HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Modell Baukasten Supply-Chain

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2020

Autoren:	Fabienne Lienhard, Jana Kravarik
Betreuer:	Prof. Dr. Andreas Rinkel
Abgabe:	29.05.2020

ABSTRACT

Ausgangslage und Problem

Eine Supply Chain hat zum Ziel, komplexe Vorgänge einer Lieferkette zu beschreiben. Darunter fallen der Informations-, Waren-, Geld- und Personenfluss. Die Analyse und Optimierung von Supply Chains ist ein wichtiger Punkt in der Industrie. Dafür wird oft auf Simulation zurückgegriffen, doch sind Simulationen nicht immer optimal ausgelegt dafür, die Domäne von Supply Chains darzustellen.

Die Simulationssoftware Simio enthält zum Beispiel Komponenten, die vor allem für einzelne Prozessschritte gedacht sind und daher ungeeignet dafür, komplexe Stationen einer Supply Chain abzubilden. Jedoch bietet Simio die Möglichkeit, Komponenten anzupassen oder neue zu definieren. Zum Abbilden einer solchen Supply Chain müssen also komplexere Bausteine geplant und in Simio umgesetzt werden.

Ziel

In der vorliegenden Arbeit wird ein Modellbaukastensystem für die Simulationssoftware Simio erstellt, das für die Darstellung einer einfachen Supply Chain Bausteine bereitstellt. Diese Bausteine sind erweiterbar definiert, so dass in Zukunft auch komplexere Supply Chains damit simuliert werden können.

Methode und Vorgehen

Die Arbeit setzt sich zusammen aus einem theoretischen und einem praktischen Teil. Im theoretischen Teil sind die Grundlagen von Supply Chains erarbeitet und hinsichtlich ihrer Möglichkeiten und Wichtigkeit in der Simulation mit Simio modelliert und spezifiziert. Der praktische Teil setzt dies in einem ersten Prototyp um.

Ergebnisse

Das Ergebnis der Arbeit ist ein in Simio umgesetzter «Prototyp0», der eine vereinfachte Supply Chain darstellt, die aus drei Stationen besteht: Kunde, Lager und Hersteller. Des Weiteren sind mehrere darauf aufbauende Prototypen vordefiniert, welche in der auf dieser Arbeit folgenden Bachelorarbeit weiter ausformuliert und umgesetzt werden.

Ausblick

In der auf dieser Arbeit folgenden Bachelorarbeit, werden die Stationsmodule anhand der vordefinierten Prototypen weiter ausgebaut, so dass sie das Abbilden von komplexeren Supply Chains ermöglichen.

DANKSAGUNG

An dieser Stelle möchten wir herzlichst unseren Familien und Freunden danken, die uns während dieser Arbeit immer unterstützt haben.

Auch danken wir Professor Andreas Rinkel für die kompetente Betreuung während der Studienarbeit.

INHALTSVERZEICHNIS

1 EINLEITUNG

Auftrag

Es ist ein Modell Baukasten für Supply Chains in der Simulationssoftware «Simio» zu erstellen. Das Themengebiet «Supply Chain» ist sehr gross, weswegen der Auftrag in eine mehrheitlich theoretische Studienarbeit und eine vorwiegend praktische Bachelorarbeit unterteilt wird. Die vorliegende Studienarbeit konzentriert sich auf den Entwurf eines ersten einfachen Prototyps.

Während der Bachelorarbeit werden darauf aufbauende Prototypen weiter definiert und umgesetzt. Zudem wird die Optimierung der Supply Chain anhand von überprüfbaren Key Performance Indikatoren ermöglicht.

Ziel

In dieser Studienarbeit wird die Umsetzung einer einfachen Supply Chain in der Simulationssoftware Simio geplant und umgesetzt. Es werden Möglichkeiten vorgestellt und ein Prototyp «Prototyp0» ausgearbeitet, welcher sich auf die Grundfunktionen konzentriert.

Die Umsetzung von weiteren Prototypen wird im Rahmen der auf dieser Arbeit aufbauenden Bachelorarbeit durchgeführt.

Vorgehen bei der Umsetzung

Es wird ein «Prototyp0» einer vereinfachten Supply Chain definiert. Dieser besteht aus der minimalen Anzahl von verschiedenen Stationen, welche für den Aufbau einer Supply Chain benötigt werden: Kunde, Lager und Hersteller. Dieses vereinfachte Model wird bereits erweiterbar erstellt und legt den Grundstein für alle später darauffolgenden Simulationen.

Überblick über die Arbeit

In den ersten Kapiteln der Arbeit wird mit den Supply Chain Grundlagen, dem Scor Modell und dem Lager der Stand der Technik festgehalten.

Ab Kapitel 4 liegt der Fokus auf der Umsetzung in Simio. Es wird zunächst ein Überblick über die Software vermittelt und anschliessend die eigentliche Umsetzung geplant, dokumentiert und analysiert.

Wo möglich, werden die englischen Begriffe verwendet und wo nötig, die deutschen Begriffe in Klammern angegeben.

2 SUPPLY CHAIN GRUNDLAGEN

2.1 DEFINITION SUPPLY CHAIN

Die Definition einer Supply Chain ist davon abhängig, aus welcher Perspektive sie betrachtet wird. Es wird zwischen Lieferanten- und Kundenansicht unterschieden:

Aus Sicht des Lieferanten Die Supply Chain besteht aus verschiedenen Stationen, welche jeweils Güter untereinander weiterreichen, bis sie schliesslich als «beim Kunden angekommen» gelten.

Aus Sicht des Kunden Die Transportunternehmen, Lagerhäuser und Händler in einer Supply Chain stehen im Vordergrund.

Den Ansichten gemeinsam ist, dass jeweils von einer Kette von Stationen gesprochen wird, welche Güter in eine Richtung und Informationen in die entgegengesetzte Richtung transportieren. In Abbildung 1 wird ein Ausschnitt der Supply Chain von Toilettenpapier dargestellt. Der Fokus dabei liegt auf dem Toilettenpapier-Hersteller und seinem direkten Zulieferer und Abnehmern, wobei das Toilettenpapier für **Commercial Use** und das für **Residential Use** bereits beim Hersteller verschieden verpackt und so zu zwei verschiedenen Produkten wird.

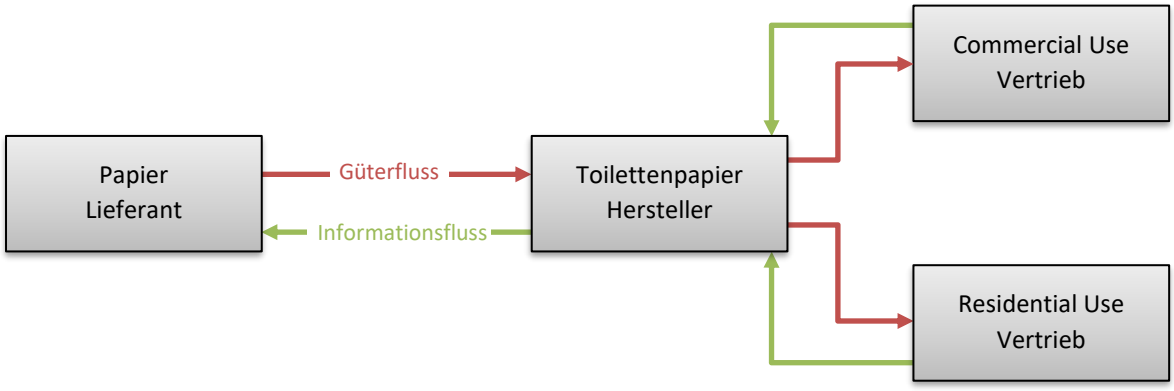


Abbildung 1: Ausschnitt aus Toilettenpapier Supply Chain (vereinfacht)

In diesem Beispiel besteht der Güterfluss aus Altpapier und Toilettenpapier. Der Informationsfluss besteht aus Bestellungen an den jeweiligen Zulieferer. Für eine bessere Veranschaulichung wird in Abbildung 2, die Toilettenpapier-Supply Chain mit möglichen Stationen bis zum Kunden ergänzt. Der Begriff «Supply Chain» umfasst dabei alle Stationen, wie diese zueinanderstehen und wie der Güter- und Informationsfluss zwischen den einzelnen Stationen erfolgt.

Güterfluss Dieser Fluss beinhaltet all das, was von einem Ende einer Supply Chain zum anderen Ende fliesst, also vom Lieferanten bis zum Kunden.

Informationsfluss Dieser umfasst alle Dokumente, Bestellungen, Reklamationen und Meldungen bezüglich eines Produkts und seinen Vorstufen.

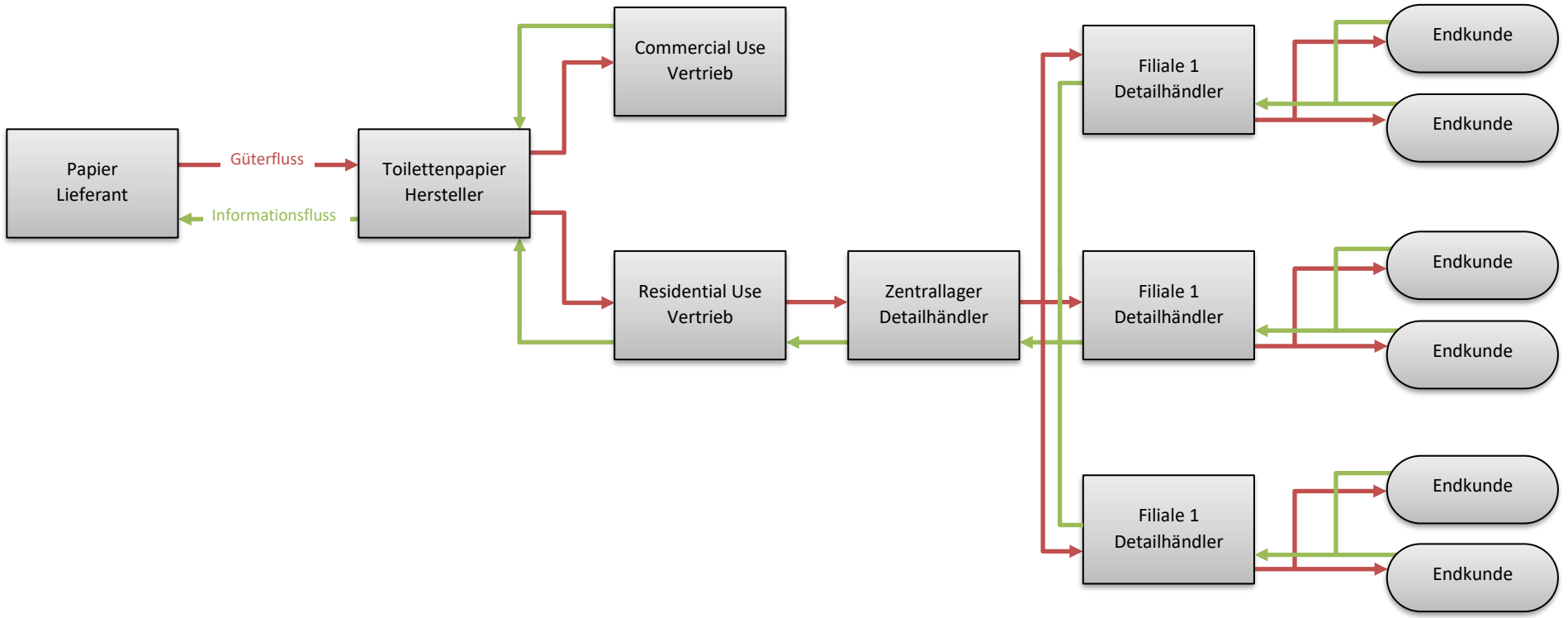


Abbildung 2: Supply Chain Toilettenpapier

Es kann zum Beispiel sein, dass einem Kunden nicht bewusst ist, aus was und/oder wie ein Produkt hergestellt wird, dies ist auch nicht relevant für ihn. Der Kunde muss lediglich wissen, dass er Toilettenpapier kaufen muss und wo und wie er das kann. Anhand Abbildung 2 würde ein Kunde zu einer Filiale eines Detailhändlers gehen und diesen darüber informieren, dass er Toilettenpapier kaufen möchte. Der Detailhändler nimmt diese Information entgegen und liefert ihm Toilettenpapier.

Unter der Annahme, dass die Supply Chain nicht optimiert wäre, würde der Detailhändler seinen Kunden solange Toilettenpapier verkaufen, bis ihm auffällt, dass er nur noch wenig Toilettenpapier hat und dann beim Zentrallager eine Anfrage nach mehr Toilettenpapier einreichen. Die Verantwortlichen beim Zentrallager wiederum beobachten ihren eigenen Stand und werden anhand dessen beim Vertrieb vom Residential Use Toilettenpapier nachbestellen. Die Station «Residential Use» bezieht ihrerseits das Papier direkt beim Hersteller.

Der zuvor geschilderte Ablauf ist nicht optimiert. Wenn zum Beispiel etwas passiert, das die Kunden dazu veranlasst, deutlich mehr Toilettenpapier zu kaufen, so gäbe es eine Verzögerung, bis der Hersteller dies bemerkt und darauf reagieren kann. Dies ist jedoch notwendig, um die Produktion so zu steigern, damit sie der erhöhten Nachfrage gerecht werden kann.

Aus diesem Grund werden Supply Chains oft als ganze Ketten und nicht nur aus direktem Zulieferer und Abnehmer dargestellt und dementsprechend optimiert.

Im Beispiel des Toilettenpapiers bedeutet dies, dass die Detailhändler die erhöhte Nachfrage als Information sofort weitergeben und nicht erst wenn ihre Lager leer sind. Dasselbe gilt für alle nachfolgenden Stationen; so erfährt der Hersteller signifikant schneller, dass mehr Toilettenpapier nachgefragt wird und er seine Produktion hochfahren muss. Diese Optimierung des Informations- und Güterflusses hat auch dazu geführt, dass neuere Systeme die Supply Chain mit einem bidirektionalen Informationsfluss darstellen. Es hat sich herausgestellt, dass der Informationsaustausch in beide Richtungen oft bessere Resultate liefert als beim unidirektionalen.

2.2 KATEGORISIERUNG

Die einzelnen Schritte innerhalb einer Supply Chain unterscheiden sich anhand des jeweiligen Produkts und dessen Herstellung. Nachfolgend einige mögliche Kategorisierungen.

2.2.1 WERTSCHÖPFENDE SCHRITTE

Die wertschöpfenden Schritte lassen sich in fünf allgemeingültige Aktionen unterteilen:
(Poluha, S. 14)

Produzieren «make»	Das Herstellen von Materialien oder Bauteilen et cetera. Zum Beispiel das Herstellen von Toilettenpapier.
Kombinieren «combine»	Das Zusammenbauen, Verpacken et cetera. Zum Beispiel das Aufrollen von Toilettenpapier auf Kartonrollen.
Bewegen «move»	Das Verteilen, Sammeln et cetera. Zum Beispiel der Transport von Hersteller zum Vertrieb.

Lagern «store»
Das Einlagern, Handeln et cetera.
Zum Beispiel die Lagerung vom Toilettenpapier im Zentrallager des Detailhändlers.

Anpassen «customize»
Die Installation, Konfiguration et cetera.

