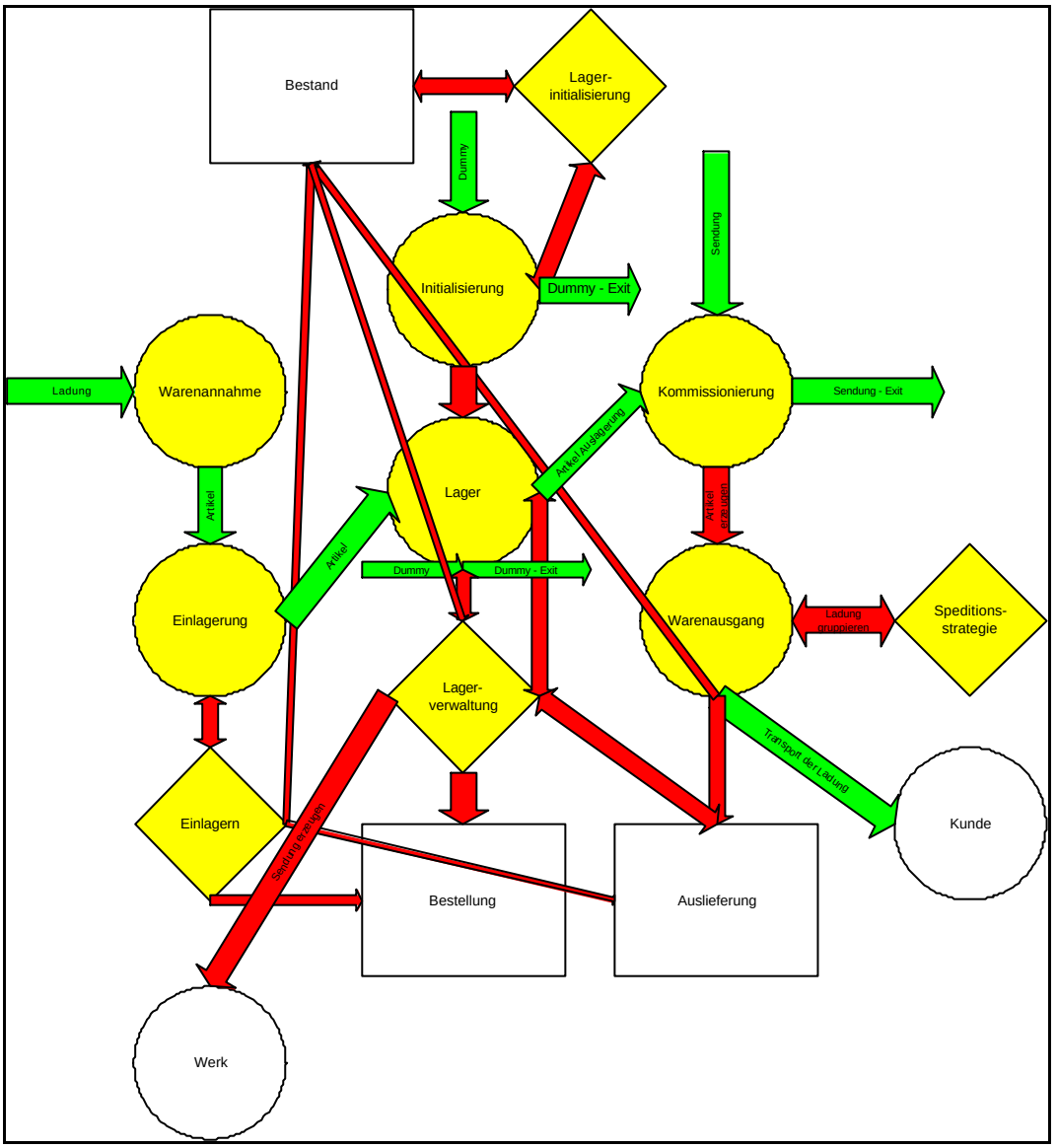


Abbildung 47: Ablauf in der Funktionseinheit „Regionallager“

Der oben symbolisierte Ablauf lässt sich wie folgt beschreiben:

Zu Beginn der Simulation wird in der „Location“ *Initialisierung* ein Objekt der Instanz *Dummy* des Modellierungselements „Entity“ erzeugt. Durch die Erzeugung dieses Objekts wird die Ausführung der Subroutine *Lagerinitialisierung* angestoßen. Diese Subroutine dient der Erzeugung von Anfangsbeständen in der Funktionseinheit „Regionallager“. Die Anfangsbestände werden dafür im jeweiligen „Array“ zur Beschreibung des Lagerbestandes (Lagerbestandsarray) angegeben. Die Subroutine ermittelt zur Erzeugung dieser Anfangsbestände die Bestandshöhe für jeden zu initialisierenden Artikel. Danach wird durch die Subroutine in der „Location“ Lager die ermittelte Artikelanzahl als Objekte der Instanz „Artikel“ des Modellierungselements „Entity“ erzeugt. Den erzeugten Objekten wird hierbei ein Attribut zur Ermittlung des jeweiligen Artikels mitgegeben.

Liegt der parametrierte Anfangsbestand eines bestimmten Artikels unter dem Meldebestand dieses Artikels, so wird bei Einsatz einer der Lagerverwaltungsstrategien drei oder vier unmittelbar nach der Initialisierung die Subroutine zur Nachbestellung der Artikel im Produktionsstandort Heidelberg angestoßen.

In der „Location“ *Kommissionierung* wird zur Übermittlung einer Bestellung durch die Funktionseinheit „Auftragsdisposition“ ein Objekt der Instanz *Sendung* des Modellierungselements „Entity“ erzeugt. Diesem Objekt wurden durch die Funktionseinheit „Auftragsdisposition“ alle für die Auftragsausführung erforderlichen Informationen als Attribute mitgegeben. Die Erzeugung des Objektes der Instanz *Sendung* in der „Location“ *Kommissionierung* führt zur Ausführung des dieser „Location“ zugeordneten Prozesses. Die in diesem Prozess programmierte Logik lagert die bestellte Anzahl an Artikeln aus dem Lager aus. Dies geschieht durch die Bewegung von Objekten der Instanz *Artikel* aus der „Location“ *Lager* auf das Objekt der Instanz *Sendung*. Der Prozess prüft vor der Objektbewegung, ob dass den Artikel beschreibende Attribut des Objektes der Instanz *Artikel* mit dem Attribut *Artikelnummer* der Bestellung übereinstimmt.

Zur Protokollierung der Auslagerereignisse wird durch den Prozess, welcher durch die Objekte der Instanz *Artikel* beim Verlassen der „Location“ *Lager* angestoßen wird, der Wert der Anzahl der Auslieferungen des betreffenden Artikels im „Array“ *Auslieferung* erhöht. Dieser Prozess vermindert auch den Wert des Bestands des betreffenden Artikels im „Array“ *Bestand*.

Wurden alle für eine Sendung benötigten Objekte der Instanz *Artikel* aus der „Location“ *Lager* ausgelagert, wurde eine Sendung somit vollständig kommissioniert, wird ein neuer Prozess angestoßen. Dieser wird durch das Objekt der Instanz *Sendung* angestoßen, welchem die ausgelagerten Objekte zugeordnet wurden. Durch den Prozess werden die Objekte der Instanz *Artikel* in der „Location“ *Warenausgabe* des bestellenden Kunden neu erzeugt. Dies hat den Vorteil, dass alle Attribute der Bestellung auf die neu erzeugten Objekte der Instanz *Artikel* kopiert werden. Das für die Kommissionierung verwendete Objekt der Instanz *Sendung* verlässt nun mit den ihm zugeordneten Objekten der Instanz *Artikel* das Modell.

Die in der „Location“ *Warenausgabe* erzeugten Objekte der Instanz *Artikel* bewirken - je nach Speditionsstrategie - die Ausführung der Subroutine *Speditionsstrategie Distribution 1 – 3*. Die programmierte Logik dieser Subroutine bewirkt, dass die in der „Location“ *Warenausgabe* vorhandenen Objekte der Instanz *Artikel* zu einem Objekt der Instanz *Ladung* des Modellierungselements „Entity“ zusammengefasst werden. Die Objektanzahl, welche zu einem neuen Objekt der Instanz *Ladung* zusammengefasst wird, variiert je nach verwendeter Strategie. Die Objektzusammenfassung erfolgt derart, dass die zusammengefasste Artikelanzahl entweder einer kompletten LKW-Ladung

entspricht oder nach Ablauf einer Mindestwartezeit der Anzahl der in der „Location“ *Warenausgabe* vorhandenen Objekte der Instanz *Artikel*.

Durch die Erzeugung des Objektes der Instanz *Ladung* wird ein Prozess angestoßen, welcher einen LKW anfordert. Die Ausführung des angestoßenen Prozesses wird nun solange unterbrochen, bis der angeforderte LKW in der Funktionseinheit „Regionallager“ eingetroffen ist. Nach Ablauf einer parametrierbaren Ladezeit wird das Objekt der Instanz *Ladung* auf das Objekt der Instanz *LKW* des Modellierungselements „Ressource“ aufgeladen. Die „Ressource“ führt anschließend den Transport des Objektes zur Funktionseinheit „Kunde“ durch.

Die Nachbestellungen der Funktionseinheiten „Regionallager“ werden bei den Lagerverwaltungsstrategien, die zu einem festen Zeitpunkt nachbestellen, durch ein Objekt der Instanz *Dummy* des Modellierungselements „Entity“ ausgelöst. Dieses Objekt wird für jeden Bestellzyklus in der Funktionseinheit Lager erzeugt. Durch die Erzeugung wird ein Prozess angestoßen, der die Subroutinen *Lagerverwaltungsstrategie 1* oder *2* ausführt. Diese Subroutinen erzeugen in den Funktionseinheiten „Produktionsstandort“ ein Objekt der Instanz *Sendung*. Diesem Objekt werden bei der Erzeugung zwei Attribute zugewiesen. Die beiden Attribute enthalten als Information die bestellte Menge und die Artikelnummer. Die bestellte Menge richtet sich dabei entweder nach der Vorgabe aus den Dispositionsdaten der Funktionseinheit „Regionallager“ oder nach der im Array *Auslieferung* gespeicherten Anzahl an ausgelieferten Artikeln. Diese werden bei einer Bestellung wieder auf den Wert „0“ gesetzt.

Erfolgt die Nachbestellung zu variablen Zeitpunkten, so wird von den Objekten der Instanz *Artikel* beim Verlassen der „Location“ *Lager* ein Prozess angestoßen, der die Subroutine *Lagerverwaltungsstrategie 3* oder *4* ausgeführt. Diese Subroutinen überprüfen, ob der Meldebestand des Artikels unterschritten ist, und ob im Array *Bestellungen* noch offene Bestellungen vorhanden sind.

Sind in dem Array keine offenen Bestellungen mehr vermerkt und wird der Meldebestand unterschritten, so führen die Subroutinen eine Nachbestellung durch. Hierzu wird durch die Subroutinen in der Funktionseinheit „Produktionsstandort Heidelberg“ eine Objekt der Instanz *Sendung* mit den Attributen der Bestellmenge und der zugehörigen Artikelnummer erzeugt. Des weiteren wird die Anzahl der bestellten Artikel im Array *Bestellungen* gespeichert.

Die „Location“ *Warenannahme* dient dazu, die durch das Objekt der Instanz *Ladung* gebildete Gruppe von Objekten der Instanz *Artikel* wieder aufzulösen. Durch diese Auflösung liegen in der „Location“ *Warenannahme* wieder einzelne Objekte der Instanz *Artikel* vor. Von dieser Location bewegen sich die Objekte in die „Location“

Einlagerung. Der dort ausgeführte Prozess wartet die parametrierte Einlagerzeit ab und ruft dann die Subroutine *Einlagern* auf.

Diese Subroutine erhöht im „Array“ *Bestand* den Bestand des jeweiligen Artikels und vermindert im „Array“ *Bestellung* die Anzahl der noch offenen bestellten Artikel. Danach bewegt sich das Objekt in die „Location“ Lager, wo es solange verweilt, bis eine Auslageranfrage gestellt wird.

Die Importartikel aus dem Produktionsstandort Paris werden bei diesen Lagerverwaltungsstrategien gesondert behandelt. Werden sie durch einen Kunden angefragt, so wird sofort ein Objekt der Instanz *Sendung* mit den Attributen der Bestellung in der Funktionseinheit „Produktionsstandort Paris“ erzeugt. Die Importartikel werden in den Regionallagern nicht bevorratet.

4.2.3 Funktionseinheit Produktionsstandort

Die Funktionseinheit „Produktionsstandort“ bildet im Modell die Produktionsstandorte der in Kapitel 3 beschriebenen Beschaffungskette ab. Die Funktionseinheit dient der Herstellung von Artikeln, welche durch die Funktionseinheiten „Regionallager“ bestellt werden. Die Funktionseinheit „Produktionsstandort Heidelberg“ umfasst neben der Produktion der Artikel auch eine Bevorratung der Artikel. In beiden modellierten Funktionseinheiten „Produktionsstandort“ sind die Strategien für die Produktion sowie die Strategien zur Sendungskommissionierung und Transportinitialisierung für die Lieferung der bestellten Artikel an die Regionallager implementiert. Der Anwender hat für die oben genannten Strategien die Möglichkeit, zwischen den in Kapitel 3.2.3 beschriebenen Produktions- und Dispositionsstrategien zu wählen.

Die Symbole der in der Funktionseinheit „Produktionsstandort“ zusammengefassten Objekte von Instanzen des Modellierungselements „Location“ sowie die grafischen Symbole zur Abgrenzung der Funktionseinheit sind in der folgenden **Abbildung 48** dargestellt.

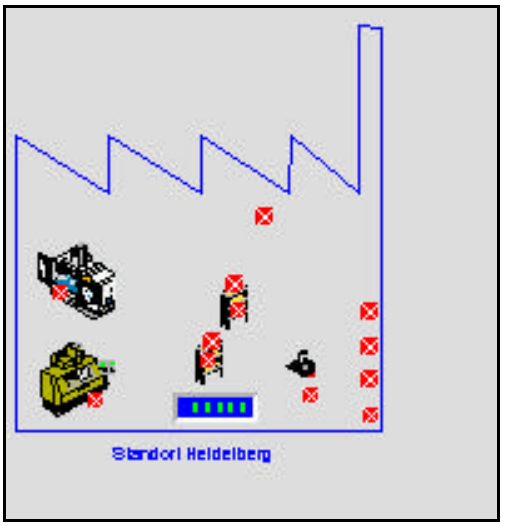


Abbildung 48: Symbol des Produktionsstandorts

Ein Produktionsstandort besteht aus Objekten der Instanzen *Werk*, *Lager*, *Initialisierung*, *Kommissionierung* und *Warenausgabe* des Modellierungselements „Location“. Aus der Instanz *Warenausgabe* wurden für die drei Regionallager drei Objekte modelliert. Der im Symbol der Funktionseinheit dargestellte Counter visualisiert den Gesamtbestand der im Werkslager bevorrateten Artikel.

Die Funktionseinheit „Produktionsstandort Paris“ verfügt über eine weiteres Objekt der Instanz *Warenausgabe* für die Belieferung des Werkslagers des Produktionsstandorts Heidelberg. Die Funktionseinheit „Produktionsstandort Heidelberg“ verfügt über ein zusätzliches Objekt der Instanz *Warenausgabe* um die Liefervorgänge in die Kundenregion Süd abzubilden. Diese Kundenregion wird direkt vom Werkslager beliefert. Außerdem verfügt das Funktionselement „Produktionsstandort Heidelberg“ über Objekte der Instanzen *Wareneingang* und *Einlagerung*. Diese Objekte sind erforderlich, um den Umschlag von Importartikeln aus dem Produktionsstandort Paris über das Werkslager des Produktionsstandorts Heidelberg abbilden zu können.

Der Ablauf in den beiden modellierten Produktionsstandorten ist in der folgenden **Abbildung 49** dargestellt.

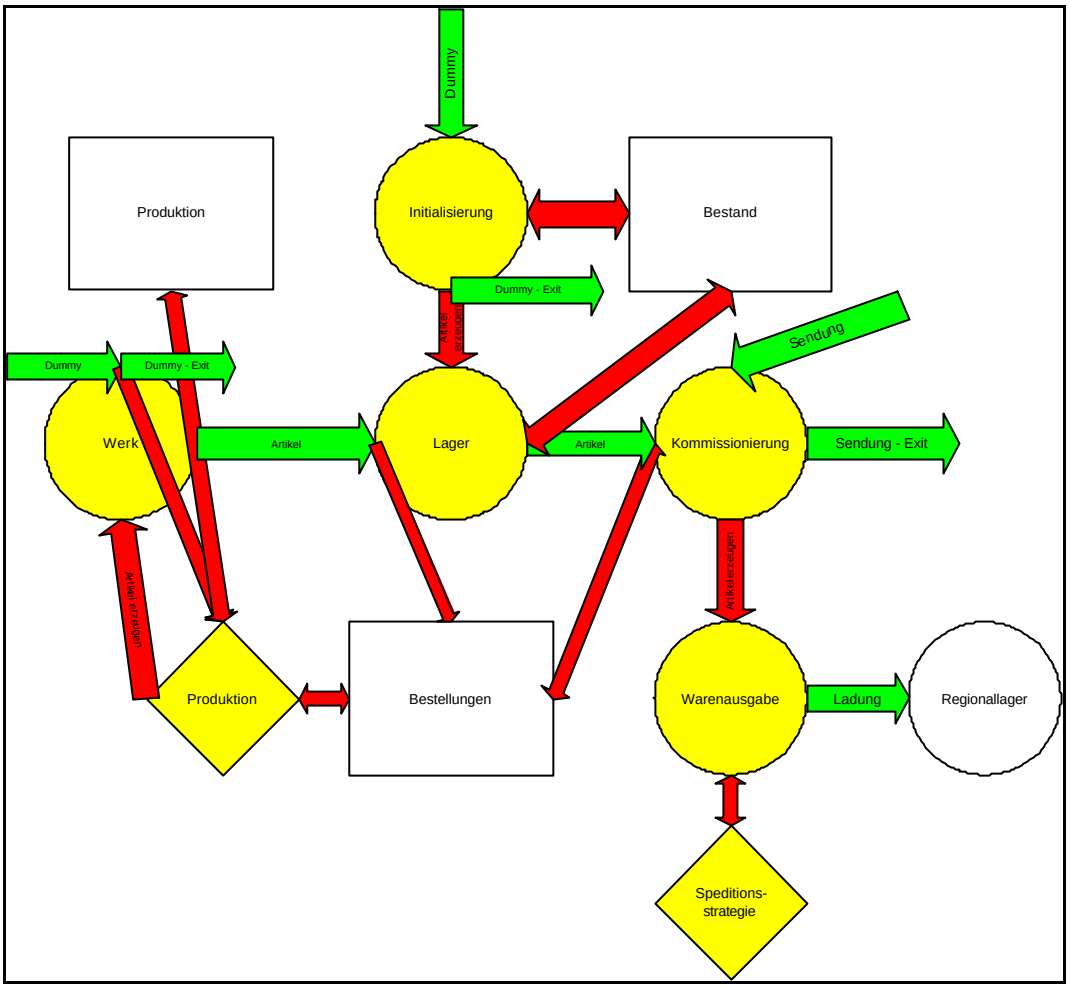


Abbildung 49: Ablauf in der Funktionseinheit „Produktionsstandort“

Die Initialisierung der Werkslager der Funktionseinheiten „Produktionsstandort“ erfolgt analog der Initialisierung der Funktionseinheiten „Regionallager“, jedoch erfolgen in den Werkslagern der Produktionsstandorte keine Nachbestellungen von Artikeln.