2.2.2 Produkt- oder Kundenzentriert

Die gesamte Supply Chain kann anhand von verschiedenen Kriterien kategorisiert werden. Zum Beispiel nach produktzentrierten und kundenzentrierten Supply Chains:

produktzentriert «product-centric»
Bei der produktzentrierten Supply Chain liegt der Fokus auf einem Produkt und aus der Supply Chain, können ein oder mehrere Produktangebote resultieren.

kundenzentriert «customer-centric»
Bei der kundenzentrierten Supply Chain wird sich hingegen auf ein spezielles Marktsegment konzentriert und die resultierenden Supply Chains werden direkt für dieses Marktsegment zugeschnitten.

2.2.3 Geschäftsstrategie

Eine weitere Möglichkeit der Kategorisierung von Supply Chains ist, sie nach Geschäftsstrategie zu unterteilen. Hier liegt der Fokus entweder auf der Beziehung hin zum Kunden oder auf der zu den Konkurrenten.

2.3 Supply chain Management

Das Hauptziel vom Supply Chain Management besteht darin, den Absatz von Gütern und Dienstleistungen an den Kunden zu steigern, und dabei gleichzeitig die Lagerbestände und die damit verbundenen Kosten zu minimieren. Um dabei eine möglichst effiziente Supply Chain zu erhalten, müssen oft sogenannte «trade-offs» zwischen den verschiedenen Zielen vereinbart werden. Es gibt dabei zwei Hauptstrategien, die für das Darstellen und Optimieren verwendet werden können:

Wertbasierte Supply Chain
«value chain strategy»
Die Wertkettenstrategie wird oft von Unternehmen eingesetzt, welche in konkurrenzstarken Märkten tätig sind. Diese konzentrieren sich darauf, innovative Produkte mit möglichst hoher Qualität zu marktfähigen Preisen als Erstes anbieten zu können. Dabei handelt es sich oft um ein grosses Unternehmen, das für jedes Produkt eine eigene Supply Chain und für jede Supply Chain einen Konkurrenten hat. Dieses zentrale Unternehmen kümmert sich um das Supply Chain Management und hat auch einen starken Einfluss auf die Zulieferer und Abnehmer. So erschafft das Unternehmen eine möglichst ertragreiche Supply Chain. Dieses System basiert auf der Erkenntnis, dass die Optimierung eines bestimmten Teiles der Supply Chain nicht unbedingt kosteneffizient für die Verbesserung der gesamten Supply Chain ist. Darum wird zentral entschieden, was an der Kette geändert werden muss, um eine möglichst effiziente und ertragreiche Supply Chain zu erhalten.

Desintegrierte Supply Chain

«de-integrated supply chain strategies»

Bei den desintegrierten Supply Chain Strategien werden vom initiierenden Unternehmen nur etwa 15% des Mehrwerts zur Supply Chain beigetragen.
Mit diesem Ansatz der Supply Chain, welche aus eng zusammenarbeitenden und doch komplett selbständigen Unternehmen besteht, werden auch kundenspezifische Serienfertigungen ermöglicht.

2.4 LEISTUNGSINDIKATOREN

Leistungsindikatoren werden definiert, um die Leistungen und Effizienz von Supply Chains oder anderen Gebilden messen und simulieren zu können. Sie bestehen aus Grenz- und Idealwerten und sollen einfach festzulegen, leicht anzuwenden und einfach zu verstehen sein. Sie werden hauptsächlich von Managern zur Prozessoptimierung und/oder schnellem Reagieren bei Änderungen verwendet.

2.5 BULLWHIP EFFECT

«Der Bullwhip-Effekt oder auch Peitscheneffekt ist ein Phänomen, das bei Schwankungen der Nachfrage entlang der gesamten Supply Chain auftritt, wobei die Schwankungen umso stärker werden, je weiter man sich in der Supply Chain vom Endkunden über die Händler und Großhändler bis hin zu den Herstellern der Güter bewegt.» (*Bullwhip Effekt – einfache Definition & Erklärung, https://www.rechnungswesen-verstehen.de/lexikon/bullwhip-effekt.php, abgerufen am 30.03.2020.)*

Wenn es nur schon zu kleinen Schwankungen in der Kundennachfrage kommt, weiten die vorhergehenden Stationen einer Supply Chain ihre Bestellmenge aus, um Lieferengpässe zu vermeiden. Dies hat zur Folge, dass die Stationen davor einer noch höheren Bestellmenge ausgesetzt sind und wiederum ihre Bestellmenge oder Produktion ebenfalls erhöhen, um einer noch höheren Nachfrage gerecht werden zu können. Der sich selbst verstärkende Effekt hat zur Folge, dass bereits kleine Schwankungen in der Endkundennachfrage zu riesigen Bestellmengen bei den einzelnen Stationen einer Supply Chain führen können.

Im Rahmen der vorliegenden Arbeit wird der Bullwhip Effekt vernachlässigt, da es sich bei diesem Phänomen hauptsächlich um einen psychologischen Effekt handelt. Für eventuell fortführende Arbeiten wäre die genauere Betrachtung dieses Effektes jedoch von Relevanz.

3 MODELLVORSTELLUNGEN

Ein Modell ist eine mögliche Abstraktion eines bestehenden Systems oder auch eines bereits vorhandenen Modells. Es dient der Zusammenfassung von Annahmen und Näherungen über bestimmte Systemverhalten. Die Betrachtung eines Modells anstatt des eigentlichen Systems bringt mehrere Vorteile mit sich. Es ist günstiger, schneller, sicherer und einfacher ein Modell zu verwenden. Oft bringt die Anwendung eines Modells auch ein besseres Verständnis für das eigentliche System.

3.1 LITERATUR

3.1.1 ANWENDUNG DES SCOR-MODELLS ZUR ANALYSE DER SUPPLY CHAIN

Poluha, R., 2010. *Anwendung Des SCOR-Modells Zur Analyse Der Supply Chain*. Lohmar: Eul.

Beschreibung In diesem Buch wird eine empirische Untersuchung der Anwendung des SCOR-Modells auf Supply Chains durchgeführt und beschrieben. Es werden Thesen aufgestellt und mit erhobenen Fragebogendaten überprüft.

Der Text enthält Erklärungen zu den verschiedenen Typen und Klassifikationen von Supply Chains. Es definiert genau, was SCOR ist und wie SCOR aufgebaut ist.

Relevanz Dieses Buch dient als erste Einführung zu den verschiedenen Supply Chain Typen und wie die Unterscheidung der einzelnen Typen zustande kam. Zusätzlich gibt es einen Überblick darüber, wie SCOR funktioniert und anzuwenden ist.

3.1.2 BAUKASTENSYSTEM FÜR SIMIO-MODELLBIBLIOTHEK

Donno, E., 2020. *Baukastensystem für Simio-Modellbibliothek.* BAWVOR

Beschreibung Die Arbeit beschäftig sich mit verschiedenen Lagertypen und wie sich diese in der Simulationssoftware «Simio» abbilden lassen. Sie gibt eine gute Übersicht über die verschiedenen Lagertypen, deren Gemeinsamkeiten und Unterschiede. Weiter werden Methoden erläutert, wie ein Baukastensystem in Simio erstellt werden kann, welches das Abbilden der verschiedenen Lagertypen erlaubt.

Relevanz Diese Quelle hat mehrere Parallelen zu der Zielsetzung der vorliegenden Arbeit. Die darin behandelten Lagertypen sind wichtige Bestandteile von Supply-Chains und können unterstützend genutzt werden, allfällige Lager-Bausteine zu definieren, die für die Umsetzung von Supply Chains in Simio gebraucht werden.

3.1.3 EINSATZ FORMALER METHODEN IN DER SIMULATION

Rinkel, A., Hollenstein, L., Version 1.5.1, 2018. *Einsatz formaler Methoden in der Simulation.*

Beschreibung Eine Zusammenarbeit der Hochschulen HSR und ZHAW, die als begleitende Einführung dient für Studenten, die Module betreffend Modellsimulation besuchen.

Relevanz Der Text dient als Nachschlagewerk, wenn es um die korrekte Schreibweise von BPMN-Modellen geht. Des Weiteren werden einfache Implementierungen von Modellen in Simio erläutert.

3.2 BUSINESS PROCESS MODEL AND NOTATION (BPMN)

Für die einheitliche Schreibweise von Prozessen in dieser Arbeit wird die BPMN Spezifikation verwendet. Es handelt sich dabei um eine Token- und Eventbasierte Beschreibungssprache für Geschäftsprozesse. Grundlage bildet der «BPMN Quick Guide»[1] sowie die Vorlesungsunterlagen zum Modul «System Modeling and Simulation».

3.2.1 FLOW OBJECTS

Flow Objects sind sogenannte Knoten in Prozessdiagrammen. Diese sind unterteilbar in Events, Activities und Gateways.

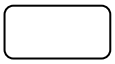
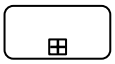
3.2.1.1 Event (Ereignis)

Ein Event ist etwas, das zu einem bestimmten Zeitpunkt passiert und den Ablauf eines Prozesses beeinflusst. Ausgelöst werden diese durch bestimmte oder unbestimmte Trigger, welche mit entsprechenden Piktogrammen genauer spezifizierbar sind. Üblich sind zum Beispiel Signale, Nachrichten oder zeitliche Bedingungen. Werden keine Piktogramme verwendet und das Eventsymbol leergelassen, bedeutet dies, dass der Auslöser nicht zwingend bekannt sein muss, um einen Prozess zu starten.

	**Start Event**	Gibt den Prozessstart an. Es werden Token erzeugt, die durch den Geschäftsprozess fliessen.
	**Intermediate Event**	Ereignisse, die nach Start und vor Ende eines Prozesses stattfinden.
	**End Event**	Geben an, wann und wie ein Prozess terminiert wird. Hierbei werden die beim Start Event erzeugten Token zerstört.

3.2.1.2 Activity (Aktivität)

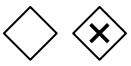
Activities beschreiben kleinere und grössere Aufgaben, die in einem Prozess durchzuführen sind.

	**Task**	Ein Task ist das kleinste Element in einem Prozess und stellt einfache Aufgaben dar.
	**Sub Process**	Eine komplexere Aktivität, die wiederum andere Aktivitäten enthält.

[1] BPMN Quick Guide: https://cloud.trisotech.com/bpmnquickguide/index.html?connecting_object_basics.htm

3.2.1.3 Gateway (Zugang)

Gateways stellen Zugangs- oder Entscheidungspunkte dar, die erlauben, den Tokenfluss zu steuern oder zu kontrollieren. Anhand von «AND-», «OR-» oder «XOR-» Gateways wird entschieden, wie mit dem Token an bestimmten Stellen weiterverfahren wird. Bei ereignisbasierten und komplexen Gateways können ebenfalls Piktogramme verwendet werden, um diese genauer zu spezifizieren. Ein Gateway vereinigt oder teilt den Fluss von Token. Bei einer Zusammenführung muss dieselbe Gateway Art verwendet werden, wie bei der Teilung der Token.



Exclusive Gateway — XOR Gateways erlauben es, den Tokenfluss zu steuern. Es wird entschieden auf welchem Pfad ein Token weitergeleitet wird.

Inclusive Gateway — OR Gateways ermöglichen, dass Token sich teilen oder vervielfachen und parallel weiterlaufen. Es werden mehrere Möglichkeiten angestossen.

Parallel Gateway — AND Gateways werden angewendet, wenn der Prozess sich teilt und parallel weiterläuft, hierfür wird das Token geklont.

3.2.2 CONNECTING OBJECTS

Stellen Verbindungen in einem Prozessdiagramm dar.

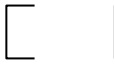
Sequence Flow — Verbindungen zwischen Flow Objects, die angeben, in welcher Reihenfolge die Prozessschritte durchgeführt werden.

Message Flow — Zeigen einen Nachrichtenaustausch oder eine Kommunikation zwischen Elementen an.

3.2.3 ARTIFACT (ARTEFAKT)

Artefakte können genutzt werden, um eine ausführlichere Dokumentation von Prozessen zu erzielen.

Data Object — Ein Objekt, welches vom Prozess bearbeitet wird. Diese können virtuell oder physisch sein. Zum Beispiel ein Datensatz oder ein Aktenordner.



Text Annotation — Ein Kommentar, der einem Element zugeordnet wird.

Group — Kann genutzt werden, um mehrere, verschiedene Elemente eines Prozesses zusammenzufassen.

3.3 SCOR MODELL UND GRUNDLAGEN

In Hinblick auf das Simulieren von Supply Chains muss erst einmal erarbeitet werden, wie diese in realen Fällen dokumentiert und ausgewertet werden. SCOR ist ein erweiterbarer Standard, der weltweit für das Dokumentieren, Planen und Auswerten von Supply Chains verwendet wird.

3.3.1 SUPPLY-CHAIN-OPERATIONS-REFERENCE-MODELL (SCOR)

SCOR wurden vom Supply Chain Council (SCC) definiert, um verschiedenen Firmen und Branchen, den Vergleich und das Analysieren von standardisiert ausgewerteten Supply Chains zu ermöglichen. Die zum jetzigen Zeitpunkt immer noch aktuelle Version 11.0 wurde im Dezember 2012 veröffentlicht.

Im SCOR-Modell gibt es fünf grundlegende Basisprozesse, vier davon – Beschaffen («Source»), Herstellen («Make»), Liefern («Deliver») und Rückliefern («Return») – sind ausführende Hauptprozesse. Der fünfte Basisprozess ist die Planung («Plan»), welche Angebot und Nachfrage zwischen den anderen Prozessen ausbalanciert.

Zusätzlich zu der Kategorisierung von Supply Chains, welche auf verschiedenem Detailgraden durchgeführt werden kann, bietet SCOR sogenannte «best practices» und Kennzahlen zum Vergleichen und Auswerten an.

3.3.2 DIE FÜNF HAUPTPROZESSE

In einer Supply Chain werden die fünf Hauptprozesse bei jedem Glied in der Kette eingesetzt. So haben sowohl Zulieferer, Hersteller und Abnehmer die Prozesse von allen fünf Kategorien als Kategorisierung zur Verfügung, wie im Bild unten dargestellt.

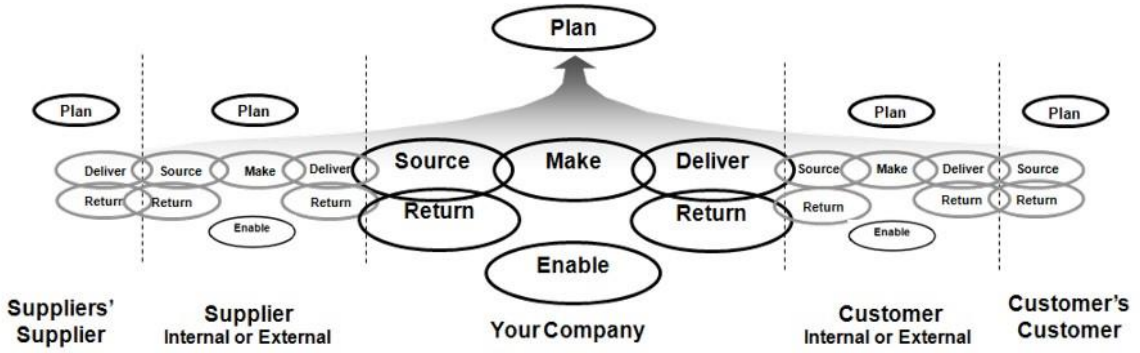


Abbildung 3: Die fünf Hauptprozesse von SCOR (SCOR-P, http://www.apics.org/credentials-education/endorsements/scor-p, 18.03.2020)

3.3.3 PROZESS STRUKTUR

Zusätzlich zu den fünf Hauptprozessen gibt es in SCOR drei Prozessarten – Planung («Planning»), Ausführung («Execution») und Ermöglichung («Enable»).

Planung Diese Prozessart wird unterschieden in weitere Prozesse, welche sich mit dem Ressourcenbedarf und der Nachfrage beschäftigen. Sie referenzieren auf den Hauptprozess «Planung».

Ausführung Ausführungsprozesse werden von der Planung oder der realen Nachfrage ausgelöst. Sie umfassen die vier Hauptprozesse «Beschaffen», «Herstellen», «Liefern» und «Rückliefern».

Ermöglichung Diese Prozessart befasst sich mit dem Pflegen, Vorbereiten und Steuern von Informationen und Beziehungen, welche für das Ausführen der anderen beiden Prozesstypen benötigt werden.

3.3.4 SCOR ALS MODELLBAUKASTEN

Um SCOR anzuwenden, werden die Prozesse in drei Ebenen unterteilt:

Ebene 1 Diese Ebene besteht aus dem Logistiknetzwerk der Firma, die genauer betrachtet wird und den jeweiligen Zulieferern und Abnehmern.

Ebene 2 Ebene 2 beinhaltet die vier Hauptprozesse «Beschaffen», «Herstellen», «Liefern», «Rückliefern» und die Ermöglicher.

Ebene 3 Die in Ebene 2 erwähnten Hauptprozesse werden auf dieser Ebene genauer ausformuliert.

Werden diese Ebenen in Bezug auf eine Simulation betrachtet, so beschreibt Ebene 1, welche Stationen verwendet werden; Ebene 2, welche Angaben bei den Stationen und den Wegen dazwischen gemacht werden müssen und Ebene 3 geht darauf ein, was diese Angaben sind.

3.4 LAGER MODELL UND GRUNDLAGEN

Funktionen Lager können in Hinblick auf die zu lagernden Produkte verschiedene Funktionen übernehmen:

- *Ausgleichende* Funktion zwischen Beschaffungsmenge und Produktionsmenge. In diesem Fall wird jeweils die Differenz der beiden Mengen eingelagert.
- *Sichernde* Funktion als Schutz gegen Lieferunfähigkeit oder Terminverschiebungen, werden mehr Produkte als kurzfristig benötigt eingelagert.
- *Spekulierende* Funktion bei extremen Preisschwankungen benötigter Produkte die dann beschafft und eingelagert werden, wenn die Preise gerade niedrig sind.
- *Veredelnde* Funktion liegt dann vor, wenn durch die Lagerung eine veredelnde Veränderung des Produkts erzielt wird.

Strategien Lagerstrategien können grob in zwei Arten unterschieden werden:

- *Vergabestrategien*: Festplatzvergabe, freie Platzvergabe, Zonung, Querverteilung, Clustering, kürzester Fahrweg oder Vorpufferung.
- *Auslagerungsstrategien*: First-In-First-Out (FiFo), Last-In-First-Out (LiFo) oder nach Anbruchmengenbevorzugung.

Typen Bei den Lagertypen kann es Sinn machen, noch in zwei weitere Arten zu unterscheiden:

- *Funktionale Prägung*: Beschaffungslager, Produktionslager, Distributionslager und Ersatzteillager.
- *Architektur*: Gliederungen nach verschiedenen Faktoren: Bauhöhe, Lagergut, Ladehilfsmittel oder Bauarten des Lagers.

3.4.1 UMSETZUNG VON LAGERMODELLEN FÜR MODELL BAUKASTEN SUPPLY CHAIN

Lager können im Gebiet Supply Chains entweder als Black Box oder White Box System betrachtet werden. Naheliegend ist ein erster Ansatz als Black Box Betrachtung, in der lediglich der Kosten- und Zeitfaktor von Lagern eine Rolle für die Simulation spielt. In einem weiterführenden Ansatz kann in Betracht gezogen werden, die Black Box Logik auf die verschiedenen Unterteilungen erweiterbar zu implementieren, um spätere White Box Betrachtungen zu ermöglichen.

3.4.2 ZUSAMMENARBEIT MIT BACHELORARBEIT ELIA DONNO

In einer gleichzeitig durchgeführten Bachelorarbeit wird bereits ein Baukastensystem für Lagertypen in Simio erstellt. Somit wird in der vorliegenden Arbeit die Lagermodellierung einfach gehalten, und in einer späteren Zusammenführung der Modelle wird das erarbeitete Lagermodell eingebunden.

4 SIMIO GRUNDLAGEN

Die Vorgabe für den praktischen Teil der vorliegenden Arbeit ist Simio. Simio ist ein Simulationsprogramm, welches für die Simulation von Abläufen, wie zum Beispiel einer Supply Chain, verwendet werden kann. Es bietet eine Standard Library, die sich fast beliebig erweitern lässt. Das Erweitern dieser Standard Library zum besseren Abbilden von Supply Chains und zum Beantworten der gestellten Fragen ist Teil der Arbeit und baut auf den folgenden Modulen dieser Standard Library auf.

4.1 STANDARD LIBRARY

Die Standard Library kann in Stationen, Begleitobjekte und Wege unterteilt werden.

4.1.1 STATIONEN

Stationen bestehen optisch aus einem Aktionsblock, einem oder mehreren Verbindungsknoten und verschiedenen Queues (Warteschlangen).

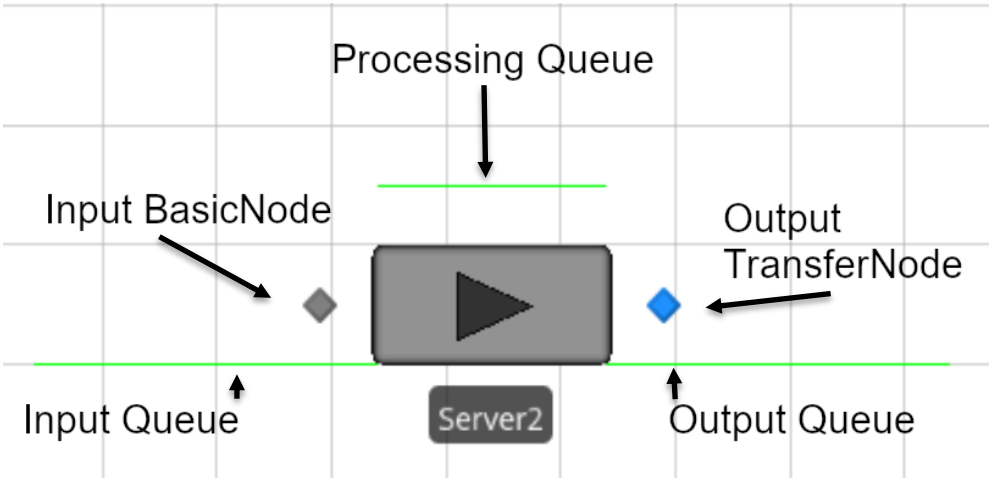
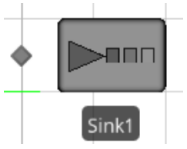


Abbildung 4: Aufbau eines Servermoduls in Simio

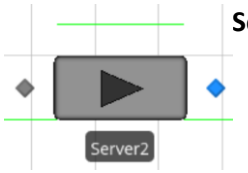
Source

Eine Source (Quelle) ist ein Startpunkt für eine Entity. Die Source kann dabei eine oder mehrere Arten von Entitys generieren.



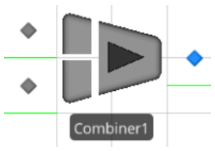
Sink

Ein Sink (Senke) ist ein Endpunkt. Alle Entitäten, die hier ankommen, verlassen durch das Sink die Simulation wieder.

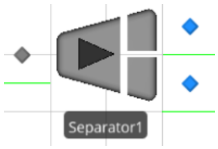


Server

Ein Server ist eine Station, an der etwas passiert. Er hat zahlreiche Einstellungsmöglichkeiten, durch die definiert werden kann, wie lange sich ein Entity hier aufhalten wird und wie viele pro Zeiteinheit den Server durchlaufen können. Es ist festgelegt, dass alle Entitäten, welche den Server betreten ihn garantiert wieder verlassen werden.



Combiner Ein Combiner nimmt Entities von zwei Wegen entgegen und setzt diese zu einem Objekt zusammen. Ein Beispiel dafür ist das Stapeln von Fässern auf einem Pallet. So kommen Fässer auf dem einen Weg und Pallets auf dem anderen rein und gehen dann als ein Objekt raus.



Separator Ein Separator ist das Gegenstück zum Combiner. Er nimmt zusammengesetzte Objekte entgegen und teilt diese wieder in separate Entities.

4.1.2 BEGLEITOBJEKTE

Begleitobjekte sind Fahrzeuge und Arbeiter, sie stellen limitierte Ressourcen dar. Zum Beispiel ein Lieferwagen, der nur einmal am Tag Waren abholt, oder die Gabelstapler, von welchen das Lager nur zwei hat.

Vehicle Vehicles werden zum Simulieren von Fahrzeugen verwendet, sofern diese limitiert sind oder eine Ein-/Ausladezeit haben.

Worker Ein Worker wird zum Simulieren von Arbeitern verwendet. Er kann an Stationen gerufen werden, dort Arbeitsschritte auszuführen oder Entities im System verschieben.

4.1.3 WEGE

Wege bestehen aus Nodes (Verbindungsknoten) und den Verbindungen dazwischen.

BasicNode Ein einfacher Verbindungspunkt von Wegen.

TransferNode Ein Verbindungspunkt, an dem Entities darauf warten, abgeholt zu werden, während sie bereits wissen wohin sie möchten. Sie werden also abgeholt und zu ihrem Ziel transferiert.

Connector Verbindung zwischen zwei Nodes, welche unmittelbar, das heisst ohne Transportzeit, zurückgelegt wird.

Path Verbindung zwischen zwei Nodes, welche von Entities basierend auf Geschwindigkeit zurückgelegt wird.

TimePath Verbindung zwischen zwei Nodes, welche von Entities basierend auf Zeit zurückgelegt wird.

Conveyor Ein Förderband oder Fördervorrichtung, die akkumulierend oder nicht akkumulierend sein kann.

4.2 REITER

Die Reiter über der Modellvorschau bieten die Möglichkeit verschiedene Einstellungen und Ansichten zu verwalten.

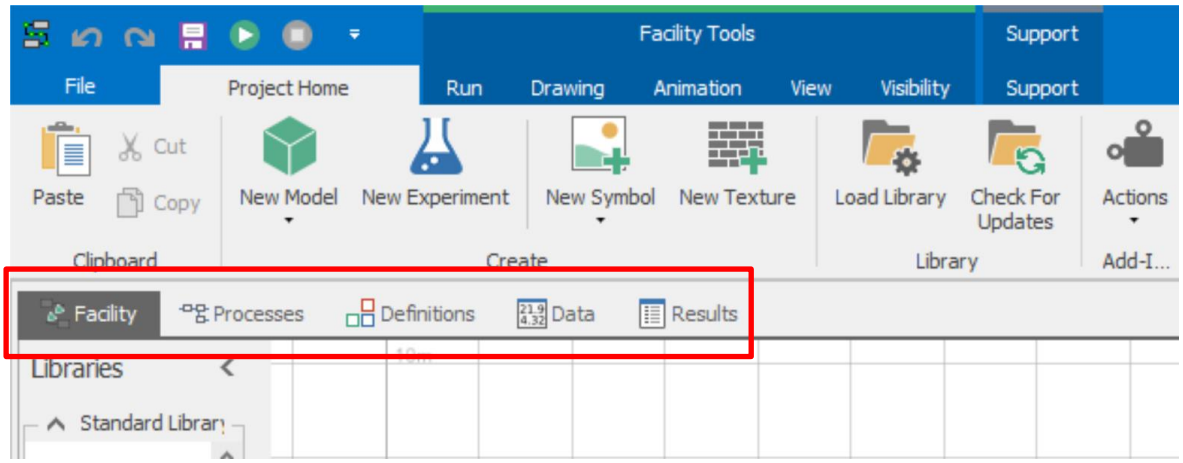


Abbildung 5: Ansicht des Simio Tools

Facility Der Reiter «Facility» ist die Hauptansicht. Hier kann aus den verschiedenen Bauteilen ein Modell erstellt werden. Alle Stationen und ihre Verbindungen werden in dieser Ansicht erstellt.

Durchzuführende Tests werden ebenfalls in dieser Ansicht festgelegt.

Processes In «Processes» werden Abläufe festgelegt, alle Default Stationen haben bereits eine Vielzahl von Abläufen, die sie standardmässig mitbringen. Diese Abläufe zu bearbeiten oder neue hinzuzufügen kann in diesem Reiter vorgenommen werden. Eine Auflistung von häufigen Schritten («Common Steps») erlaubt es, einfach eine Vielzahl von Zusammenhängen festzulegen.

Definitions In diesem Reiter können zusätzliche Parameter festgelegt werden. Das umschliesst Listen, Eigenschaften, Schnittstellen, Token und andere. Tokens sind hier besonders wichtig, da sie das Weitergeben von Informationen zwischen Stationen ermöglichen.

Data Hier können externe Datenquellen, zum Beispiel Tabellen, eingebunden werden. Es ist auch möglich, neue Datenquellen direkt zu erfassen, zum Beispiel das eines «Work Schedules» (Arbeitsplan). Diese können, wie auch die externen Tabellen, direkt in der Simulation verwendet werden.

Result In «Results» werden Information für Risiko-basiertes Planen bereitgestellt. Dies umfasst Target summary, Risk Plots, Detailed results, reports and FAQs.

5 UMSETZUNG EINER SUPPLY CHAIN IN SIMIO

In diesem Kapitel werden Überlegungen diskutiert und Möglichkeiten gezeigt, wie die Umsetzung einer einfachen Supply Chain in Simio aussehen kann.

5.1 ÜBERLEGUNGEN

Als Vorbereitung für nachfolgende Arbeiten, wird zuerst eine einfache Basis Supply Chain in Simio erarbeitet mit den wichtigsten Grundbausteinen: Kunde, Lager, Verkauf und Hersteller. Diese Bausteine sollen in zukünftigen Arbeiten erweitert werden und das Modell so zu komplexeren Supply Chains ausgebaut werden. Die Arbeit konzentriert sich auf die Konfiguration des Kunden und des Herstellerbausteins, da bereits in einer parallellaufenden Bachelorarbeit der Lagerbaustein erarbeitet wird. Für die vorliegende Arbeit wird die Lagerlogik somit vorerst vernachlässigt.

Eine mögliche Umsetzung in Simio könnte so aussehen wie in Abbildung 6 gezeigt. Der Hersteller (*Producer*) stellt Produkte her und gibt diese an das Lager (*Warehouse*) weiter, welches wiederum den Verkauf (*Sale*) beliefert. Die Kunden (*Customer*) können schliesslich beim Verkauf, welcher einer Filiale eines Detailhändlers entspricht, Produkte kaufen.

Die grünen Pfeile stellen den Informationsfluss, also den Bestellungsablauf, und die roten Pfeile den Warenfluss dar.