Auch die Kommissionierung der Bestellungen aus den Funktionseinheiten „Regionallager“ erfolgt ähnlich wie die Kommissionierung in den Regionallagern selbst. Das in der „Location“ *Kommissionierung* erzeugte Objekt der Instanz *Sendung* stößt einen Prozess an, der die Auslagerung von Objekten des angeforderten Artikels aus dem Werkslager durchführt. Die ausgelagerten Objekte werden dem Objekt der Instanz *Sendung* zugeordnet und so gruppiert, usw. Zusätzlich erhöht der Prozess den Wert der Anzahl der bestellten Artikel im „Array“ *Bestellungen*.

Im Werkslager des Produktionsstandorts Heidelberg wird mit Importartikeln aus dem Produktionsstandort Paris wie in den Regionallagern verfahren: Sie werden nicht bevorratet und bei einer Bestellung unmittelbar im Produktionsstandort Paris angefordert.

In den Funktionseinheiten „Produktionsstandort“ wird am Morgen jedes Simulationstages um 6:00 Uhr ein Objekt der Instanz *Dummy* des Modellierungsele-

ments „Entity“ erzeugt. Durch die Erzeugung dieses Objektes wird der Prozess zur Erstellung des Produktionsprogramms für den jeweiligen Tag ausgeführt. Dieser Prozess führt - je nach parametrierter Produktionsstrategie - eine der Subroutinen *Produktion 1* bis *3* aus.

Die programmierte Logik der Subroutine *Produktion 1* durchsucht das „Array“ *Produktion* nach allen an einem Tag zu produzierenden Artikeln und erzeugt alle benötigten Objekte der Instanz *Artikel* auf einmal. Die erzeugten Objekte werden mit dem Attribut Artikelnummer versehen um eine Unterscheidung verschiedener Artikel zu ermöglichen.

Die programmierte Logik der Subroutine *Produktion 2* durchsucht ebenfalls das „Array“ *Produktion* nach allen an einem Tag zu produzierenden Artikeln. Bevor die benötigten Objekte erzeugt werden, ermittelt diese Subroutine zunächst die Gesamtanzahl der an dem jeweiligen Tag zu produzierenden Artikel. Aus dieser Anzahl und der Produktionszeit errechnet die Subroutine eine Taktzeit für die Erzeugung der Objekte. Bei dieser Berechnung wird von einer Produktionszeit von 20 Stunden ausgegangen. Die benötigten Objekte werden dann durch die Subroutine mit der errechneten Taktzeit über den Tag verteilt erzeugt und mit Attributen versehen. Durch diese Methode der Objekterzeugung wird eine gleichmäßigere Ereignisdichte während der Simulation erreicht.

Bei Verwendung der Subroutine *Produktion 3* richtet sich das Produktionsprogramm nicht mehr nach den Importdaten, sondern nach den Bestellungen aus den Funktionseinheiten „Regionallager“, welche im Array *Bestellung* vermerkt sind. Die programmierte Logik der Subroutine ermittelt zuerst die Anzahl der am Vortag bestellten Artikel und berechnet daraus die nötige Taktzeit. Nun werden die Objekte zur Erzeugung der bestellten Artikel gemäß der Taktzeit erzeugt, und die Anzahl der Bestellungen im „Array“ *Bestellung* wieder auf den Wert „0“ gesetzt.

Aufgrund fehlender Eingangsdaten über die Produktionskapazitäten der beiden Produktionsstandorte wird bei einer Verwendung der Subroutinen *Produktion 2* und *Produktion 3* davon ausgegangen, dass für die Produktion genügend Kapazität zur Verfügung steht, um alle durch die Funktionseinheiten „Regionallager“ bestellten Artikel an einem Tag zu produzieren.

Nach der Erzeugung bewegen sich die Objekte der Instanz *Artikel* in die „Location“ *Werk*. In dieser „Location“ wird der Prozess zur Einlagerung der Objekte in das Werkslager angestoßen. Der Prozess erhöht zunächst im „Array“ Bestand die Anzahl der bevorrateten Artikel. Der Artikel wird jedoch nicht nur abstrakt eingelagert, sondern der Prozess bewegt das Objekt der Instanz *Artikel* in die „Location“ *Lager*. In dieser „Location“ wartet das Objekt, bis im Lager eine Auslageranfrage durch den der „Location“ *Kommissionierung* zugeordneten Prozess gestellt wird.

Nach der Kommissionierung, welche analog zu der Kommissionierung in den Funktionseinheiten „Regionallager“ erfolgt, werden die bei der Kommissionierung erstellten Objekte der Instanz *Ladung* durch Objekte der Instanz *LKW* des Modellierungselements „Ressource“ zu den modellierten Regionallagern transportiert. Die hierfür benötigten Ressourcen werden durch die Funktionseinheit „Spedition international“ bereitgestellt. Diese Funktionseinheit stellt die Ressourcen für alle Transporte der zweiten Stufe der Beschaffungskette bereit.

Einzige Ausnahme hiervon ist der Materialfluss vom Funktionselement „Produktionsstandort Heidelberg“ in die Kundenregion Süd. Da es sich hier um eine Direktbelieferung handelt, erfolgt der Transport - wie bei den Funktionseinheiten „Regionallager“ - durch die Ressourcen, die durch die Funktionseinheit „Spedition national“ bereitgestellt werden.

4.2.4 Funktionseinheit Spedition

Die Funktionseinheiten „Spedition international“ und „Spedition national“ stellen die für die Durchführung der Transporte zwischen den einzelnen Funktionseinheiten des Modells benötigten Ressourcen zur Verfügung. Diese Funktionseinheiten bilden den Logistik-Dienstleister ab, welcher die Transporte innerhalb der in Kapitel 3 beschriebenen Beschaffungskette durchführt.

Die Symbole der in der Funktionseinheit „Spedition“ zusammengefassten Objekte von Instanzen des Modellierungselements „Ressource“ sowie die grafischen Symbole zur Abgrenzung der Funktionseinheit sind in der folgenden **Abbildung 50** dargestellt.

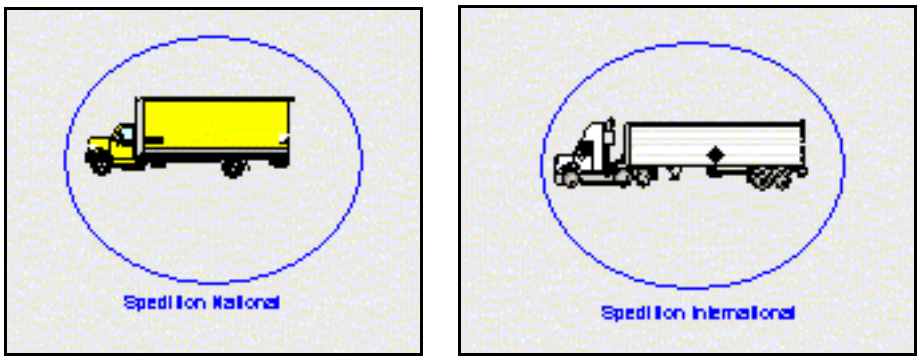


Abbildung 50: Symbole der Spedition

Die Funktionseinheiten „Spedition“ selbst beinhalten keine Objekte aus Instanzen des Modellierungselements „Location“. Sie setzen sich nur aus den Objekten der Instanz *LKW* des Modellierungselements „Ressource“ zusammen. Im Modell sind in beiden modellierten Speditionen jeweils zehn Objekte der Instanz *LKW* eingesetzt.

Die „Ressourcen“ bewegen sich auf so genannten Path Networks. Die Funktionseinheit „Spedition“ ist der „Heimatknoten“ der „Ressourcen“ auf ihrem Path Network.

Jede Funktionseinheit „Spedition“ verfügt über ein eigenes Path Network. Die Funktionseinheit „Spedition international“ verfügt über das Beschaffungsnetz und die Einheit „Spedition national“ über das Distributionsnetz.

In den Netzen sind verschiedene Entfernungen hinterlegt. Die Transportzeit ergibt sich durch die parametrierte Geschwindigkeit der Instanz *LKW*, die im Modell durch eine Normalverteilung dargestellt wird, und die zurückzulegende Entfernung. Die Berechnung der Transportzeiten erfolgt bei dem Einsatz von Objekten einer Instanz des Modellierungselements „Ressource“ automatisch durch den *promodel*-Simulator.

4.2.5 Funktionseinheit Kunde

Die Funktionseinheiten „Kunde“ repräsentiert die vier Kundenregionen der in Kapitel 3 beschriebenen Beschaffungskette.

Die Symbole der in der Funktionseinheit „Kunde“ zusammengefassten Objekte von Instanzen des Modellierungselements „Location“ sowie die grafischen Symbole zur Abgrenzung der Funktionseinheit sind in der folgenden **Abbildung 51** dargestellt.

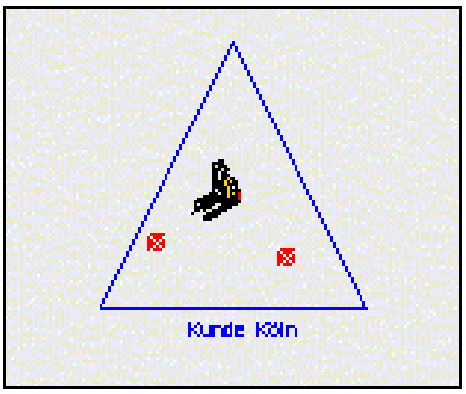


Abbildung 51: Symbol des Kunden

Die Funktionseinheiten „Kunde“ bestehen aus den „Locations“ *Warenannahme* und *Kunde*. Der Ablauf in der Funktionseinheit in und zwischen den „Locations“ ist in **Abbildung 52** dargestellt.

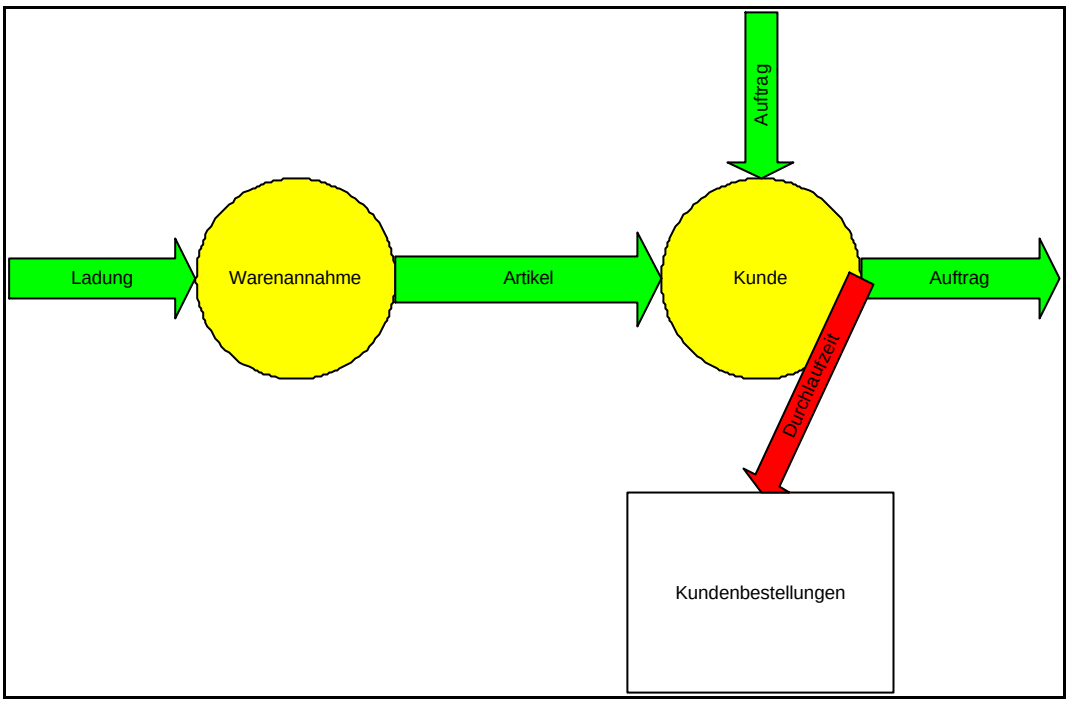


Abbildung 52: Ablauf in der Funktionseinheit „Kunde“

In der „Location“ *Warenannahme* wird die durch das Objekt der Instanz *Ladung* gebildete Gruppe von Objekten der Instanz *Artikel* des Modellierungselements „Entity“ aufgelöst, so dass wieder einzelne Objekte der Instanz *Artikel* vorliegen. Diese Objekte verbleiben in der „Location“ *Warenannahme* und warten dort, bis eine Ladeanfrage aus der „Location“ *Kunde* erfolgt.

In der „Location“ *Kunde* befinden sich die Objekte der Instanz *Auftrag*, die von der Funktionseinheit „Auftragsdisposition“ an die bestellende Funktionseinheit „Kunde“ zurückgesendet wurden. Durch diese Objekte wird ein Prozess ausgeführt, welcher die Zuordnung der Objekte der Instanz *Artikel* bewirkt, die über das Attribut Auftrags ID einem wartenden Objekt der Instanz *Auftrag* zugeordnet werden können („Load“ Befehl, der Auftrag „lädt“ die durch ihn bestellten Artikel).

Wurden einem Objekt der Instanz *Auftrag* sämtliche Objekte der Instanz *Artikel* zugeordnet, die durch dieses Objekt in der Funktionseinheit „Auftragsdisposition“ bestellt wurden, so verlässt das Objekt mit allen ihm zugeordneten Objekten der Instanz *Artikel* das System. Dabei wird ein Prozess angestoßen, der die Zeit, welche das Objekt der Instanz *Auftrag* im System verbracht hat, im „Array“ *Kundenbestellungen* speichert. Dieser Wert wird nicht nur in dem bezeichneten Array gespeichert, sondern zusätzlich mit einer Funktion des *promodel*-Simulators „geloggt“. Dieses „loggen“ der Zeit führt dazu, das die erfassten Auftragsdurchlaufzeiten bei der Auswertung der Simulationsdaten direkt im Abschlussbericht des *promodel*-Output Report vorliegen und ausgewertet werden können.

5 Beispieluntersuchungen

Allen für die Beispieluntersuchung gebildeten Szenarien liegen dieselben Modellparameter zugrunde, da nur die Auswirkungen unterschiedlicher Strategien untersucht werden. Die drei ausgewählten Szenarien unterscheiden sich nur in den gewählten Strategien für die Kommissionierung und die Transportanforderung (Speditionsstrategien).

Als verwendete Produktionsstrategie liegt allen Szenarien die Strategie 3 zugrunde, was bedeutet, dass nur die Artikel produziert werden, die am Vortag von den Regionallagern bestellt wurden.

Als Lagerverwaltungsstrategie liegt allen Szenarien die Strategie 4 zugrunde. Bei dieser Strategie erfolgt die Nachbestellung durch die Regionallager bei Unterschreitung des definierten Meldebestandes. Es wird die Differenz zwischen dem Ist-Bestand und der parametrierten Bestellgrenze beschafft.

Die Auftragsdispositionsstrategie aller Szenarien ist Strategie 3. Diese Strategie disponiert die Kundenaufträge so, dass prinzipiell das zugeordnete Regionallager gewählt wird. Nur bei Out-of-Stock Situationen, wenn also der im zugeordneten Regionallager vorhandene Bestand nicht ausreicht den Kundenwunsch zu befriedigen, werden die Aufträge an das Regionallager mit dem höchsten Bestand des bestellten Artikels weitergeleitet.

Simuliert wird ein Zeitraum von 45 Tagen, wobei an nur 31 Tagen Kundenaufträge eingehen. Die restliche Zeit dient dem Nachlaufen des Systems.

5.1 Untersuchungsszenario 1

5.1.1 Beschreibung des Untersuchungsszenarios 1

Im ersten Szenario wird als verwendete Speditionsstrategie die Strategie 1 benutzt. Die Regionallager fordern erst einen LKW an, wenn in der Warenausgabe für den entsprechenden Kunden Sendungen vorhanden sind, die eine vollständige Beladung eines LKW ermöglichen.

Es existiert keine Höchstgrenze für die Wartezeit, die Sendungen in der Warenausgabe eines Regionallagers verbringen, bis ein LKW angefordert wird.

Diese Strategie gilt sowohl für die Distributions- als auch für die Beschaffungsspedition.

5.1.2 Ziel des Untersuchungsszenarios 1

Ziel dieses Szenarios ist es, die Transportkosten der Distributionslogistik und der Beschaffungslogistik der Regionallager zu minimieren.

5.1.3 Auswertung des Untersuchungsszenarios 1

Bei Verwendung der Speditionsstrategie 1 für die Distributions- und die Beschaffungsspedition belaufen sich die Transportkosten für die Beschaffung der Regionallager auf 870.119 €. Die Transportkosten für die Distribution zum Kunden belaufen sich auf 524.899 €.

Die Lieferzeit liegt bei durchschnittlich 51,5 Stunden, jedoch ergibt sich ein hoher Maximalwert für die Lieferzeit von 586,4 Stunden.

Aufgrund des nicht Zustandekommens von vollständigen Ladungen sind 64 Aufträge nicht ausgeführt worden. Die nicht ausgeführten Aufträge lassen sich im „Array“ *Validierung Kundenbestellungen* erkennen, in welchem die Durchlaufzeit jedes einzelnen Auftrages gespeichert wird.

Die folgenden Abbildungen (**Abbildung 53** bis **Abbildung 56**) zeigen die Ergebnisdaten dieses Untersuchungsszenarios.

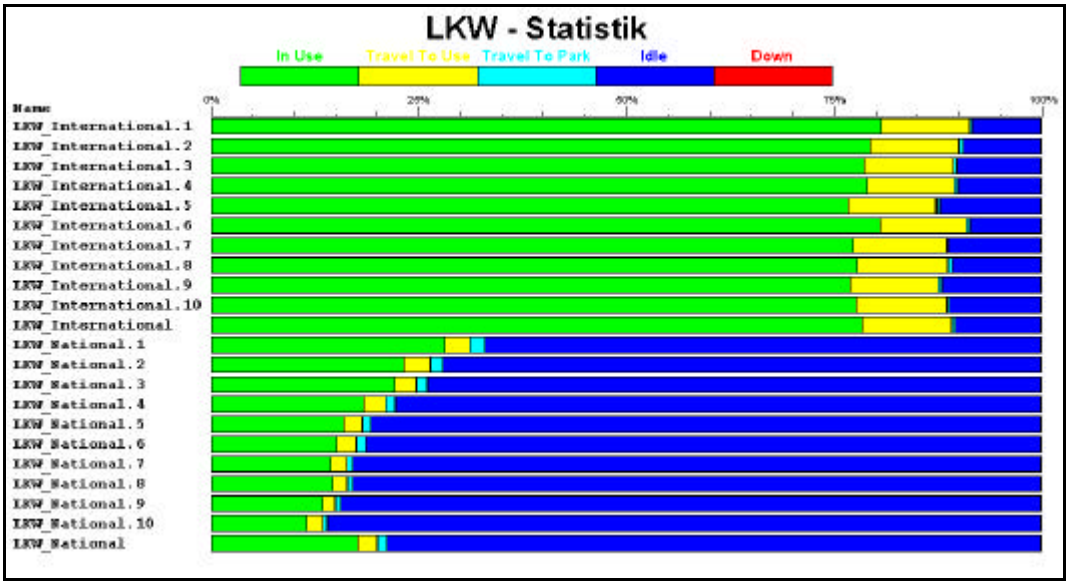


Abbildung 53: LKW – Statistik im Untersuchungsszenario 1

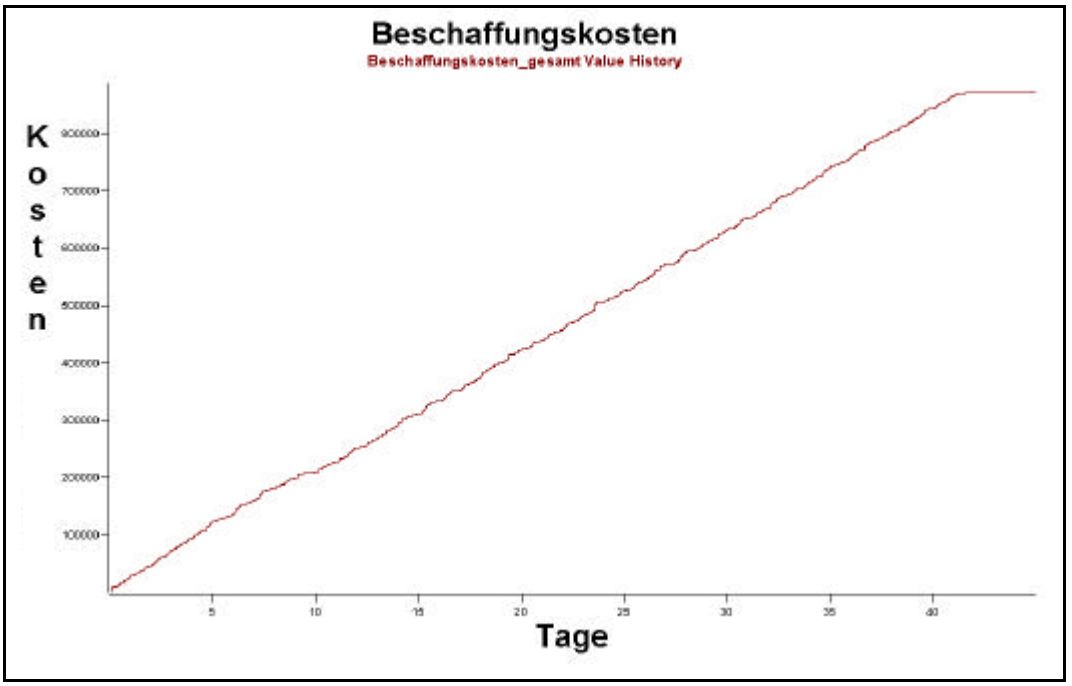


Abbildung 54: Beschaffungskosten im Untersuchungsszenario 1

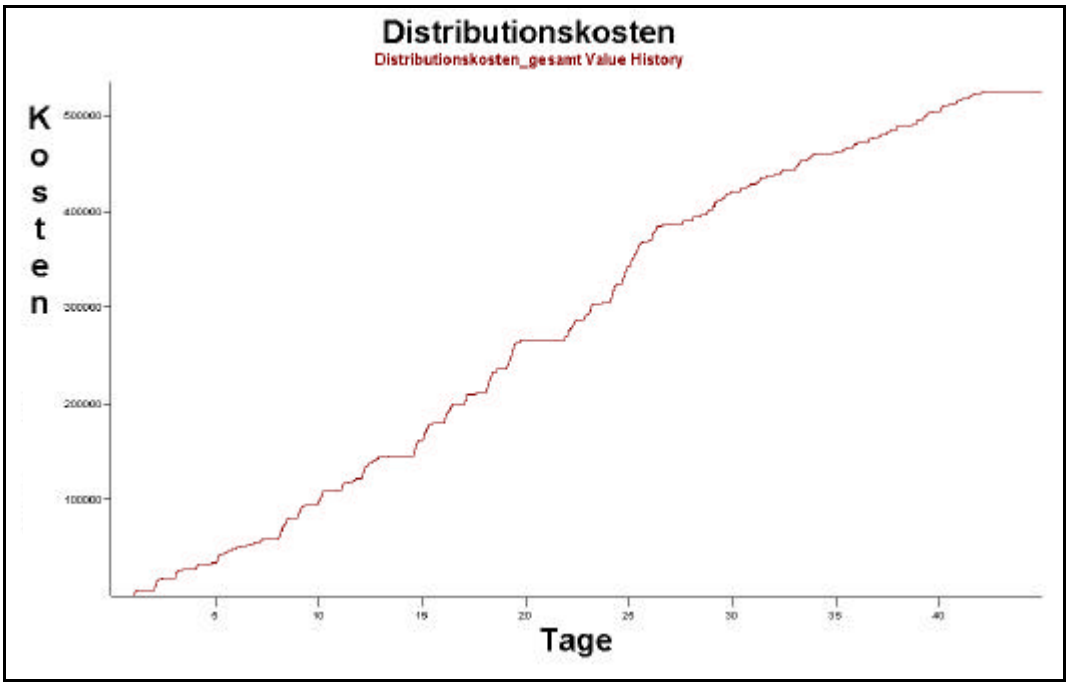


Abbildung 55: Distributionskosten im Untersuchungsszenario 1

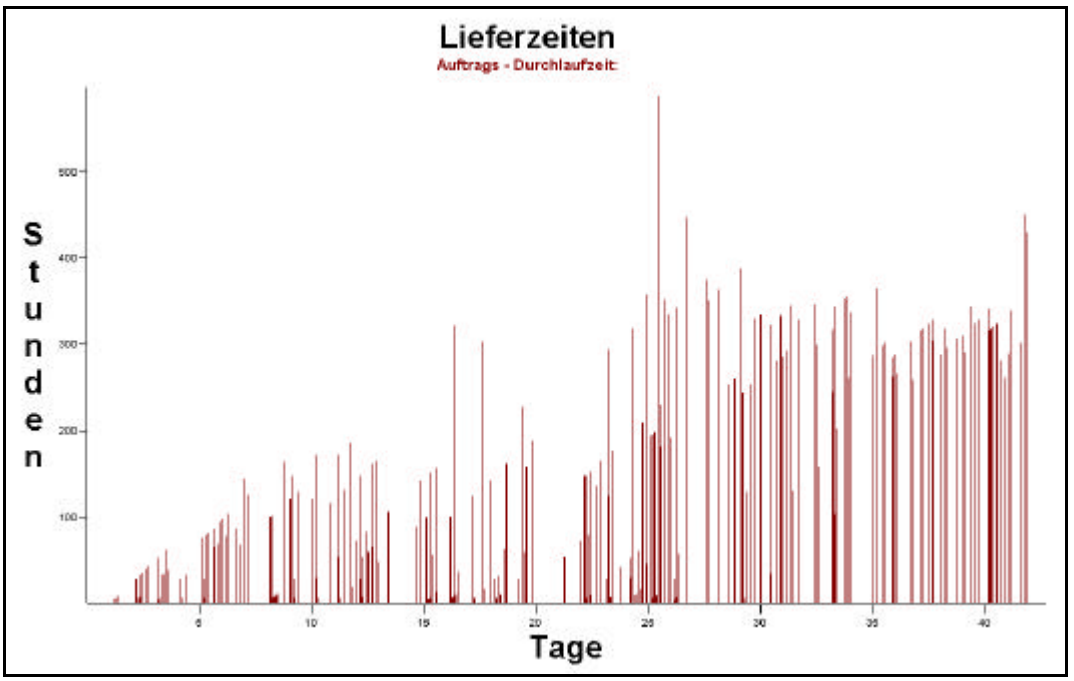


Abbildung 56: Lieferzeiten im Untersuchungsszenario 1

5.2 Untersuchungsszenario 2

5.2.1 Beschreibung des Untersuchungsszenarios 2

Im zweiten Szenario wird als Speditionsstrategie für die Transportanforderung die Strategie 2 benutzt. Bei Verwendung dieser Strategie wird durch die Regionallager erst einen LKW angefordert, wenn in der Warenausgabe für den entsprechenden Kunden eine Mindestsendungsmenge vorhanden ist und zusätzlich eine maximale Wartezeit überschritten ist.

Die Strategie gilt sowohl für die Distributions- als auch für die Beschaffungsspedition.

5.2.2 Ziel des Untersuchungsszenarios 2

Ziel des Szenarios ist es, geringe Transportkosten bei niedrigen Durchlaufzeiten zu realisieren.

5.2.3 Auswertung des Untersuchungsszenarios 2

Bei Verwendung der Speditionsstrategie 2 belaufen sich die Transportkosten für die Beschaffung der Regionallager auf 855.155 €. Die Transportkosten für die Distribution zum Kunden belaufen sich auf 549.585 €.

Die Lieferzeit liegt bei durchschnittlich 91,5 Stunden, jedoch ergibt sich ein hoher Maximalwert für die Lieferzeit von 685,5 Stunden.

Aufgrund des nicht Zustandekommens von Mindesttransportmengen sind Aufträge nicht ausgeführt worden. Die Anzahl der nicht ausgeführten Aufträge ist jedoch geringer als ihre Anzahl bei Verwendung der Speditionsstrategie 1, was zum Anstieg der maximalen und der durchschnittlichen Lieferzeiten führte.

Die nicht ausgeführten Aufträge lassen sich im „Array“ *Validierung Kundenbestellungen* erkennen, in welchem die Durchlaufzeit jedes einzelnen Auftrages gespeichert wird.

Die folgenden Abbildungen (**Abbildung 57** bis **Abbildung 60**) zeigen einige der Ergebnisdaten dieses Untersuchungsszenarios.

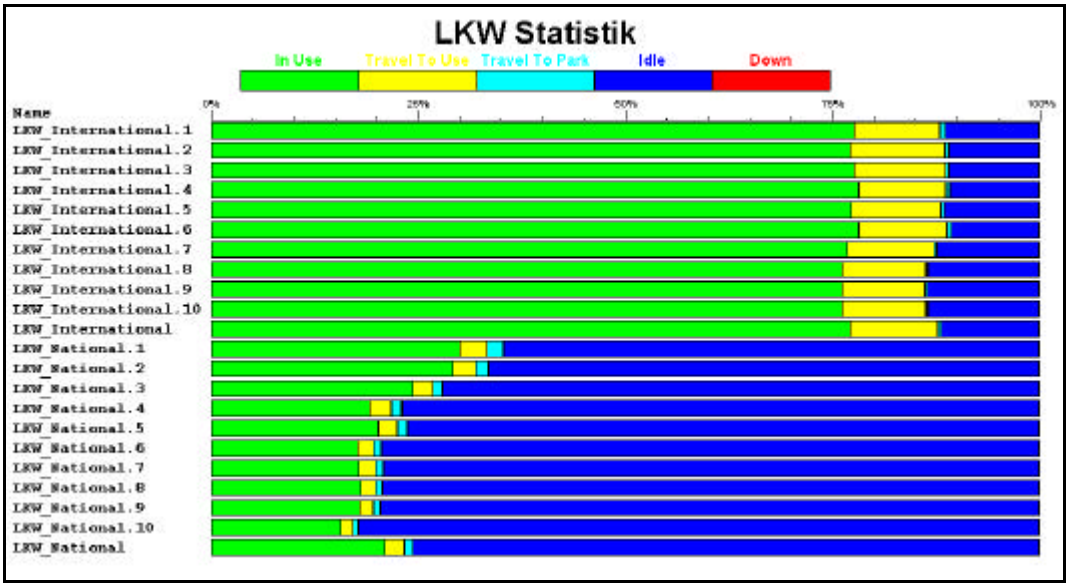


Abbildung 57: LKW Statistik im Untersuchungsszenario 2

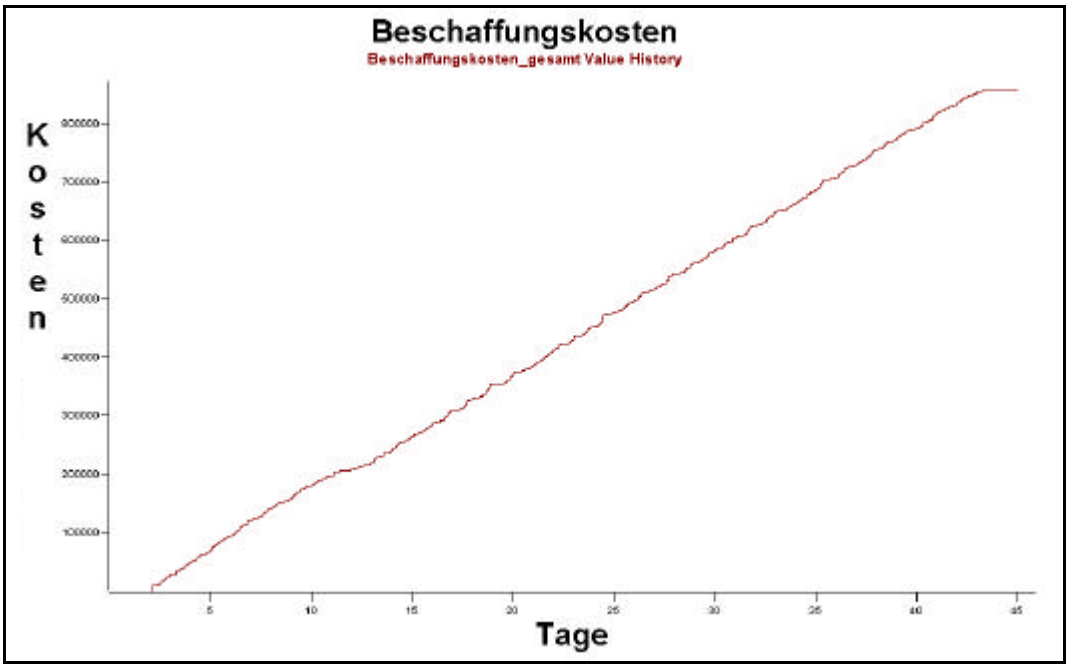


Abbildung 58: Beschaffungskosten im Untersuchungsszenario 2

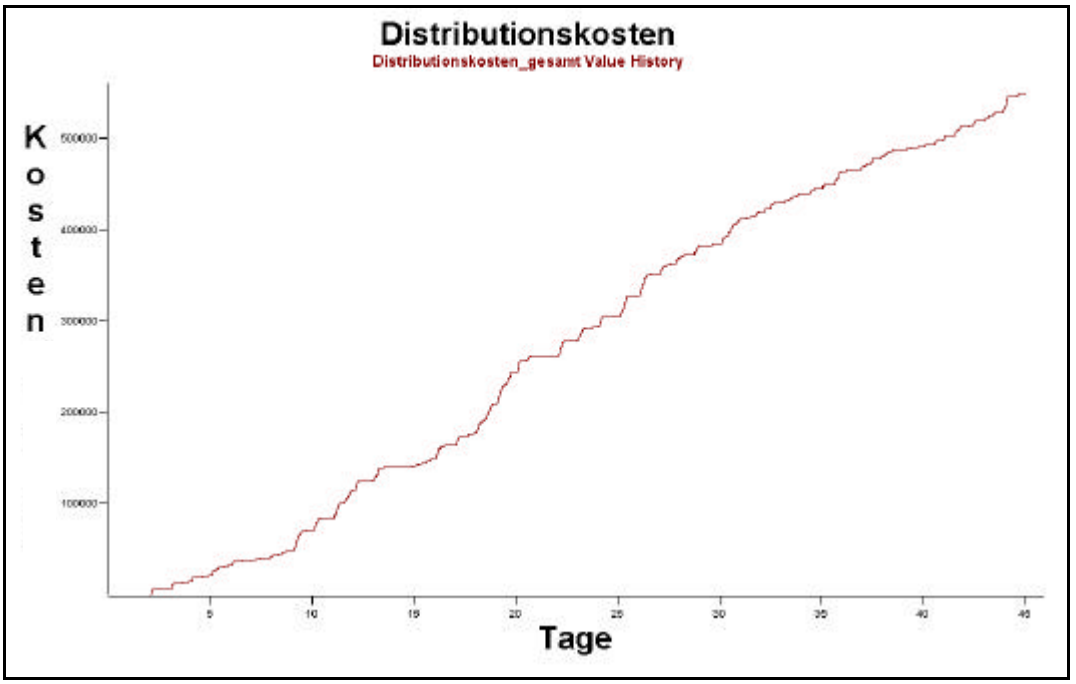


Abbildung 59: Distributionskosten im Untersuchungsszenario 2

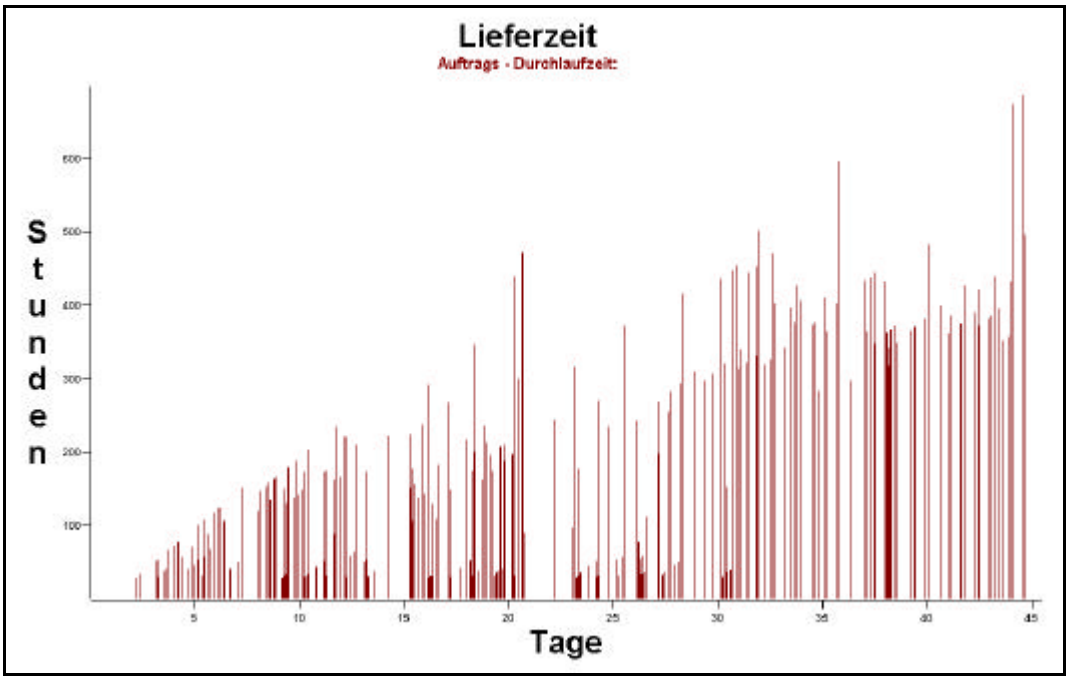


Abbildung 60: Lieferzeiten im Untersuchungsszenario 2

5.3 Untersuchungsszenario 3

5.3.1 Beschreibung des Untersuchungsszenarios 3

Im dritten Szenario wird als Speditionsstrategie die Strategie 3 benutzt. Beim Einsatz dieser Strategie wird durch die Regionallager einen LKW für den Transport

kommissionierter Sendungen angefordert, sobald eine Sendung in der Warenausgabe für den entsprechenden Kunden eine maximale Wartezeit überschritten ist.

Die Strategie gilt sowohl für die Distributions-, als auch für die Beschaffungsspedition.

5.3.2 Ziel des Untersuchungsszenarios 3

Ziel dieses Szenarios ist es, die Lieferzeiten zu optimieren und eine Abschätzung für die Entwicklung der Transportkosten zu geben.

5.3.3 Auswertung des Untersuchungsszenarios 3

Bei Verwendung der Speditionsstrategie 3 belaufen sich die Transportkosten für die Beschaffung der Regionallager auf 880.953 €. Die Transportkosten für die Distribution zum Kunden belaufen sich auf 618.533 €.

Die Lieferzeit liegt bei durchschnittlich 93 Stunden, jedoch ergibt sich ein hoher Maximalwert für die Lieferzeit von 508 Stunden.