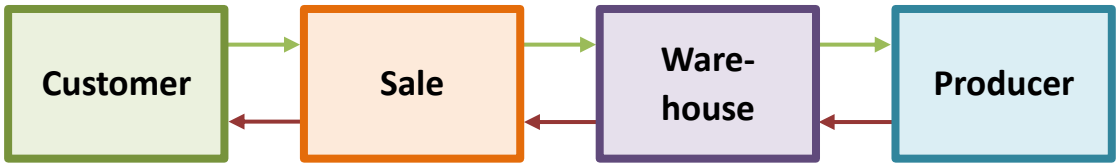


Abbildung 6: Einfache Supply Chain

Für dieses Beispiel wird davon ausgegangen, dass die Filiale intern ebenfalls über ein Lager verfügt, welches die eigentliche Verkaufsfläche mit Waren versorgt. Der dazugehörige Ablauf wird in Abbildung 7 anhand eines Sequenzdiagrammes verdeutlicht. Da der Verkauf an den Kunden wiederum als eigener Prozess betrachtet werden kann, wurde dieser in der Darstellung vernachlässigt. Anders als zum Beispiel aus der Architekturansicht, zeigen Sequenzdiagramme den zeitlich logischen Ablauf über die zu betrachtenden Entitäten.

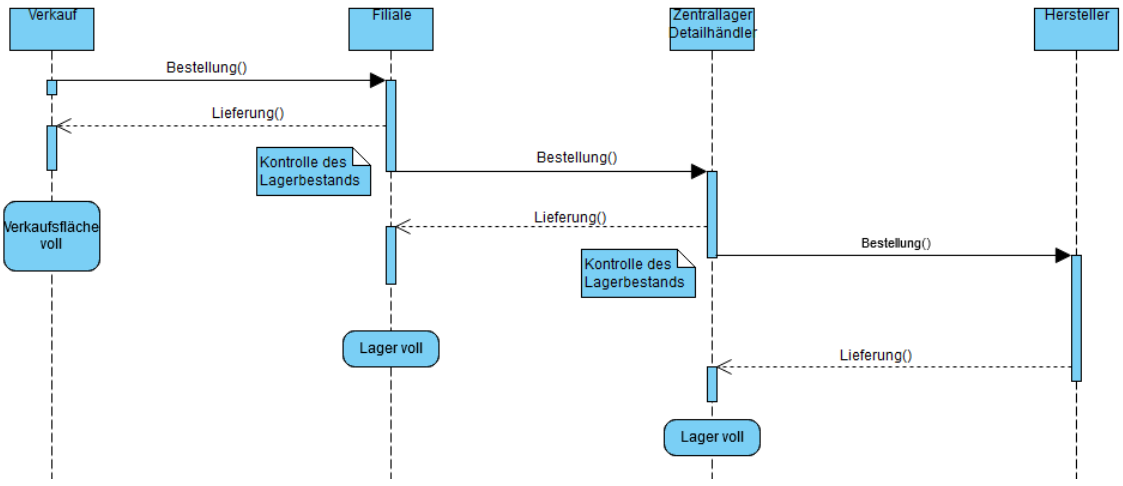


Abbildung 7: Sequenzdiagramm Bestellprozess

Sobald der Lagerstand des *Verkaufs* abnimmt, bestellt der Verkauf beim internen Lager der *Filiale* nach. Die *Filiale* liefert und überprüft den eigenen Lagerbestand. Unterschreitet dieser einen festgelegten Schwellwert so schickt die *Filiale* eine Bestellung an das *Zentrallager des Detailhändlers*. Das *Zentrallager* liefert der *Filiale* die bestellen Waren und überprüft, ob es den eigenen Schwellwert des Lagerbestands unterschritten hat. Ist dies der Fall, schickt das *Zentrallager* eine Bestellung and den *Hersteller*, welcher die Waren liefert.

Diese Prozessbeschreibung ist ein Beispiel dafür, falls jede Station nach einer Lieferung feststellt, dass der Schwellwert erreicht wurde. Dies ist jedoch nur ein möglicher Fall von vielen. Es ist wahrscheinlicher, dass die verschiedenen Lager ihren jeweiligen Schwellwert bereits so angesetzt haben, dass mehrere Lieferungen durchführbar sind, ohne dass der Lagerbestand zur Neige geht.

Weitere Möglichkeiten wären, dass Lieferwege unterbrochen sein können, Hersteller ausfallen, Ausweichprodukte geliefert werde, und so weiter.

5.1.1 BPMN Informationsfluss

In Abbildung 8 wird der Weg einer Bestellung durch das Supply Chain Modell betrachtet, dies entspricht dem Informationsfluss. Es wird davon ausgegangen, dass der Kunde eine Bestellung an den Verkauf aufgibt. Der Verkauf prüft seinen Lagerbestand darauf, ob die Bestellmenge vorhanden ist, und wenn ja, ob der Schwellwert überschritten wurde. Wenn die bestellte Menge vorhanden ist und der Schwellwert nicht unterschritten wurde ist der Prozess und der Informationsfluss der Bestellung abgeschlossen. Wenn aber der Verkauf selbst auch eine Bestellung an den Hersteller aufgeben muss, wird eine weitere Bestellung vom Verkauf an den Hersteller gesendet. Der Hersteller überprüft, ob er seine Produktionsmenge anpassen muss, um die Bestellung zu erfüllen und passt diese gegebenenfalls an. In beiden Fällen ist der Informationsfluss der Bestellung dann abgeschlossen.

Auch hier ist dies nur eine mögliche Darstellung des Prozesses, die einzelnen Schritte können noch weiter verfeinert werden und detaillierter betrachtet werden. Dies wird für die Überlegungen jedoch vorerst vernachlässigt und in einer späteren Arbeit wieder aufgegriffen.

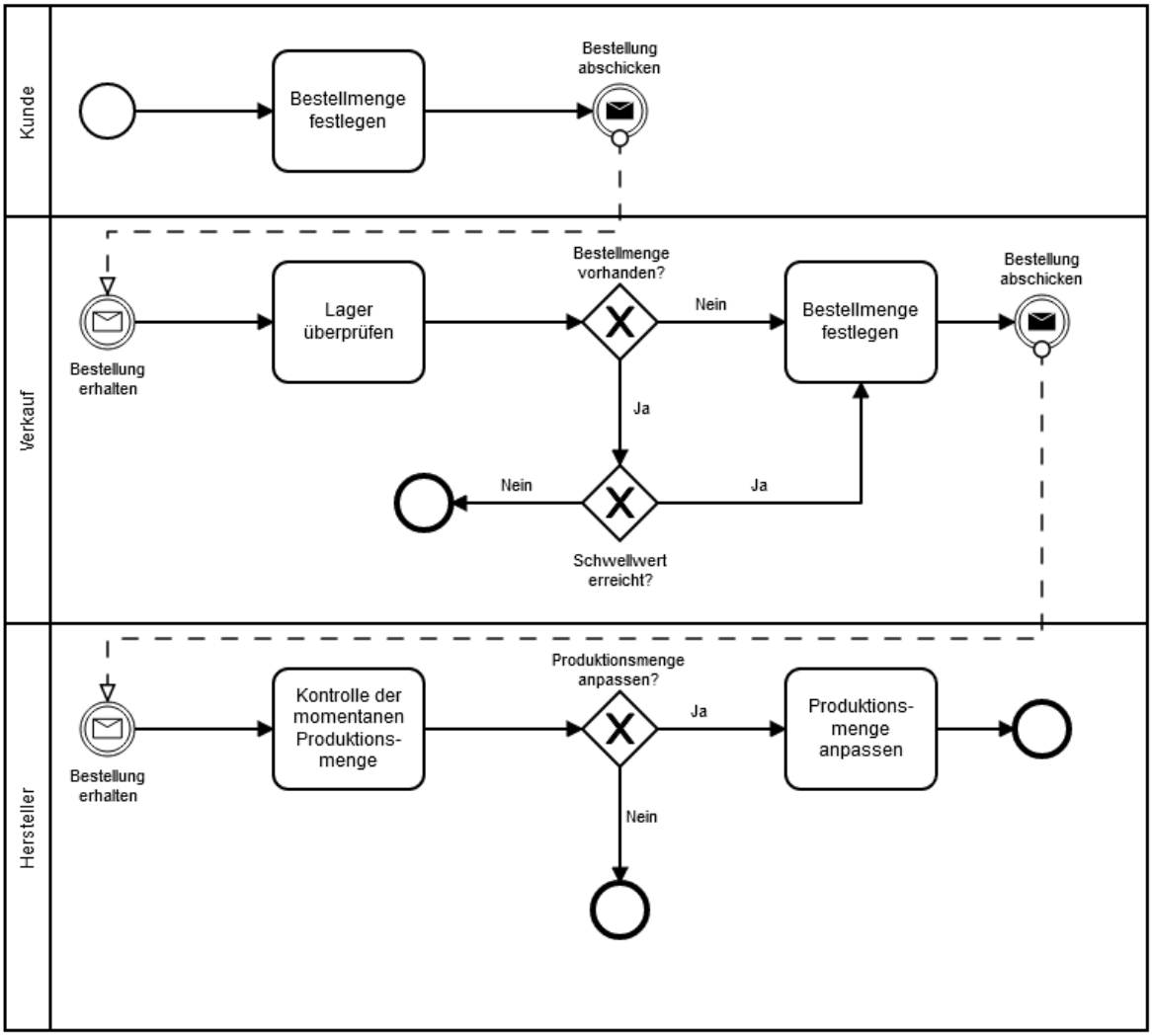


Abbildung 8: BPMN des Informationsflusses einer Supply Chain

5.1.2 BPMN Warenfluss

In der Abbildung 9 ist die Lieferung der Produkte zu sehen, dies stellt den Warenfluss dar und wird in diesem Fall aus Sicht des Herstellers betrachtet.

Der Hersteller produziert, bis die notwendige Menge erreicht wird. Die produzierten Waren werden dann im internen Lager zwischengelagert und von dort aus wird eine Lieferung an das Zentrallagers des Zwischenhändlers ausgelöst. Sind die Waren im Zentrallager angekommen, werden diese zunächst eingelagert. Sofern eine offene Bestellung einer Verkaufsfiliale besteht, überprüft das Zentrallager, ob es genug Produkte für eine Lieferung hat und sendet diese dann weiter an den Verkauf. Beim Verkauf werden bestellte Waren eingelagert und später verkauft.

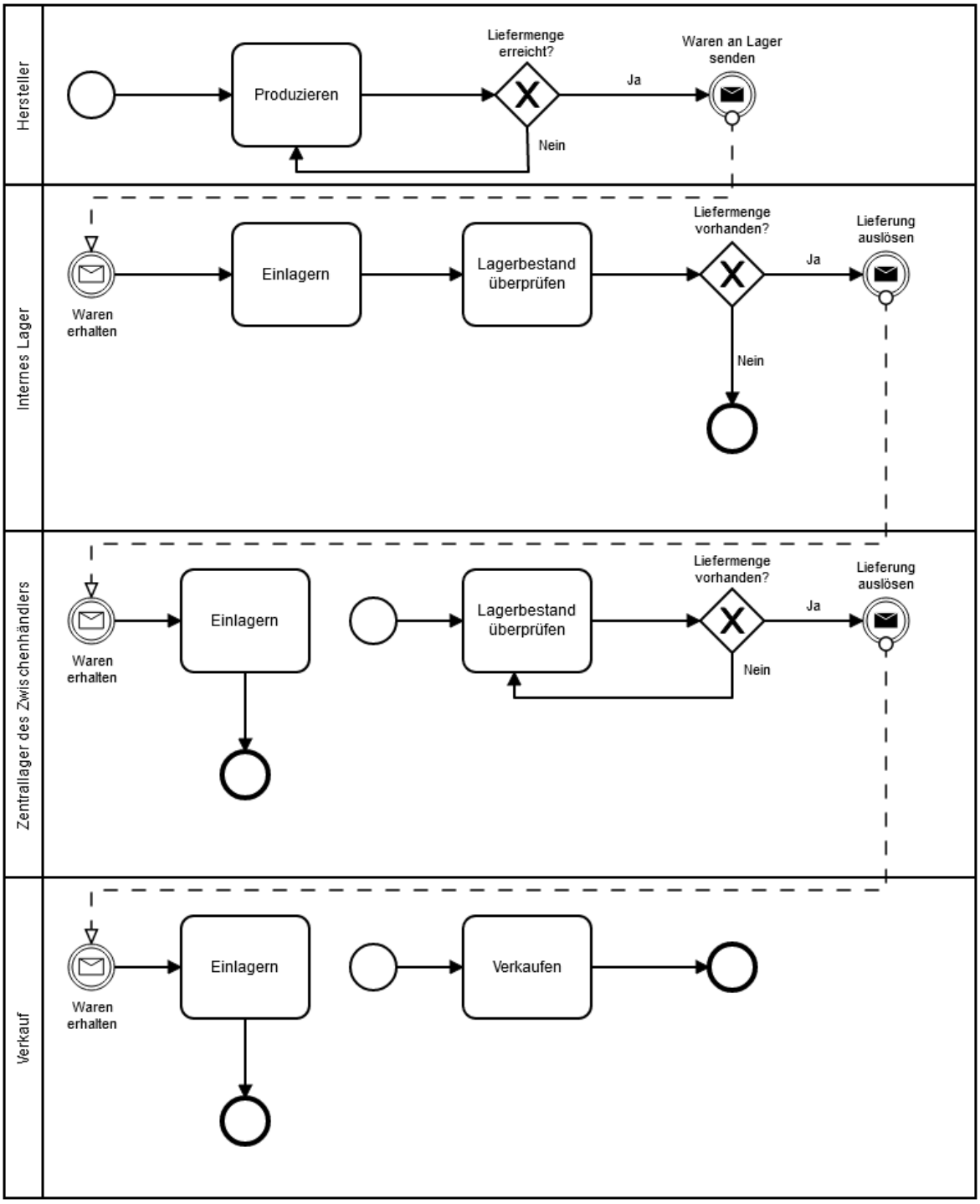


Abbildung 9: BPMN des Warenflusses einer Supply Chain

6 PROTOTYP0: EINFACHE SUPPLY CHAIN IN SIMIO

Der «Prototyp0», der eine möglichst simple Supply Chain simuliert, konzentriert sich auf die Hauptkomponenten Kunde, Lager und Verkauf und die Entities Bestellung und Produkt. Feineinstellungen für die Wege, also den Informations- sowie Warenfluss, werden in diesem Prototyp noch vernachlässigt. Ziel der nachfolgenden Kapitel ist es, zu zeigen, was wo passiert und wie es in Simio definiert ist.

Im nachfolgenden Kapitel werden für die einzelnen Komponenten und Kernelemente die englischen Begriffe, sowie Abkürzungen verwendet. Nachfolgend eine Übersicht der Begriffe.

Order	*Bestellung*	**SimpleWareHouse**	*Einfaches Lagerhaus*
Product	*Produkt*	**Store**	*Lager*
Customer	*Kunde*	**Entity**	*Entität*
Producer	*Hersteller*	**State**	*Zustand*
Property	*Eigenschaft*	**Container**	*Behälter*
Library	*Bibliothek*	**Sale**	*Verkauf*

Tabelle 1: Englisch-Deutsch Begriffe für das Simio Modell

6.1 GEPLANTER AUFBAU ANHAND SEQUENZDIAGRAMM UND BPMN

Ziel ist es, das besprochene Modell aus Kapitel 5.2 für die Simulation weiter zu vereinfachen. Customer können in dieser vereinfachten Ansicht ihre Einkäufe direkt über das Warehouse tätigen und müssen nicht zuerst über einen separaten Sale gehen, da die Logik für Sale und Warehouse für eine erste Modellbetrachtung sehr ähnlich ist. Dieses vereinfachte Modell ist ersichtlich aus Abbildung 10.