Aufgrund der nicht vorhandenen Mindesttransportmengen sind keine Aufträge aufgrund von Transportproblemen nicht ausgeführt worden. Die Anzahl der nicht ausgeführten Aufträge ist geringer als ihre Anzahl bei Verwendung der Speditionsstrategie 1, was zum Anstieg der Lieferzeiten im Mittel führt. Es werden 8 Aufträge nicht ausgeführt.

Die nicht ausgeführten Aufträge lassen sich im „Array“ *Validierung Kundenbestellungen* erkennen, in welchem die Durchlaufzeit jedes einzelnen Auftrages gespeichert wird.

Die folgenden Abbildungen (**Abbildung 61** bis **Abbildung 64**) zeigen einige der Ergebnisdaten dieses Untersuchungsszenarios.

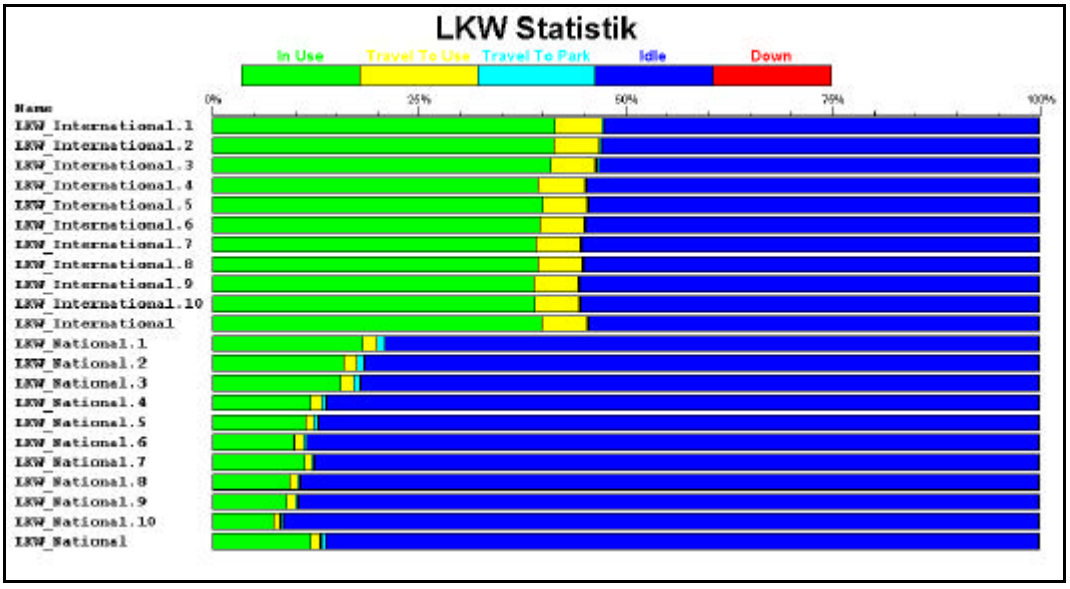


Abbildung 61: LKW Statistik im Untersuchungsszenario 3

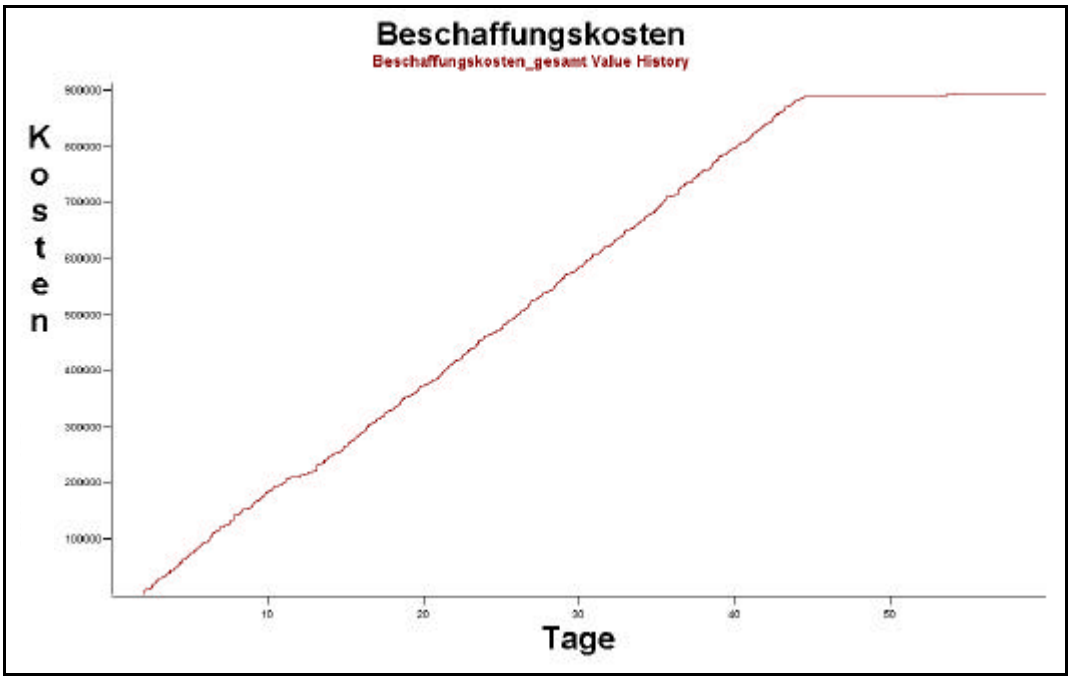


Abbildung 62: Beschaffungskosten im Untersuchungsszenario 3

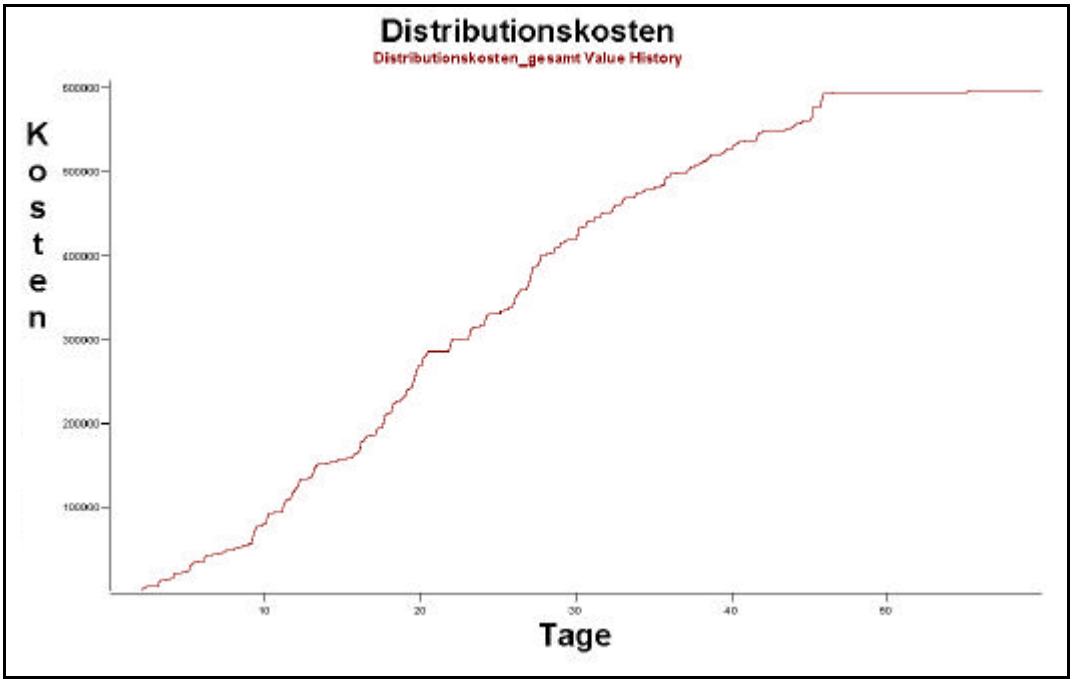


Abbildung 63: Distributionskosten im Untersuchungsszenario 3

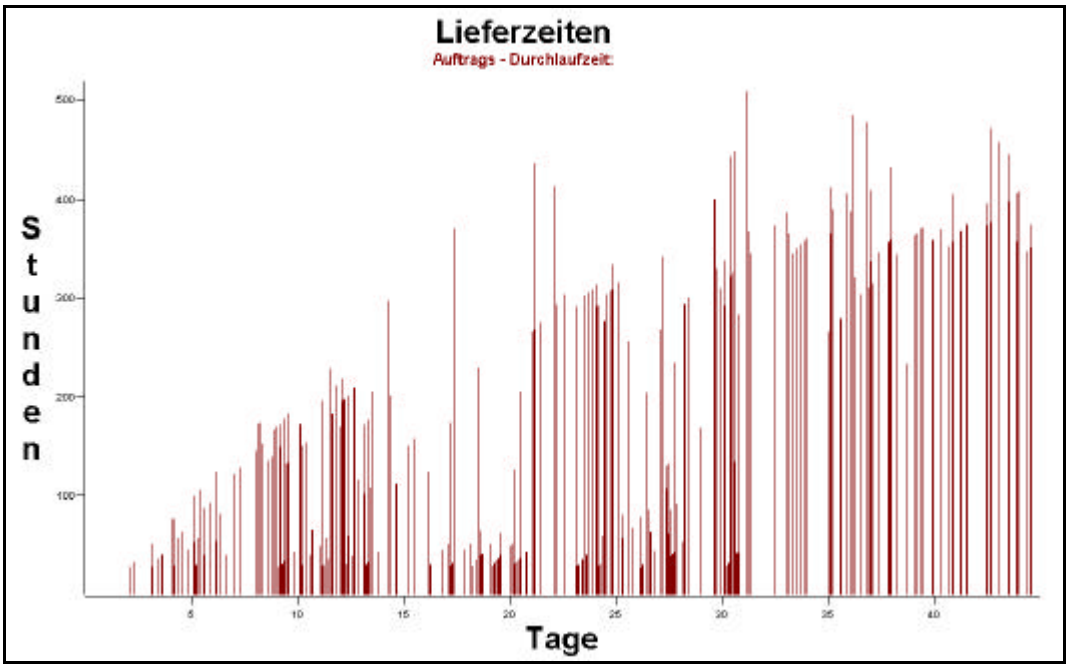


Abbildung 64: Lieferzeiten im Untersuchungsszenario 3

5.4 Beurteilung und Vergleich der Untersuchungsszenarien

Da bei der Simulation nur der kurze Zeitraum von einem Monat betrachtet wurde, ist keine abschließende Beurteilung der Szenarien möglich. Für die betrachtete Simulationszeit lässt sich feststellen, das die Verwendung der Speditionsstrategie 3 bei moderater Steigerung der Transportkosten zu einer deutlichen Glättung der Lieferzeiten führt. Zwar ist die durchschnittliche Lieferzeit kaum verändert, jedoch ist die Streuung der Lieferzeiten deutlich geringer, was sich in den Abbildungen der Histogramme der Lieferzeiten zeigt. Außerdem ist zu berücksichtigen, dass bei Strategie 3 zum Simulationsende deutlich mehr Aufträge ausgeführt werden als bei den anderen beiden Strategien, da keine Aufträge aufgrund des Nichtzustandekommens der Transportmenge oder der Mindestmenge nicht ausgeführt werden.

Bei einer Verlängerung der Simulationszeit würde es aber wahrscheinlich zu einer deutlichen Transportkostensteigerung kommen.

6 Beurteilung des Simulators

6.1 Programmumfang

Das *promodel*-Programmpaket beinhaltet nützliche Zusatzprogramme wie den *promodel*-SimRunner oder das Statistiktool *promodel*-Stat Fit. Der Funktionsumfang, der durch die einzelnen Programmkomponenten abgedeckt wird, ist nahezu vollständig. Wünschenswert wäre ein zusätzliches Programm zur 3D-Visualisierung der Modelle.

6.2 Benutzeroberfläche

Die Benutzeroberfläche entspricht weitgehend dem Windows-Standard. Wünschenswert wären anpassbare Icons für häufig verwendete Standardfunktionen.

Der *promodel*-Simulator ist gut in das Windows-Betriebssystem integriert. Zum Beispiel können „Copy & Paste"-Vorgänge programmübergreifend durchgeführt werden.

Zur Strukturierung der Einträge in den verschiedenen Funktionsfenstern gibt es eine „Move"-Funktion, welche umständlich zu bedienen und nicht in allen Fenstern verfügbar ist. Hier wäre ein Positionieren per „Drag & Drop" wünschenswert.

Die Anordnung der Fenster ist übersichtlich, da es zu keinen Überlappungen kommt. Wünschenswert wäre eine Navigationsleiste um schneller zwischen den verschiedenen Bestandteilen des Modells zu navigieren.

Eine Hierarchisierung des Modells ist nur durch das Einbinden von Teilmodellen möglich. Durch die reine Prozessorientierung von Promodel gibt es keine detaillierten und hierarchisierten Bausteine.

Die Menüführung, Hilfe und Dokumentation sind ausschließlich in der englischen Sprache verfügbar. Hier wäre eine deutsche Version wünschenswert.

Die Modellierung erfolgt nicht maßstabsgetreu.

6.3 Bausteine

Als Baustein mit vollständiger Funktionalität bietet der *promodel*-Simulator lediglich den Baustein „Conveyor", welcher sich sowohl als Förderband als auch als Staustrecke parametrieren lässt.

Die „Locations", welche sich als Bausteine auffassen lassen, verfügen lediglich über Eigenschaften wie Kapazität, Störungen und Schnittstellenregeln wie „Fifo" oder ähnliches. Hier wären weitere Standardbausteine wünschenswert.

6.4 Prozessdarstellung

Durch die Prozesse wird die Funktionalität der Bausteine programmiert. Die Darstellung der Prozesse ist übersichtlich. Durch die Unterteilung in „Operation-“ und „Move-“ Logik bieten sich flexible Gestaltungsmöglichkeiten.

Zur Steigerung der Übersichtlichkeit wäre eine Möglichkeit zur Gruppierung von Prozessen wünschenswert.

Die Erstellung von Prozessen kann sowohl grafisch als auch über einen Prozess-Editor geschehen. Zur Vereinfachung der Erstellung können auch ganze Prozesse kopiert werden. Wünschenswert wäre ein Verweis in den Prozessen, so dass Änderungen nur einmalig vorgenommen werden müssen.

6.5 Programmierung

Promodel verwendet eine Basic ähnliche Programmiersprache, die einige simulationsspezifische Funktionen enthält. Der Befehlsumfang ist überschaubar und ausreichend. Wünschenswert wäre lediglich ein „Case“-Befehl und eine Möglichkeit zum Zugriff auf Attribute von „Entities“ oder „Locations“ die nicht unmittelbar am jeweiligen Prozess beteiligt sind. Der Zugriff auf solche Attribute ist nur über den Umweg einer globalen Variable möglich.

Die Programmierung erfolgt in einem Editor, welcher die Möglichkeit zur Prüfung und Kompilierung bereitstellt. Der Editor verfügt weiterhin über die Möglichkeit, Passagen zu kopieren, auszuschneiden und einzufügen. Des weiteren ist eine Druckerschnittstelle integriert.

Während der Programmierung steht einem jederzeit ein Assistent, der „Logic Builder“ zur Verfügung. Dieser erstellt automatisch die Syntax der Befehle, was sehr hilfreich zum Erlernen der Programmiersprache ist. Der „Logic Builder“ erläutert kurz die Funktion der ausgewählten Befehle. Er ist hilfreich um sich eine Übersicht über die vorhandenen Funktionen zu verschaffen. Im „Logic Builder“ sind auch alle Elemente des Modells, wie zum Beispiel Variablen, Arrays, Locations usw. zugreifbar, so dass er bei der Programmierung auch zur Navigation oder zum „Nachschlagen“ von verwendeten Bezeichnungen genutzt werden kann.

Die von Promodel zur Verfügung gestellten Befehle waren für die Modellerstellung ausreichend. Besonders hilfreich war der Befehl LOAD IFF, welcher es ermöglicht Entities nach Kriterien wie Attributen zu laden. Dies erspart eine vorherige Sortierung.

Wünschenswert wäre die Erweiterung IFF auch bei Befehlen wie GROUP, COMBINE und beim UNLOAD Befehl. Des weiteren wäre ein Zugriff auf die Attribute von temporär gruppierten Entities wünschenswert.

6.6 Datenanbindung

Promodel unterstützt das Einlesen von Excel-Tabellen in Arrays. Das Einlesen kann aus beliebigen Arbeitsmappen erfolgen, da einzelne Arbeitsblätter und Zellenbereiche explizit ausgewählt werden können.

Durch die Verwendung von verschiedenen „Read"-Befehlen können auch Daten aus Textdateien eingelesen werden. Hierfür muss die Textdatei als „General Read File" definiert werden.

Als Hintergrundgrafiken können Bitmap, GIF, WMF und PCX Dateien importiert werden. Hier wäre eine Anbindung an das DXF Format wünschenswert. Dieselben Dateiformate können mit Hilfe des Programms *promodel*-Graphic-Editor auch für die Erstellung von „Libarys" importiert werden.

Bei Simulationsende können Arrays als Excel-Tabellen ausgegeben werden. Durch verschiedene „Write"-Befehle kann auch während der Simulation in Textdateien geschrieben werden. Die entsprechende Textdatei muss dafür als „General Write File" definiert werden.

Eine Datenbankanbindung wäre wünschenswert. Weiterhin fehlt eine Möglichkeit um online während der Simulation Daten in Excel-Tabellen zu speichern.

6.7 Datenauswertung

Die Datenauswertung erfolgt mit dem Tool *promodel*-Output Report. Nach der Simulation erhält man den General Report, welcher differenziert nach „Locations", „Entities", „Arrivals" usw. die während der Simulation gesammelten Daten enthält.

Diese Daten können nun durch verschiedene editierbare Diagramme ausgewertet werden. Leider fehlt eine Möglichkeit zum Export der Diagramme. Jedoch können die für das Diagramm selektierten Daten in eine Excel-Tabelle exportiert werden.

Wurden einmal Diagramme definiert, so können sie für verschiedene Szenarien des Modells automatisch erstellt werden.

6.8 Hilfe

Die Online Hilfe entspricht dem Windows-Standard. Sie ist umfangreich und verfügt zum Beispiel bei der Hilfe zu den Befehlen auch über Beispiele welche die Verwendung und die Syntax verdeutlichen. Es besteht die Möglichkeit nach verschiedenen Schlagworten zu suchen.

Die einzelnen Hilfetexte verfügen über Links zu den verwendeten Ausdrücken, was eine einfache Navigation in dem Hilfedokument ermöglicht.

6.9 Dokumentation

Die Dokumentation besteht aus einem User Guide, einem Reference Guide und einem Appendum zur Installation.

Der Userguide ist sehr detailliert und beschreibt ausführlich alle in *promodel* verwendeten Elemente.

Im Reference Guide ist eine alphabethische Beschreibung der Befehle der Programmiersprache enthalten.

Die Dokumentation ist gut und vollständig. Anhand der Dokumentation ist der Umgang mit dem Simulator erlernbar.

6.10 Support

Aufgrund der räumlichen Distanz kann ein Support für *promodel* nur durch E-Mail-Kommunikation erfolgen. Über die Reaktionszeiten auf eine E-Mail Anfrage liegen jedoch keine Daten vor, so dass eine Bewertung des Supports nicht möglich ist. Wünschenswert wäre jedoch eine Niederlassung in Deutschland als Ansprechpartner.

6.11 Gesamteindruck

Der Gesamteindruck des *promodel*-Programmpakets ist positiv. Das Produkt ist ausgereift, was sich durch eine hohe Stabilität zeigt.

Lediglich bei der Deaktivierung der Animation und gleichzeitigem Zoomen kam es zu Darstellungsfehlern.

Promodel ermöglicht durch die Prozessorientierung ein flexibles Modellieren. Der Simulator eignet sich besonders zur Abbildung von speziellen, nicht-Standard Modellen.

Bei der Modellierung von Standardabläufen ist der Arbeitsaufwand aufgrund der fehlenden Standardbausteine jedoch sehr hoch.

Als unübersichtlich gestaltet sich die Parametrierung der Prozesse, da diese nur über Macros erfolgen kann, welche sich nicht gruppieren lassen. Die Eingabemaske der Macros, das Run Time Interface ist für eine gut gestaltete Parametrierung zu beschränkt.

Ein wesentliches Manko für den Einsatz zur Modellierung von Beschaffungsketten ist die fehlende Hierarchisierbarkeit von Modellen. Es ist somit nicht möglich eine eigene „Bausteinbibliothek“ für verschiedene Beschaffungsketten aufzubauen. Modelle müssen daher immer wieder von Grund auf erstellt werden.

6.12 Vergleichende Bewertung des Simulators

Für die Modellierung von Beschaffungsketten ist die Erfüllung bestimmter Anforderungen an den eingesetzten Simulator besonders hervorzuheben. Insbesondere bei der experimentellen Modellierung von Beschaffungsketten ist es von besonderer Bedeutung, dass der Simulator eine Modellhierarchisierung derart unterstützt, dass Funktionseinheiten gebildet werden können und instanziiert in ein Gesamtmodell der Beschaffungskette eingesetzt werden können. Diese Anforderung ist von besonderer Bedeutung, da insbesondere bei den experimentellen Untersuchungen, wie sie im Teilprojekt A2 des Sonderforschungsbereichs 559 der Universität Dortmund durchgeführt werden, nicht nur eine spezielle Beschaffungskette untersucht wird. Der Schwerpunkt liegt vielmehr darin, dass durch die Variation von Strategien und physischen Funktionseinheiten allgemeingültige Aussagen über bestimmte Typen von Beschaffungsketten durch den Einsatz der Simulation ermittelt werden.

Neben dieser zentralen Anforderung an den Simulator ist auch die Unterstützung der Programmierung spezieller Strategien innerhalb einer Beschaffungskette eine wesentliche Anforderung, die an den eingesetzten Simulator gestellt werden muss.

Neben diesen beiden Kriterien können die eingesetzten Simulatoren unter anderem hinsichtlich des Umfanges an transport- und lagerorientierten Bausteinen, der Simulationsgeschwindigkeit, der Bedienbarkeit und der Güte der Support-Unterstützung bewertet werden.

Eine vergleichende Bewertung des untersuchten Simulationswerkzeugs *promodel* zu anderen Simulatoren ist in **Abbildung 65** dargestellt. Die zum Vergleich herangezogenen Simulatoren sind die drei Bausteinorientierten Simulatoren Dosimis-3, SimPro und eM-Plant. Für die Simulatoren Dosimis-3 und SimPro wurden im Vorfeld ähnliche Untersuchungen der Eignung wie für den Simulator *promodel* durchgeführt. Die Bewertung des Simulators eM-Plant basiert auf den Erfahrungen aus dem Einsatz dieses Simulators in zahlreichen Projekten, wobei hier sowohl Beschaffungsketten, als auch Systeme der innerbetrieblichen Logistik abgebildet wurden. Die Gewichtung der einzelnen Bewertungskriterien basiert auf den Erfahrungen während der Modellierung der abgebildeten Beschaffungskette.

Bewertung		Simulatoren							
		promodel		Dosimis-3		Simpro		eM-Plant	
Kriterium	Gewichtung	Punkte (0-10)	Wert	Punkte (0-10)	Wert	Punkte (0-10)	Wert	Punkte (0-10)	Wert
Modellierung	**50**		**25,6**		**9,8**		**28,6**		**42,8**
Hierarchisierbarkeit von Modellen	18	3	5,4	0	0	7	12,6	10	18
Möglichkeiten der Programmierung individueller Strategien	15	8	12	2	3	5	7,5	9	13,5
Möglichkeit der Abbildung individueller Parameter	5	8	4	4	2	6	3	9	4,5
Möglichkeit der Erstellung individueller Parametermasken	5	5	2,5	0	0	1	0,5	8	4
Umfang an transport- und lagerorientierten Bausteinen	4	2	0,8	6	2,4	8	3,2	4	1,6
Modellierungsaufwand	3	3	0,9	8	2,4	6	1,8	4	1,2
Simulation	**15**		**9,9**		**12,7**		**10,4**		**9,9**
Stabilität während der Simulation	8	8	6,4	8	6,4	6	4,8	8	6,4
Geschwindigkeit der Simulation	7	5	3,5	9	6,3	8	5,6	5	3,5
Auswertung	**17**		**12,6**		**7,6**		**9,6**		**12,2**
Umfang der statischen Ausgaben	7	8	5,6	8	5,6	8	5,6	6	4,2
Möglichkeiten der Programmierung individueller Statistikausgaben	10	7	7	2	2	4	4	8	8
Bedienbarkeit	**10**		**6,2**		**8,2**		**3,8**		**7**
Einbindung des Simulators in MS-Windows	4	8	3,2	7	2,8	5	2	7	2,8
Bedienbarkeit des Simulators	6	5	3	9	5,4	3	1,8	7	4,2
Support	**5**	**5**	**2,5**	**10**	**5**	**10**	**5**	**6**	**3**
Dokumentation	**6**	**6**	**3,6**	**5**	**3**	**4**	**2,4**	**8**	**4,8**
Zusätzliche Features	**3**	**8**	**2,4**	**3**	**0,9**	**3**	**0,9**	**3**	**0,9**
Summe	**100**		**59,2**		**44,2**		**58,3**		**75,8**

Abbildung 65: Vergleichende Bewertung des Simulators

Aufgrund der mangelnden Hierarchisierbarkeit von Modellen bei der Modellierung mit dem Simulator *promodel* ist dieser Simulator nicht für die experimentelle Modellierung von Beschaffungsketten geeignet. Für die Abbildung und Untersuchung einer bestimmten Beschaffungskette mit ausschließlicher Strategievariation kann der Simulator *promodel* eingesetzt werden, da in einem solchen Fall die Hierarchisierung und Instanziierung der Modelle nicht erforderlich ist. Für die experimentelle Untersuchung wie sie im Rahmen des Teilprojektes A2 des Sonderforschungsbereichs 559 der Universität Dortmund durchgeführt wird, ist der Einsatz des Simulators eM-Plant zu empfehlen, da dieser Simulator von den vier betrachteten Simulatoren den höchsten Erfüllungsgrad der zwei wesentlichen Punkte „Hierarchisierbarkeit“ und „Strategieprogrammierung“ aufweist.

7 Zusammenfassung und Ausblick

Die Basis der Systemmodellierung bildet eine real existierende zweistufige Beschaffungskette. Diese Kette wurde vor der Modellierung hinsichtlich der vorhandenen Materialflussstrategien erweitert. Hierdurch ergab sich für die Simulation die Möglichkeit, Experimente auf der Basis von Strategievariationen durchzuführen.

Die für die Systemabbildung erforderlichen Modellierungsarbeiten haben gezeigt, dass der Simulator *promodel* für die Abbildung von Beschaffungsketten geeignet ist. Diese Eignung liegt insbesondere darin begründet, dass der Simulator aufgrund der Prozessorientierung über eine eigene Programmierumgebung verfügt. Diese ermöglicht die Abbildung von Materialflussstrategien, da der Objektfluss während der Simulation durch die Programmierung direkt gesteuert werden kann.

Für den Einsatz der Untersuchungen des Teilprojektes A2 des Sonderforschungsbereichs 559 der Universität Dortmund kann der Simulator *promodel* jedoch nicht empfohlen werden. Ziel des Teilprojektes ist es, durch den Einsatz der Simulation allgemeingültige Aussagen über Beschaffungsketten zu evaluieren. Die hierfür durchzuführenden Experimente bedürfen nicht nur einer Variation der Materialflussstrategien, sondern auch einer Variation der Topologien von Beschaffungsketten. Diese Variationen stellen Änderungen des Modells hinsichtlich der modellierten Bausteine dar. Die Durchführung dieser Änderungen wird durch die Möglichkeit der Modellhierarchisierung deutlich erleichtert. Wird eine Hierarchisierung und Instanziierung der Modelle durch den eingesetzten Simulator unterstützt, so können in einem ersten Schritt komplexe Funktionseinheiten modelliert werden. Die Modellierung einer gesamten Beschaffungskette erfolgt in einem zweiten Schritt durch die Modellierung mit Instanzen der zuvor abgebildeten Funktionseinheiten. Diese Modellhierarchisierung und Instanziierung wird durch den Simulator *promodel* nicht unterstützt.

Die vergleichende Bewertung verschiedener Simulatoren hat gezeigt, dass der Simulator eM-Plant besonders geeignet für den Einsatz zur experimentellen Untersuchung von Beschaffungsketten ist. Dieser Simulator besitzt neben der Unterstützung der Modellhierarchisierung und der Möglichkeit der Programmierung von Materialflussstrategien weitere positiv zu bewertende Merkmale hinsichtlich der experimentellen Untersuchung von Beschaffungsketten. So ist es möglich, beliebige Parameter in ein Modell aufzunehmen, und für diese Parameter individuelle Parametermasken zu erstellen. Hierdurch können z.B. für die Bewertung relevante Kosten als Parameter in ein Modell integriert werden. Durch die Möglichkeit der Erstellung individueller Parametermasken können diese Parameter während der Experimentdurchführung komfortabel durch den Benutzer variiert werden. Neben dieser Variationsmöglichkeit

bietet der Simulator eM-Plant zusätzliche Schnittstellen zu MS-Excel oder zu Datenbanken auf der Basis von SQL-Abfragen.

Die im Rahmen dieser Studie durchgeführte Modellierung einer Beschaffungskette beinhaltete die Programmierung vielfältiger Materialflussstrategien. Die Umsetzung dieser Strategien in anderen Simulatoren, welche ebenfalls die Einbindung individueller Programmierungen unterstützen, wird durch die Erfahrungen aus der Modellierung mit dem Simulator *promodel* unterstützt. Eine direkte Portierung der Strategien in eine andere Simulatorumgebung ist aufgrund des individuellen Befehlssatzes des Simulators *promodel* nicht möglich. Die Unterstützung bei der Umsetzung der Materialflussstrategien in anderen Simulatoren ergibt sich jedoch aus der vorhandenen Ähnlichkeit der materialflussorientierten Programmiersprachen der einzelnen Simulatoren.

8 Anhang

8.1 Verwendete Macros

8.1.1 Strategiemacros

Die Macros

- Produktionsstrategie
- Lagerverwaltungsstrategie
- Speditionsstrategie Beschaffung
- Speditionsstrategie Distribution
- Auftragsdispositionsstrategie

dienen zur Auswahl der jeweils verwendeten Strategie. Der Standardwert für diese Macros ist 1.

8.1.2 Distributionsmacros

Die Macros

- Transportmenge Distribution (50)
- Mindestmenge Distribution (25)
- Maximale Wartezeit Distribution (1440)
- Ladedauer Distribution (N(30,5))
- Entladedauer Distribution (N(25,3))

werden bei der Ausführung der Speditionsstrategien für die Distribution verwendet. Die Standardwerte sind in den Klammern angegeben. Die Mengen sind in Stück Reifen angegeben. Die Zeiten sind in Minuten angegeben, wobei die Ladezeiten normalverteilt sind.

8.1.3 Beschaffungsmacros

Die Macros

- Transportmenge Beschaffung (100)
- Mindestmenge Beschaffung (50)
- Maximale Wartezeit Beschaffung (2880)
- Ladedauer Beschaffung (N(40,1))
- Entladedauer Beschaffung (N(35,2))

werden bei der Ausführung der Speditionsstrategien für die Beschaffung verwendet. Die Standardwerte sind in den Klammern angegeben. Die Mengen sind in Stück Reifen

angegeben. Die Zeiten sind in Minuten angegeben, wobei die Ladezeiten Normalverteilt sind.

8.1.4 Kostenmacros

Die Macros

- LKW National Anfahrpauschale (0)
- LKW National Wartepreis (8)
- LKW National Fahrtpreis (2)
- LKW International Anfahrpauschale (0)
- LKW International Wartepreis (8)
- LKW International Fahrtpreis (2)

werden zur Berechnung der Transportkosten verwendet. Die Preise LKW National beziehen sich auf die Distributionsspedition, die Preise LKW International auf die Beschaffungsspedition. Die Preise sind in € angegeben. Die Standardwerte stehen in den Klammern.

Die Anfahrpauschale wird fällig, sobald ein durch eine Funktionseinheit „Regionallager“ ein LKW angefordert wird.

Der Wartepreis wird berechnet, wenn der LKW am Lager oder beim Kunden wartet, z.B. während er be- und entladen wird. Der Preis wird in € / min angegeben.

Der Fahrtpreis wird in € / km angegeben.

8.1.5 Sonstige Macros

Das Macro Auftragsbearbeitungsdauer (N120,5) MIN wird von der Auftragsdisposition verwendet. Es dient zur Parametrierung der Zeit, die vergeht bis ein Auftrag von der Auftragsdisposition an eine Funktionseinheit „Regionallager“ weitergeleitet wird.

Die Macros Einlagerzeit (N(1,0.1)) MIN und Auslagerzeit (N(2,0.1)) MIN dienen zur Parametrierung der Funktionseinheiten „Regionallager“. Die Zeiten werden in Minuten angegeben, und beziehen sich auf den Ein- bzw. Auslagervorgang eines Reifens.

8.2 Verwendete Attribute

8.2.1 Attribut Auftrags ID

Das Attribut Auftrags ID bezieht sich auf die Objekte der Instanzen *Auftrag, Sendung* und *Artikel* des Modellierungselements „Entity“. Jedem *Auftrag* wird bei seiner Erzeugung eine Auftrags ID zugewiesen. Durch das Erzeugen einer *Sendung* in einem Regionallager erhält auch das erzeugte Objekt *Sendung* und später die mit diesem

Objekt gruppierten (geladenen) Objekte der Instanz *Artikel* die jeweilige Auftrags ID. Entscheidend ist die Auftrags ID in der Funktionseinheit „Kunde“, da anhand dieses Attributes entschieden wird, zu welchem Auftrag ein gelieferter Artikel gehört und die Ausführungszeit für den Auftrag ermittelt werden kann.

8.2.2 Attribut Artikelnummer

Das Attribut Artikelnummer bezieht sich auf die Objekte der Instanz *Artikel* des Modellierungselements „Entity“. Es wird jedem Objekt der Instanz *Artikel* bei der Initialisierung und der Erzeugung während der Produktion die jeweilige Artikelnummer zugewiesen. Das Attribut Artikelnummer dient zur Unterscheidung der Objekte der Instanz *Artikel*, da für alle Produkte nur eine Instanz *Artikel* modelliert wurde.

8.2.3 Attribute Menge , Kunde , Wunschlieferant

Diese Attribute beziehen sich ebenfalls auf die Objekte der Instanzen *Auftrag* und *Sendung* des Modellierungselements „Entity“. Bei der Auftragserzeugung werden in diesen Attributen der erzeugten Objekte die entsprechenden Auftragsdaten gespeichert. Die Attribute Kunde und Wunschlieferant werden bei der Auftragsdisposition verwendet. Das Attribut Menge steuert in der Funktionseinheit „Regionallager“ die Anzahl der auszulagernden Artikel. Das Attribut Kunde wird in der Kommissionierung zum Routen der ausgelagerten Objekte der Instanz *Artikel* zur entsprechenden Warenausgabestation verwendet.

8.2.4 Attribut Tag

Das Attribut Tag bezieht sich auf die Instanzen *Artikel* und *Auftrag* des Modellierungselements „Entity“. In diesem Attribut wird der Erzeugungstag eines Objektes der Instanzen *Artikel* und *Auftrag* gespeichert.

8.2.5 Attribut Auftragsnummer

Das Attribut Auftragsnummer bezieht sich auf das modellierte Objekt der Instanz *Auftragsdisposition* des Modellierungselements „Location“. Es wird in der Funktionseinheit „Auftragsdisposition“ verwendet, um eine fortlaufende Nummerierung der Aufträge zu ermöglichen. Die Auftragsnummer entspricht dem Wert des Attributs Auftrags ID des jeweils zuletzt erzeugten Objektes der Instanz *Auftrag*. Die Auftragsnummer dient als Pointer für den Zugriff auf das „Array“ *Kundenbestellungen*.

8.2.6 Attribut Sendungszähler

Das Attribut Sendungszähler bezieht sich auf die modellierten Objekte der Instanz *Warenausgabe* des Modellierungselements „Location“. Es wird in den Warenausgabe-

stationen verwendet um die Anzahl der zu sendenden Artikel zu speichern. Anhand des Sendungszählers entscheidet sich, wann ein LKW zum Transport der Artikel bestellt wird.

8.2.7 Attribut Produktionsnummer

Das Attribut Produktionsnummer bezieht sich auf die modellierten Objekte der Instanz *Werk* des Modellierungselements „Location“. Es wird in den Produktionsstandorten Paris und Heidelberg verwendet, um eine fortlaufende Nummerierung der produzierten Artikel zu realisieren. Die Produktionsnummer dient als Pointer für den Zugriff auf das Array *Produktion*.

8.3 Verwendete Variablen

8.3.1 Variable Simulationstag

Die Variable Simulationstag dient zur Speicherung des „gerundeten“ Simulationstages. Jeder neue Tag beginnt um 6:00 Uhr morgens. Die Variable dient als Vergleichskriterium beim Durchsuchen der Arrays *Produktion* und *Kundenbestellungen* um die am jeweiligen Tag zu erzeugenden Aufträge beziehungsweise Artikel zu bestimmen.

8.3.2 Variablen für die Erfassung der Distributionskosten

Die Variablen

- Anfahrkosten Distribution
- Wartekosten Distribution
- Fahrtkosten Distribution
- Distributionskosten gesamt

dienen zur Ermittlung der jeweiligen Distributionskosten. In den Variablen werden die einzelnen verursachten Kosten nach Typ getrennt aufaddiert. Die Variable Distributionskosten gesamt ist die Summe der einzelnen Kostentypen. Der Wert der Variablen entspricht den verursachten Kosten in €.

8.3.3 Variablen für die Erfassung der Beschaffungskosten

Die Variablen

- Anfahrkosten Beschaffung
- Wartekosten Beschaffung
- Fahrtkosten Beschaffung
- Beschaffungskosten gesamt

dienen zur Ermittlung der jeweiligen Beschaffungskosten. In den Variablen werden die einzelnen verursachten Kosten nach Typ getrennt aufaddiert. Die Variable Beschaffungskosten gesamt ist die Summe der einzelnen Kostentypen. Der Wert der Variablen entspricht den verursachten Kosten in €.

8.4 Verwendete Arrays

8.4.1 Arrays zur Verwaltung der Bestände

Die Arrays

- Bestand Bielefeld
- Bestand Koeln
- Bestand Magdeburg
- Bestand Heidelberg
- Bestand Paris

sind die Lagerbestandsarrays der jeweiligen Standorte. Die Arrays haben die Form *Bestand_xxx [Artikelnummer,Menge]*.

Die Anfangsbestände werden bei der Initialisierung aus der Excel-Arbeitsmappe *BeDis\Importdaten\Anfangsbestände.xls,* vom entsprechenden Blatt aus den Zellen B4:B49 eingelesen.

In den Bestandsarrays werden die entsprechenden Zu- und Abgänge an Artikeln gespeichert.