Abbildung 10: Einfache Supply Chain V2

Es gibt einen Customer, ein Warehouse, welches gleichzeitig auch als Sale dient und einen Producer. Die grünen Pfeile stellen den Informationsfluss, also den Weg der OrderEntities dar und die roten Pfeile den Warenfluss. Der Customer gibt beim Warehouse eine Bestellung auf und das Warehouse kann entweder sofort liefern oder muss beim Producer nachbestellen, also ebenfalls eine Bestellung an den Producer aufgeben. Ein möglicher Ablauf wird in Abbildung 11 anhand eines Sequenzdiagrammes aufgezeigt.

Abgebildet wird der Ablauf, wie ein *Kunde* beim *Lager* bestellt, welches ihm die gewünschten Waren liefern kann. Anschliessend überprüft das *Lager*, ob der Lagerbestand noch über einem bestimmten Schwellwert liegt. Tut er das nicht, so bestellt das *Lager* beim *Hersteller* neue Produkte, um das *Lager* wieder aufzufüllen. Während der laufenden Bestellung an den *Hersteller* nimmt das Lager weitere Bestellungen an und führt diese, wenn möglich, aus.

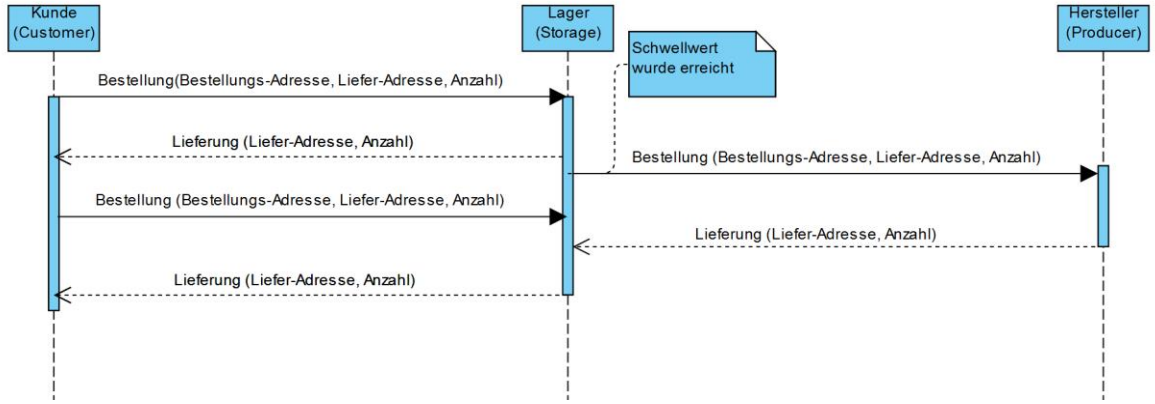


Abbildung 11: Sequenzdiagramm Bestellprozess V2

Am geplanten BPMN Modell aus Abbildung 9 wurde für die Umsetzung in Simio ebenfalls eine Änderung vorgenommen. Aus Abbildung 12 ist ersichtlich, dass das *interne Lager des Herstellers* herausgenommen wurde. Für eine erste einfache Supply Chain Betrachtung in Simio, hat der *Hersteller* selbst kein eigenes Lager, sondern produziert lediglich die bestellten Produkte.

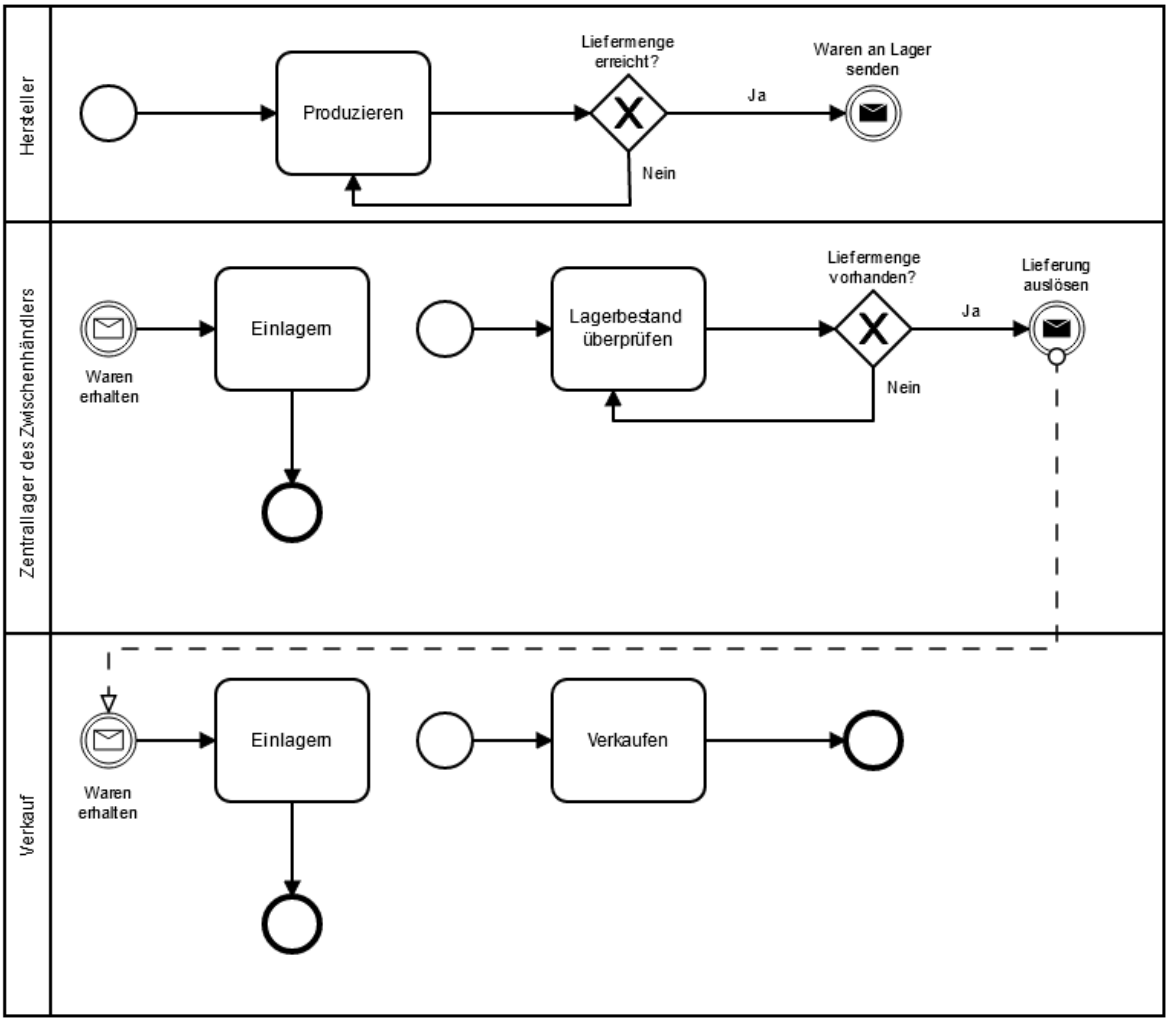


Abbildung 12: Vereinfachter Produktionsprozess BPMN

6.2 AUFBAU DES MODELLS IN SIMIO

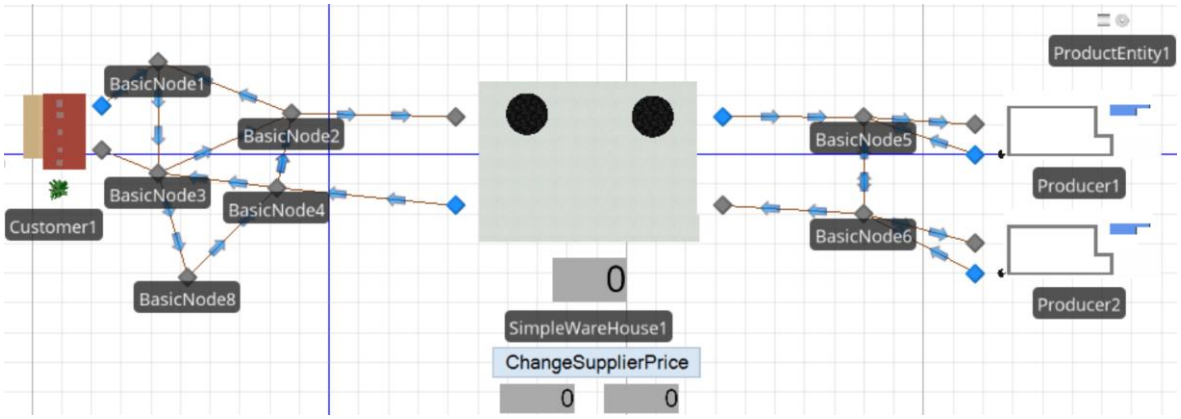


Abbildung 13: Prototyp0, Komplettansicht

Zur Veranschaulichung in Simio soll die Datei «Prototyp0.spfx» geöffnet werden. Diese Datei benötigt zudem die Library «Product_lib.spfx».

Der finale «Prototyp0» besteht aus den 3 Hauptkomponenten Customer, SimpleWareHouse und Producer, deren individueller Aufbau in den nachfolgenden Kapiteln genauer definiert ist. Es werden 2 Entities benötigt: OrderEntities und ProductEntities. Die OrderEntities werden vom Customer und vom SimpleWarehouse erzeugt und Richtung Producer geschickt. Die ProductEntities werden nur vom Producer erzeugt und werden Richtung SimpleWareHouse und Customer geschickt, wenn sie nachgefragt werden.

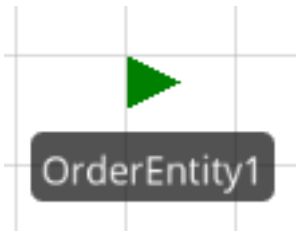
Die Verbindungswege zwischen den Stationen mit den verschiedenen BasicNodes zeigen, dass die Order- und ProductEntities den Weg zu ihren Liefer- und Bestelladressen finden.

Für den «Prototyp0» gibt es nur eine ProductEntity und eine OrderEntity und es ist nur möglich ein einziges SimpleWareHouse im Modell zu haben. Die blauen und grauen Nodes stellen Schnittstellen zu den anderen Modulen im Modell dar. Die grauen sind jeweils Output und die blauen Input.

6.3 ENTITIES

Es werden zwei Arten von Entitys benötigt: OrderEntities, die den Informationsfluss darstellen und ProductEntities, die den Warenfluss simulieren.

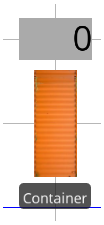
6.3.1 ORDERENTITY



Orders sind Entities, welche bei allen Stationen ausser dem Producer erstellt werden können. Sie entsprechen der Datenübermittlung des Informationsflusses. In der Realität kann eine solche Bestellung in den verschiedensten Formaten bestehen, zum Beispiel in Form eines Bestellformulars,einer E-Mail der einer mündlichen Bestellung per Telefon. Im «Prototyp0» wird ein Standardformat verwendet, welches vergleichbar mit einer Bestellung per Post ist.

Jedes OrderEntity wird mit drei Angaben erstellt, der Bestell-Adresse, der Liefer-Adresse und der Anzahl der bestellten Produkte. Für die Umsetzung wurden, wie in der Tabelle unten aufgeführt, drei States und drei Properties erstellt. Der Name setzt sich zusammen aus «initial» und dem jeweiligen «State-Name». Die Properties dienen dabei als Input-Methode für den Initialwert und die

States als Speicher des aktuellen Wertes. Beim Erstellen und Initialisieren der Bestellung werden die Initialwerte in die entsprechenden States geschrieben. Von dort aus werden sie im weiteren Verlauf verwendet.



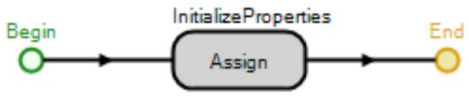
Des Weiteren werden beim Producer «Container» verwendet, welche einer leicht veränderten OrderEntity entsprechen. Verändert wird das Aussehen, um einen Container besser darzustellen und es wird ein Info-Feld hinzugefügt. Dieses Feld zeigt die Anzahl Produkte an, welche sich im Container befinden. Zu beachten ist, dass es sich dabei nicht um die eigentliche Anzahl, sondern um den Bestellwert handelt.

6.3.1.1 Einstellungen

Name	Einstellungsname	Initialwert
Bestelladresse	*[Initial]OrderAdd_Node*	*Input_Management@SimpleWareHouse1*
(State, Property)	Legt fest, wohin die Order geschickt wird. Als Initialwert wird der Bestelleingang vom Lager (Storage / SimpleWarehouse) verwendet.	
Lieferadresse	*[Initial]DeliveryAdd_Node*	*Input@Customer1*
(State, Property)	Legt fest, wohin die Produkte geliefert werden. Dies entspricht im «Prototyp0» immer dem Wareneingang der Station, welche die Bestellung abgeschickt hat. Der Initialwert entspricht somit dem Wareneingang des Customer.	
Anzahl bestellte Produkte	*[Initial]NumberOrdered*	*1*
(State, Property)	Legt fest, wie viele Produkte mit einer Bestellung bestellt werden. Initialwert ist eins.	

Tabelle 2: Einstellungen der OderEntity

6.3.1.2 Prozesse

OnCreated *und* **OnRunInitialized**	
	
InitializeProperties	Weist erzeugten OrderEntities die Lieferadresse, sowie die bestellte Stückzahl zu.

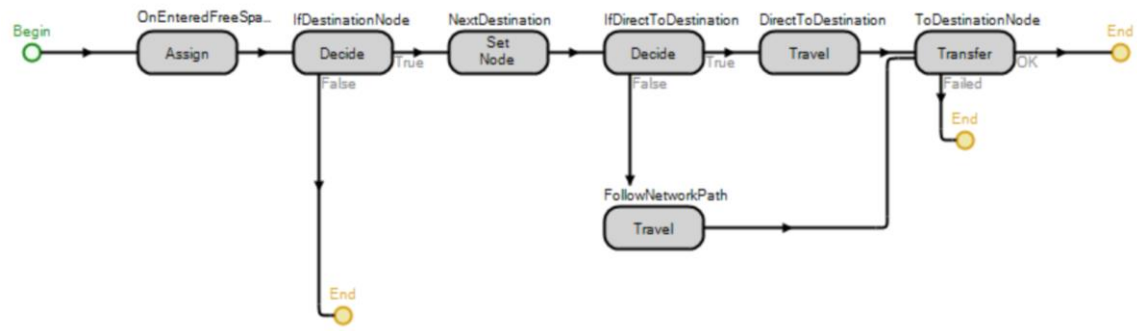
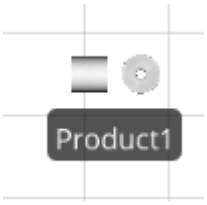
OnEnteredFreeSpace
 Dies ist ein Standard Verhaltensprozess, der von Simio bereits erzeugt wurde und auf einer ModelEntity hinterlegt ist.

Tabelle 3: OrderEntity Prozesse

6.3.2 PRODUCTENTITY



Dieses Entity stellt das bestellte Produkt dar. Es wird vom Producer erstellt, nachdem eine Bestellung dafür eingegangen ist und bewegt sich auf vorgegebenen Pfaden anhand der Angabe, was die nächste Station ist.

Im «Prototyp0» hat das Produkt das Aussehen von zwei WC-Papier rollen.