Zum Simulationsende werden die Lagerbestände in den Excel-Arbeitsmappen *BeDis\Exportdaten\Endlagerbestand xxx.xls* gespeichert. Die Zahlen in Spalte A entsprechen der im Lager vorhandenen Menge des jeweiligen Artikels in aufsteigender Artikelnummerreihenfolge. Die Zeilennummer minus eins entspricht der Artikelnummer.

8.4.2 Arrays zur Verwaltung der Artikelstammdaten

Das Array Artikelstammdaten enthält die Stammdaten der 46 betrachteten Artikel. Den Artikeln werden Wert und Gewicht zugeordnet. Das Array hat die Form *Artikelstammdaten [Artikelnummer,(Wert,Gewicht)]*.

Die Artikelstammdaten werden bei der Initialisierung aus der Excel-Arbeitsmappe *BeDis\Importdaten\Artikelstammdaten.xls,* Blatt Artikelstammdaten aus den Zellen C2:D47 importiert.

Zu Simulationsende wird der Inhalt des Arrays zur Validierung in der Excel-Arbeitsmappe *BeDis\Exportdaten\Validierung* Artikelstammdaten.xls gespeichert. Die

Zeilennummer minus eins entspricht der Artikelnummer, der Wert in Spalte A dem jeweiligen Wert des Artikels und der Wert in Spalte B dem Gewicht des jeweiligen Artikels.

8.4.3 Arrays zur Verwaltung der Dispositionsdaten

Die Arrays

- Disposition Bielefeld
- Disposition Koeln
- Disposition Magdeburg

enthalten die Dispositionsdaten der drei Regionallager. In den Arrays werden für das jeweilige Lager die Meldebestände, die Bestellgrenzen, die Nachbestellmengen, die Mengen pro Transporteinheit, die ersten Bestelltage und die Bestellperioden für die 46 betrachteten Artikel gespeichert.

Die Arrays haben die Form *Disposition_xxx [Artikelnummer, (Meldebestand, Bestellgrenze, Nachbestellmenge, Menge pro Transporteinheit, erster Bestelltag, Bestellperiode)]*.

Die Dispositionsdaten werden bei der Initialisierung aus der Excel-Arbeitsmappe *BeDis\Importdaten\Dispositionsdaten.xls* vom jeweiligen Blatt aus den Zellen D2:I47 eingelesen.

Zu Simulationsende werden die Dispositionsdaten zur Validierung in der Excel-Arbeitmappe *BeDis\Exportdaten\Dispositionsvalidierung xxx.xls* gespeichert. Die Zeilennummer minus eins entspricht der Artikelnummer, Die Spalte A dem Meldebestand, die Spalte B der Bestellgrenze, die Spalte C der Nachbestellmenge, die Spalte D der Menge pro Transporteinheit, die Spalte E dem ersten Bestelltag und die Spalte F der Bestellperiode.

8.4.4 Arrays zur Verwaltung der Kundenbestellungen

Das Array Kundenbestellungen enthält die Daten der Kundenbestellungen. Es gehen an 31 Tagen Bestellungen ein, die den Bestelltag, die Wunschlieferantennummer (1: Bielefeld, 2: Koeln, 3: Magdeburg, 4:Heidelberg), die Kundennummer (1: Kunde Bielefeld, 2: Kunde Koeln, 3: Kunde Magdeburg, 4: Kunde Heidelberg), die Artikelnummer und die Bestellmenge enthalten.

Das Array hat die Form *Kundenbestellungen [Auftragsnummer, (Bestelltag, Wunschlieferantennummer, Kundennummer, Artikelnummer, Bestellmenge)]*.

Die Daten werden bei der Initialisierung aus der Excel-Arbeitsmappe *BeDis\Importdaten\Kundenbestellungen.xls* vom Blatt *Summe_Kundenbest* aus den Zellen G2:K1204 importiert.

Zur Validierung wird der Inhalt des Arrays zum Simulationsende in der Excel-Arbeitsmappe *BeDis\Exportdaten\Validierung Kundenbestellungen.xls* gespeichert. Die Zeile minus eins entspricht der Auftragsnummer, Spalte A dem Bestelltag, Spalte B der Wunschlieferantennummer, Spalte C der Kundennummer, Spalte D der Artikelnummer und Spalte E der Bestellmenge.

8.4.5 Arrays zur Verwaltung der Produktion

Die Arrays

- Produktion Heidelberg
- Produktion Paris

enthalten die Produktionsdaten der beiden Produktionsstandorte Paris und Heidelberg. Die Produktion richtet sich nach dem Produktionstag, der Artikelnummer und der zu produzierenden Menge. Die Arrays haben die Form *Produktion_xxx [Produktionsnummer, (Tag, Artikelnummer, Menge)]*.

Die Daten werden bei der Initialisierung aus der Excel-Arbeitsmappe *BeDis\Importdaten\Produktionsdaten.xls* vom jeweiligen Blatt aus den Zellen B2:D182 für den Standort Heidelberg und aus den Zellen B2:D132 für den Standort Paris eingelesen.

Bei Simulationsende werden die Inhalte der Arrays zur Validierung in den Excel-Arbeitsmappen *BeDis\Exportdaten\Produktion xxx.xls* gespeichert. Die Zeilennummer minus eins entspricht der Produktionsnummer, Spalte A dem Tag, Spalte B der Artikelnummer und Spalte C der Menge.

8.4.6 Arrays zur Verwaltung des Artikeldurchsatzes

Das Array Artikeldurchsatz dient zur Speicherung aller zu den Kunden ausgelieferten Artikel. Das Array hat die Form *Artikeldurchsatz [Artikelnummer, Menge]*.

Ist ein Artikel beim Kunden angekommen, so wird die Menge des jeweiligen Artikels im Array Artikeldurchsatz um eins erhöht.

Bei Simulationsende wird der Inhalt des Arrays in der Excel-Arbeitsmappe *BeDis\Exportdaten\Artikeldurchsatz.xls* gespeichert. Die Zeilennummer minus eins entspricht der Artikelnummer und die Werte in Spalte A der jeweils durchgesetzten Menge.

8.4.7 Arrays zur Verwaltung der Auslieferungsdaten

Die Arrays

- Auslieferung Bielefeld
- Auslieferung Koeln
- Auslieferung Magdeburg
- Auslieferung Heidelberg
- Auslieferung Paris

dienen zur Speicherung der Anzahl der Ausgelieferten Artikel. Die Arrays haben die Form *Auslieferung_xxx [Artikelnummer, (Auslieferung seit letzter Bestellung, Auslieferung gesamt)]*.

Zum Simulationsende werden die Inhalte der Arrays in den Excel-Arbeitsmappen *BeDis\Exprortdaten\Auslieferung xxx.xls* gespeichert. Die Zeilennummer minus eins entspricht der Artikelnummer, die Werte in Spalte A der Auslieferungsmenge seit der letzten Bestellung und die Werte in Spalte B der Auslieferungsmenge insgesamt.

8.4.8 Arrays zur Verwaltung der Bestellungen

Die Arrays

- Bestellung Bielefeld
- Bestellung Koeln
- Bestellung Magdeburg

dienen zur Speicherung der von den jeweiligen Regionallager nachbestellten Artikelmengen. Die Arrays haben die Form *Bestellung_xxx [Artikelnummer, nachbestellte Menge]*

Zum Simulationsende werden die Inhalte der Arrays in den Excel-Arbeitsmappen *BeDis\Exportaten\ Bestellung xxx.xls* gespeichert. Die Zeilennummer minus eins entspricht der Artikelnummer, der Wert in Spalte A der noch offenen, nachbestellten Artikelanzahl.

Die Arrays

- Bestellung Heidelberg
- Bestellung Paris

dienen zur Speicherung der von den Regionallagern in den Produktionsstandorten nachbestellten Artikelmengen. Die Arrays haben die Form

Bestellung_xxx [Artikelnummer, (Aktuell nachbestellte Menge, Insgesamt nachbestellte Menge, nachbestellte Menge vom Vortag)].

Bei Simulationsende wird der Inhalt der Array in den Excel-Arbeitsmappen *Bestellung xxx.xls* gespeichert. Die Zeilennummer minus eins entspricht der Artikelnummer, die Werte in Spalte A den aktuell nachbestellten Mengen, die Werte in Spalte B den insgesamt nachbestellten Mengen und die Werte in Spalte C den am Vortag nachbestellten Mengen.

8.5 Verwendete Subroutinen

8.5.1 Subroutine Auftragserzeugung

```
#******************
#*Auftragserzeugung*
#******************

#*****************************
#* Berechnung des aktuellen Tages *
#*****************************

Simulationstag = TRUNC(Clock(Day) + 1.25)

#***********************************************
#* Erzeugung der am aktuellen Tag eingehenden Aufträge *
#***********************************************
WHILE Simulationstag = Kundenbestellungen[Auftragsnummer,1]
        DO
BEGIN
        AuftragsID = Auftragsnummer
        Tag = Kundenbestellungen[Auftragsnummer,1]
        Wunschlieferant = Kundenbestellun-
        gen[Auftragsnummer,2]
        Kunde = Kundenbestellungen[Auftragsnummer,3]
        Artikelnummer = Kundenbestellun-
        gen[Auftragsnummer,4]
        Menge = Kundenbestellungen[Auftragsnummer,5]
        Auftragsnummer = Auftragsnummer + 1
        ORDER 1 Auftrag TO Auftragsdisposition
```

```
END
```

8.5.2 Subroutine Lagerinitialisierung

```
#************************
#* Lagerinitialisierung *
#************************

INT i

#*********************************
#* Initialisierung Lager Bielefeld *
#*********************************
IF Lagernummer = 1 THEN
BEGIN
          i = 0
          WHILE i < 46 DO
          BEGIN
               INC i ,1
               Artikelnummer = i
               IF Bestand_Bielefeld[i,1] <> 0 THEN
               BEGIN
                    Order Bestand_Bielefeld[i,1] Artikel TO
          Lager_Bielefeld
               END
               #*********************************************
               #* Nachbestellung bei Lagerverwaltungsstrategie 3 oder 4
          *
               #*********************************************
IF Bestand_Bielefeld [Artikelnummer,1] < Disposition_Bielefeld
          [Artikelnummer,1] THEN
               {
                    IF Lagerverwaltungsstrategie = 3 THEN Lagerverwal-
          tung_3 (1)
                    IF Lagerverwaltungsstrategie = 4 THEN Lagerverwal-
          tung_4 (1)
               }
          END
END

#*****************************
```

```
#* Initialisierung Lager Koeln *
#*****************************
IF Lagernummer = 2 THEN
BEGIN
         i = 0
         WHILE i < 46 DO
         BEGIN
              INC i ,1
              Artikelnummer = i
              IF Bestand_Koeln[i,1] <> 0 THEN
              BEGIN
                   Order Bestand_Koeln[i,1] Artikel TO Lager_Koeln
              END
              #*********************************************
              #* Nachbestellung bei Lagerverwaltungsstrategie 3 oder 4
         *
              #*********************************************
              IF Bestand_Koeln [Artikelnummer,1] < Disposition_Koeln
         [Artikelnummer,1] THEN
              {
                   IF Lagerverwaltungsstrategie = 3 THEN Lagerverwal-
         tung_3 (2)
                   IF Lagerverwaltungsstrategie = 4 THEN Lagerverwal-
         tung_4 (2)
              }
         END
END
#*********************************
#* Initialisierung Lager Magdeburg *
#*********************************
IF Lagernummer = 3 THEN
BEGIN
         i = 0
         WHILE i < 46 DO
         BEGIN
              INC i ,1
              Artikelnummer = i
              IF Bestand_Magdeburg[i,1] <> 0 THEN
              BEGIN
                   Order Bestand_Magdeburg[i,1] Artikel TO
         Lager_Magdeburg
              END
```

```
          #*********************************************
          #* Nachbestellung bei Lagerverwaltungsstrategie 3 oder 4
     *
          #*********************************************
          IF Bestand_Magdeburg [Artikelnummer,1] < Dispositi-
     on_Magdeburg [Artikelnummer,1]
                    THEN
          {
               IF Lagerverwaltungsstrategie = 3 THEN Lagerverwal-
     tung_3 (3)
               IF Lagerverwaltungsstrategie = 4 THEN Lagerverwal-
     tung_4 (3)
          }
     END
END

#***********************************
#* Initialisierung Lager Heidelberg *
#***********************************
IF Lagernummer = 4 THEN
BEGIN
     i = 0
     WHILE i < 46 DO
     BEGIN
          INC i ,1
          Artikelnummer = i
          IF Bestand_Heidelberg[i,1] <> 0 THEN
          BEGIN
               Order Bestand_Heidelberg[i,1] Artikel TO
     Lager_Heidelberg
          END
     END
END
#*******************************
#* Initialisierung Lager Paris *
#*******************************
IF Lagernummer = 5 THEN
BEGIN
     i = 0
     WHILE i < 46 DO
     BEGIN
          INC i ,1
```

```
              Artikelnummer = i
              IF Bestand_Paris[i,1] <> 0 THEN
              BEGIN
                   Order Bestand_Paris[i,1] Artikel TO Lager_Paris
              END
         END
END
```

8.5.3 Subroutine Kommissionieren

```
#*********************
#* Kommissionierung *
#*********************
INT tmp = Artikelnummer
Auftragsnummer = AuftragsID
LOAD Menge IFF Artikelnummer = tmp
```

8.5.4 Subroutine Auftragsdisposition 1

```
#*********************************
#* Auftragsdispositionsstrategie 1 *
#*********************************

IF Wunschlieferant = 1 THEN ORDER 1 Sendung TO Kommissionie-
         rung_Bielefeld
IF Wunschlieferant = 2 THEN ORDER 1 Sendung TO Kommissionierung_Koeln
IF Wunschlieferant = 3 THEN ORDER 1 Sendung TO Kommissionie-
         rung_Magdeburg
IF Wunschlieferant = 4 THEN ORDER 1 Sendung TO Kommissionie-
         rung_Heidelberg
```

8.5.5 Subroutine Auftragsdisposition 2

```
#*********************************
#* Auftragsdispositionsstrategie 2 *
#*********************************

#************************************************
#* Ermittlung des Lagers mit dem höchsten Bestand *
#************************************************
```

```
IF Bestand_Bielefeld [Artikelnummer,1] > Bestand_Koeln [Artikelnum-
         mer,1] AND
         Bestand_Bielefeld [Artikelnummer,1] > Bestand_Magdeburg
         [Artikelnummer,1] THEN
         {
         ORDER 1 Sendung TO Kommissionierung_Bielefeld
         GOTO L1
         }
IF Bestand_Koeln [Artikelnummer,1] > Bestand_Bielefeld [Artikelnum-
         mer,1] AND
         Bestand_Koeln [Artikelnummer,1] > Bestand_Magdeburg
         [Artikelnummer,1] THEN
         {
         ORDER 1 Sendung TO Kommissionierung_Koeln
         GOTO L1
         }
IF Bestand_Magdeburg [Artikelnummer,1] > Bestand_Bielefeld
         [Artikelnummer,1] AND
         Bestand_Magdeburg [Artikelnummer,1] > Bestand_Koeln
         [Artikelnummer,1] THEN
         {
         ORDER 1 Sendung TO Kommissionierung_Magdeburg
         GOTO L1
         }

#*****************************************************************
#* Ermittlung des geographisch näherliegenden Lagers bei identischen
         Beständen *
#*****************************************************************

IF Bestand_Bielefeld [Artikelnummer,1] = Bestand_Koeln [Artikelnum-
         mer,1] AND
         Bestand_Bielefeld [Artikelnummer,1] > Bestand_Magdeburg
         [Artikelnummer,1] THEN
         {
         IF Wunschlieferant = 1 THEN ORDER 1 Sendung TO Kommissionie-
         rung_Bielefeld
         IF Wunschlieferant = 2 THEN ORDER 1 Sendung TO Kommissionie-
         rung_Koeln
         IF Wunschlieferant = 3 THEN ORDER 1 Sendung TO Kommissionie-
         rung_Bielefeld
```

```
          IF Wunschlieferant = 4 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Koeln
          GOTO L1
          }
IF Bestand_Bielefeld [Artikelnummer,1] = Bestand_Magdeburg
          [Artikelnummer,1] AND
          Bestand_Bielefeld [Artikelnummer,1] > Bestand_Koeln
          [Artikelnummer,1] THEN
          {
          IF Wunschlieferant = 1 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Bielefeld
          IF Wunschlieferant = 2 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Bielefeld
          IF Wunschlieferant = 3 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Magdeburg
          IF Wunschlieferant = 4 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Bielefeld
          GOTO L1
          }
IF Bestand_Koeln [Artikelnummer,1] = Bestand_Magdeburg [Artikelnum-
          mer,1] AND
          Bestand_Koeln [Artikelnummer,1] > Bestand_Bielefeld
          [Artikelnummer,1] THEN
          {
          IF Wunschlieferant = 1 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Koeln
          IF Wunschlieferant = 2 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Koeln
          IF Wunschlieferant = 3 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Magdeburg
          IF Wunschlieferant = 4 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Koeln
          GOTO L1
          }
IF Bestand_Koeln [Artikelnummer,1] = Bestand_Magdeburg [Artikelnum-
          mer,1] AND
          Bestand_Koeln [Artikelnummer,1] = Bestand_Bielefeld
          [Artikelnummer,1] THEN
          {
          Auftragsdisposition_1
          GOTO L1
          }
```

```
L1:
```

8.5.6 Subroutine Auftragsdisposition 3

```
#*********************************
#* Auftragsdispositionsstrategie 3 *
#*********************************

#*********************************************************************
          ***
#* Wenn Bestand > Bestellmenge Wunschlieferant, sonst Auftragsdisposi-
          tionsstrategie 2 *
#*********************************************************************
          ***
IF Wunschlieferant = 1 THEN
          {
          IF Bestand_Bielefeld [Artikelnummer,1] > Menge THEN
          ORDER 1 Sendung TO Kommissionierung_Bielefeld
          ELSE Auftragsdisposition_2
          }
IF Wunschlieferant = 2 THEN
          {
          IF Bestand_Koeln [Artikelnummer,1] > Menge THEN
          ORDER 1 Sendung TO Kommissionierung_Koeln
          ELSE Auftragsdisposition_2
          }
IF Wunschlieferant = 3 THEN
          {
          IF Bestand_Magdeburg [Artikelnummer,1] > Menge THEN
          ORDER 1 Sendung TO Kommissionierung_Magdeburg
          ELSE Auftragsdisposition_2
          }
IF Wunschlieferant = 4 THEN
          {
          IF Bestand_Heidelberg [Artikelnummer,1] > Menge THEN
          ORDER 1 Sendung TO Kommissionierung_Heidelberg
          ELSE Auftragsdisposition_2
          }
```

8.5.7 Subroutine Distribution 1

```
#************************************
#* Speditionsstrategie Distribution 1 *
#************************************

Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
#*********************************
#* Warten bis Transportmenge erreicht *
#*********************************
WAIT UNTIL Sendungszähler >= Transportmenge_Distribution
Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
GROUP Transportmenge_Distribution AS Ladung
```

8.5.8 Subroutine Distribution 2

```
#************************************
#* Speditionsstrategie Distribution 2 *
#************************************

INT tmp
Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
#*****************************************
#* Warten bis maximale Wartezeit abgelaufen *
#*****************************************
WAIT maximale_Wartezeit_Distribution MIN
Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
#*********************************************
#* Warten bis Mindesttransportmenge erreicht ist *
#*********************************************+*****
WAIT UNTIL Sendungszähler >= Mindestmenge_Distribution
Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
IF CONTENTS (LOC(LOCATION()),Artikel)  <= Transportmenge_Distribution
         THEN
{
         tmp = CONTENTS (LOC(LOCATION()),Artikel)
         GROUP tmp AS Ladung
}


IF CONTENTS (LOC(LOCATION()),Artikel)  > Transportmenge_Distribution
         THEN
```

```
{
          GROUP Transportmenge_Distribution AS Ladung
}
```