6.3.2.1 Einstellungen

Name	Einstellungsname	Initialwert
Lieferadresse	*DeliveryAddNode*	*Kein initialer Wert*
(State)	Legt fest, wohin das Produkt versendet wird. Vor dem Verlassen einer Station wird dieser State als nächste Station festgelegt.	

Tabelle 4: Einstellung der ProductEntity

6.4 MODULE

Für die Simulation einer einfachen Supply Chain wurde sich auf die Komponenten Customer, Producer, und SimpleWarehouse konzentriert.

6.4.1 CUSTOMER

Der Customer ist der Start- und Endpunkt der Supply Chain, er steht für den Endkunden der abzubildenden Kette und dient sowohl als Quelle von Bestellungen als auch als Endpunkt für Produktlieferungen. Er beeinflusst massgebend die Nachfrage nach dem Produkt.

6.4.1.1 Abläufe

Beim Customer passieren die folgenden Abläufe:

- Erstellen von OrderEntities
- Setzen der initialen Bestelladresse als nächste Station, wenn Bestellungen das System verlassen
- Entgegennehmen und Zerstören von ProductEntities

6.4.1.2 Aufbau

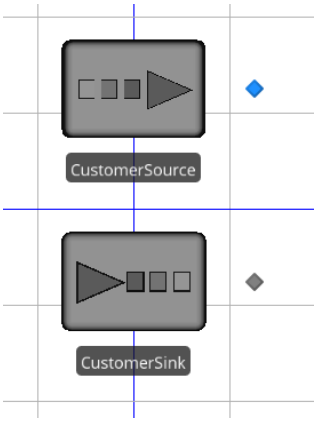
Interner Aufbau	Externe Modellansicht
	
Der Customer besteht aus einer CustomerSource, welche OrderEntities generiert und verschickt, und einer CustomerSink, welche ProductEntities entgegennimmt und aus der Simulation entfernt. Dieser Vorgang simuliert den Konsum der Produkte.	

Tabelle 5: Aufbau des Customer-Moduls

6.4.1.3 Einstellungen

Auf dem Customer im «Prototyp0» wird nur eingestellt, was für eine OrderEntity versendet wird, mit welcher Rate diese verschickt werden und was die maximale Anzahl Bestellungen ist. Für die Einstellungen auf der CustomerSource werden die Initialwerte verwendet.

Name	Einstellungsname	Initialwert
Bestellungstyp	*Order_EntityType*	*OrderEntity1*
(Property)	Legt die OrderEntity fest.	
Bestellrate	*Order_InterarrivalTime*	*Random.Exponential(.25)*
(Property)	Legt fest, wie viele OrderEntities erstellt werden. Der Initialwert beschreibt dabei, dass die Bestellungen Zufällig exponentiell mit einer viertel Minute Zwischenankunftszeit erstellt werden.	
Maximale Bestellrate	*Order_MaximumArrivals*	*Infinity*
(Property)	Definiert eine Obergrenze, wie viele Bestellungen erstellt werden können. Der Initialwert ist unbegrenzt.	

Tabelle 6: Einstellungen des Customer-Moduls

6.4.1.4 Schnittstellen

Der Customer hat zwei Schnittstellen, welche als Verbindungspunkte für die Supply Chain verwendet werden.

Bestellausgang Über diesen Knoten verlassen die OrderEntities den Kunden.

Bestelleingang Bei diesem Knoten werden die ProductEntities entgegengenommen

6.4.2 PROZESSE

Auf dem Customer gibt es einen neu erstellten Prozess, der auf dem Output der Source1 hinterlegt ist.

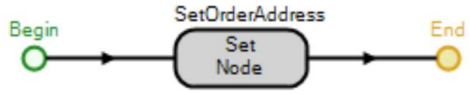
Output_Source1_Entered	
	
SetOrderAddress	Übergibt den erzeugten OrderEntities die initiale Lieferadresse.

Tabelle 7: Customer Prozess

6.4.3 PRODUCER

Der Producer erstellt ProductEntities, verpackt diese in Container und verschickt diese. Die Produkte werden erst erstellt, wenn eine OrderEntity mit einer Bestellung ankommt. Diese löst dann das Erstellen der bestellten Menge von Produkten aus. Der Producer entspricht dem äussersten Lieferanten in der abgebildeten Supply Chain, welcher für die abgebildete Kette keine Bestellungen absetzt, sondern nur Produkte liefert.

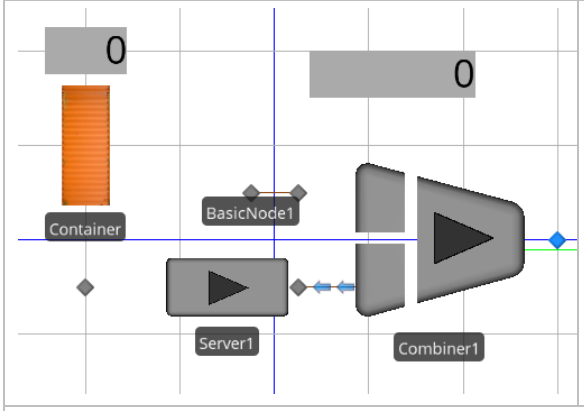
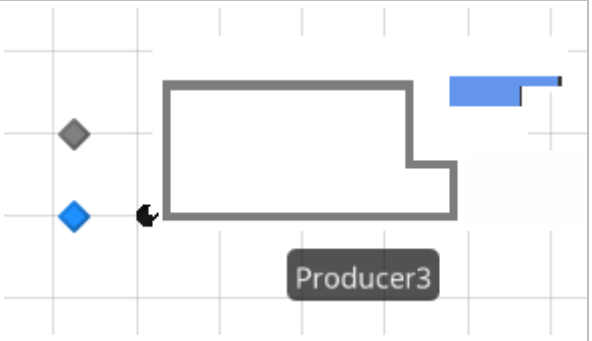
6.4.3.1 Abläufe

Beim Hersteller passieren die folgenden Abläufe.

- Entgegennehmen und Zerstören von OrderEntities
- Erstellen von Container und ProductEntities, wobei die ProductEntities in der von einer Bestellung verlangten Menge hergestellt werden
- Verpacken von ProductEntities in Container
- Verschicken von mit Produkten beladenen Containern an die Lieferadresse, welche bei der ursprünglichen Bestellung angegeben wurde.

Damit die Container am Schluss and die Lieferadresse geschickt werden können, muss die Lieferadresse beim Erhalten der Bestellung zwischengespeichert werden. Das gleiche passiert mit der Anzahl bestellten Produkte, beide Angaben werden zwischengespeichert und dann als erstes nach dem Erstellen des Containers auf diesen übertragen.

6.4.3.2 Aufbau

Interner Aufbau	Externe Modellansicht
	

Der Producer besteht aus einem Container-Lager, einem Server, welcher die ProductEntities erstellt und einem Combiner, welcher die ProductEntities in einen Container verpackt.

Kommt eine Order an, wird ein Container erstellt, mit der Lieferadresse und der Anzahl bestellter Produkte beschrieben und an den Combiner weitergegeben. Gleichzeitig wird beim Server der Auftrag erteilt, die bestellte Anzahl Produkte herzustellen und an an den Combiner zu schicken.

Beim Combiner werden die Container mit der benötigten Anzahl ProductEntities bestückt und and die Lieferadresse geschickt.

Tabelle 8: Aufbau des Producer-Moduls

6.4.3.3 Einstellungen

Um den Prototypen 0 einfach zu halten, wird auf dem Producer nur eingestellt, welche ProductEntities und mit welcher Produktionszeit diese hergestellt werden. Bei der Herstellung handelt es sich um einen «Pull», der durch eine OrderEntity ausgelöst wird.

Name	Einstellungsname	Initialwert
Produkttyp	*Product_EntityType*	*Product1*
(Property)	Legt fest, welches ProductEntity der Producer herstellt. In «Prototyp0» gibt es nur eine Art: Product.	
Produktionszeit	*Server1_ProductionTime*	*Random.Triangular(.1,.2,.3)*
(Property)	Legt fest, wie schnell ProductEntities hergestellt werden. Für den «Prototyp0» ist die Produktionszeit für ein Produkt zwischen 0.1 Minuten und 0.3 Minuten. Die Verteilung der Produktionszeit ist dabei Triangular mit einem Modus von 0.2 Minuten.	
Zwischenspeicher Lieferadresse	*RememberDeliverAddress*	*Kein initialer Wert*
(State)	In diesem State wird die Lieferadresse zwischengespeichert, nachdem eine Order erhalten wurde. Nach dem Erstellen des Containers wird die Lieferadresse aus diesem State heraus auf den Container geschrieben.	

Zwischenspeicher Bestellungsgrösse	*RememberOrderSize*	*Kein initialer Wert*
(State)	Dies dient als Zwischenspeicher für die Bestellgrösse. Nach dem Eingang einer Order wird die Anzahl der bestellten Produkte hier zwischengespeichert. Nach dem Erstellen des Containers wird dieser Wert auf den Container geschrieben.	

Tabelle 9: Einstellungen des Producer-Moduls

6.4.3.4 Schnittstellen

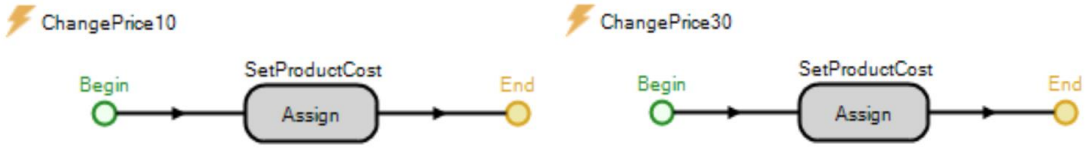
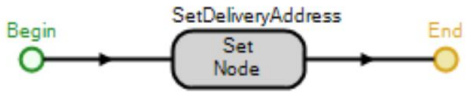
Der Hersteller hat zwei Schnittstellen, welche für das Annehmen von Bestellungen und das Ausschicken von beladenen Containern verantwortlich sind. Dabei ist jede Schnittstelle nur für eine der beiden Funktionen nutzbar.

Bestelleingang OrderEntities werden hier entgegengenommen.

Warenausgang Container verlassen den Producer über diesen Knoten.

6.4.3.5 Prozesse

Für den Producer wurden die nachfolgenden Prozesse definiert:

OnRunInitialized	
	
InitializeProductCost	Weist zu Beginn der Simulation den Initialwert für die Produktkosten zu.
Process1 *und* **Process2**	
	
SetProductCost	Ein Event, um während der Simulation den Verkaufspreis zu verändern entweder zu 10 oder zu 30.
Output_Combiner1_Entered	
	
SetDeliveryAddress	Schreibt die Lieferadresse auf die Container, welche im Combiner1 mit den bestellten ProductEntities vereint werden.

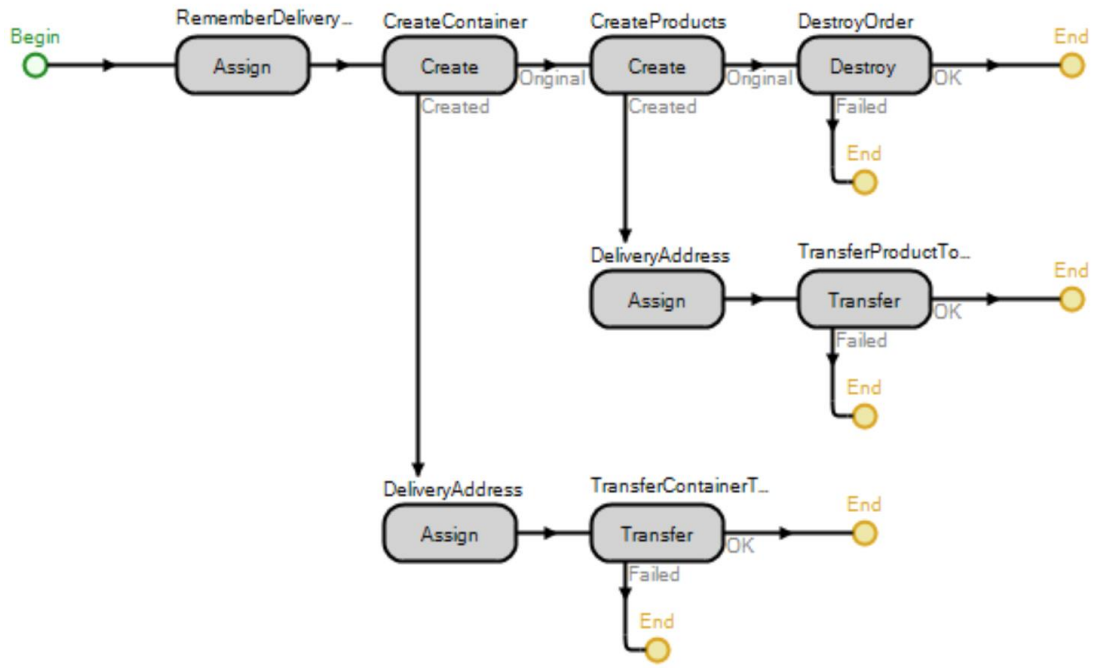
ProductionProcess	
	
RememberDelivery-Address	Dient dazu, die Lieferadresse und die Bestellmenge zwischenzuspeichern in den States «RememberDeliveryAdress» und «RememberOrderSize»
CreateContainer	Erzeugt für jede Bestellung einen Container.
CreateProducts	Erstellt die bestellte Menge an ProductEntities
DestroyOrder	Zerstört die OrderEntity, also die eigentliche Bestellung.
DeliveryAddress	Schreibt die zuvor zwischengespeicherte Lieferadresse auf den Container und das Produkt.
TransferProductTo-Combiner	Verschickt die erzeugten Produkte an den Combiner.
TransferContainerTo-Combiner	Verschickt die erzeugten Container an den Combiner.

Tabelle 10: Producer Prozesse

6.4.4 SIMPLEWAREHOUSE

Das Lager wurde im «Prototyp0» als vereinfachtes Lager (SimpleWareHouse) umgesetzt. Es deckt dabei alle Grundfunktionen eines Lagers ab, ist aber noch nicht komplett eigenständig. Es kann noch nicht dazu verwendet werden, eine beliebig lange Supply Chain aufzubauen. In seiner momentanen Form dient es als einzelner Knotenpunkt zwischen einem Kunden und einem Hersteller. Es wurde allerdings bereits darauf ausgelegt, dass es in einem nächsten Schritt, dem Prototyp 1, um weitere Funktionen ergänzt werden kann. Sodass das SimpleWareHouse als ein oder mehreres aufeinanderfolgenden Modulen von beliebig langen Supply Chains verwendet werden kann.