8.5.9 Subroutine Distribution 3

```
#************************************
#* Speditionsstrategie Distribution 3 *
#************************+*************

INT tmp
#*********************************************
#* Warten bis maximale Wartezeit abgelaufen ist *
#*********************************************
WAIT maximale_Wartezeit_Distribution MIN
Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
IF Sendungszähler <= Transportmenge_Distribution THEN
{
          tmp = Sendungszähler
          GROUP tmp AS Ladung
}
IF Sendungszähler > Transportmenge_Distribution THEN
{
          GROUP Transportmenge_Distribution AS Ladung
}
```

8.5.10 Subroutine Lagerverwaltung 1

```
#*****************************
#* Lagerverwaltungsstrategie 1 *
#*****************************

INT i = 0
#*********************************
#* Nachbestellung Lager Bielefeld *
#*********************************
IF Lager = 1 THEN
{
          #*******************************************************
          #* Auslesen des Arrays Disposition Bielefeld und
          Sendungserzeugung *
```

```
          #*********************************************************
          WHILE i <  22 DO
          BEGIN
               INC i , 1
               Artikelnummer = i
               Menge = Disposition_Bielefeld[i,3]
               Kunde = 1
               Auslieferung_Bielefeld [i,1] = 0
               IF Menge > 0 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Heidelberg
          END
}
#*****************************
#* Nachbestellung Lager Koeln *
#*****************************
IF Lager = 2 THEN
{
          #*********************************************************
          #* Auslesen des Arrays Disposition Koeln und Sendungserzeu-
          gung *
          #*********************************************************
          WHILE i <  22 DO
          BEGIN
               INC i , 1
               Artikelnummer = i
               Menge = Disposition_Koeln[i,3]
               Kunde = 1
               Auslieferung_Koeln [i,1] = 0
               IF Menge > 0 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Heidelberg
          END
}

#*********************************
#* Nachbestellung Lager Magdeburg *
#*********************************
IF Lager = 3 THEN
{
          #*********************************************************
          #* Auslesen des Arrays Disposition Magdeburg und
          Sendungserzeugung *
          #*********************************************************
```

```
          WHILE i <  22 DO
          BEGIN
               INC i , 1
               Artikelnummer = i
               Menge = Disposition_Magdeburg[i,3]
               Kunde = 3
               Auslieferung_Magdeburg [i,1] = 0
               IF Menge > 0 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Heidelberg
          END
}
```

8.5.11 Subroutine Lagerverwaltung 2

```
#*******************************
#* Lagerverwaltungsstrategie 2 *
#*******************************

INT i = 0

#*********************************
#* Nachbestellung Lager Bielefeld *
#*********************************
IF Lager = 1 THEN
{
          #*******************************************
          #* Nachbestellung der ausgelieferten Artikel *
          #*****************************+***************
          WHILE i <  22 DO
          BEGIN
               INC i , 1
               Artikelnummer = i
               Menge = Auslieferung_Bielefeld [Artikelnummer,1]
               Kunde = 1
               Auslieferung_Bielefeld [Artikelnummer,1] = 0
               IF Menge > 0 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Heidelberg
          END
}

#*****************************
```

```
#* Nachbestellung Lager Koeln *
#*****************************
IF Lager = 2 THEN
{
          #*****************************************
          #* Nachbestellung der ausgelieferten Artikel *
          #*****************************************
          WHILE i <  22 DO
          BEGIN
               INC i , 1
               Artikelnummer = i
               Menge = Auslieferung_Koeln [Artikelnummer,1]
               Kunde = 2
               Auslieferung_Koeln [Artikelnummer,1] = 0
               IF Menge > 0 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Heidelberg
          END
}

#*********************************
#* Nachbestellung Lager Magdeburg *
#*********************************
IF Lager = 3 THEN
{
          #*****************************************
          #* Nachbestellung der ausgelieferten Artikel *
          #*****************************************
          WHILE i <  22 DO
          BEGIN
               INC i , 1
               Artikelnummer = i
               Menge = Auslieferung_Magdeburg [Artikelnummer,1]
               Kunde = 3
               Auslieferung_Magdeburg [Artikelnummer,1] = 0
               IF Menge > 0 THEN ORDER 1 Sendung TO Kommissionie-
          rung_Heidelberg
          END
}
```

8.5.12 Subroutine Lagerverwaltung 3

```
#******************************
#* Lagerverwaltungsstrategie 3 *
#******************************


#***************************
#* Nachbestellung Lager Bielefeld *
#***************************
IF Lager = 1 THEN
{
          #*****************************************************
          #* Nachbestellung der Nachbestellmenge des auslösenden
          Artikels *
          #*****************************************************
          IF Artikelnummer <= 22 THEN
          {
               Menge = Disposition_Bielefeld [Artikelnummer,3]
               Kunde = 1
               Auslieferung_Bielefeld [Artikelnummer,1] = 0
               Bestellung_Bielefeld [Artikelnummer,1] = Menge
               ORDER 1 Sendung TO Kommissionierung_Heidelberg
          }
}

#****************************
#* Nachbestellung Lager Koeln *
#****************************
IF Lager = 2 THEN
{
          #*****************************************************
          #* Nachbestellung der Nachbestellmenge des auslösenden
          Artikels *
          #*****************************************************
          IF Artikelnummer <= 22 THEN
          {
               Menge = Disposition_Koeln [Artikelnummer,3]
               Kunde = 2
               Auslieferung_Koeln [Artikelnummer,1] = 0
               Bestellung_Koeln [Artikelnummer,1] = Menge
               ORDER 1 Sendung TO Kommissionierung_Heidelberg
          }
}
```

```
#*********************************
#* Nachbestellung Lager Magdeburg *
#*********************************
IF Lager = 3 THEN
{
          #*****************************************************
          #* Nachbestellung der Nachbestellmenge des auslösenden
          Artikels *
          #*****************************************************
          IF Artikelnummer <= 22 THEN
          {
               Menge = Disposition_Magdeburg [Artikelnummer,3]
               Kunde = 3
               Auslieferung_Magdeburg [Artikelnummer,1] = 0
               Bestellung_Magdeburg [Artikelnummer,1] = Menge
               ORDER 1 Sendung TO Kommissionierung_Heidelberg
          }
}
```

8.5.13 Subroutine Lagerverwaltung 4

```
#*******************************
#* Lagerverwaltungsstrategie 4 *
#*******************************

#*********************************
#* Nachbestellung Lager Bielefeld *
#*********************************
IF Lager = 1 THEN
{
          #*************************************************************
          ****************
          #* Nachbestellen der Differenz zwischen Istbestand und
          Bestellgrenze des auslösenden Artikels *
          #*************************************************************
          ****************
          IF Artikelnummer <= 22 THEN
          {
               Menge = (Disposition_Bielefeld [Artikelnummer,2] -
          Bestand_Bielefeld [Artikelnummer,1])
```

```
            Kunde = 1
            Auslieferung_Bielefeld [Artikelnummer,1] = 0
            Bestellung_Bielefeld [Artikelnummer,1] = Menge
            ORDER 1 Sendung TO Kommissionierung_Heidelberg
        }
}
#*****************************
#* Nachbestellung Lager Koeln *
#*****************************
IF Lager = 2 THEN
{
        #*************************************************************
        *****************
        #* Nachbestellen der Differenz zwischen Istbestand und
        Bestellgrenze des auslösenden Artikels *
        #*************************************************************
        *****************
        IF Artikelnummer <= 22 THEN
        {
            Menge = (Disposition_Koeln [Artikelnummer,2] -
        Bestand_Koeln [Artikelnummer,1])
            Kunde = 2
            Auslieferung_Koeln [Artikelnummer,1] = 0
            Bestellung_Koeln [Artikelnummer,1] = Menge
            ORDER 1 Sendung TO Kommissionierung_Heidelberg
        }
}
#*********************************
#* Nachbestellung Lager Magdeburg *
#*********************************
IF Lager = 3 THEN
{
        #*************************************************************
        ******************
        #* Nachbestellen der Differenz zwischen Istbestand und
        Bestellgrenze des auslösenden Artikels *
        #*************************************************************
        ******************
        IF Artikelnummer <= 22 THEN
        {
                Menge = (Disposition_Magdeburg[Artikelnummer,2] -
        Bestand_Magdeburg[Artikelnummer,1])
```

```
            Kunde = 3
            Auslieferung_Magdeburg [Artikelnummer,1] = 0
            Bestellung_Magdeburg [Artikelnummer,1] = Menge
            ORDER 1 Sendung TO Kommissionierung_Heidelberg
        }
}
```

8.5.14 Subroutine Auslagerung

```
#*************
#* Auslagern *
#*************

#***********************************
#* Auslagern aus dem Lager Bielefeld *
#***********************************
IF Lager = 1 THEN
{
        Bestand_Bielefeld [Artikelnummer,1] = (Bestand_Bielefeld
        [Artikelnummer,1] - 1)
        Auslieferung_Bielefeld [Artikelnummer,1] = (Ausliefe-
        rung_Bielefeld[Artikelnummer,1] +1)
        Auslieferung_Bielefeld [Artikelnummer,2] = (Ausliefe-
        rung_Bielefeld[Artikelnummer,2] +1)
}

#*******************************
#* Auslagern aus dem Lager Koeln *
#*******************************
IF Lager = 2 THEN
{
        Bestand_Koeln [Artikelnummer,1] = (Bestand_Koeln
        [Artikelnummer,1] - 1)
        Auslieferung_Koeln [Artikelnummer,1] = (Ausliefe-
        rung_Koeln[Artikelnummer,1] +1)
        Auslieferung_Koeln [Artikelnummer,2] = (Ausliefe-
        rung_Koeln[Artikelnummer,2] +1)
}

#*********************************
#* Auslagern aus dem Lager Magdeburg *
```

```
#************************************
IF Lager = 3 THEN
{
          Bestand_Magdeburg [Artikelnummer,1] = (Bestand_Magdeburg
          [Artikelnummer,1] - 1)
          Auslieferung_Magdeburg [Artikelnummer,1] = (Ausliefe-
          rung_Magdeburg[Artikelnummer,1] +1)
          Auslieferung_Magdeburg [Artikelnummer,2] = (Ausliefe-
          rung_Magdeburg[Artikelnummer,2] +1)
}

#************************************
#* Auslagern aus dem Lager Heidelberg *
#************************************
IF Lager = 4 THEN
{
          Bestand_Heidelberg [Artikelnummer,1] = (Bestand_Heidelberg
          [Artikelnummer,1] - 1)
          Auslieferung_Heidelberg [Artikelnummer,1] = (Ausliefe-
          rung_Heidelberg[Artikelnummer,1] +1)
          Auslieferung_Heidelberg [Artikelnummer,2] = (Ausliefe-
          rung_Heidelberg[Artikelnummer,2] +1)
}

#*********************************
#* Auslagern aus dem Lager Paris *
#*********************************
IF Lager = 5 THEN
{
          Bestand_Paris [Artikelnummer,1] = (Bestand_Paris
          [Artikelnummer,1] - 1)
          Auslieferung_Paris [Artikelnummer,1] = (Auslieferung_Paris
          [Artikelnummer,1] +1)
          Auslieferung_Paris [Artikelnummer,2] = (Auslieferung_Paris
          [Artikelnummer,2] +1)
}
```

8.5.15 Subroutine Beschaffung 1

```
#************************************
#* Speditionsstrategie Beschaffung 1 *
```

```
#***********************************

Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
#*******************************
#* Warten bis Transportmenge erreicht *
#*******************************
WAIT UNTIL Sendungszähler >= Transportmenge_Beschaffung
Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
GROUP Transportmenge_Beschaffung AS Ladung
```

8.5.16 Subroutine Beschaffung 2

```
#************************************
#* Speditionsstrategie Beschaffung 2 *
#************************************

INT tmp
Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
#*************************************************
#* Warten bis maximale Wartezeit abgelaufen ist *
#*************************************************
WAIT maximale_Wartezeit_Beschaffung MIN
Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
#*************************************************
#* Warten bis Mindesttransportmenge erreicht ist *
#*************************************************
WAIT UNTIL Sendungszähler >= Mindestmenge_Beschaffung
IF CONTENTS (LOC(LOCATION()),Artikel)  <= Transportmenge_Beschaffung
          THEN
{
          tmp = CONTENTS (LOC(LOCATION()),Artikel)
          GROUP tmp AS Ladung
}
IF CONTENTS (LOC(LOCATION()),Artikel)  > Transportmenge_Beschaffung
          THEN
{
          GROUP Transportmenge_Beschaffung AS Ladung
}
```

8.5.17 Subroutine Beschaffung 3

```
#************************************
#* Speditionsstrategie Beschaffung 3 *
#************************************

INT tmp
#*********************************************
#* Warten bis maximale Wartezeit abgelaufen ist *
#*********************************************
WAIT maximale_Wartezeit_Beschaffung MIN
Sendungszähler = CONTENTS (LOC(LOCATION()),Artikel)
IF Sendungszähler <= Transportmenge_Beschaffung THEN
{
          tmp = Sendungszähler
          GROUP tmp AS Ladung
}
IF Sendungszähler > Transportmenge_Beschaffung THEN
{
          GROUP Transportmenge_Beschaffung AS Ladung
}
```

8.5.18 Subroutine Einlagern

```
#*************
#* Einlagern *
#*************

#*******************************
#* Einlagern im Lager Bielefeld *
#*******************************
IF Lager = 1 THEN
{
          Bestand_Bielefeld [Artikelnummer,1] = (Bestand_Bielefeld
          [Artikelnummer,1] + 1)
              IF Artikelnummer < 23 THEN Bestellung_Bielefeld
          [Artikelnummer,1] = (Bestellung_Bielefeld[Artikelnummer,1] -
          1)
}

#***************************
#* Einlagern im Lager Koeln *
#***************************
```

```
IF Lager = 2 THEN
{
          Bestand_Koeln [Artikelnummer,1] = (Bestand_Koeln
          [Artikelnummer,1] + 1)
          IF Artikelnummer < 23 THEN Bestellung_Koeln [Artikelnummer,1]
          = (Bestellung_Koeln [Artikelnummer,1] -1)
}

#*******************************
#* Einlagern im Lager Magdeburg *
#*******************************
IF Lager = 3 THEN
{
          Bestand_Magdeburg [Artikelnummer,1] = (Bestand_Magdeburg
          [Artikelnummer,1] + 1)
          IF Artikelnummer < 23 THEN Bestellung_Magdeburg [Artikelnum-
          mer,1] = (Bestellung_Magdeburg [Artikelnummer,1] -1)
}

#*******************************
#* Einlagern im Lager Heidelberg *
#*******************************
IF Lager = 4 THEN
{
          Bestand_Heidelberg [Artikelnummer,1] = (Bestand_Heidelberg
          [Artikelnummer,1] + 1)
}
#****************************
#* Einlagern im Lager Paris *
#****************************
IF Lager = 5 THEN
{
          Bestand_Paris [Artikelnummer,1] = (Bestand_Paris
          [Artikelnummer,1] + 1)
}
```

8.5.19 Subroutine Produktion 1

```
#************************
#* Produktionsstrategie 1 *
#************************
```

```
INT Produktionsmenge

#************************************
#* Produktion im Standort Heidelberg *
#************************************
IF Werk = 1 THEN
{
          #*****************************************
          #* Berechnung des aktuellen Simulationstages *
          #*****************************************
          Simulationstag = TRUNC(Clock(Day) + 1.25)
#*********************************************************************
          ********************
#* Durchsuchen des Arrays Produktion Heidelberg nach am aktuellen
          Simulationstag zu erzeugenden Artikeln*
#*********************************************************************
          ********************
          WHILE Produktion_Heidelberg [Produktionsnummer,1] =
          Simulationstag DO
          BEGIN
               Tag = Produktion_Heidelberg [Produktionsnummer,1]
               Artikelnummer = Produktion_Heidelberg [Produktionsnum-
          mer,2]
               Produktionsmenge = Produktion_Heidelberg [Produktions-
          nummer,3]
               Produktionsnummer = Produktionsnummer + 1
               #************************
               #* Erzeugung der Artikel *
               #************************
               ORDER Produktionsmenge Artikel TO Werk_Heidelberg
          END
}

#*********************************
#* Produktion im Standort Paris *
#*********************************
IF Werk = 2 THEN
{
          #************************************+********
          #* Berechnung des aktuellen Simulationstages *
          #*****************************************
```

```
          Simulationstag = TRUNC(Clock(Day) + 1.25)
#*****************************************************************
          ***************
#* Durchsuchen des Arrays Produktion Paris nach am aktuellen
          Simulationstag zu erzeugenden Artikeln *
#*****************************************************************
          ****************
          WHILE Produktion_Paris [Produktionsnummer,1] = Simulationstag
          DO
          BEGIN
               Tag = Produktion_Paris [Produktionsnummer,1]
               Artikelnummer = Produktion_Paris [Produktionsnummer,2]
               Produktionsmenge = Produktion_Paris [Produktionsnum-
          mer,3]
               Produktionsnummer = Produktionsnummer + 1
               #************************
               #* Erzeugung der Artikel *
               #************************
               ORDER Produktionsmenge Artikel TO Werk_Paris
          END
}
```

8.5.20 Subroutine Produktion 2

```
#**************************
#* Produktionsstrategie 2 *
#**************************
REAL Takt = 0
INT Zähler = 0
INT Produktionsmenge = 0
INT Artikelmenge = 0
#*********************************
#* Produktion im Werk Heidelberg *
#*********************************
IF Werk = 1 THEN
{
          #*****************************************
          #* Berechnung des aktuellen Simulationstages *
          #*****************************************
          Simulationstag = TRUNC(Clock(Day) + 1.25)
```

```
          #*************************************************************
          **
          #* Ermittlung  der Anzahl aller am Simulationstag zu
          produzierenden Artikel *
          #*************************************************************
          ***
          Zähler = Produktionsnummer
          WHILE Produktion_Heidelberg [Zähler,1] = Simulationstag DO
          BEGIN
               Produktionsmenge = Produktionsmenge + Produkti-
          on_Heidelberg[Zähler,3]
               INC Zähler , 1
          END

          IF Produktionsmenge > 0 THEN
          {
#*************************************************************
          #* Berechnung der zur Produktion der Artikel nötigen Taktung
          der Produktion *
          #*************************************************************
          ****
               Takt = 86000 / Produktionsmenge
#*************************************************************************
          ********************
#* Durchsuchen des Arrays Produktion Heidelberg nach am aktuellen
          Simulationstag zu erzeugenden Artikeln*
#*************************************************************************
          ********************
               WHILE Produktion_Heidelberg [Produktionsnummer,1] =
          Simulationstag DO
               BEGIN
                    Tag = Produktion_Heidelberg [Produktionsnummer,1]
                    Artikelnummer = Produktion_Heidelberg [Produktions-
          nummer,2]
                    Artikelmenge = Produktion_Heidelberg [Produktions-
          nummer,3]
                    Produktionsnummer = Produktionsnummer + 1
                    Zähler = 0
                    #*****************************************
                    #* Erzeugung der zu produzierenden Artikel *
                    #*****************************************
                    WHILE Zähler < Artikelmenge DO
```

```
                    BEGIN
                         ORDER 1 Artikel TO Werk_Heidelberg
                         INC Zähler , 1
                         WAIT Takt SEC
                    END
               END
          }
}
#***************************
#* Produktion im Werk Paris *
#***************************
IF Werk = 2 THEN
{
          #*****************************************
          #* Berechnung des aktuellen Simulationstages *
          #***************************+*****************
          Simulationstag = TRUNC(Clock(Day) + 1.25)
          #*************************************************************
          **
          #* Ermittlung  der Anzahl aller am Simulationstag zu
          produzierenden Artikel *
          #*************************************************************
          **
          Zähler = Produktionsnummer
          WHILE Produktion_Paris [Zähler,1] = Simulationstag DO
          BEGIN
               Produktionsmenge = Produktionsmenge + Produktion_Paris
          [Zähler,3]
               INC Zähler , 1
          END
          IF Produktionsmenge > 0 THEN
          {
          #*************************************************************
          ***
          #* Berechnung der zur Produktion der Artikel nötigen Taktung
          der Produktion *
          #*************************************************************
          ***
               Takt = 86000 / Produktionsmenge
#*****************************************************************
          ***************
```

```
#* Durchsuchen des Arrays Produktion Paris nach am aktuellen
          Simulationstag zu erzeugenden Artikeln*
#*********************************************************************
          ***************
               WHILE Produktion_Paris [Produktionsnummer,1] =
          Simulationstag DO
               BEGIN
                    Tag = Produktion_Paris [Produktionsnummer,1]
                    Artikelnummer = Produktion_Paris [Produktionsnum-
          mer,2]
                    Artikelmenge = Produkti-
          on_Paris[Produktionsnummer,3]
                    Produktionsnummer = Produktionsnummer + 1
                    #*****************************************
                    #* Erzeugung der zu produzierenden Artikel *
                    #*****************************************
                    Zähler = 0
                    WHILE Zähler < Artikelmenge DO
                    BEGIN
                         ORDER 1 Artikel TO Werk_Paris
                         INC Zähler , 1
                         WAIT Takt SEC
                    END
               END
          }
}
```

8.5.21 Subroutine Produktion 3

```
#*************************
#* Produktionsstrategie 3 *
#*************************

REAL Takt = 0
INT Zähler = 0
INT Zähler2 = 0
INT Produktionsmenge = 0
INT Artikelmenge

#*******************************
#* Produktion im Werk Heidelberg *
```

```
#*********************************
IF Werk = 1 THEN
{
          Zähler = 1
          #*************************************************************
          **
          #* Ermittlung  der Anzahl aller am Simulationstag zu
          produzierenden Artikel *
          #*************************************************************
          **
          WHILE Zähler <= 46 DO
          BEGIN
               Produktionsmenge = Produktionsmenge + Bestel-
          lung_Heidelberg[Zähler,1]
               Bestellung_Heidelberg [Zähler,3] = Bestellung_Heidelberg
          [Zähler,1]
               Bestellung_Heidelberg [Zähler,1] = 0
               INC Zähler , 1
          END
          Zähler = 1
          IF Produktionsmenge > 0 THEN
          {
          #*************************************************************
          ****
          #* Berechnung der zur Produktion der Artikel nötigen Taktung
          der Produktion *
          #*************************************************************
          ****
               Takt = 86000 / Produktionsmenge

               #*******************************************************
          **********
               #* Durchsuchen des Arrays Bestellung Heidelberg nach zu
          erzeugenden Artikeln*

               #*******************************************************
          **********
               WHILE Zähler <= 46 DO
               BEGIN
                    Tag = Simulationstag
                    Artikelnummer = Zähler
                    Artikelmenge = Bestellung_Heidelberg [Zähler,3]
```

```
                    Bestellung_Heidelberg [Zähler,3] = 0
                    #***********************************
                    #* Erzeugung der zu produzierenden Artikel *
                    #***********************************
                    Zähler2 = 0
                    WHILE Zähler2 < Artikelmenge DO
                    BEGIN
                         ORDER 1 Artikel TO Werk_Heidelberg
                         INC Zähler2 , 1
                         WAIT Takt SEC
                    END
                    INC Zähler , 1
               END
          }
}

#****************************
#* Produktion im Werk Paris *
#****************************
IF Werk = 2 THEN
{
          Zähler = 1
          #************************************************************
          ***
          #* Ermittlung  der Anzahl aller am Simulationstag zu
          produzierenden Artikel *
          #************************************************************
          ***
          WHILE Zähler <= 46 DO
          BEGIN
               Produktionsmenge = Produktionsmenge + Bestellung_Paris
          [Zähler,1]
               Bestellung_Paris [Zähler,1] = Bestellung_Paris
          [Zähler,3]
               Bestellung_Paris [Zähler,1] = 0
               INC Zähler , 1
          END
          Zähler = 1
          IF Produktionsmenge > 0 THEN
          {
```

```
     #*****************************************************
*********
     #* Berechnung der zur Produktion der Artikel nötigen
Taktung der Produktion *

     #*****************************************************
*********
     Takt = 86000 / Produktionsmenge

     #*****************************************************
*****
     #* Durchsuchen des Arrays Bestellung Paris nach zu
erzeugenden Artikeln*

     #*****************************************************
*****
     WHILE Zähler <= 46 DO
     BEGIN
          Tag = Simulationstag
          Artikelnummer = Zähler
          Artikelmenge = Bestellung_Paris[Zähler,3]
          Bestellung_Paris[Zähler,3] = 0
          Zähler2 = 0
          #****************************************
          #* Erzeugung der zu produzierenden Artikel *
          #****************************************
          WHILE Zähler2 < Artikelmenge DO
          BEGIN
               ORDER 1 Artikel TO Werk_Paris
               INC Zähler2 , 1
               WAIT Takt SEC
          END
          INC Zähler , 1
     END
     }
}
```

8.6 Importdaten

Sämtliche Dateien aus denen Daten importiert werden befinden sich im Verzeichnis *BeDi\Importdaten*. Die einzelnen Importdaten werden im Folgenden beschrieben:

8.6.1 Anfangsbestände

Die Anfangsbestände sind in der Datei *Anfangsbestände.xls* gespeichert. Die Excel-Arbeitsmappe enthält für jedes Lager ein eigenes Blatt. In Spalte A werden die Artikelnummern angegeben und in Spalte B die jeweils vorhandenen Mengen.

8.6.2 Dispositionsdaten

Die Dispositionsdaten sind in der Datei *Dispositionsdaten.xls* gespeichert. Die Excel-Arbeitsmappe enthält für jedes der drei Regionallager ein eigenes Blatt. Importiert werden die Spalten D bis I.

Die Spalte C enthält die Artikelnummer, die Spalte D die zugehörige Nachbestellmenge in Stück, die Spalte E die Bestellgrenze, d.h. die Kapazitätsgrenze des Lagers für den jeweiligen Artikel. In Spalte F steht die jeweilige Nachbestellmenge in Stück, in Spalte G befindet sich die Menge pro Transporteinheit. Die Spalten H und I enthalten den ersten Bestelltag und die Bestellperiode jeweils in Simulationstagen.

8.6.3 Kundenbestellungen

Die Kundenbestellungen sind in der Datei *Kundenbestellungen.xls* gespeichert. Die Excel-Arbeitsmappe enthält ein Blatt in welchem alle Kundenbestellungen aufgelistet sind und das Blatt *Summe_Kundenbest*, in welchem die einzelnen Bestellungen kumuliert wurden. Importiert werden Daten vom Blatt *Summe_Kundenbest* aus den Spalten G bis K.

In Spalte G wird der Simulationstag angegeben, an welchem der Auftrag erzeugt wird. Spalte H enthält den Wunschlieferanten, welcher sich aus dem Schlüssel 1 = Bielefeld, 2 = Koeln, 3 = Magdeburg, 4 = Heidelberg und 5 = Paris ergibt. Spalte I enthält den Kunden, welcher sich aus dem Schlüssel 1 = Bielefeld, 2 = Koeln, 3 = Magdeburg und 4 = Heidelberg ergibt.

Spalte J enthält die Artikelnummer des bestellten Artikels und Spalte K die bestellte Menge in Stück.

8.6.4 Produktionsdaten

Die Produktionsdaten sind in der Datei *Produktionsdaten.xls* gespeichert. Die Excel-Arbeitsmappe enthält für jeden der beiden Standorte Paris und Heidelberg ein eigenes Blatt. Importiert werden jeweils die Spalten B bis D.

Spalte B enthält dem Simulationstag, an welchem die Artikel erzeugt werden. Spalte C enthält die Artikelnummer und Spalte D die jeweils zu produzierende Menge in Stück.

8.6.5 Artikelstammdaten

Die Artikelstammdaten sind in der Datei *Artikelstammdaten.xls* gespeichert. Importiert werden die Spalten B bis D. Spalte B enthält die Artikelnummer, Spalte C den Wert des Artikels in € und Spalte C das Gewicht in Kilogramm.

8.7 Exportdaten

Sämtliche Dateien in denen Daten geschrieben werden befinden sich im Verzeichnis *BeDis\Exportdaten*. In diesem Verzeichnis sind die Daten für die einzelnen Szenarien in den Szenario-Unterverzeichnissen gespeichert. Neue Daten werden immer im Hauptverzeichnis gespeichert. Die einzelnen Exportdaten werden im Folgenden Beschrieben:

8.7.1 Artikeldurchsatz

Die Datei *Artikeldurchsatz.xls* enthält den Durchsatz der einzelnen Artikel. Hier wird ein Artikel vermerkt, sobald er beim Kunden angekommen ist. Die Zeilennummer minus eins entspricht der Artikelnummer. In Spalte A ist die Anzahl der jeweils durchgesetzten Artikel in Stück angegeben.

8.7.2 Auslieferung

Die aus den einzelnen Lagern ausgelieferten Artikelmengen sind in den Dateien *Auslieferung xxx.xls* gespeichert. Auch hier entspricht die Zeile minus eins der Artikelnummer. Spalte A enthält die Anzahl der jeweils seit der letzten Nachbestellung ausgelieferten Artikel und Spalte B die Anzahl der insgesamt ausgelieferten Artikel.

8.7.3 Bestellung

Die Dateien *Bestellung_Bielefeld.xls*, *Bestellung_Koeln.xls* und *Bestellung_Magdeburg.xls* enthalten die nachbestellten, aber noch nicht vom Werk gelieferten Mengen der jeweiligen Artikel. Diese Mengen stehen in Spalte A. Die Zeilennummer minus eins entspricht der Artikelnummer.

Die Dateien *Bestellung_Heidelberg.xls* und *Bestellung_Paris.xls* enthalten die Artikelmengen, die von den Regionallagern in den beiden Produktionsstandorten nachbestellt wurden. Die Zeilennummer minus eins entspricht der Artikelnummer. In Spalte A stehen die am aktuellen Simulationstag eingegangenen und noch nicht produzierten Bestellmengen, in Spalte B die insgesamt bestellten Mengen und in Spalte C die am Vortag bestellten und noch zu produzierenden Artikelmengen.

8.7.4 Dispositionsdatenvalidierung

Die Dateien *Dispositionsvalidierung xxx.xls* dienen zur Validierung der Dispositionsdatenarrays. Die Zeilennummer minus eins entspricht der Artikelnummer, Die Spalte A dem Meldebestand, die Spalte B der Bestellgrenze, die Spalte C der Nachbestellmenge, die Spalte D der Menge pro Transporteinheit, die Spalte E dem ersten Bestelltag und die Spalte F der Bestellperiode.

8.7.5 Endlagerbestände

Die Dateien *Endlagerbestand xxx.xls* geben die Anzahl der sich zum Simulationsende im jeweiligen Lager befindenden Artikel an. Die Zeilennummer minus eins entspricht der Artikelnummer und die Werte in Spalte A der jeweiligen Artikelanzahl.

8.7.6 Validierung Kundenbestellungen

Die Datei *Validierung Kundenbestellungen.xls* dient zur Validierung des Kundenbestellungsarrays. Die Zeile minus eins entspricht der Auftragsnummer, Spalte A dem Bestelltag, Spalte B der Wunschlieferantennummer, Spalte C der Kundennummer, Spalte D der Artikelnummer und Spalte E der Bestellmenge. In Spalte F ist zusätzlich die Auftragsdurchlaufzeit in Stunden gespeichert.

8.7.7 Validierung Produktion

Die Dateien *Validierung Produktion Heidelberg.xls* und *Validierung Produktion Paris.xls* dienen zur Validierung der jeweiligen Produktionsarrays. Die Zeilennummer minus eins entspricht der Produktionsnummer, Spalte A dem Tag, Spalte B der Artikelnummer und Spalte C der Menge.

8.7.8 Validierung Artikelstammdaten

Die Datei Validierung Artikelstammdaten.xls dient zur Validierung des Artikelstammdatenarrays. Die Zeilennummer minus eins entspricht der Artikelnummer, die Werte in Spalte A dem Artikelwert in € und die Werte in Spalte B dem Gewicht des jeweiligen Artikels in Kilogramm.

9 Literaturverzeichnis

/AMBE93/ Amberger, P.: Spedition – Heute und morgen. Aufgaben und Leistungen in der Zukunft. In: VDI Berichte 1050. Spedition. Glied der Logistikkette. Hrsg.: VDI, VDI Verlag, Düsseldorf 1993.

/ASIM97/ N.N.: ASIM-Mitteilungen aus den Fachgruppen. Leitfaden für Simulationsbenutzer in Produktion und Logistik. Heft Nr. 58, o.V., 1997.

/BICH79/ Bichler, K.: Beschaffung und Lagerhaltung im Handelsbetrieb Teil 2. Gabler Studientexte, 1979

/BISC95/ Bischof, K. D.; u.a.: Speditionsbetriebslehre. 4. Aufl., Stam Verlag, Köln München 1995.

/BUCH98/ Buchholz, J.; Clausen, U.; Vastag, A. (Hrsg.): Handbuch der Verkehrslogistik. Springer, Berlin u.a. 1998.

/BULL94/ Bullinger, H.-J.: Planung der Materialbereitstellung in der Montage. Teubner, Stuttgart 1994.

/DUDE90/ N.N.: Duden. Das Fremdwörterbuch. 5. Aufl., Bibliographisches Institut & F.A. Brockhaus, Mannheim u.a. 1990.

/GUDE00a/ Gudehus, T.: Logistik 1. Grundlagen, Verfahren und Strategien. Springer, Berlin u.a. 2000

/GUDE00b/ Gudehus, T.: Logistik 2. Netzwerke, Systeme und Lieferketten. Springer, Berlin u.a. 2000

/HAMM86/ Hammann, P.; Lohrberg, W.: Beschaffungsmarketing. O.V., Stuttgart 1986.

/HELL99/ Hellingrath, B.; Hellmann, A.; Felder, A.: Simulation von Fabriksystemen. Unterlagen zur Vorlesung Fabrikmodellierung II. Sommersemester 1999., Universität Dortmund, Dortmund 1999.

/HUGH00/ Hughes, J.; Ralf, M.; Michels, B.: Supply Chain Management. So steigern Sie die Effizienz Ihres Unternehmens durch perfekte Organisation der Wertschöpfungskette. Verlag moderne Industrie, Landsberg/Lech 2000.

/JÜNE89/ Jünemann, R.: Materialfluß und Logistik. Springer, Berlin u.a. 1989.

/JÜNE99/ Jünemann, R.; Schmidt, T.: Materialflusssysteme. 2. Aufl., Springer, Berlin u.a. 1999.

/KLEI72/ Kleine, O.; Melzow, W.: Disponieren in der modernen Beschaffung. Bedarfsermittlung Mathematische Bestellmengenrechnung Elektronische Datenverarbeitung. Beuth-Vertrieb, Berlin Köln Frankfurt (Main) 1972.

/KUHN95/ Kuhn, A.: Prozeßketten in der Logistik. Praxiswissen, Dortmund 1995.

/MERT86/ Mertens, J.: Beiträge zum Beschaffungsmarketing Band 6. Konstellationsadäquate Beschaffungspolitik im Handel. Fördergesellschaft Produkt-Marketing e.V., Köln 1986.

/MEYE86/ Meyer, C.: Beschaffungsziele. o.V., Köln 1986.

/MIEH98/ Miehler, G.: Zeitcontrolling indirekter Prozessketten. Gabler, Wiesbaden 1998.

/NOCH89/ Noche, B.: Simulation in Produktion und Materialfluß. Entscheidungsorientierte Simulationsumgebung. Dissertation, Universität Dortmund, Verlag TÜV Rheinland, Dortmund 1989.

/NOCH93/ Noche, B.; u.a.: Simulationsinstrumente im Überblick. In: Handbuch Simulationsanwendungen in Produktion und Logistik. Fortschritte der Simulationstechnik; Band 7. Hrsg.: Kuhn, A.; Reinhardt, A.; Wiendahl, H.-P., Vieweg, Braunschweig Wiesbaden, 1993, S. 267 – 307.

/NOCH01/ Noche, B.: Standards in der Logistiksimulation. In: Frontiers in Simulation. Simulationstechnik 15. Symposium in Paderborn September 2001. Hrsg.: Panreck, K.; Dörrscheidt, F., SCS-Europe BVBA, Ghent 2001.

/PALU98/ Palupski, R.: Management von Beschaffung, Produktion und Absatz. Leitfaden mit Praxisbeispielen. Gabler, Wiesbaden 1998.

/PION94/ Piontek, J.: Beschaffungscontrolling. Oldenbourg Verlag, München 1994.

/PROM98a/ N.N.: promodel. users guide. PROMODEL Corporation, Orem 1998.

/PROM98b/ N.N.: promodel. reference guide. PROMODEL Corporation, Orem 1998.

/PROM98c/ N.N.: promodel. addendum. PROMODEL Corporation, Orem 1998.

/SCHU99/ Schulte, C.: Lexikon der Logistik. Oldenbourg-Verlag, München Wien 1999.

/VDI70/ VDI (Hrsg.): Richtlinie VDI 2411: Begriffe und Erläuterungen im Förderwesen. Beuth, Berlin 1970.

/VDI73/ VDI (Hrsg.): Richtlinie VDI 3300: Anleitungen für Materialflußuntersuchungen. Beuth, Berlin 1973.

/VDI00/ VDI (Hrsg.): Richtlinie VDI 3633: Simulation von Logistik-, Materialfluß- und Produktionssystemen – Grundlagen. Blatt 1, Entwurf. In: VDI-Handbuch Materialfluß und Fördertechnik, Band 8. Hrsg.: VDI, Beuth, Berlin 2000.

/WEBE01/ Weber, J.; Blum, H.: Logistik-Controlling. Konzept und empirischer Stand. In: Schriftenreihe Advanced Controlling, Bd. 20, Vallendar, 2001.

/WEBE02/ Weber, J.: Logistikkostenrechnung. Kosten-, Leistungs- und Erlösinformationen zur erfolgsorientierten Steuerung der Logistik. 2. Aufl., Springer-Verlag, Berlin Heidelberg New York 2002.

/WEID02/ Weidt, S.: Neue Geschäftsprozesse in der Beschaffung. Potenziale durch den Einsatz von E-Lösungen. In: Logistik Jahrbuch 2002. Hrsg.: Hossner, R., Verlagsgruppe Handelsblatt GmbH, Düsseldorf 2002, S. 88 – 91.

/WENZ00a/ Wenzel, S.; Noche, B.: Simulationsinstrumente in Produktion und Logistik – eine Marktübersicht. In: The new simulation in production and logistics: Prospects, views and attitudes. Hrsg.: Mertins, K.; Rabe, M., IPK Eigenverlag, Berlin 2000, S. 423 – 432.

/WENZ00b/ Wenzel, S.: Frontiers in Simulation. Referenzmodelle für die Simulation in Produktion und Logistik. SCS-Europe BVBA, Ghent 2000.

/WILD87/ Wildemann, H.: Produktion und Zulieferung auf Abruf: Das Just-In-Time Konzept. Universität Passau, Passau 1987.