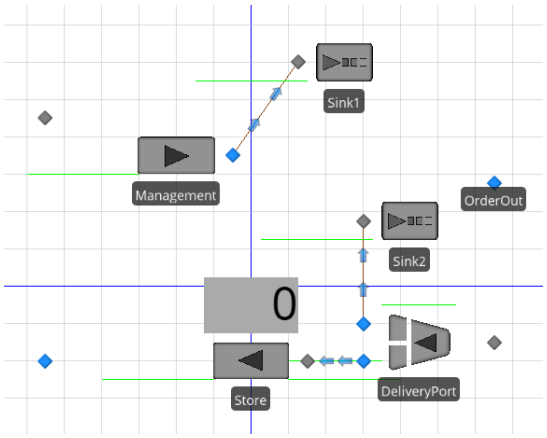
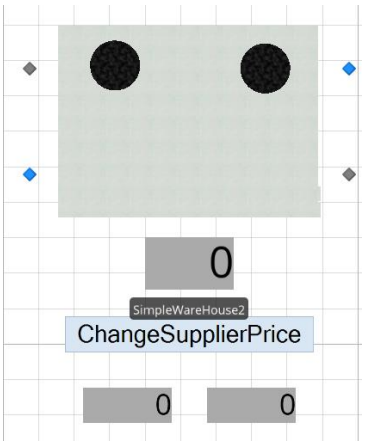
6.4.4.1 Abläufe

Das SimpleWareHouse umfasst die folgenden Abläufe:

- Entgegennehmen und Zerstören von OrderEntities.
- Überprüfen, ob die bestellte Menge im Lager vorhanden ist und basierend darauf, die Entscheidung, ob Waren geliefert werden können.
- Überprüfen, ob der Lagerbestand einen festgelegten Schwellwert erreicht hat, und das Auslösen einer Bestellung, falls dieser erreicht wurde.
- Beim Auslösen einer Bestellung wird überprüft, welcher von zwei Producern der günstigere Anbieter ist und anhand dessen wird entschieden, bei wem bestellt wird.
- Liefern von Produkten aus dem Lager an eine, von der Bestellung vorgegebene, Lieferadresse.
- Entgegennehmen von Produkten, welche in Containern verpackt ankommen und das Weiterleiten dieser in das interne Lager.

Im «Prototyp0» kann das vereinfachte Lager nur zwischen zwei Producern aussuchen und wenn beide die Produkte zum gleichen Preis anbieten, wird die Bestellung and den Producer 1 gesendet.

6.4.4.2 Aufbau

Interner Aufbau	Externe Modellansicht
	

Das SimpleWareHouse besteht aus:

- einem Management, implementiert als Server
- zwei Sinks, eine für das Zerstören von OrderEntities, die andere für Container
- einem Store, implementiert als Server
- einem DeliveryPort, implementiert als Separator

Eingehende OrderEntities werden vom Managment entgegen genommen, welches überprüft, ob noch genügend Produkte im Store vorhanden sind, um die Bestellung zu erfüllen. Ist dies der Fall wird dem Store der Auftrag erteilt, die bestellte Menge an die Lieferadresse zu senden.

Nach dem Erfüllen jeder Bestellung wird das Lager zusätzlich geprüft, ob der festgelegte Schwellwert erreicht wurde. Ist dies der Fall, erstellt das Management eine OrderEntity und schickt diese an den günstigeren Producer. Im «Prototyp0» gibt es nur 2 Producer und ihre Werte werden direkt im SimpleWareHouse festgelegt und verändert.

Das SimpleWareHouse hat noch eine weitere Funktion: es kann gelieferte Waren entgegennehmen. Dabei kommen diese vom Producer verpackt in Containern an und werden dann vom DeliveryPort auseinander genommen, Produkte werden weitergeleitet an den Store und Container werden zum Zerstören an die Sink2 geschickt.

Tabelle 11: Aufbau des SimpleWareHouse-Moduls

6.4.4.3 Einstellungen

Beim SimpleWareHouse gibt es mehrere verschiedene Einstellungen. Ein Teil davon ist spezifisch für den «Prototyp0», wie zum Beispiel die Producer-spezifischen Einstellungen, andere werden für weitere Prototypen um zusätzliche Funktionen erweitert werden.

Name	Einstellungsname	Initialwert
Produkt Typ	*ProductType*	*Product1*
(Property)	Legt fest, was für ein Produkttyp in diesem Lager gelagert und nachbestellt werden kann.	
Anzahl Produkte	*InitialNumberofProducts*	*50*
(Property)	Legt den Anfangsbestand des SimpleWareHouses fest, also wie viele Produkte beim Beginn der Simulation im Lager sind.	
Schwellwert	*[Initial]ReorderPoint*	*45*
(Property)(State)	Legt den Schwellwert fest, also ab wie vielen Produkten im Lager nachbestellt wird. Ist der Schwellwert 45 so wird bei der Bestellung, welche den Lagerbestand auf 45 oder darunter bringt, eine OrderEntity an die nächste Station geschickt. Im «Prototyp0» ist diese Bestellung immer für 35 Produkte und geht an den günstigeren Producer. Wenn die Kosten der Producer gleich sind, so geht die Bestellung an den Producer1.	
Hersteller 1	*Producer1*	*Input_Server1@Producer1*
(Property)	Legt die Adresse fest, an welche die Bestellung geschickt wird, um Producer1 zu erreichen. Dieser Wert entspricht dem Bestellungseingang vom Producer1.	
Hersteller 2	*Producer2*	*Input_Server1@Producer2*
(Property)	Legt die Adresse fest, an welche die Bestellung geschickt wird, um Producer2 zu erreichen. Dieser Wert entspricht dem Bestellungseingang vom Producer2.	
Kosten Hersteller 1	*[Initial]CostProducer1*	*15*
(Property) (State)	Im Property InitialCostPoducer1 wird festgelegt, was beim Start der Simulation der Einkaufspreis von Producer1 sind.	
Kosten Hersteller 2	*[Initial]CostProducer2*	*15*
(Property) (State)	Im Property InitialCostPoducer2 wird festgelegt, was beim Start der Simulation der Einkaufspreis von Producer2 sind.	

Zwischenspeicher Lieferadresse	*RememberDeliveryA-ddress*	
(State)	In diesem State wird die Lieferadresse zwischengespeichert, um die Lieferadresse von der Bestellung auf die zu liefernden Produkte übertragen zu können.	
Offene Bestellung	*OpenOrder*	*0*
(State)	Dieser State wird dafür gebraucht, dass das SimpleWareHouse überprüfen kann, ob es bereits eine Bestellung gesendet hat, die noch nicht erfüllt wurde. Es verhindert mehrere aufeinanderfolgende Bestellungen aufgrund der gleichen Überschreitung des Schwellwertes.	
Hersteller Preise ändern	*ChangeSupplierPrices*	
(Event)	Dieses Event gehört zum Button «ChangeSupplierPrice», welcher der vereinfachten Preisanpassung von den zwei Producern dient.	

Tabelle 12: Einstellungen des SimpleWareHouse-Moduls

6.4.4.4 Schnittstellen

Das SimpleWareHouse verfügt über vier Schnittstellen mit der restlichen Supply Chain. Rein technisch gesehen, könnten diese auf einen Eingang und einen Ausgang reduziert werden. Allerdings werden dann der Information- und Warenfluss nicht mehr sauber getrennt, was die Fähigkeit die Simulation auszubauen beschränken könnte. Daher wurden die Schnittstellen nicht reduziert.

Bestelleingang Der Bestelleingang wird beim SimpleWareHouse auch als Management Eingang bezeichnet. Er nimmt Bestellungen entgegen und leitet diese zur Bearbeitung ans Management weiter.

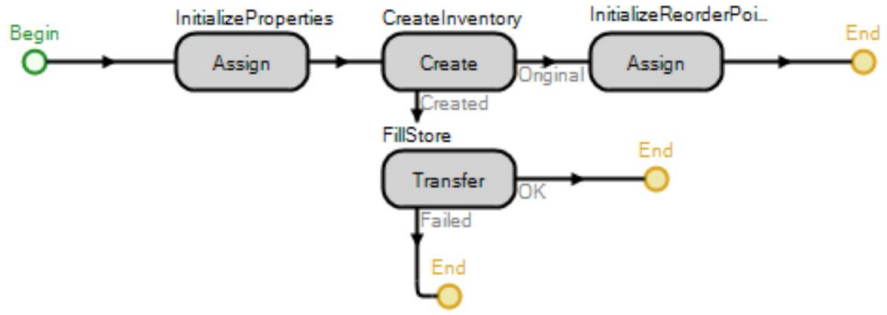
Warenausgang Produkte verlassen das SimpleWareHouse über diesen Knoten.

Wareneingang Container werden über diesen Knoten entgegengenommen.

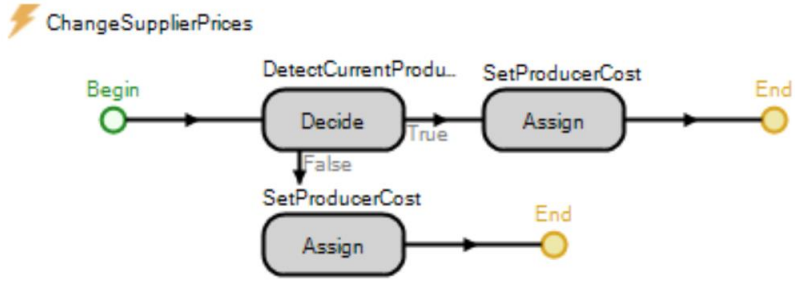
Bestellausgang Bestellungen werden über diesen Knoten verschickt.

6.4.4.5 Prozesse

OnRunInitialized



InitializeProperties	Weist dem Producer1 einen initialen Warenpreis zu.
CreateInventory	Erzeugt den Anfangsbestand der Produkte im Lager.

InitializeReorderPoint	Legt den Schwellwert fest.
FillStore	Verschiebt die erzeugten Produkte, also den Anfangsbestand an das InputNode des Lagers.
Process1	
	
DetectCurrentProducer	Überprüft ob Producer1 günstiger ist als Producer2.
SetProducerCost	Verändert die Warenkosten für Producer1 oder Producer2 und macht entweder Producer1 oder Producer2 teurer als den anderen.
DeliveryPort_ParentExited	
	
ResetContainerLabel	Sobald eine Warenlieferung das Lager erreicht, wird die Stückzahl auf einem Container auf 0 gesetzt.
Store_Entered	
	
SetOpenOrder	Sobald eine Warenlieferung das Lager erreicht, wird der Wert Open-Order auf 0 gesetzt.
Management_Processing	
Abbildung des Prozesses zu sehen auf Seite 37	
ReorderpointCrossed	Überprüft, ob der Schwellwert erreicht wurde.
CheckIfOpenOrder	Überprüft, ob bereits eine Bestellung an einen Producer offen ist.
CreateOrder	Sofern keine Bestellung offen ist, wird der Prozess gestartet, eine neue Bestellung zu eröffnen.
DecideOnProducer	Es wird geprüft, ob Producer1 günstiger als Producer2 ist.
SetStatesToProducer1	Sofern Producer1 günstiger ist, wird die Bestellungsadresse der Bestellung auf Producer1 gesetzt.

SetStatesToProducer2	Falls Producer2 günstiger ist, wird die Bestellungsadresse der Bestellung auf Producer2 gesetzt.
SetOpenOrder	Der Wert der OpenOrder wird auf 1 gesetzt.
SetOrderAddress	Die Lieferadresse wird auf das SimpleWareHouse gesetzt.
SendOutOrder	Die OrderEntity wird an die OutputNode des SimpleWareHouses geschickt.
EnoughInStore	Überprüft, ob noch genug Waren an Lager sind.
RememberDeliveryAddress	Speichert die Lieferadresse der OrderEntity.
StartSendingProcess	Sucht die bestellte Menge im Lager und nimmt diese raus.
DeliveryAddress	Ordnet den gefundenen Waren die zuvor gespeicherte Lieferaddresse zu.
SetDeliveryAddress	Setzt die nächste Node auf die Lieferadresse.
SendOutProduct	Versendet die Produkte.

Tabelle 13: SimpleWareHouse Prozesse

Management_Processing

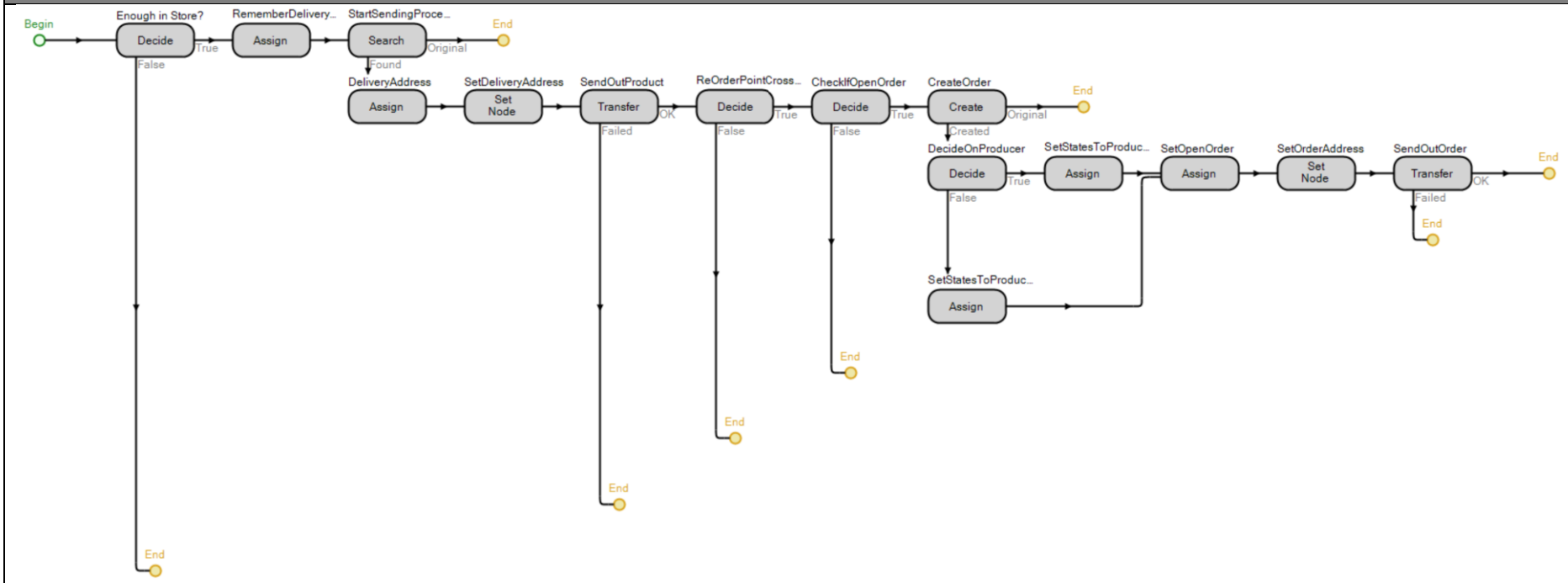


Abbildung 14: Management_Processing Prozess

7 AUSBLICK AUF ZUKÜNFTIGE UMSETZUNGEN

In diesem Kapitel werden die auf dem «Prototyp0» aufbauenden Prototypen 1 bis 3 kurz vorgestellt. Diese sollen in der auf diese Studienarbeit folgenden Bachelorarbeit genauer erarbeitet und praktisch umgesetzt werden. Beim Prototyp 3 handelt es sich zunächst um ein optionales Ziel, welches nur bei genügend Zeit umgesetzt wird.

Zusätzlich wird bereits eine Liste von möglichen Key Performance Indikatoren aufgeführt, die später aus der Simulation herauslesbar sein werden.

7.1 PROTOTYP 1

«Prototyp1» konzentriert sich darauf, die Chain unendlich erweiterbar zu machen. Es gibt zwei Ansätze, wie dies umgesetzt werden kann. Bei Beiden werden weiterhin ein oder mehrere *Customer* und *Producer* als Endstationen der Chain verwendet.

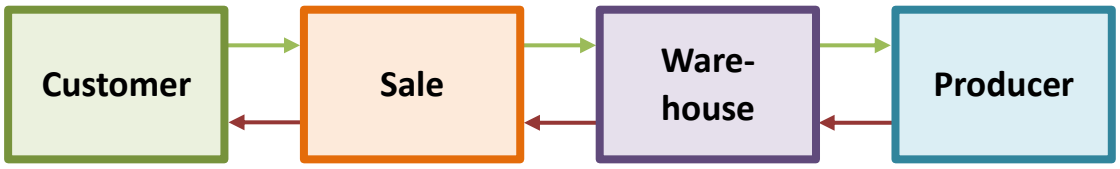


Abbildung 15: Prototyp 1, Ansatz 1

Der erst Ansatz sieht vor, dass ein zweiter Typ vom *Warehouse* definiert wird. Dieser verschickt die Produkte ebenfalls in Containern, beliefert aber nur andere *Warehouses*. Abbildung 15 zeigt wie das bestehende Warehouse-Modul zu *Sale* wird und ein zusätzlichen *Warehouse* eingefügt wird. Dieses zusätzliche *Warehouse-Modul*, hier in Violett, kann dann mehrmals hintereinander eingefügt werden. Dabei werden die Waren auf den Wegen zwischen *Producer*, *Warehouse* und *Sale* in Containern verschickt und zwischen *Sale* und *Customer* in Boxen verpackt.

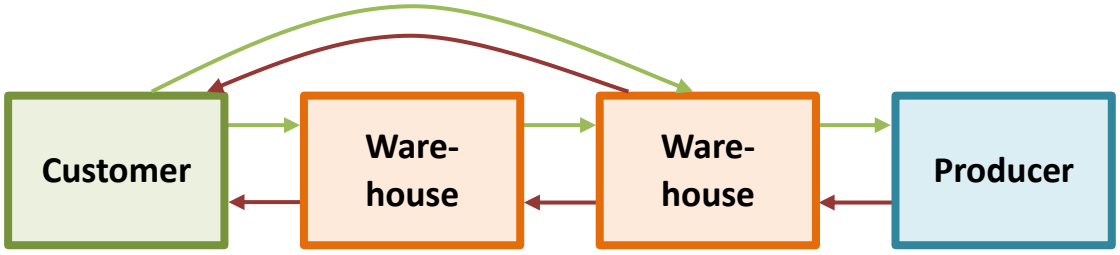


Abbildung 16: Prototyp 1, Ansatz 2

Im zweiten Ansatz wird das bestehende *Warehouse* soweit ausgebaut, dass es Produkte sowohl an den *Customer* als auch an andere *Warehouses* senden kann. Dafür muss neu auf den OrderEntities festgelegt werden, ob die Lieferadresse ein *Customer* oder ein *Warehouse* ist. Das *Warehouse* entscheidet anhand dessen, ob die Produkte in Container oder in Boxen verpackt werden sollen. Diese Variante sieht eine Anpassung von *Customer*, *Warehouse* und OrderEntity vor. Und es ermöglicht ein unbegrenztes Erweitern der Supply Chain mit dem Hintereinanderschalten von *Warehouses*.

In beiden Varianten sollen zusätzlich die Container jeweils zurückgeschickt und wiederverwertet werden.

7.2 PROTOTYP 2

Auch bei «Prototyp2» gibt es zwei Ansätze, welche für die zukünftige Umsetzung in Betracht gezogen werden. Anders als bei «Prototyp1» ist es möglich, beide Ansätze umzusetzen, da sie weniger gegensätzlich sind.

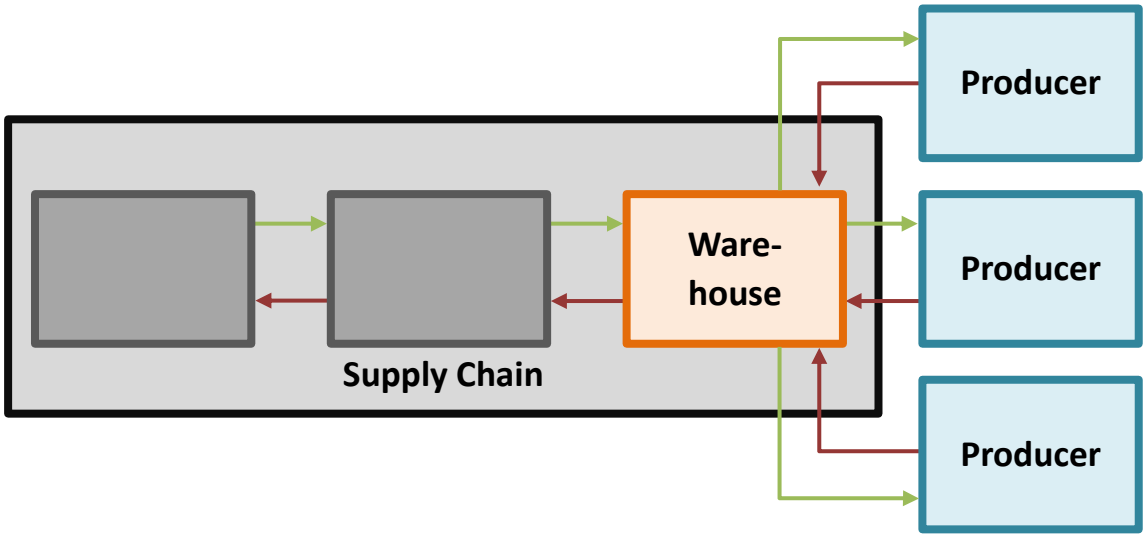


Abbildung 17: Prototyp 2, Ansatz 1

Ansatz 1 konzentriert sich auf die Erweiterbarkeit der *Warehouse*-Logik. Diese soll so erweitert werden, dass sie bei mehreren *Producern* anhand einer Matrix entscheiden kann, bei welchem *Producer* bestellt wird. Die Kriterien für die Entscheidungsmatrix sollen in einer Maske einfüllbar und veränderbar sein. Diese Kriterien umfassen Werte wie Preis/Menge, Verfügbarkeit, Qualität, Lieferzeit und so weiter.

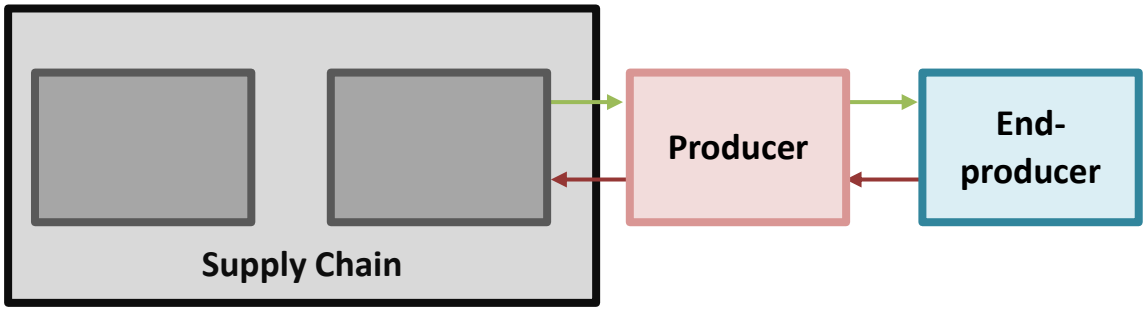


Abbildung 18: Prototyp 2, Ansatz 2

Im zweiten Ansatz geht es darum, einen *Producer* der Supply Chain zuzufügen, welcher verschiedene Produkte von Endproducern bezieht und aus diesen ein anderes Produkt erstellt, welches dann in die Supply Chain fliesst. Für diese Umsetzung werden mindestens zwei neue Produktarten benötigt und für zukünftige Erweiterbarkeit soll auch auf allen Stationen eingestellt werden können, welche Produkte sie bestellen und lagern.

Des Weiteren soll für beide Varianten ein Bestellformular zur Verfügung gestellt werden, welches es dem Simulationsbenutzer erlauben, Bestellmengen anzugeben und auszulesen.

7.3 PROTOTYP 3

Im «Prototyp3» sollen die *Warehouses* dann für das Handling von mehreren verschiedenen Produkten ausgebaut werden. Der Fokus in diesem Prototyp liegt auf den Ressourcen. Dafür werden die Lieferwege detaillierter implementiert, zum Beispiel indem eine begrenzte Anzahl Lieferwagen verwendet werden. Zusätzlich zu den Fahrzeugen sollen Arbeiter als Ressource in den *Warehouses* und *Producern* verwendet werden. Diese Änderungen sollen zur besseren Simulation von Lieferzeiten und Verzögerungen, zum Beispiel durch Arbeiterausfälle, verhelfen. Auch Mehrkosten beim Transport und verzögerte Transporte sollen so simuliert werden können.

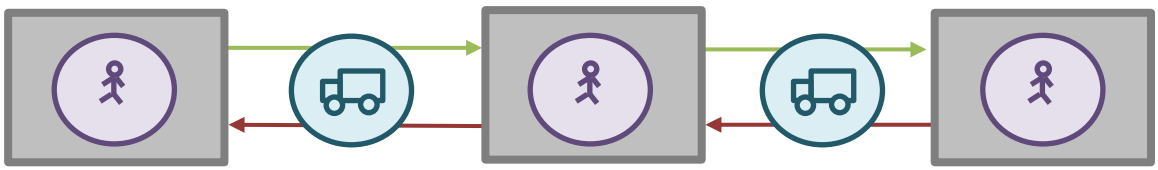


Abbildung 19: Prototyp 3, Ansatz 1

7.3.1 KEY PERFORMANCE INDIKATOREN

Nachfolgend eine nicht abgeschlossene Liste von möglichen Key Performance Indikatoren, die in der nachführenden Bachelorarbeit weiter definiert werden. Die Werte geben Auskunft darüber, wie effizient die simulierte Supply Chain ist und zeigen an, wie man die Supply Chain weiter optimieren kann. Einige dieser Indikatoren sind weiter unterteilbar in minimale, durchschnittliche und maximale Werte.

KPI	Beschreibung
Due Time	Wie lange dauert es von der Bestellung bis zur Lieferung.
Product Availability	Ist ein Produkt beim Lager oder Verkauf direkt verfügbar oder muss es zuerst noch bestellt werden bei einem anderen Lager oder dem Hersteller.
Inventory Turnover	Wie lange dauert es, bis das gesamte Inventar eines Lagers verkauft wurde, respektive jeder Artikel im Inventar durch einen neu bestellten ersetzt wird.
Delay Time	Die Abweichung zum Durchschnittswert der *Due Time*.
Days in Inventory	Wie lange liegt ein Produkt durchschnittlich im Lager.

Tabelle 14: Key Performance Indikatoren einer Supply Chain

8 ABSCHLUSS

Ergebnisse

In dieser Arbeit ist ein einfaches Supply Chain Model für Simio erarbeitet und umgesetzt worden. Es besteht aus den Kernkomponenten Customer, Simple-Warehouse und Producer. Dieser «Prototyp0» dient als Grundlage für zukünftige, komplexere Prototypen.

Die Grundkomponenten sind möglichst erweiterbar konfiguriert worden, jedoch sind sie zum jetzigen Zeitpunkt nicht alle mehrfach ins Modell einfügbar. So benötigt das Hinzufügen eines zusätzlichen Producers zum Beispiel noch manuelle Anpassungen am Modell selbst, um einwandfrei funktionieren zu können. Auch ist es mit den jetzigen Einstellungen nur möglich ein einziges Simple-Warehouse in der Simulation zu haben.

Das erarbeitete Modell bildet ein gutes Fundament für zukünftige Vertiefungsarbeiten, viele Funktionen für die geplanten Prototypen aus Kapitel 7 sind aber noch nicht implementiert.

Probleme

Aufgrund der Einschränkungen der Studentenlizenz für Simio gab es zwischendurch Probleme. Da mehrere neue Module definiert und erstellt wurden, war der Umfang der Lizenz schnell ausgeschöpft. Dies konnte teilweise umgangen werden, indem aus dem Modell heraus eine neue Library erstellt wurde für die ProductEntity. Das war jedoch nicht immer so einfach möglich, zum Beispiel hat das Auslagern der OrderEntity in eine Library nicht funktioniert, da andere Prozesse auf die OrderEntity zugreifen und eine saubere Trennung so nicht möglich ist.

Da diese Arbeit während der weltweiten Coronakrise verfasst wurde, war in den ersten Wochen der Durchführung eine Umstellung auf einen rein virtuellen Austausch mit dem Betreuer und der Verfasser untereinander notwendig. Diese Umstellung hat jedoch gut funktioniert.

Weiterführende Fragen

Nach der Umsetzung der bereits geplanten Prototypen aus Kapitel 7 ist zudem die Betrachtung von Sonderfällen der Supply Chain interessant. Zum Beispiel Pandemie-spezifische Anpassungsmöglichkeiten für die Simulation von Supply Chains während Krisenzeiten: Kurzfristig stark erhöhte Nachfrage, die sich langfristig dann aber wieder stark verringert (Beispiel Toilettenpapier Supply Chain).

Empfehlung

Für zukünftige Arbeiten, die sich mit Baukastensystemen für Simio beschäftigt, empfiehlt es sich von Anfang an eine umfangreiche Lizenz zu erwerben. Ebenfalls sollen von Anfang an aussagekräftige Namen für Prozessschritte, States und Namen verwendet werden. Einfache Modelle können sehr schnell sehr komplex werden, vor Allem wenn Prozessdefinitionen erstellt werden.

9 VERZEICHNISSE

9.1 BEGRIFFE UND DEFINITIONEN

Residental Use Wohnzwecke, Wohngebrauch in diesem Fall als «wird in Haushalten verbraucht» verwendet.

Commercial Use Kommerzielle Nutzung von Produkten.

SCOR Supply-Chain-Operations-Reference-Modell

State Zustand, welcher als solcher auf einem Simio Objekt erfasst werden kann.

Supply Chain Lieferkette

Akkumuliert ansammeln, anhäufen, zusammentragen

9.2 ABBILDUNGSVERZEICHNIS

9.3 TABELLENVERZEICHNIS

9.4 LITERATURVERZEICHNIS

Holweg, M. and Bicheno, J., 2002. Supply chain simulation – a tool for education, enhancement and endeavour. *International Journal of Production Economics*, 78(2), pp.163-175.

Ivanov, D., 2019. Disruption tails and revival policies: A simulation analysis of supply chain design and production-ordering systems in the recovery and post-disruption periods. *Computers & Industrial Engineering*, 127, pp.558-570.

Poluha, R., 2010. *Anwendung Des SCOR-Modells Zur Analyse Der Supply Chain*. Lohmar: Eul.

Donno, E., 2020. *Baukastensystem für Simio-Modellbibliothek,* BAWVOR

Rinkel, A., Hollenstein, L., Version 1.5.1, 2018. *Einsatz formaler Methoden in der Simulation.*

Bullwhip Effekt – einfache Definition & Erklärung, https://www.rechnungswesen-verstehen.de/lexikon/bullwhip-effekt.php, abgerufen am 30.03.2020

BPMN Quick Guide, https://cloud.trisotech.com/bpmnquickguide/index.html?connecting_object_basics.htm, abgerufen am 04.04.2020

Log in Visual Paradigm Online. Online.visual-paradigm.com. https://online.visual-paradigm.com/w/pqtfonxt/diagrams/#proj=0&type=SequenceDiagram. Published 2020, abgerufen am 06.05.2020

Toilet paper 3d model. Modelplusmodel.com. https://www.modelplusmodel.com/accessories/bath-accessories/f01-tolet-paper.html. Published 2020, abgerufen am 28.04.2020